管理員指南

Sun™ ONE Application Server

版本 7

817-7255-10
2003 年 9 月

目錄

關於本指南 19

本指南內容 19
本指南的組成部分 20
　第一部分：伺服器基本原理與管理全域設定 20
　第二部分：管理個別伺服器實例 20
　第三部分：管理 HTTP 伺服器功能與虛擬伺服器 21
　第四部分：附錄 21
文件慣例 22
　一般慣例 22
　參考目錄的慣例 23
產品系列簡介 24
　平台版 24
　標準版 24
　企業版 25
使用文件 25
產品支援 27

第 1 部分　伺服器基本原理與管理全域設定 29

第 1 章　Sun ONE Application Server 管理快速入門 31
關於 Sun ONE Application Server 32
配置隨附式 Solaris 版本 33
　建立管理網域 33
　啓動管理伺服器 34

建立應用程式伺服器實例 34
部署應用程式 34
使用管理介面 35
存取管理介面 35
使用標籤 37
使用按鈕 39
存取線上說明 39
退出管理介面 40
使用指令行介面 41
存取管理伺服器 41
存取應用程式伺服器實例 41
使用 Sun ONE Studio 42
關於配置檔案 42
使用授權指令 42

第 2 章 設定管理伺服器偏好設定 45
關於管理伺服器 46
啓動管理伺服器 47
使用 startserv 程序檔 47
使用指令行介面 48
使用 [服務] 視窗 (Windows) 48
使用 [開始] 功能表 (Windows) 48
關閉管理伺服器 49
使用管理介面關機 49
使用 stopserv 程序檔關機 49
使用指令行介面關機 50
使用 [服務] 視窗關機 (Windows) 50
存取管理伺服器設定 51
檢視管理伺服器控制設定 52
將變更套用至管理伺服器 52
編輯管理伺服器的 HTTP 偵聽程式設定 53
設定 SNMP、記錄和安全性偏好設定 53

第 3 章 配置管理網域 55
關於管理網域 55
執行管理網域 56
目錄結構 56
程序 / 連接埠結構 56

配置網域 56
建立網域 57
範例：在預設位置建立領域 57
範例：在某個位置 (非預設位置) 建立領域 57
範例：爲其他使用者建立領域 (僅用於 UNIX) 57
UNIX 平台上的使用者許可權 58
刪除網域 58
範例：刪除領域 58
列示網域 59
範例：列示本端機器上的領域 59
範例：使用遠端選項列示本端機器上的領域 59
啓動網域 59
範例：啓動機器上的唯一領域： 59
停用網域 60
範例：停用領域中除管理伺服器實例之外的所有實例 60
重新建立網域註冊 60

第 2 部分 管理個別伺服器實例 61

第 4 章 使用應用程式伺服器實例 63
關於應用程式伺服器實例 64
啓動和停止應用程式伺服器實例 65
使用管理介面中的 [Start] 和 [Stop] 按鈕 65
使用 start-instance 與 stop-instance 指令 66
使用 Windows 服務 (Windows) 66
使用 startserv 和 stopserv 程序檔 67
在除錯模式下啓動應用程式伺服器實例 68
設定終止逾時 68
自動重新啓動應用程式伺服器實例 (UNIX) 69
關於自動重新啓動 69
以 /etc/inittab 自動重新啓動 (UNIX) 70
以系統 RC 程序檔自動重新啓動 (UNIX) 70
手動重新啓動應用程式伺服器實例 (UNIX) 70
使用 [Restart] 按鈕重新啓動伺服器實例 (UNIX) 71
使用 restart-instance 指令重新啓動伺服器實例 (UNIX) 71
使用 restartserv 程序檔重新啓動伺服器實例 (UNIX) 71
關於監視程式 72
加入應用程式伺服器實例 72
刪除應用程式伺服器實例 73
將變更套用至應用程式伺服器實例 74
檢視應用程式伺服器實例狀況 75

配置 JVM 設定 76
配置一般設定 76
配置路徑設定 77
配置 JVM 選項 78
配置 JVM 測量程式 78
使用指令行介面配置 JVM 設定 79
配置記錄設定和監視設定 79
變更應用程式伺服器實例進階設定 80

第 5 章 使用記錄功能 81
關於記錄 82
在 UNIX 和 Windows 平台上記錄 82
server.log 內的預設記錄 83
server.log 的範例 83
使用 syslog 進行記錄 84
配置 syslog 85
配置 syslog 的步驟： 85
syslog 訊息範例 87
使用 Windows 事件日誌記錄 88
使用日誌層級 88
關於日誌層級 88
用於 syslog 配置的日誌層級 90
關於虛擬伺服器和記錄 90
關於記錄程式 92
關於用戶端記錄 94
重新導向應用程式與伺服器日誌輸出 94
日誌檔管理 95
內部常駐程式日誌旋轉 96
基於排程程式的日誌旋轉 96
使用 Solaris logadm 公用程式的旋轉 97
使用 Solaris「cron」公用程式的旋轉 99
關於 crontab 項目格式 100
使用 Solaris cron 公用程式排程 logadm 的執行 100
透過指令行介面配置記錄 101
透過管理介面配置記錄 102
配置日誌服務 102
爲應用程式伺服器元件和子系統配置記錄 105
指定日誌層級的步驟 105
指定日誌檔的步驟：(虛擬伺服器) 105
指定作業事件日誌位置的步驟： (Java Transaction Service) 106
配置錯誤記錄的指令 106
檢視存取日誌檔 106

檢視事件日誌檔 108
設定日誌偏好設定 110
執行日誌分析程式 111
檢視事件 (Windows 2000 Pro) 113

第 6 章 監視 Sun ONE Application Server 115
關於監視 Sun ONE Application Server 115
統計資料 116
SNMP 116
監視 HTTP 伺服器 117
監視應用程式元件與子系統 117
監視容器子系統 118
監視 ORB 服務 118
監視作業事件服務 118
服務品質 (QOS) 119
使用 CLI 擷取監視資料 119
list --monitor 指令 120
get --monitor 指令 120
CLI 名稱對映 121
Petstore 範例 122
可監視物件類型 124
可監視屬性名稱 126
HTTP 伺服器的可監視物件 130
可監視的 HTTP 伺服器元素 131
可監視的 HTTP 伺服器屬性 132
使用 CLI 管理作業事件服務 138
使用 HTTP 服務品質 139
服務品質的範例 139
配置服務品質 (QOS) 140
對 obj.conf 檔案的必要變更 143
服務品質的已知限制 143
關於 SNMP 144
網路管理站 (NMS) 145
管理資訊庫 (MIB) 物件 146
SNMP 訊息 150
SNMP 陷阱目標 150
SNMP 代理程式團體 151
設定 SNMP 151
使用 SNMP 代理程式 (UNIX/Linux) 152
安裝 SNMP 代理程式 153
啓動 SNMP 代理程式 153
重新啓動本端 SNMP 常駐程式 154
安裝 SNMP 主代理程式 154

啓用與啓動 SNMP 主代理程式 157
在其他連接埠上啓動主代理程式 157
手動配置 SNMP 主代理程式 158
編輯主代理程式 CONFIG 檔案 158
定義變數 sysContact 與 sysLocation 158
配置 SNMP 子代理程式 159
啓動 SNMP 主代理程式 161
手動啓動 SNMP 主代理程式 161
使用管理伺服器啓動 SNMP 主代理程式 162
啓用子代理程式 163

第 7 章 配置 Web 伺服器外掛程式 165
關於 Web 伺服器外掛程式 165
處理用戶端請求 166
HTTP 基本原理 166
請求處理程序中的步驟 168
Web 伺服器外掛程式配置 169
Web 伺服器外掛程式 SAF 參考 170
init-passthrough 170
auth-passthrough 170
service-passthrough 171
check-passthrough 172
使用 Web 伺服器外掛程式 173
對 Sun ONE Web Server 的變更 173
對 Sun ONE Application Server 的變更 174
配置 Microsoft IIS 以使用 Web 伺服器外掛程式 175
配置用於 IIS 的 Web 伺服器外掛程式 176
配置 IIS 以使用 Web 伺服器外掛程式 177
配置多重伺服器儲存區 178
sun-passthrough.properties 檔案範例 178
配置 Apache Web 伺服器 180
最低需求 180
使用 mod_proxy 模組編譯 Apache 181
修改 httpd.conf 檔案 182
啓動與停止 Apache 183

第 8 章 配置 J2EE 容器 185
關於 Web 容器 185
瞭解 Web 容器的角色 187
Web 應用程式配置 187
虛擬伺服器屬性 187
Web 模組屬性 188

Web 應用程式部署 189
動態重新部署與熱部署 189
單一登入功能 190
記錄 Web 容器 190
關於 EJB 容器 191
瞭解 EJB 容器的角色 192
企業 Java Bean 類型 193
關於訊息導引 Bean 195
配置 EJB 容器 196
執行一般配置 197
配置 EJB 設定 199
配置 MDB 儲存區設定 201

第 9 章 使用作業事件服務 203
何爲作業事件？ 204
J2EE 中的作業事件 204
作業事件資源管理員 205
資料庫 205
JMS 提供者程式 206
J2EE 連接器 206
本機作業事件與分散式作業事件 206
容器管理作業事件 208
作業事件屬性 209
Required 209
RequiresNew 210
Mandatory 210
NotSupported 210
Supports 210
Never 211
屬性摘要 211
設定作業事件屬性 211
回轉容器管理作業事件 212
同步化階段作業 Bean 的實例變數 213
容器管理作業事件中禁用的方法 214
Bean 管理式的作業事件 214
作業事件服務管理 216
使用管理介面管理作業事件 216
使用指令行介面管理作業事件 218
列示執行中的作業事件 219
管理作業事件 219
凍結作業事件服務 219
監視作業事件 220

第 10 章 配置命名與資源 221
關於 J2EE 命名服務與資源 221
JDBC 資料來源 222
Java 郵件階段作業 222
JMS 目標 222
關於 Java 命名與目錄介面 (JNDI) 223
JNDI 架構 223
J2EE 命名服務 224
命名參考與連結資訊 224
J2EE 標準部署描述元中的命名參考 225
應用程式環境項目 226
EJB 參考 226
資源管理程式連線 Factory 的參考 227
資源環境參考 228
UserTransaction 參考 229
初始命名環境 229
COSNaming 服務 229
JNDI 連線 Factory 230
建立自訂資源的步驟 232
建立外部 JNDI 資源 233
存取外部 JNDI 儲存庫 235
對映應用程式資源參考 235
關於 URL 連線 Factory 資源 236
對映應用程式資源環境參考 236
對映 EJB 參考 237
關於持續性管理程式資源 237
什麼是持續性？ 238
持續性管理程式的角色 239
預先部署 Bean 配置 239
建立新的持續性管理程式 241
關於 JDBC 資源 242
關於 JDBC API 243
JDBC API 的功能？ 243
關於資料庫存取模型 244
關於 JDBC 資料來源 244
DataSource 物件的特性 245
註冊 JDBC 資源 246
關於 JDBC 連線 248
關於 JDBC URL 249
配置 JDBC 連線區 250
關於連線匯集 258
監視 JDBC 連線匯集 259
關於連線共用 259

關於 JDBC 作業事件 260
關於 Java 郵件資源 261
關於 JavaMail 訊息處理程序 262
關於 JavaMail 的架構元件 263
Message 類別 263
訊息儲存與擷取 264
訊息構成與傳輸 264
關於 JavaBean 啓動框架 (JAF) 264
關於 JavaMail 配置參數 265
JavaMail 階段作業參考的 J2EE 部署描述元 267
Sun ONE Application Server 部署描述元中的項目 267
建立新的 JavaMail 階段作業 268
配置進階資源特性 269

第 11 章 使用 JMS 服務 271
關於 JMS 272
基本訊息傳送系統概念 272
訊息 272
訊息服務架構 273
訊息發送模型 273
JMS 規格 274
JMS 訊息結構 274
JMS 程式設計模型 274
管理物件：提供者獨立性 275
訊息導引 Bean 276
內建 JMS 服務 278
關於 Sun ONE Message Queue (MQ) 278
MQ 訊息伺服器 279
MQ 用戶端執行期間 280
MQ 管理物件 281
MQ 管理工具 281
整合 MQ 與 Sun ONE Application Server 282
內建 JMS 服務架構 282
停用內建 JMS 服務 283
內建 JMS 服務的管理 284
配置 JMS 服務 285
管理實體目標 287
建立佇列或主題目標 288
列示實體目標 289
刪除實體目標 289

管理受管理物件資源 290
管理物件屬性 291
管理物件資源管理工作 291
使用指令行介面管理內建 JMS 服務 296

第 12 章 配置 CORBA/IIOP 用戶端伺服器 297
關於 CORBA/IIOP 用戶端支援 297
關於互用性 298
關於 ORB 298
關於 RMI/IIOP 功能性 298
關於認證程序 299
配置 ORB 299
執行一般 ORB 配置 300
配置 ORB 的 IIOP 偵聽程式 302

第 13 章 部署應用程式 307
關於 J2EE 模組 308
關於 J2EE 應用程式 309
J2EE 標準描述元 309
Sun ONE Application Server 描述元 310
命名標準 311
部署目錄結構 312
運行時間環境 313
模組執行環境 313
應用程式執行環境 314
配置 server.xml 以使用 FastJavac 編譯器 315
關於類別載入器 316
部署模組與應用程式 316
部署名稱與錯誤 317
部署生命週期 317
動態部署 317
停用已部署的應用程式或模組 317
動態重新載入 318
部署工具 319
asadmin 公用程式 319
管理介面 320
Sun ONE Studio 321
部署模組或應用程式 321
部署 WAR 模組 322
部署 EJB JAR 模組 322

部署生命週期模組 322
asadmin 公用程式 323
管理介面 323
部署 RMI/IIOP 用戶端 324
部署 J2EE CA 資源介面 325
部署靜態內容 325
存取共用框架 325
應用程式部署描述元檔案 325

第 3 部分 管理 HTTP 伺服器功能與虛擬伺服器 327

第 14 章 配置 HTTP 功能 329
關於 HTTP 功能 329
配置檔案快取記憶體 330
微調伺服器以提昇效能 330
配置 HTTP 服務品質 331
加入與使用執行緒儲存區 333
編輯進階設定 333
配置 MIME 類型 334

第 15 章 使用虛擬伺服器 337
虛擬伺服器簡介 337
HTTP 偵聽程式 338
虛擬伺服器 339
虛擬伺服器類型 340
基於 IP 位址的虛擬伺服器 340
基於 URL 主機的虛擬伺服器 340
預設虛擬伺服器 341
obj.conf 檔案 341
用於請求處理的虛擬伺服器選取 342
文件根 342
以虛擬伺服器使用 Sun ONE Application Server 功能 343
配合虛擬伺服器使用 SSL 343
使用存取日誌檔和伺服器日誌檔 344
配合虛擬伺服器使用存取控制 344
配合虛擬伺服器使用 CGI 344
建立和配置 HTTP 偵聽程式 344
建立 HTTP 偵聽程式 345
編輯 HTTP 偵聽程式設定 346
刪除 HTTP 偵聽程式 347

建立和配置虛擬伺服器 347
建立虛擬伺服器 347
必需的設定 348
選擇性一般設定 348
Web 應用程式設定 349
CGI 設定 350
HTTP 服務品質設定 350
編輯虛擬伺服器設定 351
使用管理介面編輯一般設定 351
使用指令行介面編輯一般設定 351
編輯 CGI 設定 352
編輯文件處理設定、文件目錄設定以及 HTTP/HTML 設定 352
刪除虛擬伺服器 352
部署虛擬伺服器 353
範例 1 ：預設配置 353
範例 2 ：安全伺服器 355
範例 3 ：企業網路主機作業 356
範例 4 ：大量主機作業 358

第 16 章 管理虛擬伺服器內容 361
變更文件根 362
設定附加文件目錄 362
啓動遠端檔案處理 363
使用 htaccess 364
限定符號式連結 (UNIX) 364
自訂使用者公用資訊目錄 (UNIX) 365
配置公用資訊目錄 365
限定內容發佈 366
啓動時載入整個密碼檔案 367
設定文件偏好設定 367
輸入索引檔名 367
選取目錄索引 368
指定伺服器首頁 368
指定預設 MIME 類型 368
自訂錯誤回應 369
變更國際字元集 369
設定文件註腳 371
配置 URL 轉寄 372
設定伺服器剖析的 HTML 372
設定快取控制指令 373
使用更強密碼 374

第 4 部分 附錄 375

附錄 A 使用指令行介面 377
關於指令行介面 377
關於 asadmin 公用程式 377
關於 Ant 工作 378
關於其他指令行公用程式 378
使用 asadmin 379
瞭解指令語法 379
指令 380
選項 380
布林選項 380
運算元 380
語法範例 381
使用單模式與多重模式 381
單模式 381
多重模式 381
多重模式內的多重模式 382
使用互動式與非互動式選項 382
使用環境指令 383
使用密碼檔案選項 385
本機或遠端運行 asadmin 385
使用指令行呼叫 386
從指令行使用 asadmin 386
以檔案 (程序檔) 中的輸入，使用 asadmin 387
以標準輸入 (管道) 使用 asadmin 387
使用逸出字元 387
在單模式下的 UNIX 中之逸出字元 388
在單模式下的 Windows 中之逸出字元 388
在單模式下的所有平台中之逸出字元 389
在多重模式下的所有平台中之逸出字元 389
使用 get 與 set 指令 389
get 與 set 指令範例 390
取得並設定多個值範例 391
使用 get 與 set 指令監視 391
使用輔助說明 392
檢視輸出與錯誤 392
檢視結束狀況 392
檢視用法 394
安全性考量 394
並行存取考量 395
指令參考 395
指令清單 395

點名稱與屬性清單 399
在 asadmin 中使用的點名稱 400
服務名稱 400
資源名稱 401
應用程式名稱 401
其他名稱 402
屬性 402
jms-service 403
transaction-service 403
mdb-container 404
ejb-container 405
web-container 406
java-config 407
orb 或 iiop-service 408
orblistener 或 iiop-listener 409
log-service 410
security-service 411
http-service 411
jdbc-resource 412
jndi-resource 413
jdbc-connection-pool 413
custom-resource 414
jms-resource 415
persistence-manager-factory-resource 416
mail-resource 416
application 417
ejb-module 418
web-module 419
connector-module 420
http-listener 或 http-server.http-listener 421
mime 422
acl 423
virtual-server 423
auth-db 425
authrealm 425
lifecycle-module 426
profiler 427
server configuration (伺服器實例名稱) 427
長選項格式與短選項格式、預設值與對等的環境變數 428

附錄 B　協力廠商版權備註 433

詞彙表 435

索引 457

關於本指南

本指南描述配置與管理 Sun™ ONE Application Server 7 的方式。本指南是針對公司企業中的資訊技術管理員而編寫的，希望透過全球資訊網將用戶端及伺服器應用程式推介給更多的讀者。

此前言包括以下章節：

- 本指南內容
- 本指南的組成部分
- 文件慣例
- 產品系列簡介
- 使用文件
- 產品支援

本指南內容

本指南解釋配置與管理 Sun ONE Application Server 的方式。配置您的伺服器之後，請使用本指南協助維護此伺服器。

本指南的組成部分

本指南分爲四個部分，另外包括一個綜合索引部分。從第 1 部分「伺服器基本原理與管理全域設定」開始，該部分簡單介紹此產品。第 2 部分「管理個別伺服器實例」向您介紹如何使用管理伺服器，以及其他伺服器功能 (影響所有伺服器實例) 的資訊。

瞭解了使用管理伺服器的基本資訊之後，可以參考第 3 部分「管理 HTTP 伺服器功能與虛擬伺服器」，該部分提供使用程式和配置樣式的資訊。

最後，附錄論述描述各種主題的特定參考主題，其中包括國際化問題、伺服器擴充以及 Sun ONE Application Server 指令行介面說明文件。

第一部分：伺服器基本原理與管理全域設定

此部分簡單介紹 Sun ONE Application Server。其中包括以下各章：

- 第 1 章「Sun ONE Application Server 管理快速入門」簡單介紹 Sun ONE Application Server。
- 第 2 章「設定管理伺服器偏好設定」描述管理管理伺服器的方式。
- 第 3 章「配置管理網域」描述使用多重網域的方式。

第二部分：管理個別伺服器實例

此部分提供關於配置、管理以及使用伺服器實例的概念與程序詳細資訊。其中包括以下各章：

- 第 4 章「使用應用程式伺服器實例」描述配置 Sun ONE Application Server 伺服器個人喜好的方式。
- 第 5 章「使用記錄功能」描述 Sun ONE Application Server 中的記錄基礎以及記錄功能。
- 第 6 章「監視 Sun ONE Application Server」包含有關監視以及在 Sun ONE Application Server 中可用的簡單網路管理協定 (SNMP) 功能的資訊。
- 第 7 章「配置 Web 伺服器外掛程式」解釋 Sun ONE Application Server 處理 HTTP 請求的方式，以及配置 Web 伺服器外掛程式並使之與 Sun ONE Application Server 協同工作的方式。
- 第 8 章「配置 J2EE 容器」解釋如何配置與使用爲 J2EE 應用程式元件 (例如，企業 Java Bean [EJB] 與訊息導引 Bean [MDB]) 提供運行時間支援的容器。
- 第 9 章「使用作業事件服務」解釋資料庫作業事件以及管理它們的方式。

- 第 10 章「配置命名與資源」解釋配置 J2EE 資源的方式。
- 第 11 章「使用 JMS 服務」提供瞭解與管理由 Sun ONE Message Queue (原生 JMS 提供者) 提供的內建 JMS 服務所需的資訊。
- 第 12 章「配置 CORBA/IIOP 用戶端伺服器」解釋在 Sun ONE Application Server 環境中使用 RMI/IIOP 協定，配置對基於 CORBA 的用戶端的支援之方式。
- 第 13 章「部署應用程式」描述將應用程式部署至 Sun ONE Application Server 的方式。

第三部分：管理 HTTP 伺服器功能與虛擬伺服器

此部分提供使用管理介面存取程式與配置樣式的資訊。其中包括以下各章：

- 第 14 章「配置 HTTP 功能」描述配置 Sun ONE Application Server 的 HTTP 相關功能之個人喜好的方式。
- 第 15 章「使用虛擬伺服器」描述使用 Sun ONE Application Server 設定與管理虛擬伺服器的方式。
- 第 16 章「管理虛擬伺服器內容」描述您配置與管理伺服器內容的方式。

第四部分：附錄

此小節包括您要複查的各種參考材料附錄。此章節包括以下附錄：

- 附錄 A「使用指令行介面」提供使用指令行公用程式代替使用者介面螢幕的說明。
- 附錄 B「協力廠商版權備註」包含附加版權資訊。

文件慣例

此章節描述整個指南中所使用的慣例類型：

- 一般慣例
- 參考目錄的慣例

一般慣例

本指南使用下列一般慣例：

- **檔案與目錄路徑**以 UNIX® 格式給定 (使用正斜線分隔目錄名稱)。對於 Windows 版本，目錄路徑相同，但是使用反斜線分隔目錄。
- **URL** 的給定格式爲：

  http://*server.domain*/*path*/*file*.html

  在這些 URL 中，*server* 爲運行應用程式的伺服器名稱；*domain* 爲您的網際網路網域名稱；*path* 爲伺服器的目錄結構；*file* 爲個別檔案名稱。URL 中的斜體項目爲版面配置區。
- **字型慣例**包括：
  - 固定間距字型用於範例碼和程式碼清單、API 和語言元素 (例如函式名稱與類別名稱)、檔案名稱、路徑名稱、目錄名稱以及 HTML 標籤。
  - **斜體**類型用於程式碼變數。
  - **斜體**類型也用於書名、強調、變數與萬用字元，以及取其字面意思的單字。
  - **粗體**類型在段落開頭使用，或指出取其字面意思的單字。
- 本文件中的 *install_dir* 指出了大多數平台的**安裝根目錄**。第 23 頁的「參考目錄的慣例」中對異常進行了備註。

  依預設，在**大多數**平台上，*install_dir* 的位置爲：
  - Solaris 8 非封裝式評估安裝：

    user's home directory/`sun/appserver7`
  - Solaris 非隨附，非評估安裝：

    `/opt/SUNWappserver7`
  - Windows，所有安裝：

    `C:\Sun\AppServer7`

對於上面列示的平台，*default_config_dir* 與 *install_config_dir* 等同於 *install_dir*。請參閱第 23 頁的「參考目錄的慣例」以瞭解異常並取得附加資訊。

- 本文件中的 *instance_dir* 指出了**實例根目錄**，instance_dir 爲以下目錄的縮寫：

  default_config_dir/`domains`/*domain*/*instance*

- 除了特別提及 Linux 的地方之外，本手冊中所有 **UNIX 相關說明**均適用於 Linux 作業系統。

參考目錄的慣例

依預設，當使用 Solaris 8 與 9 封裝式安裝和隨附式 Solaris 9 安裝時，應用程式伺服器檔案會位於數個根目錄下。此章節中描述這些目錄。

- **對於隨附式 Solaris 9 安裝**，本指南將針對所提供的各種預設安裝目錄，使用下列文件慣例：
  - *install_dir* 是指 `/usr/appserver/`，其包含安裝影像的靜態部分。所有構成應用程式伺服器的公用程式、可執行程式以及程式庫均駐留在此位置中。
  - *default_config_dir* 是指 `/var/appserver/domains`，這是建立任何網域的預設位置。
  - *install_config_dir* 是指 `/etc/appserver/config`，其包含安裝配置資訊，如授權，以及爲此安裝配置的管理網域主清單。
- **對於 Solaris 8 與 9 的封裝式非評估、非隨附式安裝**，本指南將針對所提供的各種預設安裝目錄，使用下列文件慣例：
  - *install_dir* 是指 `/opt/SUNWappserver7`，其包含安裝影像的靜態部分。所有構成應用程式伺服器的公用程式、可執行程式以及程式庫均駐留在此位置中。
  - *default_config_dir* 是指 `/var/opt/SUNWappserver7/domains`，其爲建立任何網域的預設位置。
  - *install_config_dir* 是指 `/etc/opt/SUNWappserver7/config`，其包含安裝配置資訊，例如授權，以及爲此安裝配置的管理網域主清單。

產品系列簡介

Sun ONE Application Server 7 是相容於 J2EE 1.3 規格的應用程式伺服器，該伺服器還支援新的 Java Web 服務標準以及標準 HTTP 伺服器程式設計功能。此應用程式伺服器的三個版本提供了滿足生產與開發環境的各種需求。

- 平台版
- 標準版
- 企業版

平台版

平台版形成了 Sun ONE Application Server 7 產品系列的核心。此免費用於生產的產品提供高效能、小機體 J2EE 1.3 規格相容的執行環境，是基本作業部署以及嵌入協力廠商應用程式的最佳環境。可用的 Web 服務，平台版包括經過 Sun ONE Web Server 與 Sun ONE Message Queue 產品檢驗的內建技術。

平台版部署只限於單一的應用程式伺服器實例 (例如，Java 平台的單一虛擬機器 [「Java 虛擬機器」或「JVM™」])。平台版支援多層部署技術，但是 Web 伺服器層代理不執行載入平衡。在平台版中，管理公用程式僅用於本機用戶端。

平台版已整合於 Solaris 9 中。

標準版

「**管理員指南**」重點介紹此版本。標準版在平台版的基礎上加入了增強的遠端管理功能。增強的管理功能、遠端指令行與基於 Web 的管理均作爲標準版的一部分。此版本同時還包括由 Web 伺服器層代理分隔 Web 應用程式通訊的功能。針對每台機器，標準版支援配置多重應用程式伺服器實例 (JVM)。

企業版

企業版以更高的可用性、載入平衡與叢集功能 (適於最高需求的基於 J2EE 應用程式部署) 來增強核心應用程式伺服器平台。標準版的管理功能在企業版中進行了延伸，負責多重實例與多重機器部署。

叢集支援包括已複製的應用程式伺服器實例 (可以向其載入平衡的用戶端請求) 之易於配置的群組。此版本支援外部載入平衡程式與載入平衡 Web 層式代理。HTTP 階段作業、有狀態階段作業 Bean 實例與 Java Message Service (JMS) 資源故障恢復包含在企業版中。已註冊專利的「一直開啓」高效資料庫技術構成了企業版中 HA 持續性儲存的基礎。

如需更多的產品資訊，請參閱 Sun Microsystems 網站中的 Sun ONE Application Server 頁面。

使用文件

Sun ONE Application Server 手冊可以作爲可攜式文件格式 (PDF) 和超文件標示語言 (HTML) 格式表示的線上檔案，在下列網站上進行檢視：

`http://docs.sun.com/`

下表列示 Sun ONE Application Server 手冊中描述的工作與概念。左欄列示工作與概念，右欄列示其相應的手冊。

表 1 Sun ONE Application Server 文件路線圖

如需相關資訊	請參閱以下
有關軟體與說明文件的最新資訊	版次注意事項
受支援的平台與環境	平台摘要
應用程式伺服器的介紹，包括新功能、評估安裝資訊，以及架構概論。	入門指南
安裝 Sun ONE Application Server 與其各種元件 (範例應用程式、管理介面、Sun ONE Message Queue）。	**安裝指南**
建立並實施遵循 Sun ONE Application Server 7 中開放式 Java 標準模型的 J2EE 應用程式。包含有關應用程式設計、開發人員工具、安全性、組合、部署、除錯，以及建立生命週期模組的一般資訊。	**開發人員指南**

表 1 Sun ONE Application Server 文件路線圖 (續)

如需相關資訊	請參閱以下
建立並實施遵循 Sun ONE Application Server 7 中 Web 應用程式的開放式 Java 標準模型的 J2EE 應用程式。討論 Web 應用程式設計概念和工作，並提供範例碼、實施提示以及參考材料。	*Developer's Guide to Web Applications*
建立並實施遵循 Sun ONE Application Server 7 中企業 Bean 的開放式 Java 標準模型的 J2EE 應用程式。討論 EJB 程式設計概念和工作，並提供範例碼、實施提示以及參考材料。	*Developer's Guide to Enterprise JavaBeans Technology*
建立存取 Sun ONE Application Server 中的 J2EE 應用程式之用戶端	*Developer's Guide to Clients*
建立 Web 服務	*Developer's Guide to Web Services*
J2EE 功能，如 JDBC、JNDI、JTS、JMS、JavaMail、資源以及連接器	*Developer's Guide to J2EE Features and Services*
建立自訂 NSAPI 外掛程式	*Developer's Guide to NSAPI*
執行下列管理工作： • 使用管理介面與指令行介面 • 配置伺服器偏好設定 • 使用管理領域 • 使用伺服器實例 • 監視並記錄伺服器狀態 • 配置 Web 伺服器外掛程式 • 配置 Java Messaging Service • 使用 J2EE 功能 • 爲基於 CORBA 的用戶端配置支援 • 配置資料庫連接性 • 配置作業事件管理 • 配置 Web 容器 • 部署應用程式 • 管理虛擬伺服器	**管理員指南**
編輯伺服器配置檔案	*Administrator's Configuration File Reference*

表 1 Sun ONE Application Server 文件路線圖 (續)

如需相關資訊	請參閱以下
配置並管理 Sun ONE Application Server 7 作業環境的安全性。包括有關一般安全性、證書，以及 SSL/TLS 加密的資訊。同時還提出了基於 HTTP 伺服器的安全性。	*Administrator's Guide to Security*
爲 Sun ONE Application Server 7 的 J2EE CA 連接器配置並管理服務提供者實施。包含有關管理工具與 DTD 的資訊，並提供範例 XML 檔案。	J2EE CA Service Provider Implementation Administrator's Guide
將您的應用程式從 Netscape Application Server 版本 2.1 移轉至新的 Sun ONE Application Server 7 程式設計模型中，包括移轉 Sun ONE Application Server 所提供的 Online Bank 應用程式的一個範例。	*Migrating and Redeploying Server Applications Guide*
使用 Sun ONE Message Queue。	Sun ONE Message Queue 說明文件位於： http://docs.sun.com/

產品支援

如果您的系統發生問題，請使用下列任何一種機制聯絡客戶支援：

- 線上支援網站，其位址爲：

  `http://www.sun.com/supportraining/`

- 與維護合約相關的電話派送號碼

請在聯絡支援之前備妥下列資訊。這將有助於確保我們的支援人員可以更有效地協助您解決問題：

- 對問題進行描述，包括問題發生的情形以及對作業的影響
- 機器類型、作業系統版本以及產品版本，包括任何可能對問題造成影響的修補程式及其他軟體
- 您的詳細操作步驟以重現問題
- 所有的錯誤日誌或核心傾印

伺服器基本原理與管理全域設定

第 1 章「Sun ONE Application Server 管理快速入門」

第 2 章「設定管理伺服器偏好設定」

第 3 章「配置管理網域」

第 1 章

Sun ONE Application Server 管理快速入門

本章描述管理 Sun ONE Application Server 的基本原理：尋找關於伺服器一般資訊的位置、存取伺服器使用者介面的方式以及存取線上說明的方式。

本章包含下列小節：

- 關於 Sun ONE Application Server
- 配置隨附式 Solaris 版本
- 使用管理介面
- 使用指令行介面
- 存取管理伺服器
- 存取應用程式伺服器實例
- 使用 Sun ONE Studio
- 關於配置檔案
- 使用授權指令

關於 Sun ONE Application Server

Sun ONE Application Server 爲在廣泛的伺服器、用戶端、裝置範圍內開發、部署以及管理電子商業應用程式服務，提供了一個牢固的 J2EE 平台。主要功能包括作業事件管理、效能、可延伸性、安全性以及企業應用程式整合。

Sun ONE Application Server 支援的服務包含從 Web 發佈到企業範圍內的作業事件處理，同時可讓開發人員基於 JavaServer Pages (JSP™)、Java Servlet 以及企業 JavaBeans™ (EJB™) 技術建立應用程式。

它爲管理員提供下列基本工具：

- 允許不同的管理員建立並管理其各自應用程式伺服器實例集的多重管理領域
- 一個提供管理工具的管理伺服器 (每個領域具有一個管理伺服器)
- 一個用於伺服器管理的圖形使用者介面 (管理介面)
- 一個可以與管理介面執行相同工作的指令行介面

使用這些工具可執行所有的管理功能，包括：

- 管理領域
- 管理伺服器實例
- 部署應用程式
- 監視伺服器
- 使用日誌檔
- 管理資源
- 管理訊息佇列伺服器
- 使用作業事件服務
- 使用 Corba/IIOP 用戶端
- 配置 Web 伺服器外掛程式
- 配置 J2EE 容器
- 管理 HTTP 伺服器的功能

如需有關 Sun ONE Application Server 架構與功能的更多資訊，以及需要使用 Sun ONE Application Server 的初始步驟，請參閱「 *Sun ONE Application Server* 入門指南」。

配置隨附式 Solaris 版本

本手冊提及兩種用於 Solaris 的 Sun ONE Application Server 安裝版本：隨附式 Solaris 9 版本與非隨附式 Solaris 9 版本。如果您收到作爲 Solaris 9 安裝一部分的 Sun ONE Application Server 複本，您便取得了隨附式 Solaris 版本。如果您收到 Sun ONE
Application Server 的獨立複本，您便取得了非隨附式版本。

注意 如果您使用的是非隨附式 Solaris 版本的 Sun ONE Application Server，或使用 Windows 版本，請略過此節，並移往第 35 頁的「使用管理介面」。

如果您使用的是隨附式 Solaris 9 版本的 Sun ONE Application Server，則需要先執行一些附加的配置步驟，然後才能開始使用伺服器。

當您將非隨附式版本的 Sun ONE Application Server 作爲安裝程序的一部分進行安裝時，便會自動建立領域、管理伺服器以及伺服器實例。

使用隨附式 Solaris 9 版本時，您必須手動建立這些項目，並手動執行其他步驟。執行完這些初始步驟之後，您便可使用 Sun ONE Application server 的全部功能，包括加入附加的管理領域與伺服器實例。

本章節論述以下主題：

- 建立管理網域
- 啓動管理伺服器
- 建立應用程式伺服器實例
- 部署應用程式

建立管理網域

多重管理領域允許不同的管理使用者建立並管理其各自的領域。一個領域就是一個實例集，它是使用單一系統內已安裝的二進位制共用集合建立的。每個領域均具有一個管理伺服器。

建立新領域時，請指定：

- 管理伺服器的連接埠號。安裝非隨附式版本時，預設的連接埠號爲 4848。
- 管理使用者名稱與密碼。存取管理伺服器 (存取管理介面或執行指令行介面) 時需要這些密碼。

- 領域位置。

您必須使用指令行介面內 `asadmin` 公用程式的 `create-domain` 指令建立網域。如需有關建立管理網域的更多資訊，請參閱第 3 章 「配置管理網域」。

啟動管理伺服器

建立管理領域時，您建立了管理伺服器。管理伺服器是 Sun ONE Application Server 的特殊實例，其服務於管理介面，並爲指令行介面提供管理工具。

若要使用管理介面或使用指令行介面上的多個指令，您必須擁有執行中的管理伺服器。如需有關啓動管理伺服器的資訊，請參閱第 47 頁的 「啓動管理伺服器」。

建立應用程式伺服器實例

建立領域並啓動管理伺服器之後，您需要建立應用程式伺服器實例。每個應用程式伺服器實例均有其專有的 J2EE 配置、J2EE 資源、應用程式部署區域以及伺服器配置設定。

您可以透過管理介面或指令行介面建立應用程式伺服器實例。伺服器實例建立於領域內的資料夾中。

對於非隨附式版本，安裝時建立的伺服器實例稱爲 `server1`。在此文件內，您將經常看到在範例中使用 `server1`。

如需關於建立應用程式伺服器實例的更多資訊，請參閱第 72 頁的 「加入應用程式伺服器實例」。

部署應用程式

建立領域、啓動管理伺服器並加入應用程式伺服器實例之後，您便可以部署該實例的應用程式。如需更多資訊，請參閱第 13 章 「部署應用程式」。

使用管理介面

使用管理介面配置伺服器的各個方面。本章節包含下列主題：

- 存取管理介面
- 使用標籤
- 使用按鈕
- 存取線上說明
- 退出管理介面

注意 伺服器配置與相應管理介面的某些方面仍在變更之中。在隨後的產品發行版本中，不穩定的介面可能會被移除，並取而代之以更爲清晰、穩定的版本。多數伺服器配置與管理介面將保持不變或做些相容的變更，但是，在某些方面可能會進行不相容的變更。今後發行的產品說明文件會在這些不相容情況確實發生時對其進行清楚地描述。

存取管理介面

Sun ONE Application Server 的管理介面在名爲「管理伺服器」的 HTTP 伺服器上執行。如果您使用的是非隨附式版本，管理伺服器會在您安裝 Sun ONE Application Server 的時候隨之一同安裝。如果您安裝的是隨附式版本，則必須建立管理領域與管理伺服器。如需更多資訊，請參閱第 33 頁的「配置隨附式 Solaris 版本」。

管理伺服器必須處於執行狀態，這樣您才能使用管理介面。如需有關啓動管理伺服器的資訊，請參閱第 47 頁的「啓動管理伺服器」。

安裝 Sun ONE Application Server 或建立領域之後，就要爲管理伺服器選擇一個連接埠號，或使用預設的連接埠號 4848。若要存取管理介面，請在 Web 瀏覽器中鍵入：

http://*hostname.domain*:*port*/

例如：

http://austen.sun.com:4848/

如果您要從安裝 Sun ONE Application Server 的機器上存取管理伺服器，可以使用：

http://*localhost*:4848

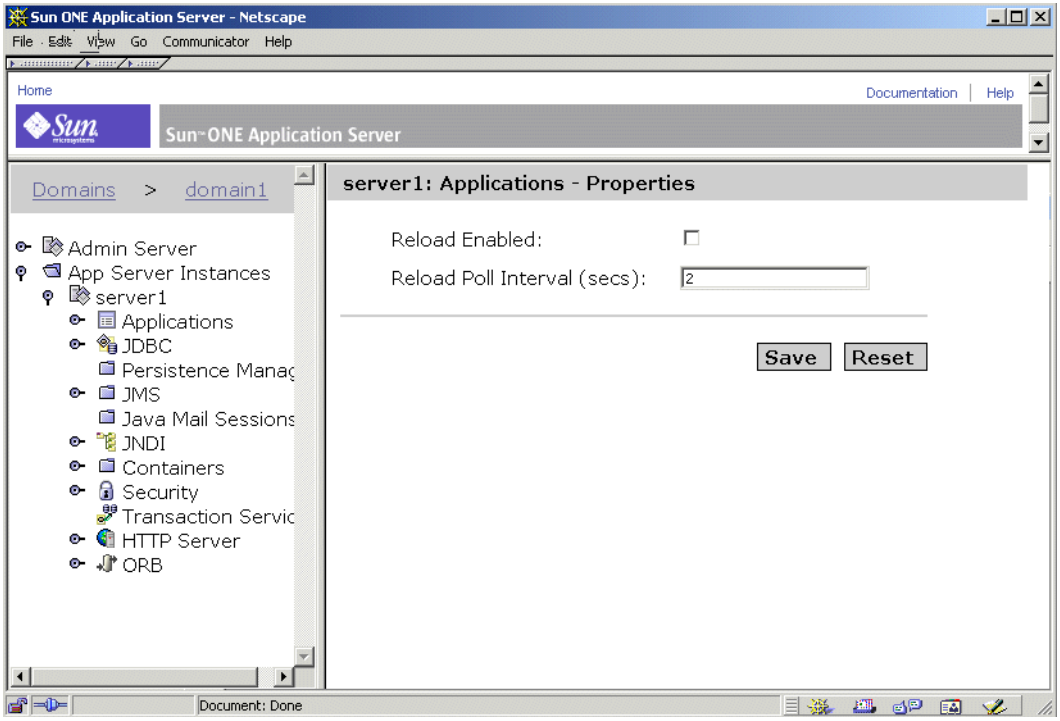
系統將會提示您輸入使用者名稱與密碼，這些名稱與密碼由執行安裝的使用者在安裝過程中設定，或在您建立領域和管理伺服器時設定。

只要您存取了瀏覽器，便可以從遠端位置存取管理介面。您可以經由任何一台能在網路上存取伺服器的機器來存取該介面。

在 Windows 上，您可以透過以下方法存取管理介面：從 [開始] 功能表中選擇 [程式集]，然後選擇 [Sun Microsystems]，接著選擇 [Sun ONE Application Server 7]，最後選擇 [Start Admin Console]。

下圖顯示管理介面。

圖 1-1 Sun ONE Application Server 管理介面



左窗格中爲可以在 Sun ONE Application Server 中進行配置的所有項目之樹檢視。若要使用管理介面，請按一下左窗格中的某個項目。右窗格會顯示與該項目關聯的頁面。

如果窗格內某個項目的旁邊具有開啓/關閉符號，您可以透過按一下開啓/關閉符號，展開該項目來檢視其子項目。如果樹項目尚未展開，則符號的形狀爲：

圖 1-2 關閉符號

如果樹項目已經展開，則符號的形狀爲：

圖 1-3 開啓符號

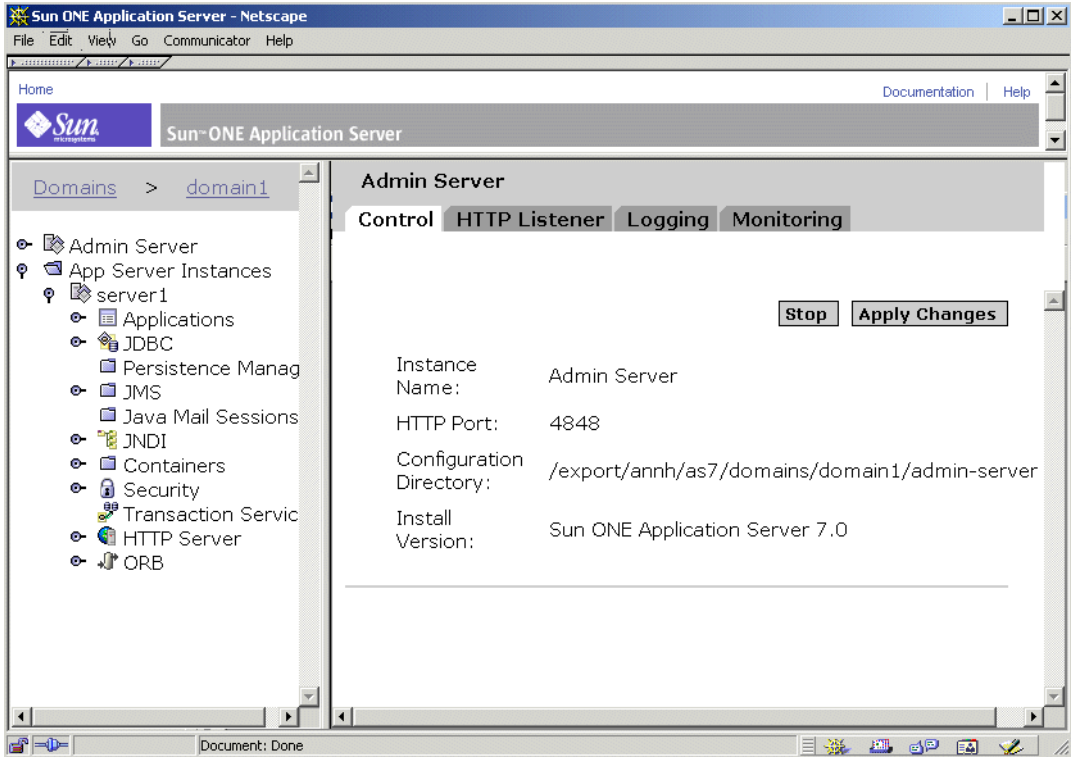
使用標籤

管理介面的某些部分中含有您用於導覽至其他頁面的標籤。這些標籤位於右窗格頂部的一個單獨窗格中。

若要使用標籤，請按一下標籤名稱。某些標籤可以直接將您導向一個頁面，而另一些標籤名稱的下面會出現一個可用頁面清單，供您選擇要檢視的頁面。按一下要檢視的頁面名稱，移往該頁面。

下圖顯示帶有標籤的管理介面：

圖 1-4 帶有標籤的管理介面



使用按鈕

管理介面中具有下列標準按鈕。

表格 1-1 標準管理介面按鈕

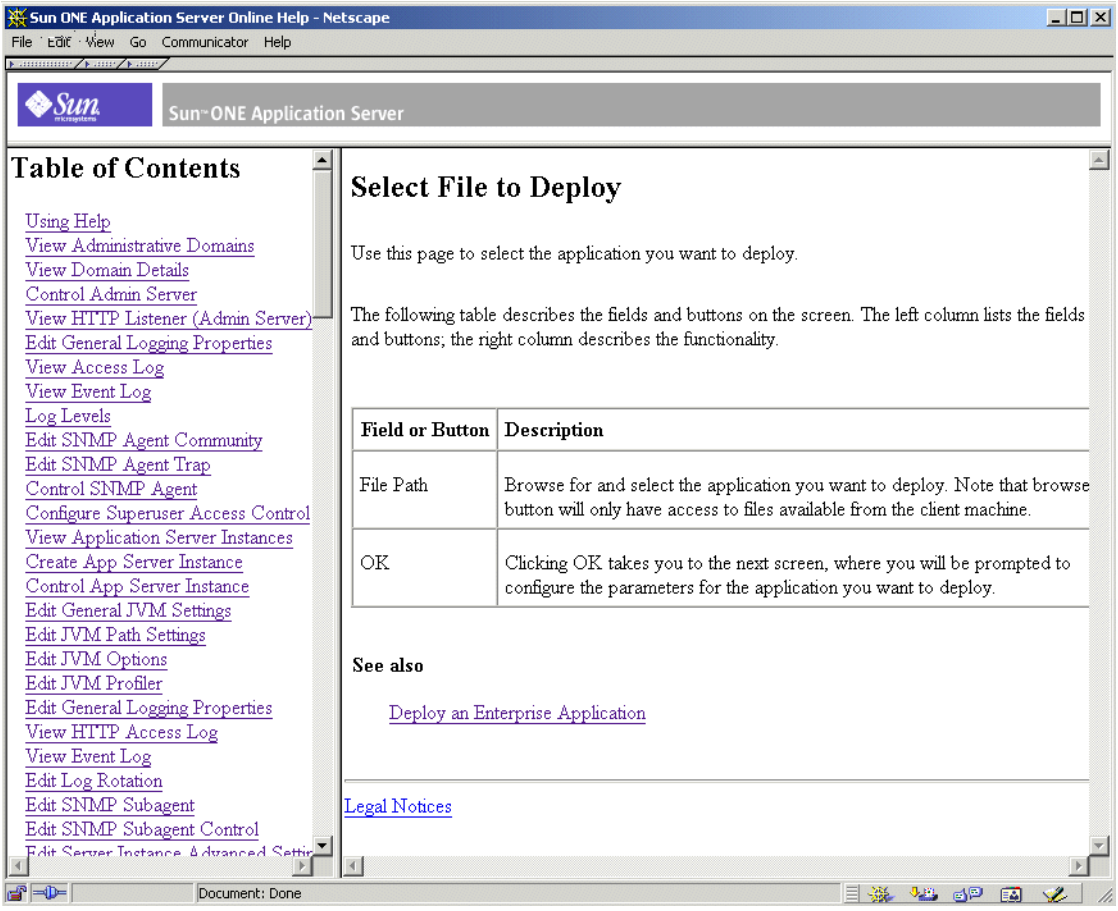
按鈕	執行的動作
Cancel	不儲存所做變更，直接取消並返回上一頁。
Delete	刪除一個項目。您經常按一下某個項目旁邊的 [Select]，來指出按一下 [Delete] 時要刪除的內容。
New	顯示用於建立新項目的頁面。例如，按一下 [Application Server Instances] 頁面上的 [New] 會顯示 [Create New Instance] 頁面。
OK	儲存您輸入的內容。如果您變更了 Sun ONE Application Server 實例的配置，則必須套用變更以使其生效。 如需更多資訊，請參閱第 74 頁的「將變更套用至應用程式伺服器實例」。
Reset	將頁面中的值重設爲預設值。
Save	儲存您輸入的內容。如果您變更了 Sun ONE Application Server 實例的配置，則必須套用變更以使其生效。 如需更多資訊，請參閱第 74 頁的「將變更套用至應用程式伺服器實例」。

根據特定螢幕的需求，也可使用其他按鈕。

存取線上說明

您可以透過按一下管理介面頂端標題中的 [Help] 按鈕，來存取管理介面內任意頁面的輔助說明。線上輔助說明描述所存取頁面的用法，並提供有關本頁面欄位內所有輸入內容的資訊。

圖 1-5 線上輔助說明



您可以在輔助說明視窗中，使用左窗格內的目錄導覽至其他頁面的輔助說明。如需有關使用輔助說明的說明，請參閱線上輔助說明目錄中的第一個主題「Using Help」。

退出管理介面

在管理介面中，沒有明確的退出或登出按鈕。若要退出，請關閉正用於存取管理介面的瀏覽器。同時關閉該瀏覽器中可能正在機器上執行的其他實例。

使用指令行介面

Sun ONE Application Server 包含指令行介面。您可以使用 `asadmin` 公用程式和與其關聯的指令來執行一組也可以在管理介面中執行的工作。例如，您可以啓動與停止應用程式伺服器實例、配置伺服器以及部署應用程式。

您可以經由 Shell 中的指令提示使用這些指令，或經由其他程序檔和程式呼叫這些指令。您可以使用這些指令自動執行重複的管理工作。

如需關於使用指令行介面的更多資訊，或需要指令清單，請參閱附錄 A「使用指令行介面」。

存取管理伺服器

管理伺服器是 Sun ONE Application Server 的特殊實例，其服務於管理介面，並爲管理介面與指令行介面提供管理工具。該伺服器管理這些介面與工具的配置、部署和監視設備，因此必須處於執行狀態，以使這些設備正常工作。

若要配置管理伺服器的特性，請存取管理介面。按一下左窗格中的 [Admin Server]，以顯示管理伺服器的配置設定。

如需有關管理伺服器的更多資訊，請參閱第 2 章「設定管理伺服器偏好設定」。

存取應用程式伺服器實例

您可以具有多個 Sun ONE Application Server 實例。每個應用程式伺服器實例均具有其各自的配置、資源以及應用程式部署區域，它們獨立於其他應用程式伺服器實例。變更一個應用程式伺服器實例的配置，不會引起其他應用程式伺服器實例配置的變更。管理介面會爲您建立的每個應用程式伺服器實例設定一個項目。如果您使用的是非隨附式版本的軟體，則會在安裝軟體時建立一個應用程式伺服器實例。如果需要，您可以建立其他應用程式伺服器實例。

如果您使用的是隨附式 Solaris 9 版本，則應用程式伺服器實例不會自動建立。如需更多資訊，請參閱第 33 頁的「配置隨附式 Solaris 版本」。

如需有關應用程式伺服器實例的更多資訊，請參閱第 4 章「使用應用程式伺服器實例」。

使用 Sun ONE Studio

若要部署應用程式，除了使用管理介面與指令行介面之外，您也可以使用 Sun ONE Studio 4。如需有關 Sun ONE Studio 的更多資訊，請參閱「*Sun ONE Studio 4, Enterprise Edition for Java with Application Server 7 Tutorial*」，其位於 http://docs.sun.com。

關於配置檔案

當您經由管理介面或指令行介面變更伺服器的設定時，管理伺服器便會變更基本的配置檔案。這些檔案包含管理伺服器以及個別應用程式伺服器實例的設定。

如需有關這些檔案及其包含的設定的更多資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

使用授權指令

購買 Sun ONE Application Server 之後，會在安裝該軟體期間自動安裝相應類型的授權。如需有關授權類型及其限制的更多資訊，請參閱「*Sun ONE Application Server* **安裝指南**」。

Sun ONE Application Server 包含在安裝後管理授權的指令行公用程式。

對於安裝 Windows 的機器而言，若要在安裝之後對授權進行升級，您可以使用 `asadmin` 公用程式中的 `install-license` 指令。該指令必須在已安裝了 Sun ONE Application Server 的本端機器上執行。安裝授權之前，必須首先停用所有執行中的應用程式伺服器。

其語法如下：

```
asadmin install-license
```

對於非隨附式 Solaris 封裝式安裝版本，安裝授權時使用下列語法：

`pkgadd -d` *full_path* SUNWaslco

例如，

pkgadd -d /*install_dir*/pkg SUNWaslco

對於隨附式 Solaris 9 授權安裝版本，安裝授權時使用下列語法：

`pkgadd -d` *full_path* SUNWaslc

若要取得有關您的授權之資訊，請使用 `display-license` 指令，其語法如下：

`asadmin display-license [--user` *admin_user*`] [--password` *admin_password*`] [--passwordfile` *password_file*`] [--host` *localhost*`] [--port` *admin_port*`] [--local=`*true/false*`]`

該指令可以在本機執行或從遠端執行，具體取決於本機選項值。例如，若要從本端機器執行指令，採用主機與連接埠號的預設值，語法如下：

```
asadmin display-license --local
```

輸出的內容描述目前安裝的授權類型 (例如，評估版)、該授權的過期日期 (如果有)、授權允許每個管理伺服器中包含的實例數，以及是否允許遠端管理。

如需有關指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

第 2 章

設定管理伺服器偏好設定

管理伺服器是 Sun ONE Application Server 的特殊實例，其服務於管理介面，並爲管理介面與指令行介面提供管理工具。它可以管理用於 Sun ONE Application Server 的配置、部署和監視工具。本章說明配置管理伺服器的方式。

本章包含以下主題：

- 關於管理伺服器
- 啓動管理伺服器
- 關閉管理伺服器
- 存取管理伺服器設定
- 檢視管理伺服器控制設定
- 將變更套用至管理伺服器
- 編輯管理伺服器的 HTTP 偵聽程式設定
- 設定 SNMP、記錄和安全性偏好設定

關於管理伺服器

管理伺服器是 Sun ONE Application Server 的特殊實例，其爲管理介面和指令行介面提供管理工具。它可以管理用於這些介面的配置、部署和監視工具。管理伺服器服務於管理介面頁面。它必須正常執行，您才可以使用管理介面並在指令行介面中執行大多數指令。

在您安裝 Sun ONE Application Server (非隨附式版本) 或建立新領域時，會安裝管理伺服器。雖然您可以建立由一個管理伺服器管理的多重 Sun ONE Application Server 實例，但每個領域僅可以有一個管理伺服器。透過管理伺服器，您可以存取用於每個應用程式伺服器實例的配置設定、已部署應用程式及其他伺服器功能。

如需有關管理網域的更多資訊，請參閱第 3 章「配置管理網域」。

如果您使用的是非隨附式版本的 Sun ONE Application Server，安裝 Sun ONE Application Server 後，您要選擇管理伺服器的連接埠號，或使用預設連接埠 4848。

如果您使用的是隨附式 Solaris 9 版本的 Sun ONE Application Server，必須手動建立一個領域和一個管理伺服器。如需關於配置隨附式 Solaris 9 版本的更多資訊，請參閱第 33 頁的「配置隨附式 Solaris 版本」。

若要存取管理介面，請在 Web 瀏覽器中鍵入：

http://*hostname.domain*:*port*/

請注意，此處的 *domain* 不是您的 Sun ONE Application Server 管理領域，而是您的領域名稱。

例如：

http://austen.sun.com:4848/

如果您要從安裝 Sun ONE Application Server 的機器上存取管理伺服器，可以使用：

http://localhost:4848

系統會提示您提供使用者名稱和密碼。

啟動管理伺服器

若要啓動或重新啓動管理伺服器，請使用下列主題中描述的方法之一：

- 使用 startserv 程序檔
- 使用指令行介面
- 使用 [服務] 視窗 (Windows)
- 使用 [開始] 功能表 (Windows)

使用 startserv 程序檔

若要使用啓動程序檔來啓動管理伺服器，如果伺服器在號碼低於 1024 (UNIX) 的連接埠上運行，則請以超級使用者身份登入；否則，請以超級使用者身份登入或使用伺服器的使用者帳號登入。在指令行提示處移往目錄：

install_dir/domains/*domain_dir*/admin-server/bin

其中，*install_dir* 是您安裝伺服器的目錄，*domain_dir* 是管理領域目錄。

對於 UNIX，請鍵入：

```
./asadmin startserv
```

您可以在行結尾處使用選擇性參數 `-i`。伺服器通常作爲後台程序執行。`-i` 選項可防止伺服器將其自身放置在後台中。如果您使用 `/etc/inittab` 執行伺服器，則該選項很有用。

對於 Windows，請執行 `startserv.bat` 檔案。

注意 如果伺服器已經在執行，`startserv` 指令將失敗。您必須首先停止伺服器，然後再使用 `startserv` 指令。並且，如果伺服器啓動失敗，您應該終止程序，然後再嘗試重新啓動。

使用指令行介面

指令行介面的 `asadmin` 公用程式有一個 `start-domain` 指令，您可以使用該指令啓動應用程式伺服器以及所有關聯的 Sun ONE Application Server 實例。您只能在本機執行該指令，即從安裝您 Sun ONE Application Server 的機器上執行。該指令不需要引數。

您也可以使用指令 `start-domain` 來啓動管理領域內的所有實例。該指令使用以下語法：

`asadmin start-domain [--domain` *domain-name*`]`

如需關於使用指令行介面的更多資訊，請參閱指令行介面的線上說明和附錄 A「使用指令行介面」。

使用 [服務] 視窗 (Windows)

若要使用 Windows 中的服務控制台來啓動伺服器，請執行以下步驟：

1. 在 [控制台] 中，按一下 [管理工具]。
2. 按一下 [服務]。
3. 捲動服務清單並按兩下 [Application Server 7.0 Administration Server] 服務。

   選取用於給定領域的管理伺服器。[服務] 視窗中的 [名稱] 欄顯示管理伺服器的名稱 Sun App Server Admin Server (*your_domain_name*:admin-server)。
4. 按一下 [啓動]。
5. 按一下 [確定]。

當您啓動電腦時，[Application Server 7.0 Administration Server] 服務便自動啓動。

使用 [開始] 功能表 (Windows)

若要使用 Windows [開始] 功能表來啓動伺服器，請執行以下步驟：

1. 從 [開始] 功能表中，選擇 [程式集]。
2. 選擇 [Sun Microsystems]。
3. 選擇 [Sun ONE Application Server 7]。
4. 按一下 [Start Application Server]。

關閉管理伺服器

一旦啓動管理伺服器，它將持續執行，偵聽並接受請求。您可能想要停止並重新啓動伺服器，例如，當變更管理伺服器記錄個人喜好或管理伺服器 HTTP 偵聽程式進行偵聽的連接埠時。

當您停止管理伺服器時，它會停止接受新連線。然後等待所有未完成的連線完成。管理伺服器停止後，您便無法使用管理介面或指令行介面。

您可以使用下列主題中描述的一種方法來停止伺服器：

- 使用管理介面關機
- 使用 stopserv 程序檔關機
- 使用指令行介面關機
- 使用 [服務] 視窗關機 (Windows)

在您關閉伺服器後，伺服器可能需要幾秒鐘完成關機程序。

使用管理介面關機

若要使用管理介面關閉管理伺服器，請執行下列步驟：

1. 在左窗格中，按一下 [Admin Server]。
2. 按一下 [Control] 標籤。
3. 按一下 [Stop]。

   按一下此連結，管理伺服器將立即關閉。沒有附加螢幕。

使用 stopserv 程序檔關機

若要手動停止管理伺服器，請在指令提示處移至目錄：

install_dir/domains/*domain_dir*/admin-server/bin

其中，*install_dir* 是您安裝伺服器的目錄，*domain_dir* 是領域目錄。

對於 UNIX，請鍵入：

```
./asadmin stopserv
```

您必須以執行伺服器的使用者身份執行該指令。

如果您使用 /etc/inittab 檔案重新啓動伺服器，您必須從 /etc/inittab 中移除啓動伺服器的行，並鍵入 kill -1，然後嘗試停止伺服器。否則，伺服器在停止後會自動重新啓動。

對於 Windows，請執行 stopserv.bat 檔案。

使用指令行介面關機

您可以使用指令行介面 asadmin 公用程式的 shutdown 指令，停止管理伺服器。該指令不需要引數，並可以在本機或遠端執行。

您也可以使用指令行介面 asadmin 公用程式的 stop-appserv 指令，停止管理伺服器及所有關聯的 Sun ONE Application Server 實例。您只能在本機執行該指令，即從安裝您 Sun ONE Application Server 的機器上執行。該指令不需要引數。

您也可以透過關閉領域 (使用 stop-domain 指令)，來關閉管理伺服器。依預設，該指令可關閉領域中的所有實例，包括管理伺服器。您也可以將其配置爲關閉領域中除管理伺服器之外的所有實例。該指令使用以下語法：

```
asadmin stop-domain [--user admin_user] [--password admin_password]
[--host admin_host] [--port admin_port] [--local=true/false] [--domain
domain_name] [--adminserv=true/false] [--passwordfile file_name] [--secure |
-s]
```

如果您使用 local 選項，該指令將於本機執行。如果您使用 --adminserv=false 選項，該指令將不停止管理伺服器。不過，依預設 --adminserv 設定爲 true，因此將依預設停止管理伺服器。

如需關於使用指令行介面的更多資訊，請參閱指令行介面的線上輔助說明和附錄 A「使用指令行介面」。

使用 [服務] 視窗關機 (Windows)

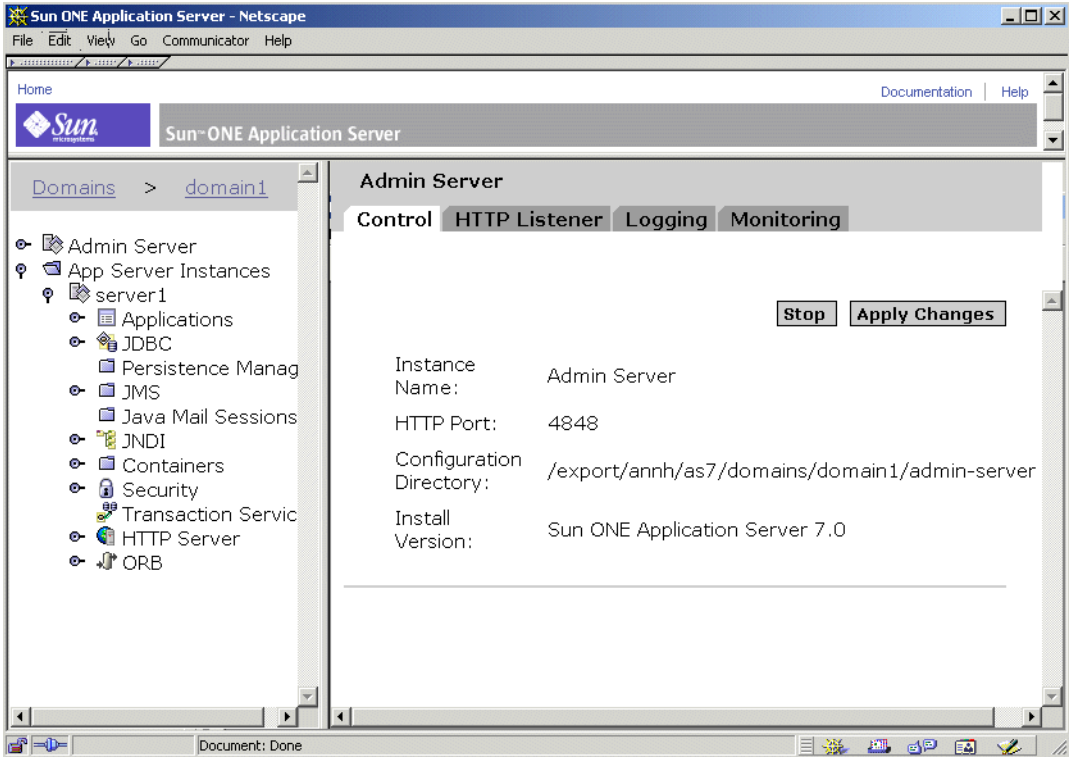
若要使用 [服務] 視窗關閉管理伺服器，請執行下列步驟：

1. 在 [控制台] 中，按一下 [管理工具]。
2. 按一下 [服務]。
3. 捲動服務清單並按兩下 [Sun Application Server 7 Admin Server] 服務。
4. 按一下 [Stop]。
5. 按一下 [OK]。

存取管理伺服器設定

若要存取管理伺服器設定，請在管理介面的左窗格中，按一下 [Admin Server]。管理伺服器的設定會顯示在右窗格中，其架構爲一個標籤集。

圖 2-1 管理伺服器使用者介面



按一下某一標籤，可存取特定功能區域的設定。有時按一下標籤，可向您直接展示一個頁面；有時按一下標籤，會爲您提供可從中選擇的頁面清單。

本章涵蓋 [Control] 和 [HTTP Listener] 標籤。如需關於監視和 SNMP 設定的資訊，請參閱第 6 章 「監視 Sun ONE Application Server」如需關於記錄的資訊，請參閱第 5 章 「使用記錄功能」。

檢視管理伺服器控制設定

管理伺服器控制設定顯示實例名稱 (管理伺服器)、執行您管理伺服器的連接埠、包含配置檔案的目錄、您所執行的 Sun ONE Application Server 軟體版本。

檢視這些設定的步驟：

1. 在左窗格中，按一下 [Admin Server]。
2. 按一下 [Control] 標籤。

將變更套用至管理伺服器

當使用管理介面或指令行介面變更管理伺服器的配置資訊時，您可能需要套用您的變更，以使變更生效。這也稱爲重新配置伺服器。當您套用變更後，自您上次套用變更以來所作的所有配置變更均生效。

如果您已對需要套用變更的管理伺服器配置進行了變更，在左窗格樹檢視中的 [Admin Server] 旁邊會出現一個黃色圖示。

圖 2-2 警告圖示

套用配置變更的步驟：

1. 在左窗格中，按一下 [Admin Server]。
2. 按一下 [Control] 標籤。
3. 按一下 [Apply Changes]。

   套用變更後，螢幕會顯示一條訊息。

編輯管理伺服器的 HTTP 偵聽程式設定

在伺服器可以處理請求之前，其必須經由 HTTP 偵聽程式接受請求。

藉由非隨附式版本的 Sun ONE Application Server，在您安裝時將自動建立用於管理伺服器的 HTTP 偵聽程式 `http-listener-1`。該 HTTP 偵聽程式使用 IP 位址 0.0.0.0 以及您在安裝期間指定的作爲管理伺服器連接埠號的連接埠號。預設值爲 4848。您無法刪除預設 HTTP 偵聽程式。

對於管理伺服器實例而言 (在領域中)，僅 HTTP 偵聽程式具有 `http-listener-1` id。換言之，如果您建立管理伺服器實例，則僅一個 HTTP 偵聽程式可以及時地在任何點上以 HTTP 或 HTTPS 協定執行。(這也意味著您無法以 HTTP 和 HTTPS 兩種連線同時連線至管理伺服器。) 如需關於配置隨附式 Solaris 9 版本的更多資訊，請參閱第 33 頁的「配置隨附式 Solaris 版本」。

HTTP 偵聽程式是您爲管理伺服器啓動和配置 SSL/TLS 安全性設定的程式。

此外，您可以指定 HTTP 偵聽程式中接受者執行緒 (有時稱爲接受執行緒) 的數目。接受執行緒是等待連線的執行緒。執行緒接受連線並將它們放入佇列中，在佇列中將由工作執行緒接受這些連線。看起來理想的做法爲，您想擁有足夠多的接受執行緒，以便在發生新的請求時總有一個可用，但又需要數目相當少，以免對系統造成太重負擔。預設值爲 1。最佳規則是讓系統上的每個 CPU 有一個接受執行緒。如果您發現效能受到損害，可以調整此值。如需有關效能的更多資訊，請參閱「*Sun ONE Application Server Performance Tuning and Sizing Guide*」。

編輯管理伺服器 HTTP 偵聽程式設定的步驟：

1. 在左窗格中，按一下 [Admin Server]。
2. 按一下 [HTTP Listeners] 標籤。
3. 進行所需的變更，然後按一下 [OK]。

如需有關 HTTP 偵聽程式的更多資訊，請參閱線上輔助說明。

設定 SNMP、記錄和安全性偏好設定

如需關於 SNMP 設定的資訊，請參閱第 6 章「監視 Sun ONE Application Server」如需關於記錄的資訊，請參閱第 5 章「使用記錄功能」如需關於安全性設定的資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。

第 3 章

配置管理網域

本章解釋透過 Sun ONE Application Server 設定與管理管理網域的方式。

本章包含以下主題：

- 關於管理網域
- 配置網域
- 重新建立網域註冊

關於管理網域

管理領域提供一種基本的安全結構，不同的管理員可以藉此管理一台機器上應用程式伺服器實例的特定群組 (領域)。透過這種方法分割應用程式伺服器實例，便有可能在不同的組織之間共用一台機器，而且每個組織都有自己的管理員。

在 Sun One Application Server 內，每個應用程式伺服器實例都是某個領域中的成員。不需要存在多個領域，但如果需要，也支援多個領域，這是一項很有用的功能。

使用作業系統的基本安全性機制 (即，通過檔案許可權)，可以爲本機指令建立管理安全性。透過使用者名稱 / 密碼對與特定管理伺服器進行通訊，可以建立遠端指令安全性。管理網域不採用任何其他安全性建構。

本章節描述下列主題：

- 執行管理網域
- 目錄結構
- 程序 / 連接埠結構

執行管理網域

領域是通過檔案、作業系統程序以及連接埠來執行的。每個領域都具有唯一名稱。

目錄結構

一個安裝版本內有多個由所有領域共用的檔案 (配置檔案、可執行檔案等等)。在這裡，重點討論領域特定的檔案。

所有領域特定的檔案都共用一個根目錄，該目錄稱爲領域目錄，其名稱爲領域名稱。在領域目錄下爲每個實例對應一個目錄 (目錄依據實例命名)，而每個實例目錄下爲該實例特定的檔案。

領域目錄可以建構在檔案系統中的任意位置 (符合安全性許可權與作業系統層級的其他限制條件)。除非使用者另有選擇，否則，領域目錄將建構在預設目錄之下 (稱爲領域目錄)。不過，使用者可以選擇在任意位置建立領域目錄。

程序/連接埠結構

領域在執行中會使用作業系統程序與連接埠。具體來說，對於在網域中執行的各個實例 (包括網域的管理伺服器)，均存在一個程序與一個連接埠。

配置網域

可以使用專用指令建立、刪除、列示、啓動或停用領域。

建立、刪除、啓動領域僅可以在本機執行，而列示與停用領域則既可以在本機執行，也可以從遠端執行。

全部的刪除、啓動與停止指令會佔用一個領域名稱。在只有一個領域的情況下，該名稱是可選的。如果在具有多個已配置領域的情況下仍不給定領域，指令便會發出一則錯誤訊息。

本章節包含下列主題：

- 建立網域
- 刪除網域
- 列示網域
- 啓動網域
- 停用網域

建立網域

領域是使用 create-domain 指令建立的。該指令只可在本機使用。

提要：

```
asadmin create-domain [--path domain_path] [--sysuser sys_user]
[--passwordfile file_name] --adminport port_number --adminuser admin_user
--adminpassword password domain_name
```

範例：在預設位置建立領域

```
$ asadmin create-domain --adminport 123 --adminuser MyAdmin
--adminpassword MyPassword MyDomain
```

本範例在預設位置建立名爲 MyDomain 的領域 (即，領域目錄)。管理伺服器的偵聽連接埠爲 123，管理使用者名稱爲 MyAdmin，密碼爲 MyPassword。下面的領域目錄與檔案將由執行該指令的作業系統使用者擁有。此外，作業系統程序將作爲執行該指令的使用者來執行。

如果已經存在一個名爲 MyDomain 的網域，便會傳回一則錯誤訊息。

(請注意，您可以將密碼放入檔案，在使用 --passwordfile 選項時使其通過驗證，而不必在指令行上使用該密碼，因爲這樣做可能引發安全性問題)。

範例：在某個位置 (非預設位置) 建立領域

```
$ asadmin create-domain --path $HOME --adminport 123 --adminuser
MyAdmin --adminpassword MyPassword MyDomain
```

本範例與第一個範例類似，只是現在網域目錄位於使用者的 $HOME 目錄之下，而不是位於預設的網域目錄之下。

範例：為其他使用者建立領域 (僅用於 UNIX)

```
# asadmin  create-domain --user AnotherUser --adminport 123
--adminuser MyAdmin --adminpassword MyPassword MyDomain
```

本範例與第一個範例類似，只是領域與其檔案以及作業系統程序由名爲 AnotherUser 的使用者所擁有。

使用 --sysuser 選項，使用者可以建構其他使用者可以在以後管理的領域。該選項要求執行 create-domain 指令的使用者必須爲 root 使用者。

UNIX 平台上的使用者許可權

爲了使非 root 使用者能夠建立與刪除管理領域，您必須在具有寫入領域配置檔案許可權的 UNIX 群組內加入該使用者的 ID：

1. 建立將要套用於安裝領域配置檔案的 UNIX 群組。例如，名爲 asadmin 的 UNIX 群組。
2. 將位於 /etc/appserver 之下的安裝領域配置檔案設定爲由新建的 UNIX 群組所擁有。

   這些檔案被命名爲 `domains.bin` 與 `domains.lck`。例如，變更之後，群組被指定給下列檔案：

   ```
   -rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.bin
   -rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
   ```

3. 啓用新建立的 UNIX 群組對這些檔案的寫入存取。在本範例中，最終許可權爲：

   ```
   -rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.bin
   -rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
   ```

4. 在 UNIX 群組內加入使用者 ID。

或者，如果您不想使非 root 使用者具有寫入存取安裝配置檔案的權限，可以建立一個管理領域來代表使用者。建立新管理領域期間，指定 `--sysuser` 與 `--path` 選項來識別將擁有領域目錄和檔案的 UNIX 使用者 ID，以及識別管理領域的建立位置。如需有關範例，請參閱第 57 頁的「範例：爲其他使用者建立領域 (僅用於 UNIX)」。

一旦在使用者 ID 下建立了管理領域，則該使用者便可以建立新的應用程式伺服器實例，並在該實例上執行各種管理作業。使用者 ID 無需屬於具有管理領域配置檔案寫入權限的 UNIX 群組。只有在建立與刪除管理領域時才要求具有 UNIX 群組成員的身份。

刪除網域

領域是使用 `delete-domain` 指令刪除的。僅具有領域管理權限的作業系統使用者 (或 root 使用者) 才能成功地執行該指令。該指令只可在本機使用。

提要：

`asadmin delete-domain [`*domain_name*`]`

範例：刪除領域

```
$ asadmin delete-domain MyDomain
```

本範例刪除本端機器上名爲 MyDomain 的領域。

列示網域

使用 `list-domains` 指令可以找到在機器上建立的領域。

該指令可以在本機執行，也可以從遠端執行。

提要：

```
asadmin list-domains [--host host] [--port port] [--password password]
[--user user]
```

範例：列示本端機器上的領域

```
$ asadmin list-domains

domain1     [/opt/ias/build/domains/domain1]
```

範例：使用遠端選項列示本端機器上的領域

```
$ asadmin list-domains --user admin --password password --host
localhost --port 4848

domain1     [/opt/ias/build/domains/domain1]
```

啟動網域

可以使用 `start-domain` 指令啓動領域。這會啓動領域的管理伺服器以及領域內所有其他實例。

該指令只可在本機執行。

提要：

```
asadmin start-domain [--domain domain_name]
```

範例：啟動機器上的唯一領域：

```
$ asadmin start-domain

Instance domain1:admin-server started

Instance domain1:server1 started

Domain domain1 Started.
```

停用網域

可以使用 stop-domain 指令停用領域。使用者可以選擇停用領域內的任何一個實例，或停用管理伺服器以外的所有實例，如此便可從遠端管理領域。

該指令可以在本機執行，也可以從遠端執行。

提要：

```
asadmin stop-domain [--user admin_user] [--password admin_password]
[--host host_name] [--port port_name] [-- local=false] [--domain
domain_name] [--adminserv=true] [--passwordfile file_name] [--secure |
-s]
```

範例：停用領域中除管理伺服器實例之外的所有實例

```
$ asadmin stop-domain --user admin --password password --host
localhost --port 4848 --adminserv=false --domain domain1

DomainStoppedRemotely
```

重新建立網域註冊

爲了執行，每個領域的詳細資訊 (包括其名稱、位置、使用的連接埠等等) 都被記錄在名爲領域註冊的檔案中。

在一般作業條件下，您無須直接處理領域註冊，因爲系統管理指令完全封裝了所有對領域註冊的修改或使用。但是，由於領域註冊是一個檔案，所以可能被損壞 (例如，程序檔出錯，或某人不慎刪除了註冊等等)，在這種情況下，您可能必須重新建立檔案。

注意 您可以透過指令行介面，使用 asadmin 指令存取領域註冊。

如果註冊被損壞，請執行下列程序重新建立：

1. 取得所有網域以及網域所處目錄 (預設或非預設) 的清單。
2. 重新命名各個目錄 (例如，爲各個目錄名稱附加尾碼 .bak)。
3. 使用連接埠、密碼等的預設值，在原始位置上重新建立各個目錄。
4. 刪除各個新的領域目錄，並以原始目錄取代。
5. 如需有關各個領域的資訊，請執行 reconfig 指令。這會導致透過舊領域中的值更新領域註冊。

第 2 部分

管理個別伺服器實例

第 4 章「使用應用程式伺服器實例」

第 5 章「使用記錄功能」

第 6 章「監視 Sun ONE Application Server」

第 7 章「配置 Web 伺服器外掛程式」

第 8 章「配置 J2EE 容器」

第 9 章「使用作業事件服務」

第 10 章「配置命名與資源」

第 11 章「使用 JMS 服務」

第 12 章「配置 CORBA/IIOP 用戶端伺服器」

第 13 章「部署應用程式」

第 4 章

使用應用程式伺服器實例

本章描述建立、刪除、配置、啓動和停止 Sun ONE Application Server 實例的方式。

本章包含以下主題：

- 關於應用程式伺服器實例
- 啓動和停止應用程式伺服器實例
- 在除錯模式下啓動應用程式伺服器實例
- 設定終止逾時
- 自動重新啓動應用程式伺服器實例 (UNIX)
- 手動重新啓動應用程式伺服器實例 (UNIX)
- 關於監視程式
- 加入應用程式伺服器實例
- 刪除應用程式伺服器實例
- 將變更套用至應用程式伺服器實例
- 檢視應用程式伺服器實例狀況
- 配置 JVM 設定
- 配置記錄設定和監視設定
- 變更應用程式伺服器實例進階設定

關於應用程式伺服器實例

當您安裝非隨附式版本的軟體時，Sun ONE Application Server 會建立一個稱爲 `server1` 的應用程式伺服器實例。如果願意，您可以刪除 `server1` 實例並以其他名稱建立一個新實例。

如果要使用隨附式 Solaris 9 版本的軟體，您必須建立自己的伺服器實例。如需更多資訊，請參閱第 33 頁的 「配置隨附式 Solaris 版本」。

每個應用程式伺服器實例均有其專有的 J2EE 配置、J2EE 資源、應用程式部署區域以及伺服器配置設定。對一個應用程式伺服器實例所做的變更，不會影響其他應用程式伺服器實例。在一個管理領域內，您可以擁有多個應用程式伺服器實例。在一個領域內，所有伺服器實例使用同一個管理伺服器。如需有關網域的更多資訊，請參閱第 3 章 「配置管理網域」。

對於許多使用者而言，一個應用程式伺服器實例就符合他們的需要了。不過，依據您的環境，您可能想建立一個或多個附加的應用程式伺服器實例。例如，在開發環境下，您可以使用不同的應用程式伺服器實例來測試不同的 Sun ONE Application Server 配置，或比對和測試不同的應用程式部署。由於您可以輕易地加入或刪除應用程式伺服器實例，因而可以在開發時透過這些實例建立暫時的「沙箱」區域進行試驗。

此外，對於每個應用程式伺服器實例，您也可以建立虛擬伺服器。在單一安裝的應用程式伺服器實例內，您可以爲公司或個人提供領域名稱、IP 位址以及某些管理功能。對於使用者而言，看起來好像使用者有自己的 Web 伺服器，但沒有硬體和基本的伺服器維護功能。這些虛擬伺服器不擴充應用程式伺服器實例。如需關於虛擬伺服器的更多資訊，請參閱第 15 章 「使用虛擬伺服器」。

在作業部署中，您可以使用虛擬伺服器代替多重應用程式伺服器實例，用於多種目的。不過，如果虛擬伺服器不符合您的需要，也可以使用多重應用程式伺服器實例。

啟動和停止應用程式伺服器實例

Sun ONE Application Server 實例不會自動啓動。您啓動一個實例後，該實例將一直執行，直到您停止它。若您停止一個應用程式伺服器實例，該實例將停止接受新連線，然後等待所有未完成的連線完成。如果您的機器當機或離線，伺服器將結束，其正在處理的任何請求均可能遺失。

您可以使用下列主題中包含的幾種方法之一，啓動和停止應用程式伺服器實例：

- 使用管理介面中的 [Start] 和 [Stop] 按鈕
- 使用 start-instance 與 stop-instance 指令
- 使用 Windows 服務 (Windows)
- 使用 startserv 和 stopserv 程序檔

注意 如果您的伺服器安裝了安全性模組，將需要您輸入適當的密碼，然後再啓動或停止伺服器。

如果您安裝了伺服器證書，Sun ONE Application Server 在啓動之前，將提示管理員輸入鍵值資料庫密碼。如果希望能夠重新啓動非隨附式的 Sun ONE Application Server，您需要將該密碼儲存在 `password.conf` 檔案中。僅當您的系統受到適當保護時才能這樣做，以免洩漏該檔案和鍵值資料庫。如需關於建立和使用 `password.conf` 的更多資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

注意 在 UNIX 上，某些 Sun ONE Application Server 安裝可能需要存取比您的作業系統依預設所允許的數目更多的記憶體和/或檔案描述元。如果您無法啓動伺服器，請使用 `ulimit` 指令，檢查由您的作業系統施加的資源限制。作業系統的 `ulimit` 線上說明手冊應該會提供更多資訊。

使用管理介面中的 [Start] 和 [Stop] 按鈕

使用管理介面啓動和停止伺服器的步驟：

1. 在左窗格中，於 [App Server Instances] 下，按一下要啓動或停止的實例名稱。
2. 在右窗格中，按一下 [Start] 或 [Stop]；或在 [General] 標籤中，按一下 [Start] 或 [Stop]。
3. 當應用程式伺服器實例成功啓動或停止時，您會看到一則訊息。

使用 start-instance 與 stop-instance 指令

使用指令行介面公用程式 asadmin，您可以經由指令提示或經由程序檔啓動和停止應用程式伺服器實例。使用 start-instance 與 stop-instance 指令。

這些指令使用下列語法：

start-instance [--user *admin_user*] [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--local=*true/false*] [--domain *domain_name*] [--debug=*true/false*] [--passwordfile *file_name*] [--secure | -s] *instance_name*

stop-instance [--user *admin_user*] [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--local=*true/false*] [--domain *domain_name*] [--passwordfile *file_name*] [--secure | -s] *instance_name*

這些指令有一個 local 選項，使用該選項，您便可啓動或停止伺服器 (無需經由管理伺服器)。如果您使用 local 選項，則無需指定 host、port、user 與 password (或 passwordfile) 選項。

如需關於這些指令語法的資訊，請使用 asadmin 輔助說明。如需關於使用 asadmin 的資訊，請參閱附錄 A「使用指令行介面」。

使用 Windows 服務 (Windows)

您可以使用 Windows 中的服務控制台，啓動伺服器。

請執行以下步驟：

1. 在 [控制台] 中，按一下 [管理工具]。
2. 按一下 [服務]。
3. 捲動服務清單並按兩下用於您伺服器的服務。

   它稱爲 Sun Application Server (*domain_name:instance_name*)。例如，Sun Application Server (domain1:server1)。
4. 按一下 [Start] 或 [Stop]。
5. 按一下 [OK]。

使用 startserv 和 stopserv 程序檔

若要使用 `startserv` 和 `stopserv` 程序檔，請在指令行提示處移至目錄：

instance_dir/bin

其中，*install_dir* 是安裝伺服器的目錄，*domain_dir* 是領域目錄，而 *instance_dir* 是您要啓動的實例名稱。

對於 UNIX，請鍵入：

```
./asadmin startserv
```

如果伺服器在號碼低於 1024 的連接埠上運行，請以超級使用者身份登入；否則，請以超級使用者身份或使用伺服器的使用者帳號登入。

您可以在行結尾處使用選擇性參數 `-i`。`-i` 選項可以在 `inittab` 模式下執行伺服器，因此，如果伺服器程序被終止或當機，`inittab` 將爲您重新啓動伺服器。該選項也可防止伺服器將其本身放入後台程序中。

注意 如果伺服器已經在執行，`startserv` 指令將失敗。您必須首先停止伺服器，然後再使用 `startserv` 指令。並且，如果伺服器啓動失敗，您應該終止程序，然後再嘗試重新啓動。

對於 Windows，請鍵入：

```
startserv
```

若要手動停止伺服器，請在指令行提示處移至目錄：

instance_dir/bin

其中，*install_dir* 是您安裝伺服器的目錄，而 *instance_dir* 是您要啓動的實例名稱。

對於 UNIX，請鍵入：

```
./asadmin stopserv
```

如果使用 `/etc/inittab` 檔案重新啓動伺服器，您必須從 `/etc/inittab` 中移除啓動伺服器的行，並鍵入 `kill -1 1`，然後嘗試停止伺服器。否則，伺服器在停止後會自動重新啓動。

對於 Windows，請鍵入：

```
stopserv
```

在除錯模式下啟動應用程式伺服器實例

如果開發人員要對其 J2EE 應用程式除錯，可以在除錯模式下執行應用程式伺服器實例。

在除錯模式下啓動伺服器的步驟：

1. 存取管理介面並按一下您要在除錯模式下啓動的應用程式伺服器實例名稱。
2. 按一下 [General] 標籤。
3. 按一下 [Run in Debug Mode] 旁邊的核取方塊。
4. 重新啓動應用程式伺服器實例。

除錯模式會變更 JVM 設定。[Debug Enabled] 設定爲 true，並且除錯選項發生變更。如需關於 JVM 除錯選項的更多資訊，請參閱 Java 平台除錯程式選項說明文件 (位於 http://java.sun.com/products/jpda/doc/conninv.html)。

若要經由指令行介面啓動除錯模式中的應用程式伺服器實例，請使用 `asadmin` 公用程式的 `start-instance` 指令，並將 `debug` 選項設定爲 true。如需有關指令語法的更多資訊，請參閱指令行介面的線上輔助說明。

設定終止逾時

當您停止應用程式伺服器實例時，它會停止接受新連線。然後等待所有未完成的連線完成。伺服器在逾時前的等待時間可在 `init.conf` 檔案中配置，該檔案可在 *instance_dir*`/config/` 中找到。依預設，該值設定爲 30 秒。若要變更該值，請將以下行加入至 `init.conf` ：

```
TerminateTimeout seconds
```

其中，*seconds* 表示伺服器在逾時前將等待的秒數。

配置該值的優點是伺服器將等待更長的時間以讓連線完成。不過，由於伺服器經常開啓來自非回應用戶端的連線，因此延長終止逾時可能會延長其用於伺服器關機的時間。

自動重新啟動應用程式伺服器實例 (UNIX)

您可以使用下列方法之一，重新啓動應用程式伺服器實例：

- 經由 `/etc/inittab` 檔案自動重新啓動。

  請注意，如果使用的 UNIX 版本不源自 System V，您將不能使用 `/etc/inittab` 檔案。

- 在機器重新啓動時，以 `/etc/rc2.d` 中的常駐程式自動重新啓動實例。
- 手動重新啓動實例。請參閱第 65 頁的 「啓動和停止應用程式伺服器實例」和第 73 頁的 「刪除應用程式伺服器實例」。

本章節包含下列主題：

- 關於自動重新啓動
- 以 /etc/inittab 自動重新啓動 (UNIX)
- 以系統 RC 程序檔自動重新啓動 (UNIX)

關於自動重新啟動

由於安裝程序檔無法編輯 `/etc/rc.local` 或 `/etc/inittab` 檔案，因而您必須使用文字編輯程式編輯這些檔案。如果您不瞭解如何編輯這些檔案，請洽詢您的系統管理員或參考系統說明文件。

通常，您無法用其中的任何一個檔案啓動已啓用 SSL 的伺服器，因爲該伺服器在啓動前需要密碼。如果以純文字形式將密碼保留在檔案中，雖然您可以自動啓動啓用了 SSL 的伺服器，但建議您**不要**這樣做。

小心 以一般文字形式將啓動了 SSL 的伺服器密碼保留在該伺服器的 startserv 程序檔內，會造成很大的安全性風險。可以存取該檔案的任何使用者都可以存取啓動了 SSL 的伺服器密碼。以一般文字形式保留啓動了 SSL 的伺服器密碼前，請考慮安全性風險。

伺服器的 `startserv` 程序檔、鍵對檔案以及鍵密碼應該爲超級使用者所有 (或者，如果非超級使用者安裝了伺服器，則爲該使用者帳號)，並且僅所有者可以讀取和寫入它們。

以 /etc/inittab 自動重新啟動 (UNIX)

若要使用 `inittab` 重新啓動伺服器，請將以下文字放入檔案 `/etc/inittab` 中的一行上：

`http:2:respawn:`*install_dir*`/`*path_to_domain_dir*`/`*instance_dir*`/bin/startserv -start -i`

其中，*install_dir* 是安裝伺服器的目錄，*path_to_domain_dir* 是網域路徑，*instance_dir* 是伺服器的目錄。

`-i` 選項可防止伺服器將其自身放置在後台程序中。

您必須在停止伺服器之前移除該行，否則伺服器會自動重新啓動。

以系統 RC 程序檔自動重新啟動 (UNIX)

如果使用 `/etc/rc.local` 或系統上的對等檔案，請將下列行放入 `/etc/rc.local` 中：

install_dir`/`*path_to_domain_dir*`/`*instance_dir*`/bin/startserv`

以您安裝伺服器的目錄取代 *install_dir*，以領域路徑取代 *path_to_domain_dir*，以應用程式伺服器實例的名稱取代 *instance_dir*。

手動重新啟動應用程式伺服器實例 (UNIX)

在 UNIX 上，您可以選擇手動重新啓動伺服器實例。與停止伺服器實例然後再啓動不同，重新啓動不停止監視程式。如需關於監視程式的資訊，請參閱第 72 頁的「關於監視程式」。

注意 如果您透過編輯手動變更了配置檔案，則必須在重新啓動伺服器之前套用變更，方法爲使用管理介面中的 [Apply Changes] 按鈕或者使用 `asadmin reconfig` 指令 (同時將 `keepmanualchanges` 選項設定爲 `true`)。如需關於套用變更的更多資訊，請參閱第 74 頁的「將變更套用至應用程式伺服器實例」。

下列主題中涵蓋三種重新啓動伺服器實例的方法：

- 使用 [Restart] 按鈕重新啓動伺服器實例 (UNIX)
- 使用 restart-instance 指令重新啓動伺服器實例 (UNIX)

- 使用 restartserv 程序檔重新啓動伺服器實例 (UNIX)

使用 [Restart] 按鈕重新啟動伺服器實例 (UNIX)

使用管理介面重新啓動伺服器實例的步驟：

1. 在左窗格中，於 [App Server Instances] 下，按一下要重新啓動的實例名稱。
2. 在右窗格中，按一下 [Restart]。
3. 當應用程式伺服器實例重新啓動成功時，您會看到一則訊息。

使用 restart-instance 指令重新啟動伺服器實例 (UNIX)

使用指令行介面公用程式 `asadmin`，您可以經由指令行或經由程序檔啓動和停止應用程式伺服器實例。使用指令 `restart-instance`。該指令使用以下語法：

`restart-instance [--user` *admin_user*`] [--password` *admin_password*`] [--host` *admin_host*`] [--port` *admin_port*`] [--local=`*true/false*`] [--domain` *domain_name*`] [--passwordfile` *file_name*`] [--secure | -s]` *instance_name*

該指令有一個 `local` 選項，使用該選項，您便可重新啓動伺服器實例 (無需經由管理伺服器)。

如需關於這些指令語法的資訊，請使用 `asadmin` 輔助說明。如需關於使用 `asadmin` 的資訊，請參閱附錄 A「使用指令行介面」。

使用 restartserv 程序檔重新啟動伺服器實例 (UNIX)

若要使用 `restartserv` 程序檔，請在指令行提示處移至目錄：

instance_dir/bin

其中，*install_dir* 是安裝伺服器的目錄，*domain_dir* 是領域目錄，而 *instance_dir* 是您要啓動的實例名稱。

類型：

```
./asadmin restartserv
```

如果伺服器在號碼低於 1024 的連接埠上運行，請以超級使用者身份登入；否則，請以超級使用者身份或使用伺服器的使用者帳號登入。

關於監視程式

監視程式 (在 UNIX 上爲 `appserv-wdog`，在 Windows 上爲 `appservd-wdog.exe`) 是您 Sun ONE Application Server 隨附的程式。該程式執行下列工作：

- 啓動伺服器
- 停止伺服器
- 如果啓用了 SSL/TLS，當伺服器啓動時，將提示管理員提供可信任的資料庫密碼
- 如果伺服器當機，則重新啓動它

監視程式在後台執行，無需使用者干預。您不得配置或以其他方式變更該程式。一個監視程式可針對每個應用程式伺服器實例 (包括管理伺服器) 執行。

在 UNIX 上，每個監視程式可爲原始應用程式伺服器 (`appservd`) 程序產生一個程序，繼而該程序也會產生可接受請求的 `appservd` 程序。由於監視程序 ID 可以啓動伺服器，因此其顯示在 *instance_dir*/logs 中的 `pid` 日誌檔內。

注意 UNIX 平台上的 `appservd` 程序：雖然您會注意到，在 Window 上爲每個應用程式伺服器實例啓動了單一 appservd 程序，但在 UNIX 系統上，卻爲每個應用程式伺服器實例啓動了兩個 appservd 程序。

在 UNIX 上，一個 appservd 程序指「原始」程序，第二個 appservd 程序指「工作者」程序。工作者程序是執行應用程式請求之實際處理的程序，而原始程序則作爲起支配作用的控制程式。在未來版本的應用程式伺服器中，您將可以選擇定義每個應用程式伺服器實例的工作者程序數目。在初始版本的產品中，僅支援每個應用程式伺服器實例使用一個工作者程序。

加入應用程式伺服器實例

使用管理介面加入應用程式伺服器實例的步驟：

1. 存取管理介面，並按一下左窗格中的 [App Server Instances]。
2. 按一下 [General] 標籤。
3. 在 [Application Server Instances] 頁面中，按一下 [New]。
4. 在 [Create New Instance] 頁面中，提供實例名稱和連接埠號。

   實例名稱對於該管理伺服器和領域必須是唯一的。連接埠號不得用於機器上的任何其他程序。

如果您使用的是 UNIX，也可以指定一個 UNIX 使用者，以便實例以該使用者身份執行。

5. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

若要藉由指令行介面加入另一個應用程式伺服器實例，請使用 `asadmin` 公用程式的 `create-instance` 指令，該指令具有下列語法：

```
asadmin create-instance [--user admin_user] [--password admin_password]
[--host host] [--port port] [--sysuser sys_user] [--domain domain_name]
[--local=true/false] [--passwordfile file_name] [--secure | -s]
--instanceport instance_port instance_name
```

該指令有一個 `local` 選項，使用該選項，您便可重新啓動伺服器實例 (無需經由管理伺服器)。`sysuser` 選項僅用於 UNIX。

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

刪除應用程式伺服器實例

您可以從您的管理領域中刪除應用程式伺服器實例。在刪除之前，請確定您不再需要該應用程式伺服器實例，因爲此程序無法還原。

使用管理介面從機器中刪除應用程式伺服器實例的步驟：

1. 存取管理介面，並按一下您要移除的應用程式伺服器實例名稱。
2. 按一下 [General] 標籤。
3. 按一下 [Delete]。

如需更多資訊，請參閱線上輔助說明。

若要使用指令行介面從機器中刪除一個應用程式伺服器實例，請使用 `asadmin` 公用程式的 `delete-instance` 指令，該指令具有下列語法：

```
asadmin delete-instance [--user admin_user] [--password admin_password]
[--host admin_host] [--port admin_port] [--domain domain_name]
[--local=true/false] [--passwordfile file_name] [--secure | -s] instance_name
```

該指令有一個 `local` 選項，使用該選項，您便可刪除伺服器實例 (無需經由管理伺服器)。

如需有關指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

將變更套用至應用程式伺服器實例

使用管理介面或指令行介面變更配置資訊時，不會立即套用變更，而是將變更儲存到位於 *server_instance*/config/backup 下的特殊檔案中。管理介面和指令行介面可顯示儲存在以上目錄下檔案中的配置值。直到您套用了所做的變更，其才會生效。套用變更也稱爲重新配置伺服器。當您套用變更後，自您上次套用變更以來所作的所有配置變更均生效。請注意，重新啓動實例不會自動套用變更。

如果您已對需要套用變更的伺服器實例配置進行了變更，則會在左窗格樹檢視中的應用程式伺服器實例旁邊顯示一個黃色圖示，當您存取伺服器實例時，會在標題中以及伺服器實例的主頁面中顯示黃色圖示。

圖 4-1 警告圖示

使用管理介面套用變更至應用程式伺服器實例的步驟：

1. 存取管理介面，並按一下您要重新配置的應用程式伺服器實例名稱。
2. 按一下 [General] 標籤。
3. 按一下 [Apply Changes]。

   套用變更後，螢幕會顯示一條訊息。

若要藉由指令行介面重新配置應用程式伺服器實例，請使用 `asadmin` 公用程式的 `reconfig` 指令，該指令具有下列語法：

`asadmin reconfig --user` *admin_user* `[--password` *admin_password*`] [--host` *admin_host*`] [--port` *admin_port*`] [--passwordfile` *file_name*`] [--secure | -s] [--discardmanualchanges=`*true/false* `| --keepmanualchanges=`*true/false*`]` *instance_name*

如果您已透過手動編輯手動變更了配置檔案，在重新配置期間您必須使用 `keepmanualchanges=true` 保留這些編輯 (該選項預設爲 false)。如果您設定了 `discardmanualchanges=true`，便捨棄了手動進行的所有變更。設定 `discardmanulchanges=false` (預設) 的涵義與設定 `keepmanualchanges=true` 不同。將其設定爲 false，反而等同於不指定 `discardmanualchanges` 選項。

如需有關指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

對於某些特性，您需要重新啓動伺服器並套用變更，以讓您的變更生效。這些特性包括配置檔案 `init.conf` 和 `obj.conf` 中設定的所有特性，以及 `server.xml` 中的某些特性。如需關於這些檔案的資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

如果變更需要重新啓動，伺服器將警告您，方式爲在左窗格樹檢視中的伺服器實例旁邊放置一個黃色警告圖示，當您存取伺服器實例時，會在標題中以及伺服器實例的主頁面中放置黃色警告圖示。標題與頁面中的訊息可指示需要重新啓動。一旦您重新啓動了伺服器實例，黃色警告圖示將消失。

不需要重新啓動的 `server.xml` 設定包括以下內容：

- 部署、取消部署與重新部署 J2EE 應用程式 (EAR 檔案)、EJB 模組 (JAR 檔案)、Web 模組 (WAR 檔案)、連接器 (RAR 檔案)。請注意，這些設定也不需要套用變更。
- 啓用與停用 J2EE 應用程式 (EAR 檔案)、EJB 模組 (JAR 檔案)、Web 模組 (WAR 檔案)、連接器 (RAR 檔案)。
- 建立、更新和刪除資源。
- 針對 EJB 容器或 MDB 容器，將啓用的監視設定爲 true/false。
- 對 HTTP 與 Web 容器功能所做的變更 (即在 `server.xml` 中，對 `http-service` 與 `web-container` 及其子元素的變更)。

檢視應用程式伺服器實例狀況

使用管理介面，您可以檢視一個伺服器已經啓動還是停止，以及基本的應用程式伺服器實例設定。

檢視應用程式伺服器實例狀況的步驟：

1. 在左窗格中，按一下應用程式伺服器實例名稱。
2. 在右窗格中，按一下 [General] 標籤。

   您將看到伺服器是否在執行，也將看到主機名稱、連接埠號、安裝目錄以及 Sun ONE Application Server 軟體的版本。

若要藉由指令行介面檢視應用程式伺服器實例的狀態，請使用 `asadmin` 公用程式的 `show-instance-status` 指令。狀況爲「正在啓動」、「已啓動」、「正在停止」或「已停止」。該指令使用以下語法：

```
asadmin show-instance-status --user admin_user [--password
admin_password] [--host admin_host] [--port admin_port] [--passwordfile
file_name] [--secure | -s] instance_name
```

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

配置 JVM 設定

您可以爲您的應用程式伺服器實例配置 Java Virtual Machine (JVM) 設定。這些設定包括您的 Java Home 位置、編譯程式選項、除錯選項以及測量程式資訊。配置這些設定的一個原因是爲了提昇效能。如需有關效能的更多資訊，請參閱「*Sun ONE Application Server Performance and Tuning Guide*」。

本章節描述下列主題：

- 配置一般設定
- 配置路徑設定
- 配置 JVM 選項
- 配置 JVM 測量程式
- 使用指令行介面配置 JVM 設定

配置一般設定

在管理介面中配置 JVM 一般選項的步驟：

1. 在左窗格中，按一下應用程式伺服器實例名稱。
2. 在右窗格中，按一下 [JVM] 標籤。
3. 按一下 [General]。
4. 設定 Java Home。

   Java Home 是安裝 Java Developer's Kit (JDK) 的目錄路徑。Sun ONE Application Server 支援 Sun JDK 1.4.0_02 或更高版本。
5. 選擇啓用除錯還是設定除錯選項。

   可從 http://java.sun.com/products/jpda/doc/conninv.html#Invocation 取得除錯選項清單。

6. 選擇 rmic 選項。

   rmic 選項欄位顯示在應用程式部署時傳送到 RMI 編譯程式的 rmic 選項。`-keepgenerated` 選項可爲存根和聯繫儲存產生的來源。如需關於 rmic 指令的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to Enterprise Java Beans*」。

7. 按一下 [Save]。

配置路徑設定

在管理介面中配置 JVM 路徑設定的步驟：

1. 在左窗格中，按一下應用程式伺服器實例名稱。
2. 在右窗格中，按一下 [JVM] 標籤。
3. 按一下 [Path Settings]。
4. 選擇用於系統類別路徑的尾碼。
5. 選擇是否忽略環境類別路徑。

   如果您不忽略此類別路徑，將讀取 CLASSPATH 環境變數，並將其附加至 Sun ONE Application Server 類別路徑。CLASSPATH 環境變數於類別路徑字尾之後加入，恰在其結尾處。

   對於開發環境，應該使用類別路徑。對於生產環境，應該忽略該類別路徑以防止環境變數副作用。

6. 設定本端程式庫路徑的字首和字尾。

   本端程式庫路徑是應用程式伺服器安裝相對路徑 (用於其本端共用程式庫)、標準 JRE 本端程式庫路徑、Shell 環境設定 (UNIX 上的 LD_LIBRARY_PATH) 以及測量程式元素中指定的任何路徑的自動建構之連結。由於該本端程式庫路徑是合成的，因此其不在伺服器配置中明確出現。

7. 按一下 [Save]。

配置 JVM 選項

在管理介面中設定 JVM 指令行選項的步驟：

1. 在左窗格中，按一下應用程式伺服器實例名稱。
2. 在右窗格中，按一下 [JVM] 標籤。
3. 按一下 [JVM Options]。
4. 若要加入一個 JVM 選項，請在螢幕頂端的文字欄位中鍵入該選項，然後按一下 [Add]。
5. 若要刪除一個 JVM 選項，請按一下其旁邊的核取方塊，然後按一下 [Delete]。
6. 若要編輯一個 JVM 選項，請編輯 [JVM Option] 欄位內的文字，然後按一下 [Save]。

如需關於特定 JVM 選項的資訊，請參閱
http://java.sun.com/docs/hotspot/VMOptions.html

配置 JVM 測量程式

在管理介面中配置 JVM 測量程式的步驟：

1. 在左窗格中，按一下應用程式伺服器實例名稱。
2. 在右窗格中，按一下 [JVM] 標籤。
3. 按一下 [Profiler]。
4. 指定測量程式的名稱、其類別路徑與本端程式庫路徑，以及其是否已啓用。
5. 若要爲測量程式加入一個 JVM 選項，請在螢幕頂端的文字欄位中鍵入該選項，然後按一下 [Add]。
6. 若要刪除測量程式的一個 JVM 選項，請按一下其旁邊的核取方塊，然後按一下 [Delete]。
7. 若要編輯測量程式的一個 JVM 選項，請編輯 [JVM Option] 欄位內的文字，然後按一下 [Save]。

如需關於測量程式的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide*」。

使用指令行介面配置 JVM 設定

若要使用指令行介面的 asadmin 公用程式配置 JVM 設定，請使用以下指令：

若要從實例中取得所有屬性，請使用：

```
asadmin> get server_instance.java-config.*
```

若要在 server1 中取得名爲 classpathprefix 的屬性，請使用：

```
asadmin> get server1.java-config.classpathprefix
```

若要在 server1 中設定稱爲 classpathprefix 的屬性，請使用：

```
asadmin> set server1.java-config.classpathprefix=com.sun
```

以上範例均假定您已經在您的環境變數中設定使用者、密碼、主機和連接埠。如需完整的屬性清單，請參閱附錄 A「使用指令行介面」。

若要使用指令行介面的 asadmin 公用程式設定 JVM 選項，請使用以下指令：

```
asadmin> create-jvm-options --user admin_user [--password admin_password]
[--host host] [--port port] [--secure | -s] [--instance instance_name]
[--profiler=true/false]
(jvm_option_name=jvm_option_value) [:jvm_option_name=jvm_option_name]*
```

```
asadmin> delete-jvm-options --user admin_user [--password admin_password]
[--host host] [--port port] [--secure | -s] [--instance instance_name]
[--profiler=true/false]
(jvm_option_name=jvm_option_value) [:jvm_option_name=jvm_option_name]*
```

注意：您可以輸入多個 JVM 選項 (以冒號分隔)。如果測量程式使用這些選項，請將 --profiler 設定爲 true。

如需有關指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 asadmin 的更多資訊，請參閱附錄 A「使用指令行介面」。

配置記錄設定和監視設定

[Logging] 和 [Monitoring] 標籤上的設定是個別章內包含的記錄和監視設定。如需關於記錄的資訊，請參閱第 5 章「使用記錄功能」如需關於監視和 SNMP 設定的資訊，請參閱第 6 章「監視 Sun ONE Application Server」。

變更應用程式伺服器實例進階設定

應用程式伺服器實例還具有附加設定，顯示實例的語言環境 (可決定諸如字元集和語言的設定)、伺服器日誌檔路徑、已部署應用程式的目錄路徑，以及儲存鈍化 Bean 和永久性 HTTP 階段作業的階段作業儲存目錄路徑。

此外，您也可以啓用應用程式重新載入功能並輪詢重新載入的間隔時間。動態應用程式重新載入可自動檢查應用程式有無變更，如果已變更，它會自動服務於更新後的版本。一般而言，您應該在開發環境內 (而非生產環境) 啓用動態重新載入。輪詢間隔時間可指定應用程式伺服器檢查應用程式有無更新的間隔時間。

使用管理介面變更應用程式伺服器實例設定的步驟：

1. 在左窗格中，按一下應用程式伺服器實例名稱。
2. 在應用程式伺服器實例頁面上，按一下 [Advanced] 標籤。
3. 在欄位中輸入所需的值。
4. 按一下 [Save]。

若要藉由指令行介面的 `asadmin` 公用程式變更伺服器實例進階設定，請使用 get 和 set 指令。當您取得一個伺服器實例的所有屬性時，

若要從實例中取得所有屬性，請使用：

```
asadmin get instance_name.*
```

例如：

```
asadmin get server1.*
```

若要爲 `server1` 取得稱爲 `logRoot` 的屬性，請使用：

```
asadmin get server1.logRoot
```

若要爲 `server1` 設定稱爲 `logRoot` 的屬性，請使用：

```
asadmin set server1.logRoot=/space/log
```

以上範例均假定您已經在您的環境變數中設定使用者、密碼、主機和連接埠。如需有關指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

第 5 章

使用記錄功能

本章描述 Sun ONE Application Server 的記錄功能。另外，還論述可能使用記錄功能的元件。

本章包含以下主題：

- 關於記錄
- 在 UNIX 和 Windows 平台上記錄
- 使用日誌層級
- 關於虛擬伺服器和記錄
- 關於記錄程式
- 關於用戶端記錄
- 重新導向應用程式與伺服器日誌輸出
- 日誌檔管理
- 透過指令行介面配置記錄
- 透過管理介面配置記錄
- 配置錯誤記錄的指令
- 檢視存取日誌檔
- 檢視事件日誌檔
- 設定日誌偏好設定
- 執行日誌分析程式
- 檢視事件 (Windows 2000 Pro)

關於記錄

在應用程式中使用時，記錄是很有用的除錯和診斷工具。它也會提高開發人員的生產率。應用程式伺服器專用的日誌輸出可以幫助識別和診斷伺服器配置與部署問題。

Sun ONE Application Server 內的記錄使用的是 Java 記錄 API。Sun ONE Application Server 將記錄資訊收集和儲存在兩個日誌檔中，即 `logs` 目錄下的 `access.log` 與 `server.log`。您也可將日誌導入自己的日誌檔中。

記錄的訊息所提供的資訊比僅訊息本身提供的資訊要多。提供的附加資訊包括：

- 事件的日期與時間。
- 事件的日誌層級。應用程式伺服器指定日誌層級 ID 或名稱。
- 程序 ID (PID)。應用程式伺服器程序的 PID。
- (選擇性的) 虛擬伺服器 ID (vsid)。產生訊息的 vsid。
- 訊息 ID。子系統與四位整數。
- 訊息資料。

根據用於記錄的平台以及爲該平台啓用的記錄服務之不同，附加訊息資訊的類型和次序也不同。若要啓用虛擬伺服器 ID 以取得記錄的訊息，請參閱第 102 頁的「配置日誌服務」。

在 UNIX 和 Windows 平台上記錄

本章節論述日誌檔的建立方法。另外，本章節還包括下列主題：

- server.log 內的預設記錄
- 使用 syslog 進行記錄
- 使用 Windows 事件日誌記錄

server.log 內的預設記錄

在 UNIX 與 Windows 兩個平台上，於 log 子目錄下的 server.log 內建立日誌檔。在該單一檔案內，收集來自實例之所有伺服器元件和虛擬伺服器的日誌。

可以設定整個伺服器的預設日誌層級。不過，您也可在子系統層級上置換特定子系統的預設日誌層級。您也可將 stdout 和 stderr 重新導向至伺服器的事件日誌，並將日誌輸出導向至作業系統的系統日誌。此外，您也可將 stdout 和 stderr 內容導向至伺服器事件日誌。依預設，日誌訊息除發送至指定的伺服器日誌檔以外，也發送至 stderr。

另一個可用功能是以日誌訊息記錄虛擬伺服器 ID。使用多重虛擬伺服器將訊息記錄至同一個日誌檔時，此功能非常有用。您可以選擇將日誌訊息寫入至系統日誌。若這樣做，在 server.log 檔案上將不執行記錄。相反，UNIX 上的 syslog 記錄服務或 Windows 平台上的系統記錄服務將用於產生和管理日誌。

您也可以使用 server.xml 屬性控制該檔案的內容。如需關於 server.xml 檔案的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

server.log 的範例

以下是 server.log 的範例。

時間標記 , 日誌層級 , (PID vsid (選擇性的)): messageID: 訊息

```
[01/Aug/2002:11:39:31] INFO ( 1224): CORE1116: Sun ONE Application
Server 7.0

[01/Aug/2002:11:39:36] INFO ( 1224): CORE5076: Using [Java
HotSpot(TM) Server VM, Version 1.4.0_02-20020712] from [Sun
Microsystems Inc.]

[01/Aug/2002:11:39:50] INFO ( 1224): JMS5023: JMS service
successfully started. Instance Name = domain1_server1, Home =
[D:\install_7_29\imq\bin].

[01/Aug/2002:11:39:53] INFO ( 1224): CIS0056: Creating TCP
ServerConnection at [EndPoint
[IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

[01/Aug/2002:11:39:53] INFO ( 1224): CIS0057: Created TCP
ServerConnection at [EndPoint
[IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

[01/Aug/2002:11:39:54] INFO ( 1224): CIS0054: Creating TCP
Connection from [-] to [EndPoint
[IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]
```

注意 重新導向有關預先編譯 JSP 的日誌訊息：

有關預先編譯 JSP 的日誌訊息依預設儲存在管理伺服器日誌檔中，該日誌檔位於 {*domain_root*}/{*domain_name*}/admin-server/logs/server.log 之下。

由於所有訊息都記錄至同一檔案，因而，使用預先編譯的 JSP 部署應用程式時，拋出的異常或錯誤在普通日誌檔訊息卷次中可能會遺失。在給定的領域下，將多重應用程式部署至多重實例時，需要仔細檢查管理伺服器中的日誌訊息，以確定與特定應用程式之 JSP 相關的任何異常。這會導致冗餘。

因此，最好在伺服器實例的 server.log 檔案中，而非管理伺服器的 server.log 檔案中，記錄有關使用預先編譯 JSP 所部署的應用程式之訊息。

若要將日誌訊息重新導向至您的 Sun ONE Application Server 實例 server.log 檔案，請在管理員介面中變更此日誌檔的路徑。請參閱配置日誌服務，以取得更多資訊。

使用 syslog 進行記錄

對於需要集中記錄的穩定作業環境而言，syslog 適用。對於經常需要日誌輸出以便診斷和除錯的環境而言，個別伺服器實例或虛擬伺服器日誌可能更容易管理。

注意

- 一個檔案中針對伺服器實例和管理伺服器的所有記錄資料，可能很難讀取和除錯。建議您僅對可順利執行的已部署應用程式使用 syslog 主日誌檔。
- 記錄的訊息可與 Solaris 常駐應用程式的所有其他日誌混合。

使用 syslog 日誌檔，並結合 syslogd 以及系統日誌常駐程式，您可以將 syslog.conf 檔案配置爲：

- 將訊息記錄到適當的系統日誌中
- 將訊息寫入至系統主控台
- 將記錄的訊息轉寄至使用者清單，或透過網路將其轉寄至另一台主機上的另一個 syslogd

注意 遵循 Sun ONE Application Server 的安裝，該伺服器的日誌服務元素屬性 `use-system-logging` 未啓用。這意味著依預設，日誌未導向至 UNIX 上的 `syslog` 或 Windows 平台上的 Windows 事件日誌。您可以將記錄導向至 `syslog` 或 Windows 事件日誌，方法是在「Sun ONE Application Server Configuration File Reference」中描述的 `server.xml` 之 Server 元素上啓用此屬性。設定 `use-system-logging` 之前，請參閱第 95 頁「日誌檔管理」。

配置 syslog

若要取得更高的管理性和可讀性，您可以配置 `/etc` 目錄下的 `syslog.conf`，將準備儲存的嚴重程度較低的訊息導入個別檔案中。

配置 syslog 的步驟：

1. 若要將準備儲存的嚴重性較低的訊息導入個別檔案中，請在 Solaris 的 `syslog.conf` 檔案中加入以下指令：

   ```
   daemon.debug  /var/adm/iasdebug
   ```

注意 日誌訊息導入至 Windows 事件日誌後，僅記錄層級爲「資訊」、「警告」、「伺服器」、「警示」或「嚴重」的訊息。

2. 向 `syslogd` 發出掛斷訊號。可以使用以下指令完成此動作：

   ```
   kill -HUP <PID syslogd>
   ```

3. 移往管理介面中的管理伺服器，然後選取 [寫入系統日誌] 選項。儲存並套用變更。重新啓動管理伺服器，以使變更生效。

   隨之出現配置的 Solaris `syslog.conf` 檔案範例：

   ```
   #ident"@(#)syslog.conf1.598/12/14 SMI"/* SunOS 5.0 */
   #
   # 版權所有 (c) 1991-1998 by Sun Microsystems, Inc.
   # 保留所有權利。
   #
   # syslog 配置檔案。
   #
   ```

```
# 該檔案以 m4 處理，因此請小心引用（`'）符合 m4 保留字的名稱。
# 同時，在 ifdef's 之內，必須引用包含逗號的引數。
#
*.err;kern.notice;auth.notice/dev/sysmsg
*.err;kern.debug;mail.crit/var/adm/messages
daemon.info;daemon.err;daemon.debug;daemon.alert;daemon.crit;dae
mon.warning/var/adm/iaslog
daemon.debug/var/adm/iasdebug
#daemon.notice;/var/adm/iaslognotice
#daemon.warning;/var/adm/iaslogwarning
#daemon.alert;/var/adm/iaslogalert
#daemon.err;/var/adm/iaslogerr

#*.alert;kern.err;daemon.erroperator
#*.alertroot
*.emerg*

# 如果非日誌主機選擇將認證訊息
# 發送至日誌主機，un-comment 會出現在下列行上：
#auth.noticeifdef(`LOGHOST', /var/log/authlog, @loghost)

mail.debugifdef(`LOGHOST', /var/log/syslog, @loghost)
#
# 非日誌主機將使用下列行讓「使用者」日誌訊息在本機記錄。
#
ifdef(`LOGHOST', ,
user.err/dev/sysmsg
user.err/var/adm/messages
user.alert`root, operator'
user.emerg*
)
```

如需更多資訊，請參閱 `syslog.conf` 線上援助頁。

對 `syslog.conf` 做出的任何變更，都需要重新啓動 Sun ONE Application Server 才能讓變更生效。

由於記錄至 `syslog` 意味著所有 Sun ONE Application Server 的日誌以及其他常駐應用程式都收集在同一個檔案中，所以，藉由下列資訊增強記錄的訊息，以從特定伺服器或虛擬伺服器實例中識別 Sun ONE Application Server 特定訊息：

- 唯一的訊息 ID
- 時間標記
- 實例名稱
- 程式名稱 (`appservd` 或 `appserv-wdog`)
- 程序 ID (應用程式伺服器程序的 PID)
- 執行緒 ID (選擇性的)
- 伺服器 ID

在 `server.xml` 檔案中可以配置伺服器實例與虛擬伺服器實例的日誌服務。第 90 頁的「關於虛擬伺服器和記錄」一節中描述虛擬伺服器實例的日誌服務配置。第 102 頁的「透過管理介面配置記錄」一節中描述伺服器實例的日誌服務配置。

透過適當子系統和元件的管理介面可以配置日誌層級。

如需關於 UNIX 作業環境中使用的 `syslog` 記錄機制的更多資訊，請於終端提示處使用下列線上援助指令：

```
man syslog

man syslogd

man syslog.conf
```

syslog 訊息範例

以下是 `syslog` 訊息的範例。

時間標記 , 主機名稱 [instance_name], [子系統], [vsid], 訊息 ID, 日誌層級 , 訊息資料

```
Jul 19 14:33:18 strange /usr/lib/nfs/lockd[164]: [ID 599441
daemon.info] Number of servers not specified. Using default of 20.

Jul 19 14:33:20 strange ntpdate[181]: [ID 558275 daemon.notice]
adjust time server 192.18.56.149 offset 0.06702 6 sec

Jul 19 14:38:13 strange xntpd[248]: [ID 204180 daemon.info]
synchronisation lost
```

```
Jul 19 14:38:47 strange server1 appservd[374]: [ID 702911
daemon.info] INFO (  374): CORE1116: Sun ONE Application Server 7.0

Jul 19 14:38:48 strange server1 appservd[374]: [ID 702911
daemon.info] FINE (  374): Collecting statistics for up to 1
processes with 128 threads, 200 listen sockets, and 1000 virtual
servers
```

使用 Windows 事件日誌記錄

如需關於 Windows 作業環境使用的事件日誌機制的更多資訊，請參閱 Windows 輔助說明系統索引，查找關鍵字**事件記錄**。

使用日誌層級

本章節論述日誌層級以及如何爲每個 Sun ONE Application Server 子系統指定日誌層級。

描述下列主題：

- 關於日誌層級
- 用於 syslog 配置的日誌層級

關於日誌層級

Sun ONE Application Server 爲選擇性的資訊記錄使用標準的 JDK 1.4 日誌層級。除標準 JDK 日誌層級外，Sun ONE Application Server 還新增了一些日誌層級，用以更直觀地對映至 `server.log`，並更緊密地與 Solaris 整合。

當記錄的訊息路由至 `server.log` 時，它們也會依照第 90 頁的「對映至 server.log 的 Sun ONE Application Server 日誌層級」中所定義的那樣，對映至日誌層級。

注意　用於管理伺服器以及預設應用程式伺服器實例之 `server.log` 檔案 (或 `syslog`) 的預設日誌層級爲「資訊」。該預設日誌層級用於應用程式伺服器實例時，將記錄錯誤訊息和資訊訊息。若要避免記錄此類訊息，請在 `server.xml` 檔案內或在管理伺服器與伺服器實例的管理介面中將日誌層級變更爲「警告」或「伺服器」。

可於 log-service 元素中設定整個伺服器的預設日誌層級。這會影響將日誌層級設定爲 "default" 的所有元素。

您可以爲啓用記錄功能的每個 Sun ONE Application Server 子系統指定日誌層級。日誌層級對於合理化在執行期間記錄的訊息資訊數量很有用。可在預定子系統的 `server.xml` 檔案中指定層級。您可以藉由所選子系統的管理介面指定日誌層級，或者您可以直接編輯 `server.xml` 檔案，爲選取的子系統設定需要的日誌層級。

小心 手動編輯 `server.xml` 檔案可能會產生語法錯誤，從而導致伺服器啓動失敗。「*Sun ONE Application Server Administrator's Configuration File Reference*」的「Manually Editing Configuration Files」一節論述手動編輯配置檔案的指導原則。

透過管理介面設定日誌層級的範例在圖第 93 頁的「用於 JMS 服務的日誌層級」中展示。若要在 `server.xml` 檔案中直接設定每個子系統或元件的日誌層級，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

表第 89 頁的「日誌層級」中描述的日誌層級符合 JDK1.4 記錄 API 規格的需求。不過，日誌層級「警示」和「嚴重」僅特定用於 Sun ONE Application Server，不在 JDK1.4 記錄 API 中執行。

下表照嚴重性從低到高的次序，定義 Sun ONE Application Server 的日誌層級和訊息。左欄列示 Sun ONE Application Server 中指定的日誌層級，右欄提供每個日誌層級的簡要描述。

表格 5-1 日誌層級

日誌層級	描述
最精細 更精細 精細	這些訊息分別指示除錯訊息的冗長程度。「最精細」產生最大冗長度。
配置	訊息與各種靜態配置資訊有關，可協助對可能與特定配置關聯的問題進行除錯。
資訊	訊息實際上是有益的，通常與伺服器配置或伺服器狀況有關。這些訊息不指示需要立即採取動作的錯誤。 例如，可能記錄一條指示已收到配置變更通知的訊息；建立關於 MessageBroker 的新主題。
警告	訊息表示一條警告。該訊息可能伴有異常。
伺服器	訊息指示發生相當重要的事件，可能會防止應用程式正常執行。
警示 *	訊息警示使用者採取特定動作。
嚴重 *	訊息指示發生嚴重錯誤，發生錯誤後，建議不要執行伺服器。理論上來說，這將是伺服器當機前的最後訊息。

* 特定用於 Sun ONE Application Server 的日誌層級。

注意 日誌層級低於「資訊」(最精細、更精細、精細及配置) 的所有訊息都會提供可幫助解決有關除錯問題的資訊，並且您必須依照技術支援人員的建議啓用它們。日誌層級低於「資訊」的訊息一般不用本土化。

用於 syslog 配置的日誌層級

下表包含當使用 `syslog` 時可在 Sun ONE Application Server 內配置的日誌層級之清單。左欄列示 Sun ONE Application Server 中指定的日誌層級，右欄提供 `syslog` 設備內相應的日誌層級。

表格 5-2 對映至 server.log 的 Sun ONE Application Server 日誌層級

Sun ONE Application Server	syslog 層級
最精細	LOG_DEBUG
更精細	LOG_DEBUG
精細	LOG_DEBUG
配置	LOG_INFO
資訊 (預設)	LOG_INFO
警告	LOG_WARNING
伺服器	LOG_ERR
警示	LOG_ALERT
嚴重	LOG_CRIT

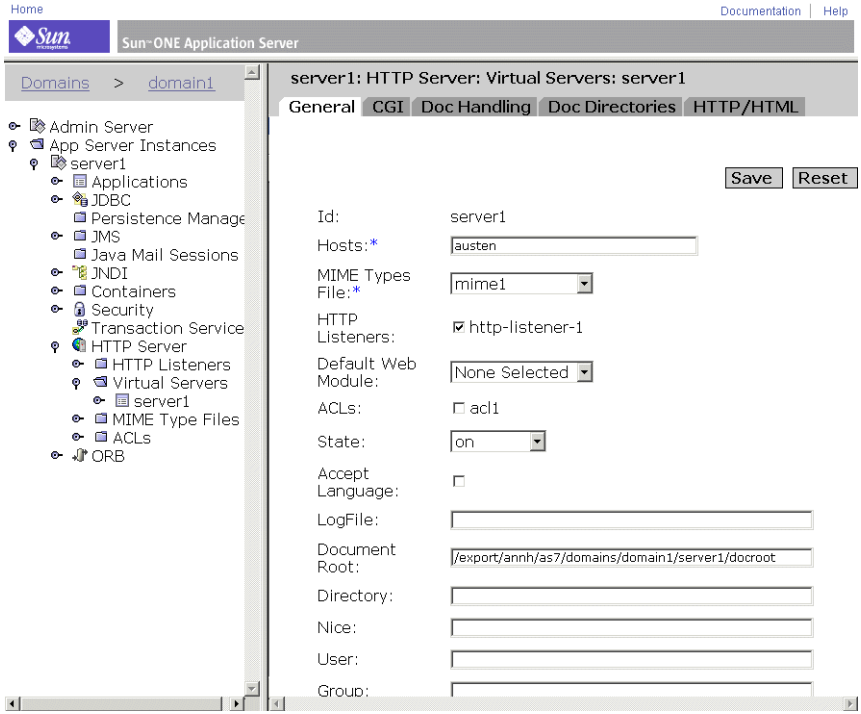
關於虛擬伺服器和記錄

Sun ONE Application Server 可以有虛擬伺服器實例。Sun ONE Application Server 實例內的每個虛擬伺服器都有自己的識別身份，還可能有自己的日誌檔。使用每個虛擬伺服器的個別日誌檔，可以幫助追蹤針對特定作業事件和資源的伺服器活動。

若要藉由管理介面指定虛擬伺服器的日誌檔名稱，請自目錄樹移往 [HTTP 伺服器] 連結，然後開啓虛擬伺服器資料夾下的伺服器實例元素，以便在右框中顯示 [一般] 標籤。您可以在 [日誌檔] 欄位中輸入該虛擬伺服器的日誌檔路徑和名稱。圖第 91 頁的「設定虛擬伺服器日誌檔名稱」顯示此設定的位置。

注意 啓用記錄功能且當應用程式正在執行時，無需虛擬伺服器 ID 便可記錄已記錄的應用程式訊息。

圖 5-1 設定虛擬伺服器日誌檔名稱



您也可以將記錄的訊息從多重虛擬伺服器導入到一個伺服器日誌檔中。若這樣做，您可能想在 `server.xml` 檔案的 `log service` 元素內啓用 `log-virtual-server-id`。這可幫助使用者識別來自不同虛擬伺服器的日誌訊息。

```
<log-service level="FINEST" log-stdout="false" log-stderr="false"
echo-log-messages-to-stderr="false" create-console="false"
log-virtual-server-id="true" use-system-logging="false">

</log-service>

    <http-listener>

        <virtual-server-class>

            <virtual-server id="server1"
http-listeners="http-listener-1" hosts="strange" mime="mime1"
state="on" accept-language="false"/>

            <virtual-server id="server2" hosts="strange"
mime="mime1"/>

        </virtual-server-class>
```

```
</http-listener>
```

在此範例中，<log-service log-virtual-server-id="true"> 用於在每條日誌訊息中納入 virtual_server_id。這可讓您鑑別來自不同虛擬伺服器的訊息。在 virtual-server 元素內缺失屬性 "log-file"，會導致所有虛擬伺服器將訊息記錄至單一檔案。

關於記錄程式

在子系統層級，可以有選擇地啓用或停用記錄功能。針對每個子系統的記錄控制方式在 server.xml 檔案內指定，如「*Sun ONE Application Server Configuration File Reference*」中所述。依照 JDK1.4 記錄 API 的需求，每個子系統都有自己的記錄程式。

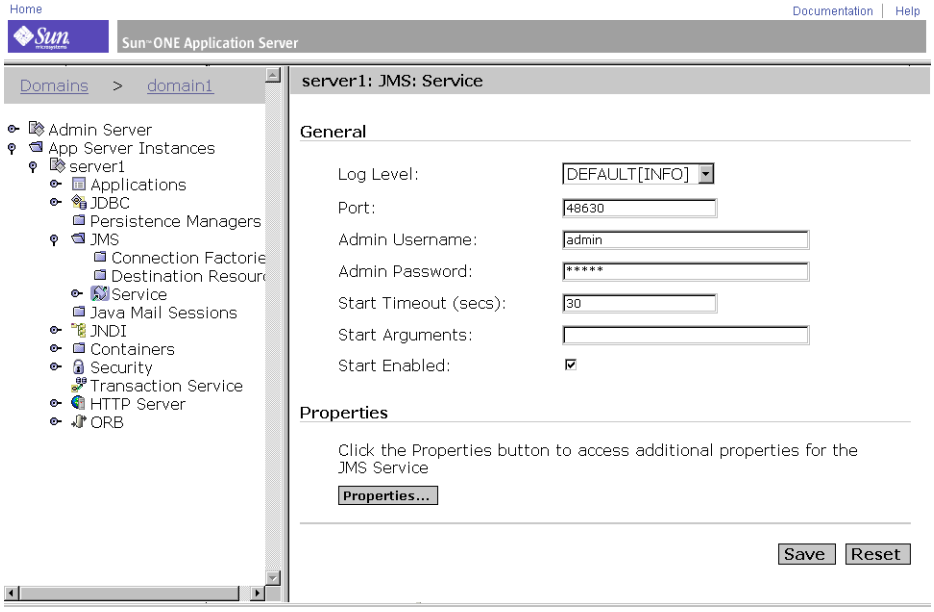
在下表中，左欄定義子系統，右欄展示每個子系統之 server.xml 檔案中的元素。

表格 5-3 Sun ONE Application Server 中的子系統與位置

子系統	元素
管理伺服器	<admin-service>
EJB 容器	<ejb-container>
Web 容器	<web-container>
MDB 容器	<mdb-container>
Sun ONE Message Queue (JMS 服務)	<jms-service>
安全性服務	<security-service>
Java Transaction Service (JTS)	<transaction-service>
物件請求仲裁介面 (ORB)	<iiop-service>
預設處理程式[1]	<log-service>

1. 預設處理程式指的是與所有 server.xml 項目 (不與特定子系統關聯，如公用程式類別) 相關聯的預設記錄程式。

圖 5-2 用於 JMS 服務的日誌層級



注意 在 Windows 平台上，如果您選擇將日誌發送至 Windows `server.log`，則僅會將層級爲「資訊」、「警告」、「伺服器」、「警示」或「嚴重」的訊息記錄至 Windows 事件日誌。

表格第 89 頁的「日誌層級」依嚴重性從低到高的次序，定義爲 Sun ONE Application Server 訊息提供的日誌層級。這些日誌層級符合 JDK1.4 記錄 API 規格的需求。另外，日誌層級 ALERT 和 FATAL 僅特定用於 Sun ONE Application Server，在 JDK1.4 記錄 API 中不受支援。

關於用戶端記錄

應用程式用戶端容器 (ACC) 有自己的日誌服務，僅能記錄至本機檔案。

ACC 通常在應用程式伺服器以外的其他主機上以自己的程序執行。它本身有自己的記錄基礎架構以及自己的日誌檔。ACC 配置保留在檔案 `sun-acc.xml` 中。

用於 ACC 的用戶端子系統記錄元素是 `log-service`。下表列示元素與屬性，每個項目都指明預設值和值範圍。

表格 5-4 ACC 記錄元素

元素	屬性	描述
`log-service`	file	若 ACC 日誌檔爲空或遺漏，將記錄至 `stdout`。
`log-service`	層級	ACC 日誌層級。

「*Sun ONE Configuration File Reference*」中提供 `sun-acc.xml` 檔案的範例。

重新導向應用程式與伺服器日誌輸出

對於開發人員而言，在應用程式元件與 J2EE 應用程式的單元測試期間，應用程式記錄和伺服器日誌必須隨時可用。在 Windows 平台上，開發人員更希望看到伺服器日誌訊息顯示在桌面上的指令視窗中。在 UNIX 平台上，許多開發人員樂意讓日誌訊息簡單地流向可啓動伺服器實例的終端機視窗中的 `stderr`，或者，使用指令尾 `-f` 查看日誌檔內寫入的日誌訊息。

`server.xml` 檔案包含可爲 `stdout` 與 `stderr` 設定的屬性，以便將記錄的訊息導向至日誌檔或終端機視窗等等。請參閱「*Sun ONE Application Server Configuration File Reference*」，以取得關於使用 `stdout` 與 `stderr` 的更多資訊。

請參閱第 102 頁的「配置日誌服務」以取得日誌服務資訊。

日誌檔管理

您可以設定您的存取和事件日誌 (`server.log`) 檔案，使其自動歸檔。有時，或者在指定的間隔時間後，您的日誌將旋轉。Sun ONE Application Server 會儲存舊日誌檔，並以含有儲存日期及時間的名稱標示已儲存的檔案。

注意 雖然您可以建立多個虛擬伺服器並爲每個虛擬伺服器關聯日誌檔，但無法支援個別虛擬伺服器的日誌旋轉設定。

例如，您可以設定您的存取日誌檔每小時旋轉一次，然後 Sun ONE Application Server 儲存並命名檔案「`access.199907152400`」，在該檔案中將日誌檔名稱、年、月、日，以及 24 小時時間一起連成一個單一字元字串。依據您設定的日誌旋轉類型的不同，日誌歸檔檔案的確切格式也會不同。

注意 這些工具主要用於非 Solaris 平台。

對於 Solaris 而言，依預設不會啓用上述工具，您必須使用本端 Solaris 作業系統日誌管理工具，如 Solaris 9 上的 `logadm`。在 Solaris 8 上，用於日誌管理的優先公用程式爲第 100 頁的「使用 Solaris cron 公用程式排程 logadm 的執行」中描述的 `cron` 工具。

依據作業系統的不同，您可以用四種不同的方法執行日誌旋轉。下面一節中論述相關內容。主題包括：

對於 UNIX 和 Windows ：

- 內部常駐程式日誌旋轉
- 基於排程程式的日誌旋轉

對於 Solaris 9

- 使用 Solaris `logadm` 公用程式。如需更多資訊，請參閱第 97 頁的「使用 Solaris logadm 公用程式的旋轉」。

對於 Solaris (所有版本)

- 使用 Solaris `cron` 公用程式。如需更多資訊，請參閱第 99 頁的「使用 Solaris「cron」公用程式的旋轉」。

內部常駐程式日誌旋轉

內部常駐程式日誌旋轉可用於 UNIX 和 Windows 兩種作業系統。內部常駐程式日誌旋轉發生於 HTTP 常駐程式內，且僅能在伺服器實例啓動之時配置。使用此方法旋轉的日誌將以下列格式儲存：

```
access.<YYYY><MM><DD><HHMM>
```

```
error.<YYYY><MM><DD><HHMM>
```

您可以指定作爲旋轉日誌檔並啓動新日誌檔基礎的時間。例如，如果旋轉開始時間是 12:00 a.m.，旋轉間隔時間爲 1440 分鐘 (一日)，則不管目前時間爲何，儲存完畢後均會立即建立新的日誌檔，並隨之收集旋轉開始時間之前的資訊。日誌檔每天在 12:00 a.m. 旋轉，存取日誌在 12:00 a.m. 標示，並另存爲 `access.199907152400`。同樣，如果您將間隔時間設定爲 240 分鐘 (四小時)，四小時的間隔時間開始於 12:00 a.m.，這樣存取日誌檔將包含從 12:00 a.m. 到 4:00 a.m.、從 4:00 a.m. 到 8:00 a.m. 收集的資訊，以此類推。

如果啓用了日誌旋轉，日誌檔旋轉將於伺服器啓動之時開始。要旋轉的第一個日誌檔將從目前時間到下一個旋轉時間收集資訊。使用上述範例，如果您將開始時間設定在 12:00 a.m.，將旋轉間隔時間設定爲 240 分鐘，目前時間爲 6:00 a.m.，則要旋轉的第一個日誌檔將包含從 6:00 a.m. 到 8:00 a.m. 收集的資訊，下一個日誌檔將包含從 8:00 a.m. 到 12:00 p.m. (正午) 的資訊，以此類推。

基於排程程式的日誌旋轉

排程程式日誌旋轉，可讓您立即歸檔日誌檔或在特定日期的特定時間使用伺服器來歸檔日誌檔。若要立即歸檔日誌檔，請從管理介面的左窗格中選取 [Admin Server]。然後，按一下位於右頁面頂端的 [Logging] 連結。接著，按一下 [Log Rotation]。最後，按一下 [Archive]。

使用排程程式方法旋轉的日誌將另存爲原始檔名，後隨旋轉該檔案的日期與時間。例如，若在 4:30 p.m. 旋轉 `access`，其可能成爲 `access.24Apr-0430PM`。

伺服器啓動時，將初始化日誌旋轉。如果開啓旋轉功能，Sun ONE Application Server 會建立有時間戳記的存取日誌檔，並且旋轉將在伺服器啓動之時開始。

開始旋轉後，若發生需要記錄到存取日誌檔或錯誤日誌檔的請求或錯誤，並且其發生在預先排程的「下一個旋轉時間」之後，則 Sun ONE Application Server 會建立新的時間戳記日誌檔。

注意　對於 Windows 平台，以及對於導向至一個檔案 (非 Solaris 上的 `syslog`) 的伺服器記錄，您必須歸檔伺服器日誌。

若要歸檔日誌檔並指定使用 schedulerd 控制方法，請從管理介面左窗格中選取 [Admin Server]。然後，按一下位於右頁面頂端的 [Logging] 連結。接著，按一下 [Scheduler based Log Rotation] 方塊。最後，按一下 [OK]。將指示 scheduler 的目前狀態。

使用 Solaris logadm 公用程式的旋轉

Solaris 9 作業系統包含可用於執行含有記錄訊息之功能陣列的公用程式 logadm。

尤其對於 Sun ONE Application Server，藉由 Solaris cron 公用程式執行該公用程式時，它可用於執行日誌旋轉工作，如第 100 頁的「使用 Solaris cron 公用程式排程 logadm 的執行」中所述。

您可以指定下列關於日誌檔的日誌旋轉詳細資訊：

- 必須旋轉的系統上之所有日誌檔名稱
- 旋轉間隔時間
- 將觸發旋轉的條件
- 要儲存的備份日誌檔數目
- 要儲存的備份日誌檔之命名慣例

上述詳細資訊在檔案 logadm.conf 中指定，該檔案位於：

```
n /etc/logadm.conf
```

logadm.conf 範例檔案如下：

```
# 版權所有 2001-2002 Sun Microsystems, Inc.。保留所有權利。
# 應依照授權條款使用。
#
# ident "@(#)logadm.conf 1.2 02/02/13 SMI"
#
# logadm.conf
#
# 系統日誌檔管理的預設設定。
# logadm(1M) 的 -w 選項是寫入至該檔案的優先方法，
# 但如果您仍然手動編輯該檔案，請使用「logadm -V」檢查錯誤。
#
```

```
# 該檔案中的行格式爲：
# < 日誌名稱 > <; 選項 >
# 對於此處列示的每個日誌名稱，均提供 logadm 的預設選項。
# logadm 指令行上提供的選項會置換該檔案中的預設選項。
# # logadm 通常藉由 root 使用者 crontab ( 參閱 crontab(1)) 中
# 的項目，於每天上午提早執行。
#
/var/log/syslog -C 8 -P 'Tue Jul 9 10:10:00 2002' -a 'kill -HUP `cat
/var/run/syslog.pid`' /var/adm/messages -C 4 -P 'Tue Jul 30 10:10:00
2002' -a
'kill -HUP `cat /var/run/syslog.pid`' /var/cron/log -c -s 512k -t
/var/cron/olog
/var/lp/logs/lpsched -C 2 -N -t '$file.$N'
#
# 下面的項目由 turnacct(1M) 使用
#
/var/adm/pacct -C 0 -N -a '/usr/lib/acct/accton pacct' -g adm -m 664
-o adm -p never
#
# 倘若 SUN One 應用程式伺服器預設日誌檔的目前大小爲 >= 512k，
# 下面的項目將每天旋轉該日誌檔。歸檔舊日誌檔之前，該項目會壓縮舊日誌檔，並在
# 30 天後刪除舊的檔案。壓縮藉由 gzip(1) 完成，壓縮後的日誌檔將含有字尾 .gz。
/var/appserver/domains/domain1/server1/logs/server.log -A 30d -s
512k -p 1d -z
```

或者，您可以透過互動式呼叫 logadm 指令，在特定檔案上啓動日誌旋轉。

下列範例會旋轉 syslog，並保留八個日誌檔。舊日誌檔位於目錄 /var/oldlogs 之下 (而非目錄 /var/log) ：

```
% logadm -C8 -t'/var/oldlogs/syslog.$n' /var/log/syslog
```

您也可以使用互動式指令行選項，在 /etc/logadm.conf 中指定的檔案上呼叫旋轉功能，但要使用不同的或修改後的選項。

如果在 /etc/logadm.conf 中及在指令行上都指定了選項，則首先套用 /etc/logadm.conf 檔案中的選項。因此，指令行選項將置換 /etc/logadm.conf 中的選項。以下爲該情況的範例：

```
% logadm /var/appserver/domains/domain1/server1/logs/server.log -p
now
```

透過以上指令，便可使用 /etc/logadm.conf. 中爲指定的檔案提供的所有選項，旋轉該檔案。

注意 若一次指定多個選項，在選項之間便會有一個隱含的 AND。這意味著必須符合所有條件，才能旋轉日誌。

如需關於 logadm 公用程式及其選項的詳細資訊，請參考其線上援助頁，如下所述：

```
% man logadm
```

或

```
% logadm -h
```

使用 Solaris 「cron」公用程式的旋轉

在 Solaris 8 上，cron 公用程式可用於執行應用程式伺服器日誌旋轉。可以使用以下指令完成此動作：

```
% crontab -e
```

該指令會啓動您最喜好的編輯程式 (由 env. 變數 $EDITOR 定義)，從而您可以提供 cron 項目清單。

注意 您結束編輯程式時，該指令也會呼叫 /etc/cron.d/logchecker 程序檔。此程序檔將爲 cron 常駐程式提供 changed/new crontab 項目。因此，cron 常駐程式將立即接受以此方式加入的項目，並且日誌旋轉立即開始。

您無需重新啓動 cron 常駐程式，便可啓用日誌旋轉。

本章節包括下列主題：

- 關於 crontab 項目格式
- 使用 Solaris cron 公用程式排程 logadm 的執行

關於 crontab 項目格式

在 crontab 檔案中，每行有六個欄位。這些欄位以空格或標籤分隔。前五個欄位爲整數型樣，指定以下內容：

- 分鐘 (0-59)
- 小時 (0-23),
- 每月日期 (1-31),
- 每年月份 (1-12),
- 一週天數 (0-6, 0=Sunday),

使用此格式，您便可指定要在日/週/月之指定時間旋轉的存取檔案和事件日誌檔，並排程以重複旋轉。例如，

0 0 * * 1-5

```
/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelogs
```

0 12 * * 1-5

```
/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelogs
```

0 * * * 1-5

```
/opt/SUNWappserver7/appserver/domains/domain1/mainserver/bin/rotate
logs
```

從週一至週五，在每天的午夜和中午，這將旋轉 server1 的存取檔案和日誌檔；同時從週一至週五，每天每小時還會旋轉主伺服器的存取檔案和日誌檔一次。

crontab 檔案儲存於 /var/spool/cron/crontabs/ 之下。您可以以終端使用者或 root 使用者身份，建立 crontab 檔案。依據您的權限，使用下列指令，您會看到 crontab 項目：

```
% crontab -l username
```

使用 Solaris cron 公用程式排程 logadm 的執行

cron 指令會啓動於指定日期和時間執行指令的過程。可以依據目錄 /var/spool/cron/crontabs 下 crontab 檔案中的說明，指定正規排程的指令。

作爲與 cron 配合使用的正規排程指令的範例，crontab 中的下列項目將在每天午夜啓動 logadm。

```
0 0 * * 0-6 logadm
```

請注意，使用者可以使用 crontab(1) 指令提交自己的 crontab 檔案。

若要保留以 cron 執行的所有動作之日誌，必須在 /etc/default/cron 檔案中指定 CRONLOG=YES (依預設)。/etc/cron.d/logchecker 是用於檢查日誌檔是否超出系統 ulimit 的程序檔。如果超出，日誌檔將移至 /var/cron/olog 下。

透過指令行介面配置記錄

對於伺服器實例及虛擬伺服器實例，您可以藉由指令行來配置記錄服務的各個方面。

注意 本章節的所有指令範例均假定已設定環境變數。

若要取得伺服器實例的所有 log-service 屬性：

```
asadmin> get instance_name.log-service.*
```

在表第 104 頁的 「日誌服務屬性」中也定義了 log-service 屬性。

將該指令與指定的伺服器實例名稱配合使用的範例如下：

```
asadmin> get server1.log-service.*
```

將傳回用於 server1 實例之記錄服務的屬性清單。使用 set 指令，可以配置列示的每個屬性。

若要爲虛擬伺服器實例啓用虛擬伺服器 ID 記錄功能，請在終端機提示處輸入下列指令：

```
asadmin> get instance_name.LogVirtualServerId
```

將傳回 LogVirtualServerId 的目前狀態。如果狀態爲 false，您可以藉由 set 指令 (如下所述) 啓用它：

```
asadmin> set instance_name.LogVirtualServerId=true
```

若要設定虛擬伺服器實例的日誌檔名稱，請使用 set 指令，如下所述：

```
asadmin> set instance_name.virtual-server.<virtual server
id>.logFile=<log file>
```

下面發佈的設定日誌檔指令用作範例：

```
asadmin> set
instance2.virtual-server.instance2.logFile=/space/IAs7se/appserver7/appse
rv/domains/domain1/instance2/logs/log
```

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。

如需關於使用 asadmin 的更多資訊，請參閱附錄 A 「使用指令行介面」。

透過管理介面配置記錄

本章節描述可以透過 Sun ONE Application Server 管理介面執行的工作，以爲 server-wide (全域) 元素、指令和應用程式元件配置可用的記錄服務選項。

本章節包括下列主題：

- 配置日誌服務
- 爲應用程式伺服器元件和子系統配置記錄
- 配置錯誤記錄的指令

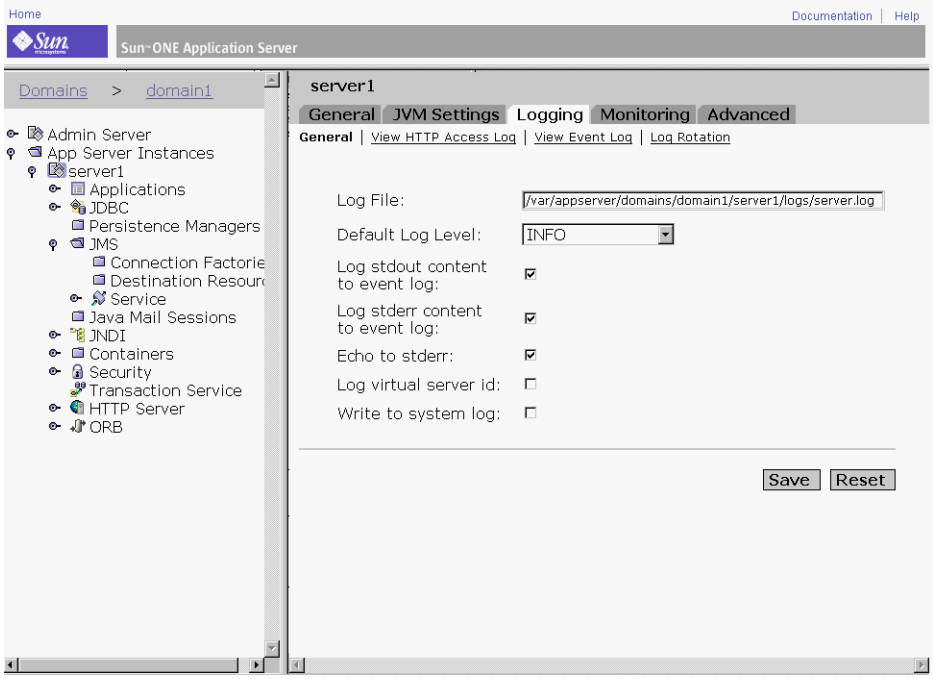
配置日誌服務

日誌服務是 `server.xml` 檔案中 J2EE 服務元素種類內的元素，如「*Sun ONE Application ServerConfiguration File Reference*」所述。日誌服務用於配置包含下列日誌檔的系統記錄服務：

- 伺服器日誌
- 存取日誌
- 作業事件日誌
- 虛擬伺服器日誌

系統記錄服務的配置包括指定日誌服務元素的屬性值。

圖 5-3 服務實例的日誌服務管理



可以透過管理介面，配置日誌服務元素的下列屬性，如圖第 103 頁的「服務實例的日誌服務管理」所示。

- 日誌檔
- 預設日誌層級
- 將標準輸出內容記錄到事件日誌中
- 將標準錯誤內容記錄到事件日誌中
- 回應標準錯誤
- 建立主控台
- 記錄虛擬伺服器 ID
- 寫入至系統日誌

在管理介面的左窗格內，從伺服器實例的擴充樹階層中可以存取 [Log Service] 連結。下表描述可以配置的每個屬性，以及允許的預設值和值範圍。

表格 5-5 日誌服務屬性

屬性	預設值	描述
file	server.log[1]	(選擇性的) 置換伺服器日誌的名稱或位置。用來執行伺服器的任何使用者帳號均必須能夠寫入保留伺服器日誌的檔案和目錄。
層級	資訊	(選擇性的) 控制由其他元素記錄到伺服器日誌中的訊息預設類型。允許值依從最高到最低的次序，列示如下：最精細、更精細、精細、配置、資訊、警告、伺服器、警示、嚴重。 每個值都記錄用於所有較低值的所有訊息，例如，「最精細」記錄所有訊息，「嚴重」僅記錄「嚴重」訊息。預設值爲「資訊」，其記錄所有「資訊」、「警告」、「伺服器」、「警示」與「嚴重」訊息。
log-stdout	True	(選擇性的) 如果爲 true，則將 stdout 輸出重新導向至伺服器日誌。合法值爲 on、off、yes、no、1、0、true、false。
log-stderr	True	(選擇性的) 如果爲 true，則將 stderr 輸出重新導向至伺服器日誌。合法值爲 on、off、yes、no、1、0、true、false。
echo-log-messages-to-stderr	True	(選擇性的) 如果爲 true，則將日誌訊息發送至伺服器日誌以及 stderr。合法值爲 on、off、yes、no、1、0、true、false。
create-console	False	(選擇性的) 如果爲 true，將在 Windows 作業系統上建立主控台視窗以輸出 stderr。合法值爲 on、off、yes、no、1、0、true、false。
log-virtual-server-id	False	(選擇性的) 如果爲 true，虛擬伺服器 ID 將顯示在虛擬伺服器日誌中。如果多個 virtual-server 元素共用同一個日誌檔，這些功能將很有用。
use-system-log	False	如果爲 true，將使用 UNIX syslog 服務或 Windows 事件記錄功能來產生和管理日誌。

1. 在伺服器元素之 log-root 屬性指定的目錄中。

為應用程式伺服器元件和子系統配置記錄

本章節描述如何爲 Sun ONE Application Server 元件和子系統啓用記錄及選取日誌層級。請注意，Java Transaction Service (JTS) 元件有多個日誌檔。由於多數元件和子系統在配置日誌層級時所採用的處理方法相同，所以選取日誌層級的程序僅一次證明，適用於指明的元件和子系統群組過程。

下列元件和子系統可以利用伺服器訊息的選擇記錄功能。您可以藉由對本指南其他主題的參考 (根據指示) 來瞭解元件和子系統。

- ORB - 配置對基於 Corba 用戶端的支援
- Web 容器 - 配置 J2EE 服務
- EJB 容器 - 配置 J2EE 服務
- MDB 容器 - 配置 J2EE 服務 (在 EJB 容器內)
- Java 作業事件服務 - 配置 J2EE 服務
- JMS 服務 - Java Message Service
- 虛擬伺服器 - 使用虛擬伺服器

指定日誌層級的步驟

若要爲 ORB、Web 容器、EJB 容器、MDB 容器 (在 EJB 容器內)、Java 作業事件服務以及 JMS 服務指定日誌層級，請執行以下程序：

1. 在管理介面的左窗格內，展開 Sun ONE Application Server 實例，以顯示您要編輯的元件和子系統。
2. 按一下所需元件或子系統的連結。
3. 在管理介面的右頁面內，從 [Log Level] 下拉式清單中選取下列日誌層級參數之一。在第 88 頁的「關於日誌層級」中描述了日誌層級。

指定日誌檔的步驟：(虛擬伺服器)

若要指定日誌檔，請執行以下程序：

1. 在管理介面的左窗格內，展開 Sun ONE Application Server 實例，以顯示 HTTP 伺服器子系統。
2. 按一下 [HTTP Server] 連結。
3. 按一下 [Virtual Server] 連結。
4. 按一下所需伺服器實例連結。

5. 在管理介面右頁面內，於 [General] 標籤下的 [Log File] 欄位中，輸入所需目錄路徑和檔名。

指定作業事件日誌位置的步驟： (Java Transaction Service)

若要指定作業事件日誌位置，請執行以下程序：

1. 在管理介面的左窗格內，展開 Sun ONE Application Server 實例，以顯示作業事件服務子系統。
2. 按一下 [Transaction Service] 連結。
3. 在管理介面右頁面內，於 [Advanced] 欄位群組下的 [Transaction Log Location] 欄位中，輸入所需目錄路徑和檔名。

配置錯誤記錄的指令

Sun ONE Application Server 包括用於 `init.conf` 檔案的錯誤記錄指令。包括下列指令：

- **錯誤記錄日期格式**。`ErrorLogDateFormat` 指令指定伺服器日誌使用的日期格式。
- **日誌清除間隔時間**。`LogFlushInterval` 可決定最大的時間間隔 (以秒計算)，在此之前，將從記憶體中清除存取日誌並放入 `access.log` 檔案中。
- **Pid 日誌**。`PidLog` 可指定用於記錄基本伺服器程序之程序 ID (pid) 的檔案。某些伺服器支援程式假定此日誌位於伺服器根下的 `logs/pid` 中。

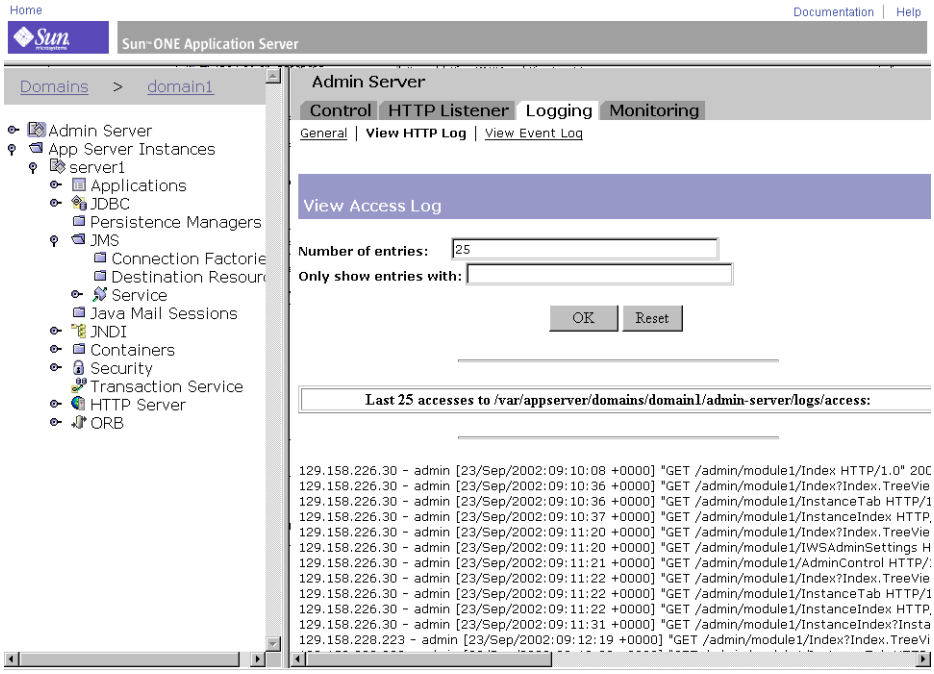
「*Sun ONE Application Server Configuration File Reference*」中詳細描述針對 `init.conf` 的所有指令。

檢視存取日誌檔

您可以檢視管理伺服器的 http 日誌檔以及 Sun ONE Application Server 實例的 http 日誌檔。

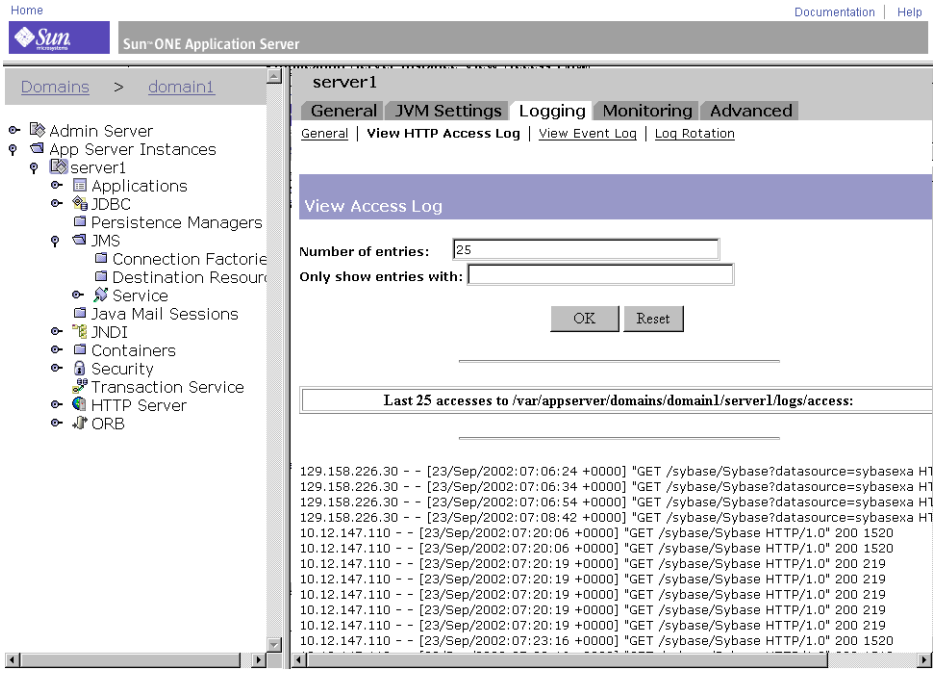
若要檢視管理伺服器的 http 日誌，請從管理介面的左窗格中選取 [Admin Server]，然後從右頁面中選擇 [Logging] 標籤。[View HTTP Access Log] 連結即會顯示。選取此連結以檢視已配置的存取日誌。圖第 107 頁的「管理伺服器檢視 HTTP 存取日誌」中展示已顯示日誌的範例。

圖 5-4 管理伺服器檢視 HTTP 存取日誌



若要檢視應用程式伺服器實例的存取日誌，請在管理介面左窗格中按一下所需的伺服器實例。按一下右窗格中的 [Logging] 標籤。按一下 [View Access Log] 連結，以顯示爲該伺服器實例配置的作用中存取日誌。圖第 108 頁的「應用程式伺服器實例檢視存取日誌」中顯示範例。

圖 5-5 應用程式伺服器實例檢視存取日誌

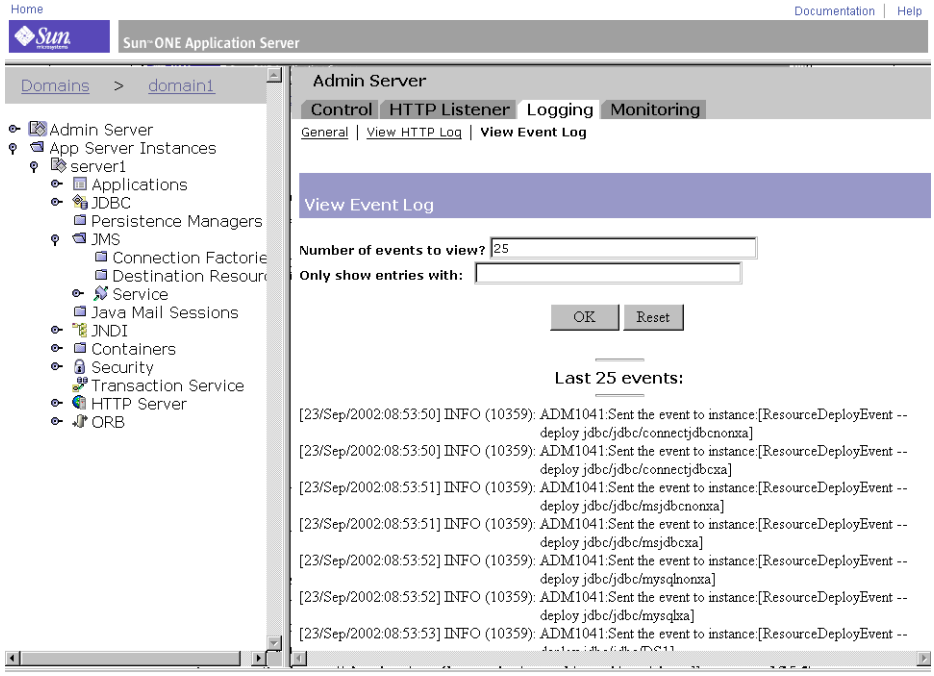


檢視事件日誌檔

您可以檢視管理伺服器的作用中事件日誌檔，以及 Sun ONE Application Server 實例的作用中事件日誌檔。

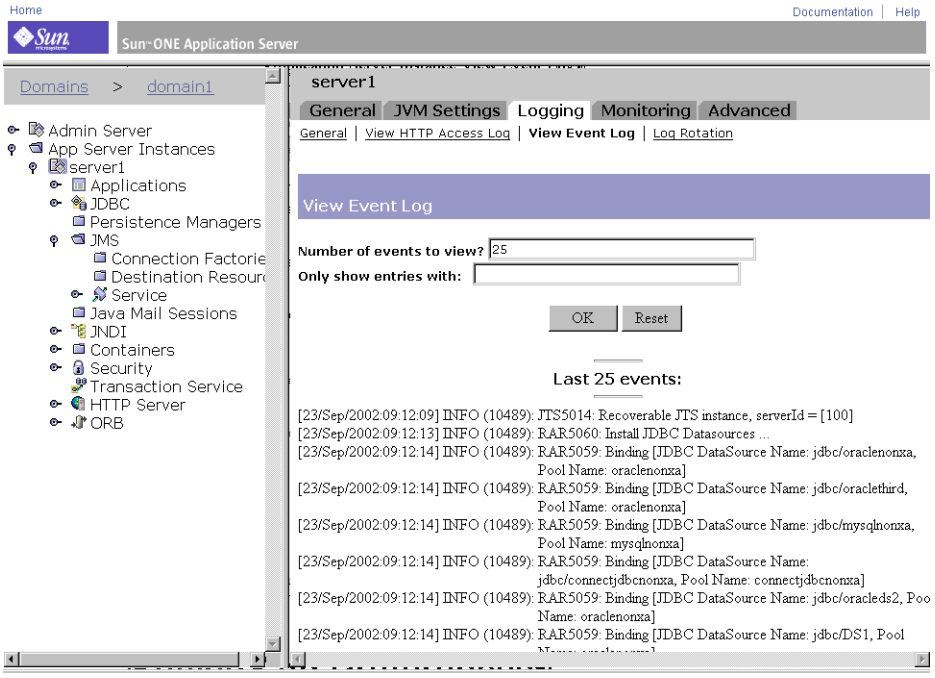
若要檢視管理伺服器的事件日誌，請從左窗格中選取 [Admin Server]，然後從右頁面中選擇 [Logging] 標籤。[View Event Log] 連結將顯示。選取此連結以檢視已配置的事件日誌。圖第 109 頁的「管理伺服器檢視事件日誌」中展示已顯示日誌的範例。

圖 5-6 管理伺服器檢視事件日誌



若要檢視應用程式伺服器實例的事件日誌，請在管理介面左窗格中按一下所需的伺服器實例，然後從右窗格中選擇 [Logging] 標籤。[View Event Log] 連結將顯示。選取此連結以檢視已配置的事件日誌。圖第 110 頁的「應用程式伺服器實例檢視事件日誌」中展示已顯示日誌的範例。

圖 5-7 應用程式伺服器實例檢視事件日誌



設定日誌偏好設定

在安裝期間，將爲伺服器建立名爲 `access` 的存取日誌檔。您可以透過指定是否記錄存取、記錄使用什麼格式，以及當用戶端存取資源時伺服器是否應花時間查找用戶端領域名稱，自訂針對任何資源的存取記錄。

若要對多個虛擬伺服器使用一個日誌檔，應該在事件日誌的 `server.xml` 檔案中開啓 `LogVsId`。請參閱「*Sun ONE Application Server Configuration File Reference*」以取得詳細資訊。或者，可在管理介面的 [Admin Server Logging] 標籤中開啓 `LogVsID`。

執行以下步驟，藉由管理介面啓用日誌虛擬伺服器 ID。重新啓動管理伺服器後，變更將生效。

1. 在管理介面左窗格中按一下 [Admin Server]。
2. 在右頁面中按一下 [Logging] 標籤。
3. 在日誌虛擬伺服器 ID 的核取方塊內按一下。

4. 按一下 [Save] 按鈕，將變更套用於 Sun ONE Application Server。

該設定需要重新啓動 Sun ONE Application Server，變更才能生效。

執行日誌分析程式

`flexanlg` 是用於日誌檔報告的日誌分析程式工具。僅當記錄導入至檔案 (非 `syslog`) 時，才能使用日誌分析程式。

使用日誌分析程式可以產生關於預設伺服器的統計資料，例如活動摘要、最常存取的 URL、一日內反復存取伺服器的次數，等等。日誌分析程式能產生預設伺服器的統計資料，但無法產生虛擬伺服器的統計資料。不過，如第 106 頁的「檢視存取日誌檔」所述，可以檢視每個虛擬伺服器的統計資料。

注意 執行日誌分析程式之前，您必須旋轉伺服器日誌。如需更多資訊，請參閱第 95 頁的「日誌檔管理」。

您可以透過執行位於目錄 *install_dir*`/bin/flexanlg` 下的工具 `flexanlg`，從指令行執行日誌分析程式。

若要執行 `flexanlg`，請於指令提示處鍵入下列指令與選項：

```
flexanlg [ -P ] [-n name] [-x] [-r] [-p order] [-i file]* [ -m
metafile ]* [ o file][ c opts] [-t opts] [-l opts] [-h help]
```

指令選項 (有 * 標記的選項可以重複)。

-i *filename*

輸入日誌檔

`-P`

代理日誌格式

-n *servername*

伺服器名稱

`-x`

以 `HTML` 輸出

`-r`

將 `IP` 位址解析爲主機名稱

`-p [ c, t, l ]`

輸出次序，預設次序爲計數、時間統計資料、清單

`-m filename`

中繼檔案

-o filename

輸出日誌檔；預設檔案爲 stdout

`-c [ h, n, r, f, e, u, o, k, c, z ]`

對這些項目計數；預設爲：`h, n, r, e , u, o, k, c`

- `h`: 全部命中
- `n`: 304 [未修改] 狀況碼（使用本機複本）
- `r`: 302 [找到] 狀況碼（重新導向）
- `f`: 404 [未找到] 狀況碼（未找到文件）
- `e`: 500 [伺服器錯誤] 狀況碼（配置不當）
- `u`: 全部的唯一 URL
- `o`: 全部的唯一主機
- `k`: 傳輸的全部千位元組
- `c`: 快取記憶體儲存的全部千位元組
- `z`: 不要對任何項目計數

`-t [ sx, mx, hx, xx, z ]`

尋找一般統計資料；預設爲：`s5m5h24x10`

- `s`（數目）: 尋找日誌的最高（數目）秒數
- `m`（數目）: 尋找日誌的最高（數目）分鐘數
- `h`（數目）: 尋找日誌的最高（數目）小時數
- `u`（數目）: 尋找日誌的最高（數目）使用者人數
- `a`（數目）: 尋找日誌的最高（數目）使用者代理程式數
- `r`（數目）: 尋找日誌的最高（數目）參考者數
- `x`（數目）: 尋找雜項關鍵字的最高（數目）
- `z`: 請勿尋找任何一般統計資料

`-l [ cx, hx ]`

製作指定子選項的清單；預設爲： c+3h5

c (x, +x): 最常存取的 URL

x: 僅列示 x 項目

+x: 如果存取次數多於 x 次，則僅列示

h (x, +x): 最經常存取您伺服器的主機或 IP 位址

x: 僅列示 x 項目

+x: 如果存取次數多於 x 次，則僅列示

z: 請勿製作任何清單

EXAMPLE: 使用 flexanlg 指令

flexanlg -i /var/opt/SUNQappserver7/domains/domain1/server1/logs/access

注意 執行日誌分析程式之前，應歸檔伺服器日誌。

檢視事件 (Windows 2000 Pro)

除了將錯誤記錄到 server.log 檔案中，Sun ONE Application Server 還會把嚴重的系統錯誤記錄到事件檢視器內。事件檢視器可讓您監視系統上的事件。使用事件檢視器，可查看開啓錯誤日誌之前可能發生的基礎配置問題所導致的錯誤。

若要使用事件檢視器，請執行下列步驟：

1. 從 [開始] 功能表中，選取 [程式集]，然後選取 [管理工具]。在 [管理工具] 程式群組中選擇 [事件檢視器]。
2. 從 [日誌] 功能表中選擇 [應用程式]。

   [應用程式] 日誌將顯示於事件檢視器中。來自 Sun ONE Application Server 的錯誤有來源標籤 https-*serverid*。
3. 從 [檢視] 功能表中選擇 [尋找]，在日誌中搜尋其中一個標籤。從 [檢視] 功能表中選擇 [重新顯示]，查看更新後的日誌項目。

如需關於事件檢視器的更多資訊，請參考您的系統說明文件。

第 6 章

監視 Sun ONE Application Server

本章包含有關監視、簡單網路管理協定 (SNMP) 功能以及 Sun ONE Application Server 功能的資訊。

本章包含下列小節：

- 關於監視 Sun ONE Application Server
- 使用 CLI 擷取監視資料
- 使用 CLI 管理作業事件服務
- 使用 HTTP 服務品質
- 關於 SNMP
- 設定 SNMP
- 啓用與啓動 SNMP 主代理程式

關於監視 Sun ONE Application Server

您可以從系統的策略資料點中收集狀態統計資料來監視 Sun ONE Application Server。統計資料會顯示伺服器所處理的請求數，以及對這些請求的處理程度。您可以針對個別虛擬伺服器檢視某些統計資料，並針對整個應用程式伺服器實例檢視另一些統計資料。可以使用 `asadmin` 公用程式或 SNMP 監視 Sun ONE Application Server。

本章節論述以下主題：

- 統計資料
- SNMP
- 監視 HTTP 伺服器

- 監視應用程式元件與子系統
- 服務品質 (QOS)

統計資料

統計資料集合總處於啓動狀態，以用於多數 Sun ONE Application Server 應用程式元件和子系統，包括 HTTP 伺服器；因此，無需啓動任何功能。但是，某些統計資料僅當該子系統上的監視處於明確的啓用狀態，或僅當啓用了相關的功能時才會被收集。這些統計資料包括下列資料點：

- EJB 方法的統計資料
- 作用中的作業事件
- 連線 (只有啓用了服務品質才可用)
- DNS (只有啓用了 DNS 快取記憶體才可用)

可以經由管理介面爲應用程式子系統或元件啓用監視，如第 117 頁的「監視應用程式元件與子系統」中所述。

如果伺服器監視器報告伺服器正在處理大量的請求，可能需要您調整伺服器配置或系統的網路核心。如需有關調整伺服器配置的更多資訊，請參閱「*Sun ONE Application Server Performance Tuning and Sizing Guide*」。

SNMP

透過使用簡單網路管理協定 (SNMP) (該協定用於在網路之間交換管理與監視資訊) 的資訊收集工具，Sun ONE Application Server 可以提供網路管理資訊。透過使用 SNMP，名爲 *agents* 的程式可以監視網路上的各種裝置 (如集線器、路由器、橋接器等等)。另一個程式從 agents 處收集資料。由監視作業建立的資料庫被稱爲**管理資訊庫** (MIB)。該資料用於檢查網路上的所有裝置是否都在正確地作業。

雖然使用 SNMP 僅可以監視 HTTP 伺服器；但使用指令行介面 (CLI) 卻可以監視所有的元件和系統

如需有關 SNMP 的更多資訊，請參閱第 144 頁的「關於 SNMP」與第 151 頁的「設定 SNMP」。

監視 HTTP 伺服器

依預設，HTTP 伺服器監視總處於啓動狀態，這就意味著無需特意開啓監視。HTTP 伺服器監視以 XML 檔案爲基礎，可以使用 `asadmin` 指令 (作爲三個可監視屬性的集合) 對其進行存取。第 131 頁的 「可監視的 HTTP 伺服器元素」與第 132 頁的 「可監視的 HTTP 伺服器屬性」中描述此 XML 檔案的元素、子元素以及屬性。

注意 使用 SNMP 僅可以取得 HTTP 伺服器統計資料。所有 Sun ONE Application Server 子系統的統計資料 (包括 HTTP 伺服器)，均可以使用指令行介面取得。

如需關於使用 `asadmin` 的更多資訊，請參閱第 377 頁的 「使用指令行介面」。

監視應用程式元件與子系統

對於 Sun ONE Application Server 中的某些子系統或元件，無需啓動監視，因爲總可以收集到相關的統計資料。例如，可以啓用或停用對應用程式元件 (如容器) 的監視。啓用監視後，除了總可以收集到的統計資料之外，還可以收集到有關所有 EJB 方法的附加統計資料。對 JDBC 連線區的監視總處於啓用狀態。首次存取連線區時便對其進行了初始化，在此之後，可以在任何時候監視相關的統計資料。

如需可監視資料點的完整清單，請參閱第 126 頁的 「可監視屬性名稱」。

您可以經由管理介面或指令行介面 (CLI)，啓用對所選應用程式元件和子系統的監視。例如，若要經由 EJB 容器的 CLI 啓用監視，請在終端機視窗中鍵入下列指令：

```
set server1.ejb-container.monitoringEnabled=true
reconfig server1
```

其中，`server1` 爲實例名稱。

在 [Containers] 節點下的管理介面上可以存取對等的功能。

本章節論述以下主題：

- 監視容器子系統
- 監視 ORB 服務
- 監視作業事件服務

監視容器子系統

至於 EJB 容器，啓用了監視之後，便可以收集與所有實體 Bean、有狀態階段作業 Bean 以及無狀態階段作業 Bean 方法有關的統計資料。這些統計資料包括：

- 錯誤總數
- 呼叫總數
- 成功總數
- 執行時間，以毫秒表示 (針對上一次的方法調用)

總能收集容器子系統的所有其他統計資料。某些受監視資料點包括的統計資料爲：

- 儲存區中初始、最少、最多的無狀態 Bean
- 快取記憶體中最少與適合的無狀態 Bean 數目和實體 Bean 數目
- 快取記憶體中最少與適合的無狀態階段作業 Bean 數目
- 建立與銷毀的 Bean 的數目
- 其他相關統計資料

監視 ORB 服務

對於 ORB 服務，受監視資料點包括爲 ORB 連線與 ORB 執行緒儲存區收集的統計資料。總可以收集到 ORB 統計資料，因此，無需啓用對 ORB 服務的監視。

監視作業事件服務

對於 Java Transaction Service (JTS) 服務，受監視資料點包括：

- 已完成的作業事件總數
- 已回轉的作業事件總數
- 執行中的作業事件總數
- 執行中的作業事件清單

請參閱第 138 頁的「使用 CLI 管理作業事件服務」，以取得進一步資訊。

服務品質 (QOS)

服務品質是指您爲伺服器實例虛擬伺服器類別或虛擬伺服器設定的效能範圍。例如，如果您是網際網路服務提供者 (ISP)，可能會依據所提供的頻寬對虛擬伺服器收取不同的費用。您可以限定兩個方面：頻寬量與連線數。

Sun ONE Application Server 提供的服務品質資訊用於確定伺服器在執行期間的效率，包括：

- 啓動時間
- 伺服器流量，以及流量對頻寬的影響
- 對活性資料與靜態資料的分析
- 其他資料元素

如需更多資訊，請參閱第 138 頁的 「使用 CLI 管理作業事件服務」。

使用 CLI 擷取監視資料

透過 `asadmin` 指令，您可以經由使用 `list` 與 `get` 指令，經由指令行介面 (CLI) 擷取受監視資料。

注意 如第 138 頁的「使用 CLI 管理作業事件服務」中所述，`set` 指令僅用於設定對作業事件服務的監視。

本章節討論下列主題：

- list --monitor 指令
- get --monitor 指令
- CLI 名稱對映
- HTTP 伺服器的可監視物件

list --monitor 指令

`list` 指令提供關於目前正在受監視的指定伺服器實例名稱應用程式元件與子系統的資訊。使用該指令，您可以檢視伺服器實例的可監視元件與子元件。

範例

```
asadmin> list --monitor server1
```

傳回下列已經啓用監視的應用程式元件與子系統清單：

```
iiop-service
transaction-service
application.converter
application.myApp
http-server
```

您也可以列示指定伺服器實例中目前被監視的應用程式。這對於使用 `get` 指令從應用程式中尋找特定的監視統計資料很有幫助。

範例

```
asadmin> list --monitor server1.application
```

傳回：

```
converter
myApp
```

如需更爲詳細的範例，請參閱第 122 頁的 「Petstore 範例」。

get --monitor 指令

該指令擷取下列受監視資訊：

- 一個元件或子系統內的全部受監視屬性
- 一個元件或子系統內特定的受監視屬性

當特定元件或子系統所需的屬性不存在時，便會傳回錯誤。同樣，當元件或子系統所需的特定屬性不在作用中時，也會傳回錯誤。

請參閱第 121 頁的 「CLI 名稱對映」，以取得有關使用 `get` 指令的更多資訊。

範例 *1*

嘗試從特定屬性的子系統中取得全部屬性：

```
asadmin> get --monitor server1.iiop-service.orb.system.orb-connection.*

total-inbound-connections=1
total-outbound-connections=1
```

範例 *2*

嘗試從 J2EE 應用程式中取得全部屬性：

```
asadmin> get --monitor server1.application.converter.*

Attribute name(s) not found
```

J2EE 應用程式層級上看不到可監視屬性，因此指令失敗。

範例 *3*

嘗試從子系統中取得特定屬性：

```
asadmin> get --monitor server1.transaction-service.inflight-tx

Attribute name = inflight-tx Value = No active transaction found.
```

範例 *4*

嘗試從一個子系統屬性中取得不明的屬性：

```
asadmin> get --monitor server1.iiop-service.orb.system.orb-connection.bad-name

              Could not get the attribute

Execution failed for the command:get --monitor
server1.iiop-service.orb-connection.bad-name
```

CLI 名稱對映

Sun ONE Application Server 使用樹狀結構追蹤可監視物件。樹上的每個節點都具有名稱和類型。如果類型爲單一型，則任何父節點下僅能有一個該類型的節點。如需有關此樹中節點類型的更多資訊，請參閱第 124 頁的「可監視物件類型」。

樹中的根物件由 Sun ONE Application Server 實例名稱表示。例如，名爲 `server1` 的實例之根監視物件爲：

```
server1
```

然後，使用點 (.) 字元作爲分隔符號來命名所有子物件。如果子節點爲單一型，則僅需要監視物件類型來命名物件；否則，會需要形式爲 `type.name` 的名稱來命名物件。

例如，http-server 爲一種有效的可監視物件類型，並且爲單一型。若要命名表示實例 server1 之 http-server 的單一型子節點，則稱爲：

```
server1.http-server
```

另一個範例，application 爲有效的可監視物件類型，但並非單一型。若要命名表示應用程式 Petstore 的非單一型子節點，名稱爲：

```
server1.application.petstore
```

CLI 名稱也可以命名可監視物件中的特定屬性。例如，http-server 具有名爲 summary 的可監視屬性。以下名稱可以命名 summary 屬性：

```
server1.http-server.summary
```

對於監視物件中的屬性名稱，沒有固定的命名慣例。

您無需瞭解 CLI 使用的有效名稱。list 指令可讓您檢查可用的可監視物件，而與萬用字元參數配合使用的 get 指令可讓您檢查任何可監視物件上的所有可用屬性。

下面的範例闡明一些用戶端名稱對映方案：

Petstore 範例

一個使用者想要檢查呼叫 Petstore 應用程式中方法的次數，該應用程式部署在名爲 server1 的 Sun ONE Application Server 實例上。配合使用 list 與 get 指令來存取所需的方法統計資料。

1. 以多種模式呼叫 CLI。
2. 設定一些有用的環境變數，以避免爲每個指令輸入這些變數：

   ```
   asadmin>export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
   ```

   ```
   asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4848
   ```

3. 列示實例 server1 的可監視元件：

   ```
   asadmin> list --monitor server1
   ```

   輸出的內容爲：

   ```
   iiop-service
   transaction-service
   application.CometEJB
   application.ConverterApp
   application.petstore
   http-server
   resources
   ```

可監視元件的清單包括 iiop-service、http-server、transaction-service、resources 以及所有已部署 (或已啓用) 的應用程式。

4. 列示 Petstore 應用程式中的可監視子元件 (可以使用 -m 代替 --monitor)：

```
asadmin>list -m server1.application.petstore
```

輸出的內容爲：

```
ejb-module.signon-ejb_jar
ejb-module.catalog-ejb_jar
ejb-module.uidgen-ejb_jar
ejb-module.customer-ejb_jar
ejb-module.petstore-ejb_jar
ejb-module.AsyncSenderJAR_jar
ejb-module.cart-ejb_jar
```

5. 列示 Petstore 應用程式之 EJB 模組 sigon-ejb_jar 中的可監視子元件：

```
asadmin>list -m server1.application.petstore.ejb-module.signon-ejb_jar
```

輸出的內容爲：

```
entity-bean.UserEJB
stateless-session-bean.SignOnEJB
```

6. 列示 Petstore 應用程式之 EJB 模組 sigon-ejb_jar 的實體 Bean UserEJB 中的可監視子元件：

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB
```

輸出的內容爲：

```
bean-method.create0
bean-method.findByPrimaryKey1
bean-method.remove2
bean-method.getUserName3
bean-method.setPassword4
bean-method.getPassword5
bean-method.matchPassword6
bean-method.remove7
bean-method.isIdentical8
bean-method.getEJBLocalHome9
bean-method.getPrimaryKey10
bean-pool
bean-cache
```

7. 列示 Petstore 應用程式之 EJB 模組 `sigon-ejb_jar` 的實體 Bean `UserEJB` 中方法 `getUserName3` 內的可監視子元件：

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB.bean-m
ethod.getUserName3
```

輸出的內容爲：

```
No monitorable entities for element
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB.bean-m
ethod.getUserName3
```

8. 沒有方法的可監視子元件。請取得方法 `getUserName3` 的全部可監視統計資料。

```
asadmin>get -m server1.application.petstore.ejb-module.
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.*
method-name = public abstract java.lang.String
com.sun.j2ee.blueprints.signon.user.ejb.UserLocal.getUserName()
total-num-errors = 0
total-num-success = 2
execution-time-millis = 1
total-num-calls = 2
```

9. 您也可以取得特定的統計資料，如執行時間。

```
asadmin>get -m server1.application.petstore.ejb-module.
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.execution-time-millis
execution-time-millis = 1
```

可監視物件類型

用於監視的物件樹包含數個節點。節點是物件樹中的特定項目，可以透過其類型、名稱以及父節點來唯一地識別。某些節點的類型爲單一型，表示父節點下僅能有一個此類型的節點。對於單一型節點，名稱是非相關的。

非單一型節點需要具有名稱。實例名稱欄對可能的名稱空間進行了描述。

下表依照各種節點類型與某些節點類型名稱空間之間可能的父子關係，對樹狀結構進行描述。

表格 6-1 監視物件類型

節點類型	單一型嗎？	葉節點嗎？	子節點類型	實例名稱
root	是	否	http-server iiop-service resources transaction-service application standalone-ejb-module	
http-server	是	否	virtual-server process	
virtual-server	是	是		
process	是	是		
iiop-service	是	是	orb	
orb	否	否	orb-connection orb-thread-pool	system 是系統 ORB 的保留名稱。所有的使用者 ORB 均會取得一個源自 TCP 端點的名稱。
orb-connection	是	是		
orb-thread-pool	是	是		
resources	是	否	jdbc-connection-pool	
jdbc-connection-pool	否	是		其名稱與建立連線區期間使用者指定的名稱相同。
transaction-service	是	是		
application	否	否	ejb-module	在 server.xml 中註冊的應用程式名稱。
ejb-module	否	否	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	EJB 模組的名稱。其源自 EJB JAR 名稱。
standalone-ejb-module	否	否	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	在 server.xml 中註冊的獨立 EJB 模組的名稱。
stateless-session-bean	否	否	bean-pool bean-method	來自於部署描述元的 Bean 名稱。

表格 6-1 監視物件類型 (續)

節點類型	單一型嗎？	葉節點嗎？	子節點類型	實例名稱
stateful-session-bean	否	否	bean-cache bean-method	來自於部署描述元的 Bean 名稱。
entity-bean	否	否	bean-cache bean-pool bean-method	來自於部署描述元的 Bean 名稱。
message-driven-bean	否	否	bean-pool bean-method	來自於部署描述元的 Bean 名稱。
bean-pool	是	是		
bean-cache	是	是		
bean-method	否	是		onMessage 用於訊息導引 Bean，帶有數字字尾的方法名稱用於其他企業 Bean 中的方法。(需要使用字尾來明確超載方法的意義。)

可監視屬性名稱

並非每個可監視物件都需要展示可監視屬性。某些物件僅用於對其他物件進行分組。對於 Sun ONE Application Server，除了節點 http-server，僅有樹上的葉節點才具有屬性。http-server 節點類型具有屬性和子節點。下表列示各種節點的可能可監視屬性名稱。

表格 6-2 http-server

屬性名稱	資料類型	描述
summary	字串 (已格式化)	HTTP 伺服器摘要。包括虛擬伺服器與程序。 注意：請參閱第 130 頁的「HTTP 伺服器的可監視物件」，以取得有關格式化字串內資料類型的更多資訊。

表格 6-3 virtual-server

屬性名稱	資料類型	描述
<vs-id>	字串 (已格式化)	虛擬伺服器資訊。每個應用程式伺服器實例都可以具有一個或多個虛擬伺服器。虛擬伺服器的 ID 清單可以從 http-server 的 summary 屬性中取得。您可以使用 server1.http-server.virtual-server.<vs-id> 形式的 get 指令參數來查找特定虛擬伺服器的統計資料。您可以使用 server1.http-server.virtual-server.* 形式的 get 指令參數來查找所有虛擬伺服器的統計資訊。 注意：請參閱第 130 頁的 「HTTP 伺服器的可監視物件」，以取得有關格式化字串內資料類型的更多資訊。

表格 6-4 process

屬性名稱	資料類型	描述
<pid>	字串 (已格式化)	程序資訊。每個應用程式伺服器實例僅有一個程序。可以從 http-server 的 summary 屬性中取得程序 ID。可以使用形式爲 server1.http-server.process.<pid> 的 get 指令參數取得程序的統計資料。 注意：請參閱第 130 頁的 「HTTP 伺服器的可監視物件」，以取得有關格式化字串內資料類型的更多資訊。

表格 6-5 orb-connection

屬性名稱	資料類型	描述
total-inbound-connections	整數	ORB 的內收連線總數。
total-outbound-connections	整數	ORB 的外送連線總數。

表格 6-6 orb-thread-pool

屬性名稱	資料類型	描述
thread-pool-size	整數	ORB 執行緒儲存區中的執行緒總數。
waiting-thread-count	整數	執行緒儲存區中等待工作的執行緒總數。

表格 6-7 jdbc-connection-pool

屬性名稱	資料類型	描述
total-threads-waiting	整數	等待 JDBC 連線的執行緒總數。
total-outbound-connections	整數	JDBC 連線驗證的失敗總數。
total-connections-timed-out	整數	逾時的連線請求總數。

表格 6-8 transaction-service

屬性名稱	資料類型	描述
total-tx-completed	整數	已完成的作業事件總數。
total-tx-rolled-back	整數	已回轉的作業事件總數。
total-tx-inflight	整數	執行中 (活性) 的作業事件總數。
isFrozen	字串	已凍結了作業事件系統嗎 (True 或 False)？
inflight-tx	字串 (已格式化)	執行中的作業事件清單。

表格 6-9 bean-pool

屬性名稱	資料類型	描述
max-pool-size	整數	儲存區內 Bean 實例的最大數目。
steady-pool-size	整數	儲存區內正常維護的 Bean 實例數目。首次建立儲存區後，將實例總裝於儲存區內，使該儲存區的大小等同於 -steadypool-size。從儲存區中移除實例時，需要隨後向該儲存區內補充實例，以使其大小等於或大於 steady-pool-size。
pool-resize-quantity	整數	儲存區增加到 max-pool-size 的增量或收縮到 steady-pool-size 的減少量。

表格 6-9 bean-pool (續)

屬性名稱	資料類型	描述
idle-timeout-in-seconds	整數	定義執行儲存區內清除執行緒的速率。檢查儲存區目前的大小是否大於固定的儲存區大小，並移除 pool-resize-quantity 元素。如果目前的儲存區大小小於 steady-pool-size，該儲存區便會增加 pool-resize-quantity，增量的上限爲 min (current-pool-size+pool + resize-quantity, max-pool-size)。僅有存取時間小於 pool-idle-timeout-in-seconds 的物件才是候選的移除物件。
num-beans-in-pool	整數	儲存區內可用 Bean 的數目。
num-threads-waiting	整數	等待自由 Bean 的執行緒數。
total-beans-created	整數	目前已建立的 Bean 的數目。
total-beans-destroyed	整數	目前已銷毀的 Bean 的數目。
jms-max-messages-load	整數	針對要服務的訊息導引 Bean 而一次載入 JMS 階段作業的最大訊息數。預設值爲 1。僅套用於訊息導引 Bean 的儲存區。

表格 6-10 bean-cache

屬性名稱	資料類型	描述
cache-resize-quantity (resize-quantity)	整數	快取記憶體中 Bean 的數目等於 max-cache-size (即發生了快取記憶體溢位) 時，快取記憶體的減少量。
cache-misses	整數	使用者請求沒有找到快取記憶體中 Bean 的次數。
idle-timeout-in-seconds	整數	排程快取記憶體清除程式執行緒的速率。該清除程式執行緒會檢查快取記憶體中的全部 Bean，並鈍化存取時間不足 cache-idle-timeout-in-seconds 的 Bean。
cache-hits	整數	使用者請求找到快取記憶體中某個項目的次數。
total-beans-in-cache	整數	快取記憶體中 Bean 的數目。這便是快取記憶體目前的大小。
max-beans-in-cache	整數	快取記憶體中最多可持有的 Bean 數目，超過此數目，便會發生快取記憶體溢位。
num-passivations	整數	鈍化的數目。僅套用於有狀態階段作業 Bean。
num-passivation-errors	整數	鈍化期間發生的錯誤數。僅套用於有狀態階段作業 Bean。

表格 6-10 bean-cache (續)

屬性名稱	資料類型	描述
num-expired-sessions-removed	整數	清除執行緒所移除的過期階段作業數目。僅套用於有狀態階段作業 Bean。
num-passivation-success	整數	鈍化成功完成的次數。僅套用於有狀態階段作業 Bean。

表格 6-11 bean-method

屬性名稱	資料類型	描述
method-name	字串	完整的方法名稱。
total-num-calls	整數	方法已經被呼叫的次數。如果爲 EJB 容器啓用了監視，便會爲有狀態階段作業 Bean、無狀態階段作業 Bean 以及實體 Bean 收集該呼叫次數；如果爲訊息導引 Bean 容器啓用了監視，則會爲訊息導引 Bean 收集該呼叫次數。
total-num-errors	整數	執行方法導致異常的次數。如果在 EJB 設定下啓用了監視，便會爲有狀態階段作業 Bean、無狀態階段作業 Bean 以及實體 Bean 收集該次數；如果在 MDB 設定下啓用了監視，則會爲訊息導引 bean 收集該次數。
total-num-success	整數	方法成功執行的次數。如果爲 EJB 容器啓用了監視，便會爲有狀態階段作業 Bean、無狀態階段作業 Bean 以及實體 Bean 收集該呼叫次數；如果爲容器啓用了監視，則會爲訊息導引 Bean 收集該呼叫次數。
execution-time-millis	長型	上次成功執行此方法的執行時間。如果在 EJB 容器上啓用了監視，便會爲有狀態階段作業 Bean、無狀態階段作業 Bean 以及實體 Bean 收集該執行時間；如果在容器上啓用了監視，則會爲訊息導引 Bean 收集該執行時間。

HTTP 伺服器的可監視物件

HTTP 伺服器的可監視屬性名稱 `summary` 會列印 `Server` 元素的屬性值，及其子元素的摘要，包括每個子元素的數目，以及每個子元素的屬性值。HTTP 伺服器的 `virtual-server` 屬性會列印 `VirtualServer` 元素的屬性值及其每個子元素的詳細資訊。`process` 屬性會列印 `Process` 元素的屬性值及每個子元素的詳細資訊。

若要啓用 NSAPI 效能設定檔，並取得有關 `Profile` 與 `ProfileBucket` 元素的統計資料，請參閱「*Sun ONE Application Server Developer's Guide to NSAPI*」。

如需有關如何使用效能微調的監視統計資料之資訊，請參閱「*Sun ONE Application Server Performance and Tuning Guide*」。

可監視的 HTTP 伺服器元素

下表列示可監視的 HTTP 伺服器元素。

表格 6-12 可監視的 HTTP 伺服器元素

元素名稱	子元素	描述
`Server`	ConnectionQueue ThreadPool Profile Process VirtualServer	一個伺服器實例。
`ConnectionQueue`	無	請求在被服務之前所處的佇列。Sun ONE Application Server 7 中僅有一個連線佇列。
`ThreadPool`	無	在 `init.conf` 檔案中定義的執行緒儲存區。
`Profile`	無	`init.conf` 檔案中定義的 NSAPI 效能設定檔儲存區。

`Process`	`ConnectionQueueBucket`
	`ThreadPoolBucket`
	`DnsBucket`
	`DnsBucket`
	`KeepaliveBucket`
	`CacheBucket`
	`Thread`
		伺服器實例內的單一伺服器程序。

`ConnectionQueueBucket`	無	追蹤有關特定 `ConnectionQueue` 的統計資料。
`ThreadPoolBucket`	`ThreadPoolBucket`	追蹤有關特定 `ThreadPool` 的統計資料。
`DnsBucket`	無	追蹤 DNS 統計資料。
`KeepaliveBucket`	無	追蹤 keepalive (持續性連線) 統計資料。
`CacheBucket`	無	追蹤檔案快取記憶體 (NSFC) 統計資料。
`Thread`	`RequestBucket` `ProfileBucket`	描述處理執行緒的請求。
`VirtualServer`	`RequestBucket` `ProfileBucket`	描述虛擬伺服器。
`RequestBucket`	無	追蹤與請求相關的統計資料。
`ProfileBucket`		追蹤有關 `Profile` 元素的統計資料。

可監視的 HTTP 伺服器屬性

下表列示可監視的 HTTP 伺服器屬性。

表格 6-13 伺服器

屬性名稱	值	描述
Id		伺服器實例 ID (例如，server1)。
VersionServer		包含 Sun ONE Application Server 版本的字串。
TimeStarted	GMT	此伺服器實例啓動的時間。
SecondsRunning		該伺服器實例啓動後的秒數。
TicksPerSecond		一秒內的滴答次數。該值的大小取決於系統。
MaxProcs		程序的最大數目。
MaxThreads		處理執行緒的最大數目。
MaxVirtualServers		追蹤的虛擬伺服器的最大數目。
FlagProfilingEnabled	0 (關閉)， 1 (開啓)	指出 NSAPI 效能設定檔是否被啓用 (開啓)。
FlagVirtualServerOverflow	0 (否)， 1 (是)	指出是否配置了多於 MaxVirtualServers 的虛擬伺服器 (是)。如果將該屬性設定爲 1，則不會爲所有的虛擬伺服器追蹤統計資料。
LoadMinuteAverage		1 分鐘負載平均值。
Load5MinuteAverage		5 分鐘負載平均值。
Load15MinuteAverage		15 分鐘負載平均值。
RateBytesTransmitted	每秒傳輸的位元組	資料在某一伺服器定義間隔時間內的傳輸速率，如果沒有該資訊，則值爲 0。
RateBytesReceived	每秒傳輸的位元組	資料在某一伺服器定義間隔時間內的接收速率，如果沒有該資訊，則值爲 0。

表格 6-14 ConnectionQueue

屬性名稱	值	描述
Id		連線佇列 ID。

表格 6-15 ThreadPool

屬性名稱	值	描述
Id		執行緒儲存區 ID。
Name		執行緒儲存區的符號名稱。

表格 6-16 Profile

屬性名稱	值	描述
Id		NSAPI 效能設定檔儲存區 ID。
Name		NSAPI 效能設定檔儲存區的符號名稱。
描述		NSAPI 效能設定檔儲存區的描述。

表格 6-17 Process

屬性名稱	值	描述
Pid		唯一地識別此程序的作業系統程序識別碼。
Mode	unknown active	當此程序在作用中時，顯示 active。
TimeStarted	GMT	此程序啓動的時間。
CountConfigurations		載入某一配置的次數，如果該資訊不存在，則值爲 0。
SizeVirtual	千位元組	此程序使用的虛擬記憶體大小。
SizeResident	千位元組	此程序使用的常駐記憶體大小。
FractionSystemMemoryUsage		此程序使用的系統記憶體部分。

表格 6-18 ConnectionQueueBucket

屬性名稱	值	描述
ConnectionQueue		ConnectionQueue 元素的 ID。
CountTotalConnection		已接受的新連線總數。
CountQueued		目前排列的連線數。
PeakQueued		同時位於佇列的最大連線數。
MaxQueued		可以位於佇列的最大連線數。
CountOverflow		佇列已滿而無法容納更多連線的的次數。
CountTotalQueued		已經排列的連線總數。可以多次排列一個給定的連線。因此，CountTotalQueued 可能大於或等於 CountTotalConnections。
TicksTotalQueued		連線在佇列中所使用的滴答總數。滴答是取決於系統的時間單位，請參閱 TicksPerSecond。

表格 6-19 ThreadPoolBucket

屬性名稱	值	描述
Thread-pool		ThreadPool 元素的 ID。
CountThreadsIdle		處理目前閒置執行緒的請求數。
CountThreads		處理執行緒的請求數。
MaxThreads		處理可並存執行緒的最大請求數。
CountQueued		透過此執行緒儲存區進行處理的佇列請求數。
PeakQueued		同時位於佇列的最大請求數。
MaxQueued		可以位於佇列的最大請求數。

表格 6-20 DnsBucket

屬性名稱	值	描述
FlagCacheEnabled	0 (關閉)，1 (開啓)	指出 DNS 快取記憶體是否被啓用 (開啓)。
CountCacheEntries		目前在快取記憶體中的 DNS 項目數。
MaxCacheEntries		快取記憶體可以容納的最大 DNS 項目數。
CountCacheHits		DNS 快取記憶體查找成功的次數。
CountCacheMisses		DNS 快取記憶體查找失敗的次數。
FlagAsyncEnabled	0 (關閉)，1 (開啓)	指出非同步 DNS 查找是否已啓用 (開啓)。
CountAsyncNameLookups		已執行之非同步 DNS 名稱查找的總數。
CountAsyncAddrLookups		已執行之非同步 DNS 位址查找的總數。
CountAsyncLookupsInProgress		正在執行之非同步 DNS 查找的總數。

表格 6-21 KeepaliveBucket

屬性名稱	值	描述
CountConnections		目前處於 keepalive 模式下的連線數。
MaxConnections		同步 keepalive 連線的最大數目。
CountHits		Keepalive 模式中的連線隨後作出有效請求的總次數。
CountFlushes		伺服器關閉 keepalive 連線的次數。
CountTimeouts		Keepalive 連線逾時的次數。
SecondsTimeouts		伺服器關閉閒置 keepalive 連線之前的秒數。
CountRefusals		伺服器拒絕 keepalive 連線的次數。

表格 6-22 CacheBucket

屬性名稱	值	描述
FlagEnabled	0 (關閉)，1 (開啓)	指出檔案快取記憶體是否已啓用 (開啓)。
SecondsMaxAge	秒數	檔案快取記憶體項目的最長存在時間。
CountEntries		目前位於檔案快取記憶體中的項目數。
MaxEntries		檔案快取記憶體可以容納的最大快取記憶體項目數。
CountOpenEntries		與開放式檔案相關聯的項目數。
MaxOpenEntries		與開放式檔案 (檔案快取記憶體可以同時容納該檔案) 相關聯的最大快取記憶體項目數。
SizeHeapCache	位元組的數目	快取的檔案內容所使用的堆疊數量。
MaxHeapCacheSize	位元組的數目	檔案快取記憶體用於快取檔案內容的最大堆疊數量。
SizeMmapCache	位元組的數目	記憶體對映檔內容所使用的位址空間總量。
MaxMmapCacheSize	位元組的數目	檔案快取記憶體用於記憶體對映檔內容的最大位址空間量。
CountHits		快取記憶體項目查找成功的次數。
CountMisses		快取記憶體項目查找失敗的次數。
CountInfoHits		檔案資訊查找成功的次數。
CountInfoMisses		檔案資訊查找失敗的次數。
CountContentHits		內容查找成功的次數。
CountContentMisses		內容查找失敗的次數。

表格 6-23 Thread

屬性名稱	值	描述
Mode	unknown、idle、DNS、request、processing、response、updating	執行緒的上一個已知狀況。
TimeStarted	GMT	此執行緒啓動的時間。
ConnectionQueue		執行緒正在服務的 ConnectionQueue 之 ID。

表格 6-24 VirtualServer

屬性名稱	值	描述
Id		虛擬伺服器 ID。
Mode	unknown、active	當此虛擬伺服器在作用中時，顯示 active。
Hosts		由此虛擬伺服器提供服務的軟體虛擬伺服器主機名稱 (例如，www.foo.com foo.com foo.isp.com)。
Interfaces		需要配置虛擬伺服器的介面 (偵聽程式) (例如，192.168.1.2:80 192.168.1.2:443)。

表格 6-25 RequestBucket

屬性名稱	值	描述
CountRequests		被服務的請求數。
CountBytesReceived		收到的位元組數目，如果該資訊不存在，則值爲 0。
CountBytesTransmitted		已傳輸的位元組數目，如果該資訊不存在，則值爲 0。
RateBytesTransmitted	每秒傳輸的位元組	資料在某一伺服器定義間隔時間內的傳輸速率，如果沒有該資訊，則值爲 0。
MaxByteTransmissionRate		資料在某一伺服器定義間隔時間內的最大傳輸速率，如果沒有該資訊，則值爲 0。
CountOpenConnections		開放式連線的數目，如果該資訊不存在，則值爲 0。
MaxOpenConnections		開放式連線的最大數目，如果該資訊不存在，則值爲 0。
Count2xx		已發送的 200 層級回應的數目。
Count3xx		已發送的 300 層級回應的數目。
Count4xx		已發送的 400 層級回應的數目。
Count5xx		已發送的 500 層級回應的數目。
CountOther		已發送的 200、300、400、或 500 層級之外的回應數目。
Count200		已發送的 200 層級回應的數目。
Count302		已發送的 302 層級回應的數目。
Count304		已發送的 304 層級回應的數目。
Count400		已發送的 400 層級回應的數目。

表格 6-25 RequestBucket (續)

屬性名稱	值	描述
Count401		已發送的 401 層級回應的數目。
Count403		已發送的 403 層級回應的數目。
Count404		已發送的 404 層級回應的數目。
Count503		已發送的 503 層級回應的數目。

表格 6-26 ProfileBucket

屬性名稱	值	描述
Profile		Profile 元素的 ID。
Countcalls		呼叫 NSAPI SAF 的次數。
CountRequests		已處理的請求數。
TicksDispatch		用於派送請求的滴答數。滴答是取決於系統的時間單位，請參閱 TicksPerSecond。
TicksFunction		用於 NSAPI SAF 的滴答數。滴答是取決於系統的時間單位，請參閱 TicksPerSecond。

使用 CLI 管理作業事件服務

您可以使用 set 指令，管理要針對 JTS 而監視的統計資料。

範例 *1*

若要在回轉清單內加入一個作業事件 (此作業會導致回轉或指定的作業事件)，請將 set 指令發佈爲：

```
set --monitor server1.transaction-service.rollback-list=txnid1
```

範例 *2*

若要凍結作業事件服務，請將 set 指令發佈爲：

```
set --monitor server1.transaction-service.freeze=true
```

下表描述可以監視以收集 JTS 統計資料的屬性。可以依據第 121 頁的「CLI 名稱對映」中描述的規則，經由指令行設定這些屬性。

如需有關 Java 作業事件服務的更多資訊，請參閱第 9 章「使用作業事件服務」。

使用 HTTP 服務品質

下列設定控制如何計算流量，以及再計算頻寬的頻率：

- 再計算間隔時間 — 指出計算頻寬的頻率 (以毫秒表示)。
- 公制間隔時間 — 在流量計算中使用資料的時間長度。

在管理介面中，您可以爲伺服器實例或某類虛擬伺服器啓用這些伺服器或類別層級設定。不過，您可以針對個別虛擬伺服器而置換這些設定。

本章節包括下列主題：

- 服務品質的範例
- 配置服務品質 (QOS)
- 對 obj.conf 檔案的必要變更
- 服務品質的已知限制

服務品質的範例

以下範例顯示如何收集與計算服務品質資訊。

- 伺服器具有的公制間隔時間爲 30 秒。
- 在 0 秒處啓動伺服器。
- 在 1 秒處，一個 HTTP 連線產生 5000 個位元組輸入或輸出伺服器的流量。
- 此後，不會增加更多的連線。在 30 秒處，前 30 秒的總流量爲 5000 個位元組。
- 在 32 秒處，放棄從 1 秒開始的流量取樣，因爲時間大於公制間隔時間 30 秒。前 30 秒的總流量現在爲 0。

再計算間隔時間的工作原理與此相似。伺服器的再計算間隔時間爲 100 毫秒。

繼續該範例，每 100 毫秒定期地再計算一次頻寬。該計算以流量和公制間隔時間爲基礎。

- 在 0 秒處，首次計算頻寬。總流量爲 0，除以公制間隔時間 30 秒，則頻寬爲 0。
- 在 1 秒處，第 10 次計算頻寬 (1000 毫秒 /100 毫秒)。總流量爲 5000 個位元組，用 5000 除以 30 秒。頻寬爲每秒 5000/30 = 166 個位元組。

- 在 30 秒處，第 300 次計算頻寬。總流量爲 5000 個位元組，用 5000 除以 30 秒。頻寬爲每秒 5000/30 = 166 個位元組。
- 在 32 秒處，第 320 次計算頻寬。流量現在爲 0 (由於產生流量的連線間隔時間太長而無法計算)，除以 30，則頻寬爲 0 個位元組 / 秒。

配置服務品質 (QOS)

經由管理介面配置伺服器實例或某類虛擬伺服器的服務品質。

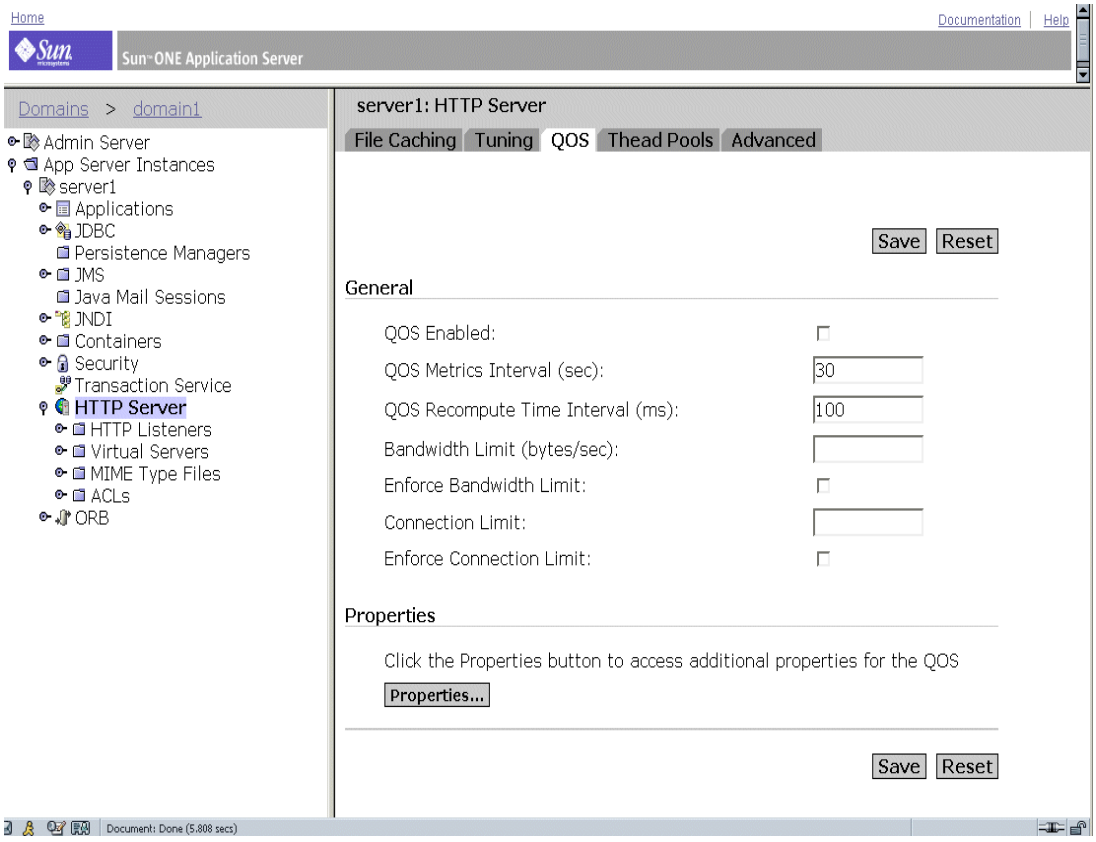
注意 若要執行您的服務品質設定，您也必須在 `obj.conf` 檔案中設定伺服器應用程式功能 (SAF)，如第 143 頁的「對 obj.conf 檔案的必要變更」中所述。

若要配置服務品質，請採取以下步驟：

1. 請在左窗格中選取 [App Server Instances] 節點。
2. 展開伺服器實例節點以顯示 [HTTP Server] 節點。
3. 按一下 [HTTP Server] 節點以顯示 [QOS] 標籤。
4. 按一下 [QOS] 標籤。

   以下頁面顯示服務品質的一般設定，設定後面跟隨有一個 [Properties] 按鈕。

圖 6-1 虛擬伺服器實例 [QOS] 標籤



5. 若要啓用此 HTTP 伺服器的服務品質，請按一下 [QOS Enabled]。

   注意：依預設，服務品質處於停用狀態。啓用服務品質會稍微增加伺服器的耗用時間。

6. 指定 [QOS Metrics Interval]。

   公制間隔時間爲在伺服器流量計算期間，以秒表示的資料取樣時間。其預設值爲 30 秒。

   如果站點一般要傳輸大量檔案，請在此欄位中使用較大的值 (數分鐘或更長)。較大的檔案傳輸可能會在較短的公制間隔時間內佔用所有可用的頻寬，如果您已經執行了最大的頻寬設定，還可能導致連線被拒絕。由於頻寬通過除以公制間隔時間來平均分配，所以，較長的間隔時間可以消除由大檔案引起的尖波。

如果頻寬限定遠小於可用的頻寬 (例如，頻寬限定爲 1 MB 每秒，但是到主幹的連線爲 1 GB 每秒)，則應該縮短公制間隔時間。

注意：如果您要進行大的靜態檔案傳輸，並且頻寬限定遠小於可用的頻寬，則必須確定要調整的條件，因爲這些問題需要相反的解決方案。

7. 指定 QOS 再計算的時間間隔。

   再計算時間間隔是指每次計算所有伺服器、類別以及虛擬伺服器頻寬之間的毫秒數。預設值爲 100 毫秒。

8. 指定頻寬限定。

   這是伺服器實例以位元組/秒表示的最大頻寬。在某種程度上，其與 QOS 公制間隔時間相互依賴。

9. 選擇是否執行最大頻寬設定。

   如果您選擇執行最大頻寬，則當伺服器達到其頻寬限定時，便會拒絕附加的連線。

   如果未執行最大頻寬，則當超過最大設定時，伺服器會在事件日誌中記錄一則訊息。

10. 指定連線限定。

    這是被處理的並行請求數。

11. 選擇是否執行連線限定設定。

    如果您選擇執行最大連線，則當伺服器達到其連線限定時，便會拒絕附加的連線。如果未執行最大連線，則當超過最大設定時，伺服器會在事件日誌中記錄一則訊息。

12. (可選)。如果要爲服務品質指定附加的名稱值對屬性，請按一下 [Properties] 按鈕。

    如需服務品質特性的可用名稱值對，請參閱線上輔助說明。

13. 按一下 [Save]，以確定對伺服器實例的變更。

14. 在左窗格中存取 [App Server Instances] 和您的伺服器實例，然後按一下 [Apply Changes]。

對 obj.conf 檔案的必要變更

若要執行服務品質，必須將指令納入您的 `obj.conf` 檔案，以呼叫下列伺服器應用程式功能 (SAF)：

- AuthTrans `qos-handler`
- Error `qos-error`

爲了正確作業，`qos-handler` AuthTrans 指令必須爲預設物件中第一個配置的 `AuthTrans`。服務品質處理程式的作用爲檢查虛擬伺服器、虛擬伺服器類別，以及全域伺服器的目前統計資料，並透過傳回錯誤來執行限定。Sun ONE Application Server 包含內建的取樣服務品質處理程式 SAF，稱爲 `qos-handler`。該 SAF 會在伺服器達到限定的時候進行記錄，並將 503 `Server busy` 錯誤傳回至伺服器，以便 NSAPI 對其進行處理。

Sun ONE Application Server 還包含內建的取樣錯誤 SAF，稱爲 `qos-error`，其會傳回錯誤頁面，該頁面將指出導致 503 錯誤的限定，以及觸發限定的統計資料值。

如需有關這些 SAF 以及如何使用它們的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to NSAPI*」。

服務品質的已知限制

當您使用服務品質功能時，請記住下列限制：

- 服務品質功能僅測量應用程式層級的 HTTP 頻寬。HTTP 頻寬可以不同於實際的 TCP 網路頻寬，原因如下：
  - 如果啓用了 SSL，訊號交換與用戶端證書交換會加入到流量中，但並未測量。
  - 如果在一個或兩個方向上啓用了塊狀式編碼，則塊狀層會移除塊標頭，這些塊標頭便不會被計入流量中。但是會計算其他標頭或協定項目。
- 服務品質功能無法精確計算來自於 `PR_TransmitFile` 呼叫的流量。對於基本的 I/O 作業，如 `PR_Send()/net_write` 或 `PR_Recv()/net_read`，可以透過頻寬管理程式迅速說明傳輸資料，因爲某個系統呼叫中傳輸的位元組數目通常等於緩衝區的大小，而且 I/O 呼叫可以立刻傳回。可以很好地測量動態內容應用程式的即時頻寬。但是，由於只有在傳輸要結束時才能知道來自於 `PR_TransmitFile` 的傳輸資料量，因此，在傳輸完成之前無法進行測量。

如果 PR_TransmitFile 很小，則可以充份地執行服務品質功能。但是，如果 PR_TransmitFile 很大，如撥號使用者下載的大型檔案，就會在完成的時候計算傳輸資料總量。當頻寬管理程式在啓動下一個再計算間隔時間之後再次計算頻寬時，由於最近下載的大型 PR_TransmitFile，要計算的頻寬會顯著增加。這樣會導致伺服器在下一個公制間隔時間之前拒絕所有請求，這時頻寬管理程式會「拒絕」傳輸檔案作業 (因爲間隔時間太長)，因此頻寬值會回落。如果您的站點經常下載大型靜態檔案，應該從預設的 30 秒開始增加公制間隔時間。

- 計算的頻寬總是爲近似值，因爲其不是即時測量的，但是會在一定時期內以規則的間隔時間多次進行計算。例如，如果預設的公制間隔時間爲 30 秒，並且伺服器閒置了 29 秒，那麼在下一秒內用戶端可能會使用 30 次頻寬限定。
- 每當動態地重新配置伺服器時，便會遺失服務頻寬統計資料。此外，如果執行緒具有過時、非作用配置上的連線，則不會在這些執行緒中執行服務品質限定。因爲頻寬管理程式執行緒僅計算作用中配置的頻寬統計資料。可能的情況是，動態地重新配置伺服器之後，服務品質限制不會限制用戶端長時間地不關閉其套接字，並保持使用中狀態，這樣伺服器不會使其逾時。
- 與計算虛擬伺服器類別以及全域伺服器實例的並行連線相比較，虛擬伺服器透過不同顆粒性來計算並行連線。剖析請求並將其路由至虛擬伺服器之後，會立即自動增加個別虛擬伺服器的連線計數器。在針對該請求的回應處理完成時，也會自動減少連線計數器。這說明虛擬伺服器連線統計資料在任何時刻都是精確的。

  然而，虛擬伺服器類別與全域伺服器實例的連線統計資料不會即時更新。它們在每個再計算間隔時間後由頻寬管理程式執行緒更新。虛擬伺服器類別的連線計數爲該類別中全部虛擬伺服器上的連線；全域伺服器實例連線計數爲所有虛擬伺服器類別上的連線。

  對這些值採取這樣的計算方式後，虛擬伺服器的連線數目總是正確，如果您已經執行了連線的限定數目，便再也不可以增加連線的數目。虛擬伺服器類別與伺服器實例值並非十分精確，因爲僅僅間或地計算它們。

關於 SNMP

簡單網路管理協定 (SNMP) 是用於在網路之間交換管理和監視資訊的協定。透過 SNMP，資料在管理裝置與網路管理站 (NMS) 之間傳輸。**管理裝置**是指執行 SNMP 的所有裝置：主機、路由器、您的 HTTP 伺服器以及您網路上的其他伺服器。

本章節討論下列主題：

- 網路管理站 (NMS)
- 管理資訊庫 (MIB) 物件
- SNMP 訊息

- SNMP 陷阱目標
- SNMP 代理程式團體

網路管理站 (NMS)

網路管理站 (NMS) 是指用於遠端管理特定網路的機器。通常，NMS 軟體會提供圖形來顯示收集到的資料，或使用此資料確定伺服器在特定的容許度下作業。

NMS 通常爲安裝了一個或多個網路管理應用程式的功能強大的工作站。諸如 HP OpenView 的網路管理應用程式以圖形的方式顯示有關管理裝置 (如您的 HTTP 伺服器) 的資訊。例如，它可以顯示您企業中工作和停用的伺服器，以及收到的錯誤訊息數目與類型。將 SNMP 與 Sun ONE Application Server 配合使用時，會通過使用下列兩類代理程式使該資訊在 NMS 與伺服器之間傳輸：子代理程式與主代理程式。

子代理程式收集有關在各種領域中執行的伺服器實例之資訊，並將此資訊傳送至主代理程式。每個 Sun ONE Application Server 安裝版本中都含有一個主代理程式和一個子代理程式。

注意 變更 SNMP 配置之後，您必須按一下 [Apply] 按鈕，然後重新啓動 SNMP 子代理程式。

主代理程式在各種子代理程式與 NMS 之間交換資訊。主代理程式隨附 Sun ONE Application Server 一同安裝。

您可以在一台主機上安裝多個子代理程式，但僅能安裝一個主代理程式。例如，如果您在同一台主機上安裝了 Sun ONE Directory Server、Sun ONE Application Server 以及 Sun ONE Messaging Server，則每個伺服器的子代理程式將與同一個主代理程式進行通訊。

NMS 或者從伺服器請求資訊，或者變更伺服器 MIB 中的變數儲存區的值。例如：

1. NMS 將一則訊息發送至管理伺服器主代理程式。該訊息可能是對資料的請求 (一則 `GET` 訊息)，或者是一條設定 MIB 內變數的指令 (一則 `SET` 訊息)。
2. 主代理程式將訊息轉寄至適當的子代理程式。
3. 子代理程式擷取資料或變更 MIB 中的變數。
4. 子代理程式將資料或狀況報告給主代理程式，然後，主代理程式將訊息轉寄回 (一則 `GET` 訊息) NMS。
5. NMS 經由其網路管理應用程式，用文字或圖形顯示資料。

管理資訊庫 (MIB) 物件

Sun ONE Application Server 儲存了有關管理與監視跨網路資訊的變數。主代理程式可存取的變數稱爲管理物件。這些物件在稱爲**管理資訊庫** (MIB) 的樹狀結構中定義。使用 MIB 可存取 HTTP 伺服器的網路配置、狀態以及統計資料。使用 SNMP，您可以經由網路管理站 (NMS) 檢視此資訊。

MIB 樹的頂層顯示出網際網路物件識別碼具有下列子樹：

- directory (1)
- mgmt (2)
- experimental (3)
- private (4)

子樹 private (4) 包含節點 enterprises (1)。Enterprises (1) 節點中的每個子樹被指定給個別的企業，該企業爲已註冊其自身特定 MIB 延伸的組織。企業然後便可以在其子樹下建立產品特定子樹。公司建立的 MIB 位於節點 enterprises (1) 之下。

每個 Sun ONE Application Server 子代理程式都會提供一個 MIB 以用於 SNMP 通訊。伺服器通過發送訊息或包含這些變數的陷阱 ，將重要的事件報告給 NMS。NMS 可以查詢伺服器的 MIB 以取得資料。

每個 Sun ONE Application Server 都具有其自身的 MIB，位於：*install_dir*`/lib`

Sun ONE Application Server 的 MIB 是一個名爲 `appserv.mib` 的檔案。該 MIB 包含有關 Sun ONE Application Server 網路管理之各種變數的定義。

Sun ONE Application Server MIB 具有物件識別碼

`appserver 1 (as appserver7 OBJECT IDENTIFIER ::= {appserver 1 })` ，該識別碼位於目錄 *install_dir*`/lib` 中。

您可以察看關於 Sun ONE Application Server 的管理資訊，並使用 Sun ONE Application Server MIB 即時監視伺服器。下表列示並描述儲存於 `appserv.mib` 檔案中的管理物件。

appserv.mib 管理物件與描述

管理物件	描述
iwsCpuID	CPU 識別碼。
iwsCpuIdleTime	CPU 閒置時間。
iwsCpuKernelTime	CPU 核心時間。
iwsCpuTable	Sun ONE Application Server CPU。
iwsCpuUserTime	CPU 使用者時間。
iwsInstanceTable	Sun ONE Application Server 實例。
iwsInstanceId	伺服器實例識別碼。
iwsInstanceVersion	字串，如 SunONE-ApplicationServer-Enterprise/7 BB1-01/24/2001 17:15 (SunOS DOMESTIC)
iwsInstanceDescription	伺服器實例的描述。
iwsInstanceOrganization	負責伺服器實例的組織。
iwsInstanceContact	伺服器實例負責人的聯絡資訊。
iwsInstanceLocation	伺服器的位置。
iwsInstanceStatus	伺服器實例的狀況。
iwsInstanceUptime	伺服器的執行時間。
iwsInstanceDeathCount	伺服器實例程序停止的次數。
iwsInstanceRequests	伺服器實例處理的請求數。
iwsInstanceInOctets	伺服器實例收到的八位元組數目。如果資訊不可用，會顯示爲 0。
iwsInstanceOutOctets	伺服器實例傳輸的八位元組數目。如果資訊不可用，會顯示爲 0。
iwsInstanceCount2xx	由伺服器實例發佈的 200 層級 (成功的) 回應的數目。
iwsInstanceCount3xx	由伺服器實例發佈的 300 層級 (重新導向) 回應的數目。
iwsInstanceCount4xx	由伺服器實例發佈的 400 層級 (用戶端錯誤) 回應的數目。
iwsInstanceCount5xx	由伺服器實例發佈的 500 層級 (伺服器錯誤) 回應的數目。
iwsInstanceCountOther	由伺服器實例發佈的其他 (既非 2xx、3xx、4xx，也非 5xx) 回應的數目。
iwsInstanceCount200	由伺服器實例發佈的 200 (確定) 回應的數目。

appserv.mib 管理物件與描述 (**續**)

管理物件	**描述**
iwsInstanceCount302	由伺服器實例發佈的 302 (暫時移動) 回應的數目。
iwsInstanceCount304	由伺服器實例發佈的 304 (未修改) 回應的數目。
iwsInstanceCount400	由伺服器實例發佈的 400 (錯誤的請求) 回應的數目。
iwsInstanceCount401	由伺服器實例發佈的 401 (未授權) 回應的數目。
iwsInstanceCount403	由伺服器實例發佈的 403 (禁止的) 回應的數目。
iwsInstanceCount404	由伺服器實例發佈的 404 (未找到) 回應的數目。
iwsInstanceLoad1MinuteAverage	執行伺服器實例的系統之 1 分鐘負載平均值。
iwsInstanceLoad5MinuteAverage	執行伺服器實例的系統之 5 分鐘負載平均值。
iwsInstanceLoad15MinuteAverage	執行伺服器實例的系統之 15 分鐘負載平均值。
iwsInstanceNetworkInOctets	網路上每秒傳輸的八位元組數目。
iwsInstanceNetworkOutOctets	網路上每秒收到的八位元組數目。
iwsVsTable	伺服器虛擬伺服器。
iwsVsId	虛擬伺服器識別碼。
iwsVsRequests	虛擬伺服器處理的請求數。
iwsVsInOctets	虛擬伺服器收到的八位元組數目。
iwsVsOutOctets	虛擬伺服器傳輸的八位元組數目。
iwsVsCount2xx	虛擬伺服器發佈的 200 層級 (成功的) 回應的數目。
iwsVsCount3xx	虛擬伺服器發佈的 300 層級 (重新導向) 回應的數目。
iwsVsCount4xx	虛擬伺服器發佈的 400 層級 (用戶端錯誤) 回應的數目。
iwsVsCount5xx	虛擬伺服器發佈的 500 層級 (伺服器錯誤) 回應的數目。
iwsVsCountOther	虛擬伺服器發佈的其他 (既非 2xx、3xx、4xx，也非 5xx) 回應的數目。
iwsVsCount200	虛擬伺服器發佈的 200 (確定) 回應的數目。
iwsVsCount302	虛擬伺服器發佈的 302 (暫時移動) 回應的數目。
iwsVsCount304	虛擬伺服器發佈的 304 (未修改) 回應的數目。
iwsVsCount400	虛擬伺服器發佈的 400 (錯誤的請求) 回應的數目。
iwsVsCount401	虛擬伺服器發佈的 401 (未授權) 回應的數目。
iwsVsCount403	虛擬伺服器發佈的 403 (禁止的) 回應的數目。

appserv.mib 管理物件與描述 (**續**)

管理物件	**描述**
iwsVsCount404	虛擬伺服器發佈的 404 (未找到) 回應的數目。
iwsProcessTable	Sun ONE Application Server 程序。
iwsProcessId	作業系統程序識別碼。
iwsProcessThreadCount	處理執行緒的請求數。
iwsProcessThreadIdle	處理目前閒置執行緒的請求數。
iwsProcessConnectionQueueCount	目前位於連線佇列的連線數。
iwsProcessConnectionQueuePeak	已經同時排列的最大連線數。
iwsProcessConnectionQueueMax	連線佇列所允許的最大連線數。
iwsProcessConnectionQueueTotal	已被接受的連線數。
iwsProcessConnectionQueueOverflows	由於連線佇列溢位而被拒絕的連線數。
iwsProcessKeepaliveCount	目前位於 keepalive 佇列的連線數。
iwsProcessKeepaliveMax	Keepalive 佇列所允許的最大連線數。
iwsProcessSizeVirtual	以千位元組表示的程序大小。
iwsProcessSizeResident	以千位元組表示的常駐程序大小。
iwsProcessFractionSystemMemoryUsage	系統記憶體中的程序記憶體部分。
iwsListenTable	Sun ONE Application Server 偵聽套接字。
iwsListenId	偵聽套接字識別碼。
iwsListenAddress	套接字偵聽的位址。
iwsListenPort	套接字偵聽的連接埠。
iwsListenSecurity	加密支援。
iwsThreadPoolCount	被拒絕的請求數。
iwsThreadPoolMax	佇列所允許的最大請求數。
iwsThreadPoolPeak	已經同時排列的最大請求數。
iwsThreadPoolTable	Sun ONE Application Server 執行緒儲存區。
iwsVsCount503	已發佈的 503 (不可用) 回應的數目。
iwsInstanceCount503	已發佈的 503 (不可用) 回應的數目。

SNMP 訊息

GET 與 SET 是由 SNMP 定義的兩類訊息。

在 MIB 內，每個物件都被指定一個唯一的識別碼。SNMP 管理程式通過發佈可指定物件唯一識別碼的 GET 與 GETNEXT 指令來存取物件。代理程式取得指定物件的值，並將其傳輸至 SNMP 管理程式。加入至日誌的事件如果滿足陷阱過濾條件，便可以產生 SNMP 陷阱。沒有產生陷阱的事件僅作爲一個項目被記錄在維護日誌表格中，由 SNMP 管理程式透過一般的 GET 與 GETNEXT 指令進行存取。

GET 與 SET 訊息由網路管理站 (NMS) 發送至主代理程式。您可以經由管理介面同時使用兩個，或使用其中的一個。

SNMP 以協定資料單元 (PDU) 的形式交換網路資訊。這些單元包含有關儲存於管理裝置上的變數之資訊，如 HTTP 伺服器。這些變數，也稱爲管理物件，具有值和標題，必要時，可以將這些值和標題報告給 NMS。由伺服器發送至 NMS 的協定資料單元被稱爲**陷阱**。在下列章節中將進一步解釋 GET、SET 指令的使用，以及陷阱訊息。

SNMP 陷阱目標

SNMP **陷阱**爲 SNMP 代理程式發送至網路管理站 (NMS) 的訊息。例如，當介面的狀況由工作中變更爲停用，則 SNMP 代理程式會發送一個陷阱。SNMP 代理程式必須瞭解 NMS 的位址，以便發送陷阱。

您可以經由 Sun ONE Application Server 管理介面爲 SNMP 主代理程式配置此陷阱目標。也可以檢視、編輯、移除已經配置的陷阱目標。使用管理介面配置陷阱目標實際上也就是在編輯 CONFIG 檔案。

發生了重要事件之後，伺服器子代理程式便會發送一則訊息或一個陷阱至 NMS。例如：

1. 子代理程式通知主代理程式伺服器已經停止。
2. 主代理程式會發送一則訊息或一個陷阱，將事件報告給 NMS。
3. NMS 經由其網路管理應用程式，用文字或圖形顯示資訊。

請參閱第 154 頁的「安裝 SNMP 主代理程式」，以取得有關設定 SNMP 陷阱連接埠的說明。

SNMP 代理程式團體

SNMP **代理程式團體**由團體字串和指定給指定團體的作業構成。團體字串爲用於網路管理站 (NMS) 名稱的文字字串，SNMP 代理程式使用它來進行授權。這表示 NMS 將訊息發送至代理程式的同時，會隨附發送一個團體字串。

指定的作業爲 `get` 和/或 `set`。SNMP 代理程式然後便可以確認 NMS 是否取得授權來執行 `get`、`set`，或同時執行 `get` 與 `set` 作業，以進行資料交換。當團體字串在 SNMP 封包中發送時，不會隱藏；字串以 ASCII 文字的形式發送。

您可以經由管理介面配置、管理團體字串和每個指定團體允許的作業。請參閱第 154 頁的「安裝 SNMP 主代理程式」，以取得有關設定 SNMP 代理程式團體的說明。

設定 SNMP

通常，若要使用 SNMP，系統上必須安裝並執行一個主代理程式和至少一個子代理程式。必須先安裝主代理程式，然後才能啓用子代理程式。請參閱第 154 頁的「安裝 SNMP 主代理程式」。

由於系統不同，因此，設定 SNMP 的程序也不盡相同。針對各種情形，下表提供要遵循的程序之概論。稍後在本章節中將詳細地描述實際程序。

如果您的伺服器符合這些條件 ...	**... 請遵循這些程序。(在後面的章節中詳細地討論)。**
目前尚未執行任何本端代理程式	1. 啓動主代理程式。 2. 爲系統上安裝的每個伺服器啓用子代理程式。
• 目前正在執行本端代理程式 • 無 SMUX • 無需繼續使用本端代理程式	1. 當您爲管理伺服器安裝了主代理程式之後，請停止本端代理程式。 2. 啓動主代理程式。 3. 爲每個伺服器實例配置 SNMP 子代理程式。
• 目前正在執行本端代理程式 • 無 SMUX • 需要繼續使用本端代理程式	1. 安裝 SNMP 代理程式。 2. 啓動 SNMP 代理程式。 3. 使用主代理程式連接埠號之外的連接埠號，重新啓動本端代理程式。 4. 啓動主代理程式。 5. 爲系統上安裝的每個伺服器啓用子代理程式。

開始之前，應該確認兩個事項：

- 您的系統是否已經在執行 SNMP 代理程式 (您作業系統的原生代理程式) ？
- 如果執行，您的本端 SNMP 代理程式是否支援 SMUX 通訊？

請參閱您的系統說明文件，以取得有關如何確認此資訊的資訊。

注意 變更管理伺服器中的 SNMP 設定，安裝新伺服器或刪除現有的伺服器之後，您必須執行下列步驟：

- (Windows 2000) 重新啓動 Windows SNMP 服務或重新開機。
- (UNIX) 使用管理伺服器重新啓動 SNMP 主代理程式與 SNMP 子代理程式。

本章節論述以下主題：

- 使用 SNMP 代理程式 (UNIX/Linux)
- 安裝 SNMP 主代理程式

使用 SNMP 代理程式 (UNIX/Linux)

已經開始執行本端代理程式之後，需要使用 SNMP 代理程式，應該繼續將其與 Sun ONE Application Server 主代理程式配合使用。在啓動之前，一定要停止本端主代理程式。(請參閱您的系統說明文件，以取得詳細資訊。)

注意 若要使用代理程式，您需要安裝該代理程式，然後啓動它。您也必須使用執行 Sun ONE Application Server 主代理程式的連接埠之外的連接埠，重新啓動原生 SNMP 主代理程式。

本章節包括下列主題：

- 安裝 SNMP 代理程式
- 啓動 SNMP 代理程式
- 重新啓動本端 SNMP 常駐程式

安裝 SNMP 代理程式

如果您的系統已經在執行 SNMP 代理程式，並且您要繼續使用本端 SNMP 常駐程式，請遵循這些章節中的步驟：

1. 安裝 SNMP 主代理程式。請參閱第 154 頁的「安裝 SNMP 主代理程式」。
2. 安裝並啓動 SNMP 代理程式，然後重新啓動本端 SNMP 常駐程式。請參閱第 152 頁的「使用 SNMP 代理程式 (UNIX/Linux)」。
3. 啓動 SNMP 主代理程式。請參閱第 157 頁的「啓用與啓動 SNMP 主代理程式」。
4. 啓用子代理程式。請參閱第 163 頁的「啓用子代理程式」。

若要安裝 SNMP 代理程式，請編輯 CONFIG 檔案 (其位於伺服器根目錄下的 *install_dir*/lib/snmp/sagt 中，您可以使用其他名稱命名該檔案)，以便使其包含 SNMP 常駐程式的偵聽連接埠。代理程式也需要包含 MIB 樹和 SNMP 代理程式要轉寄的陷阱。

以下是 CONFIG 檔案的範例：

```
AGENT AT PORT 1161 WITH COMMUNITY public
SUBTREES 1.3.6.1.2.1.1,
        1.3.6.1.2.1.2,
        1.3.6.1.2.1.3,
        1.3.6.1.2.1.4,
        1.3.6.1.2.1.5,
        1.3.6.1.2.1.6,
        1.3.6.1.2.1.7,
        1.3.6.1.2.1.8
FORWARD ALL TRAPS;
```

啟動 SNMP 代理程式

若要啓動 SNMP 代理程式，請在指令提示下輸入下列內容：

```
# sagt -c CONFIG&
```

重新啟動本端 SNMP 常駐程式

啓動 SNMP 代理程式之後，請於您在 CONFIG 檔案中指定的連接埠處，重新啓動本端 SNMP 常駐程式。

若要重新啓動本端 SNMP 常駐程式，請在指令提示下輸入下列內容：

```
# snmpd -P port_number
```

其中，*port_number* 爲在 CONFIG 檔案中指定的連接埠號。例如，在 Solaris 平台，使用過去提及的 CONFIG 檔案範例中的連接埠，您應該輸入：

```
# snmpd -P 1161
```

安裝 SNMP 主代理程式

注意 除非伺服器正在作爲 root 執行，否則否則您無法使用管理介面安裝與啓動 SNMP 主代理程式。

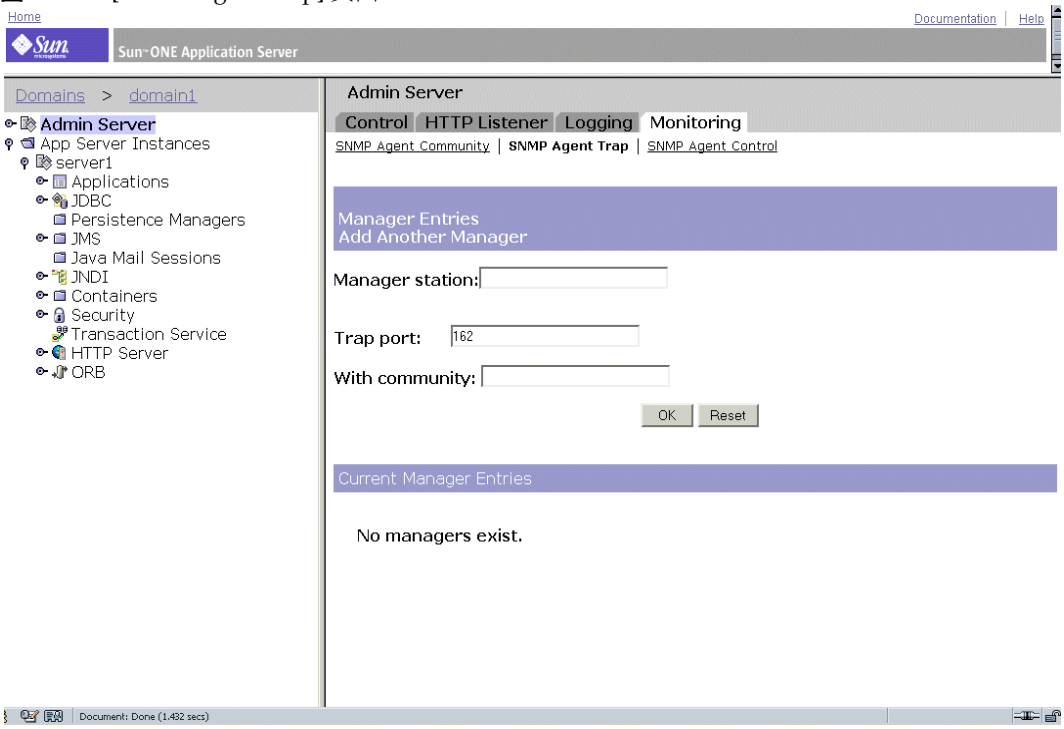
安裝 SNMP 主代理程式的步驟：

1. 以 root 身份登入。
2. 檢查連接埠 161 上是否正在執行 SNMP 常駐程式 (snmpd)。

   如果尚未執行任何 SNMP 常駐程式，請移往步驟 4。

   如果已經執行 SNMP 常駐程式，請確定您瞭解如何重新啓動該常駐程式，並瞭解其所支援的 MIB 樹。
3. 如果 SNMP 常駐程式正在執行，請終止其程序。
4. 在管理介面中，從左窗格內選取 [Admin Server] 節點。
5. 選取 [Monitoring] 標籤，以顯示 [SNMP Agent Trap] 頁面，如下圖所示：

圖 6-2 [SNMP Agent Trap] 頁面

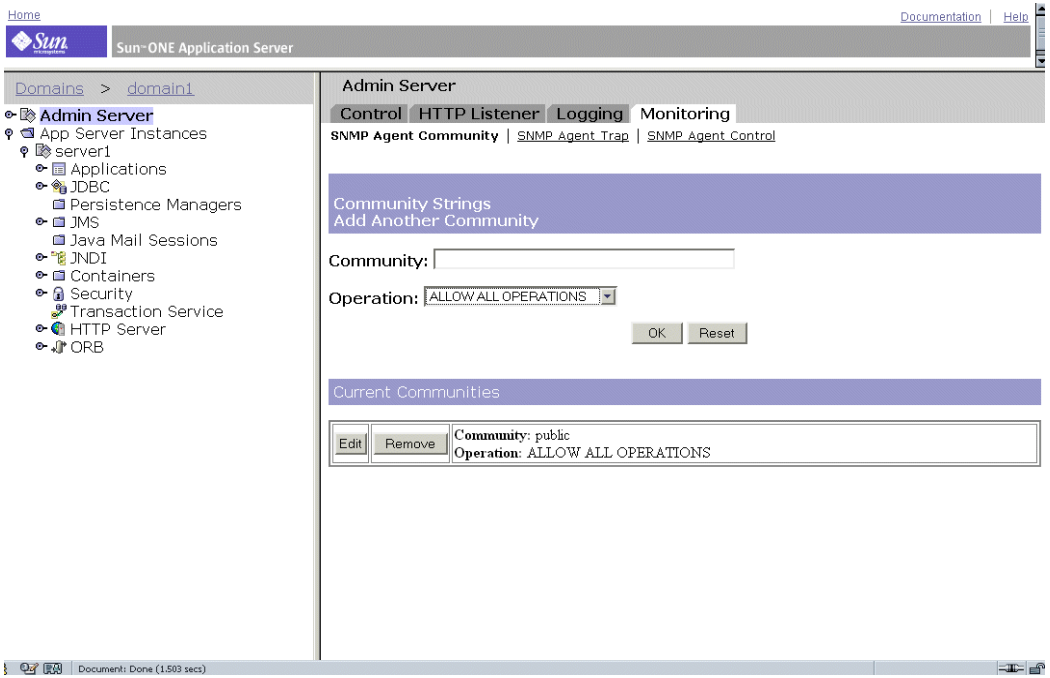


此頁面顯示管理程式項目資訊。

6. 鍵入執行網路管理軟體的系統名稱。
7. 鍵入網路管理系統偵聽陷阱的連接埠號。(眾所周知的連接埠爲 162。) 如需有關陷阱的更多資訊，請參閱第 150 頁的「SNMP 陷阱目標」。
8. 鍵入您要在陷阱中使用的團體字串。如需有關團體字串的更多資訊，請參閱第 151 頁的「SNMP 代理程式團體」。
9. 按一下 [OK]。
10. 按一下 [Monitoring] 標籤中的 [SNMP Agent Community] 連結。

    如下圖所示，螢幕上顯示出團體字串資訊。

圖 6-3 [SNMP Agent Community] 頁面



11. 爲主代理程式鍵入團體字串。
12. 爲團體選擇一個作業層級。

    建立了團體之後，您可以編輯該團體的設定，或經由該頁面上 [Current Communities] 標題中的按鈕將其移除。
13. 按一下 [OK]。
14. 在左窗格中存取 [App Server Instances] 和您的伺服器實例，然後按一下 [Apply Changes]。

啟用與啟動 SNMP 主代理程式

主代理程式作業是在名爲 CONFIG 的代理程式配置檔案中定義的，您可以手動對該檔案進行編輯。您必須先安裝 SNMP 主代理程式，然後才能啓用 SNMP 子代理程式。

注意 在重新啓動主代理程式的時候，如果您收到一個類似於 `System Error:Could not bind to port` 的連結錯誤，請使用 `ps -ef | grep snmp` 檢查是否正在執行 `magt`。如果正在執行，請使用指令 `kill -9` *pid* 結束程序。SNMP 的 CGI 將再次開始工作。

本章節包括下列主題：

- 在其他連接埠上啓動主代理程式
- 手動配置 SNMP 主代理程式
- 編輯主代理程式 CONFIG 檔案
- 定義變數 sysContact 與 sysLocation
- 配置 SNMP 子代理程式
- 啓動 SNMP 主代理程式
- 啓用子代理程式

在其他連接埠上啟動主代理程式

管理介面僅在連接埠 161 上啓動 SNMP 主代理程式。但是，您可以使用下列步驟，在其他連接埠上手動啓動主代理程式：

1. 編輯 *install_dir*`/lib/snmp/magt/CONFIG`，以指定所需的連接埠。
2. 依照下列步驟執行啓動程序檔：

```
cd instance_root/admin-server ./start -shell
install_dir/lib/snmp/magt/magt
install_dir/lib/snmp/magt/CONFIG
install_dir/lib/snmp/magt/INIT
```

然後會在所需的連接埠上啓動主代理程式。但是，管理介面能夠偵測主代理程式是否正在執行。

手動配置 SNMP 主代理程式

手動配置 SNMP 主代理程式的步驟：

1. 以 root 身份登入。
2. 檢查連接埠 161 上是否正在執行 SNMP 常駐程式 (`snmpd`)。

   如果已經執行 SNMP 常駐程式，請確定您瞭解如何重新啓動該常駐程式，並瞭解其所支援的 MIB 樹。然後終止其程序。
3. 編輯 `CONFIG` 檔案，其位於伺服器根目錄下的 `lib/snmp/magt` 中。
4. (可選) 依照第 158 頁的「定義變數 sysContact 與 sysLocation」中的描述，在 `CONFIG` 檔案內定義變數 `sysContact` 與 `sysLocation`。

編輯主代理程式 CONFIG 檔案

`CONFIG` 檔案定義將與主代理程式配合使用的團體和管理程式。管理程式的值應爲有效的系統名稱或 IP 位址。

以下是基本 `CONFIG` 檔案的範例：

```
COMMUNITY           public
                    ALLOW ALL OPERATIONS

MANAGER             manager_station_name
                    SEND ALL TRAPS TO PORT 162
                     WITH COMMUNITY public
```

定義變數 sysContact 與 sysLocation

您可以編輯 `CONFIG` 檔案來爲 `sysContact` 與 `sysLocation` 加入初始值，這些值用於指定 `sysContact` 與 `sysLocation` MIB-II 變數。在本範例中，`sysContact` 與 `sysLocation` 的字串均用引號括住。任何含有空格、行中斷、標籤等等的字串均必須用引號括住。您也可以使用十六進制表示法指定值。

以下爲定義了變數 `sysContact` 與 `sysLocation` 的 `CONFIG` 檔案的範例：

```
COMMUNITY           public
                    ALLOW ALL OPERATIONS

MANAGER             nms2
                    SEND ALL TRAPS TO PORT 162
                    WITH COMMUNITY public

INITIAL             sysLocation "Server room
901 San Antonio Road
Palo Alto CA 94303
USA"

INITIAL             sysContact "John Doe
email: jdoe@sun.com"
```

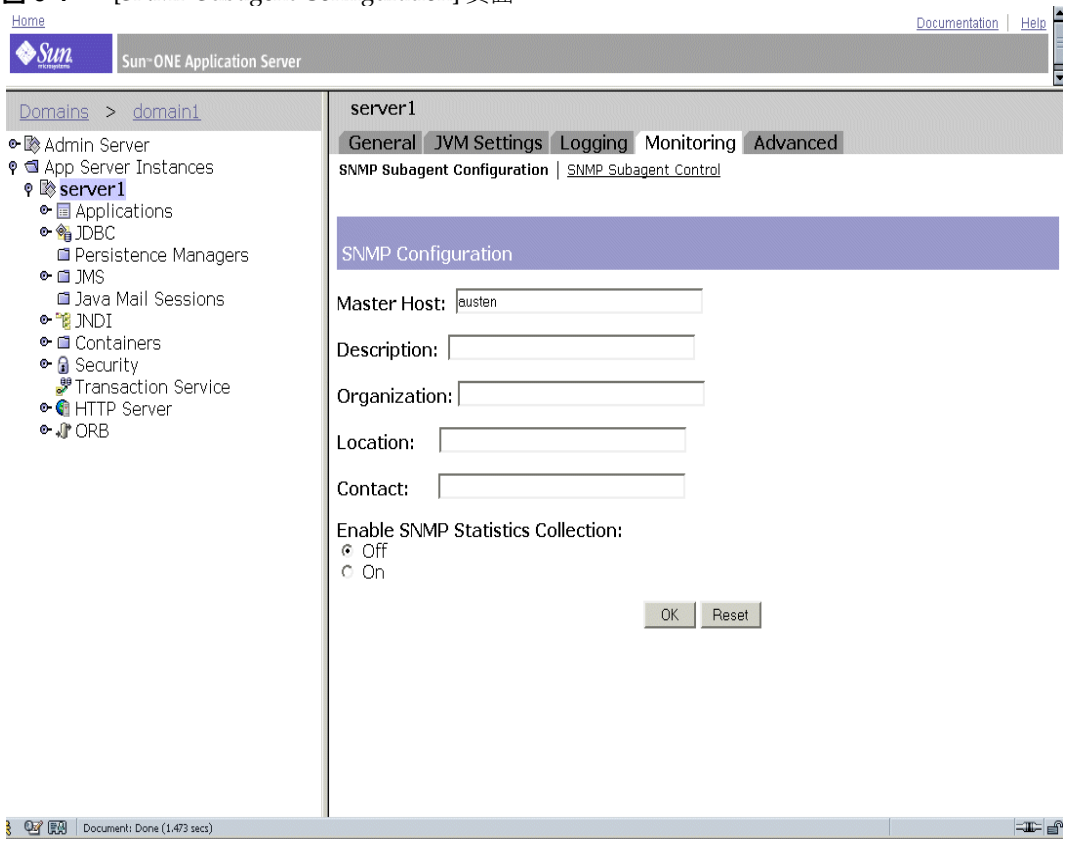
配置 SNMP 子代理程式

若要配置 SNMP 子代理程式，請執行下列步驟：

1. 從 [Admin Server] 中，選取左窗格內的 [server instance] 節點。
2. 從右窗格中，選取 [Monitoring] 標籤。
3. 選取 [SNMP Subagent Configuration] 連結。

   螢幕上會顯示以下頁面：

圖 6-4 [SNMP Subagent Configuration] 頁面



4. (僅用於 UNIX) 在 [Master Host] 欄位中輸入伺服器的名稱與領域。
5. 在 [Description] 欄位中，輸入伺服器的描述，包含作業系統資訊。
6. 在 [Organization] 欄位中，輸入負責伺服器的組織。
7. 在 [Location] 欄位中，輸入伺服器實例的位置。
8. 在 [Contact] 欄位中，輸入伺服器負責人的姓名以及該負責人的聯絡資訊。
9. 選取 [On] 以啓用 SNMP 統計資料集合。
10. 按一下 [OK]。
11. 在左窗格中存取 [App Server Instances] 和您的伺服器實例，然後按一下 [Apply Changes]。

啟動 SNMP 主代理程式

安裝了 SNMP 主代理程式之後，便可以手動啓動它，或透過使用管理介面中的管理伺服器來啓動它。

手動啟動 SNMP 主代理程式

若要手動啓動主代理程式，請在指令提示下輸入下列內容：

```
# magt CONFIG INIT&
```

INIT 檔案是一個非揮發性檔案，其包含 MIB-II 系統群組的資訊，包括系統位置和聯絡資訊。如果 INIT 檔案不存在，則在首次啓動主代理程式的時候會建立該檔案。

注意 CONFIG 檔案中的無效管理程式名稱會導致主代理程式啓動失敗。

若要在非標準的連接埠上啓動主代理程式，請使用下列兩種方法之一：

方法 1：在 CONFIG 檔案中，爲每個介面指定一個傳輸對映，在此對映上，主代理程式經由管理程式偵聽 SNMP 請求。傳輸對映允許主代理程式接受在標準連接埠與非標準連接埠處的連線。主代理程式也可以接受非標準連接埠的 SNMP 流量。目標系統對開放式套接字數目或每個程序中描述元數目的限定會限定 SNMP 的最大並行數目。以下爲傳輸對映項目的範例：

```
TRANSPORT           extraordinary   SNMP
                    OVER UDP SOCKET
                     AT PORT 11161
```

手動編輯 CONFIG 檔案之後，您應該在指令提示下鍵入以下內容來手動啓動主代理程式：

```
# magt CONFIG INIT&
```

方法 2：編輯 /etc/services 檔案，以允許主代理程式接受在標準連接埠和非標準連接埠處的連線。

使用管理伺服器啟動 SNMP 主代理程式

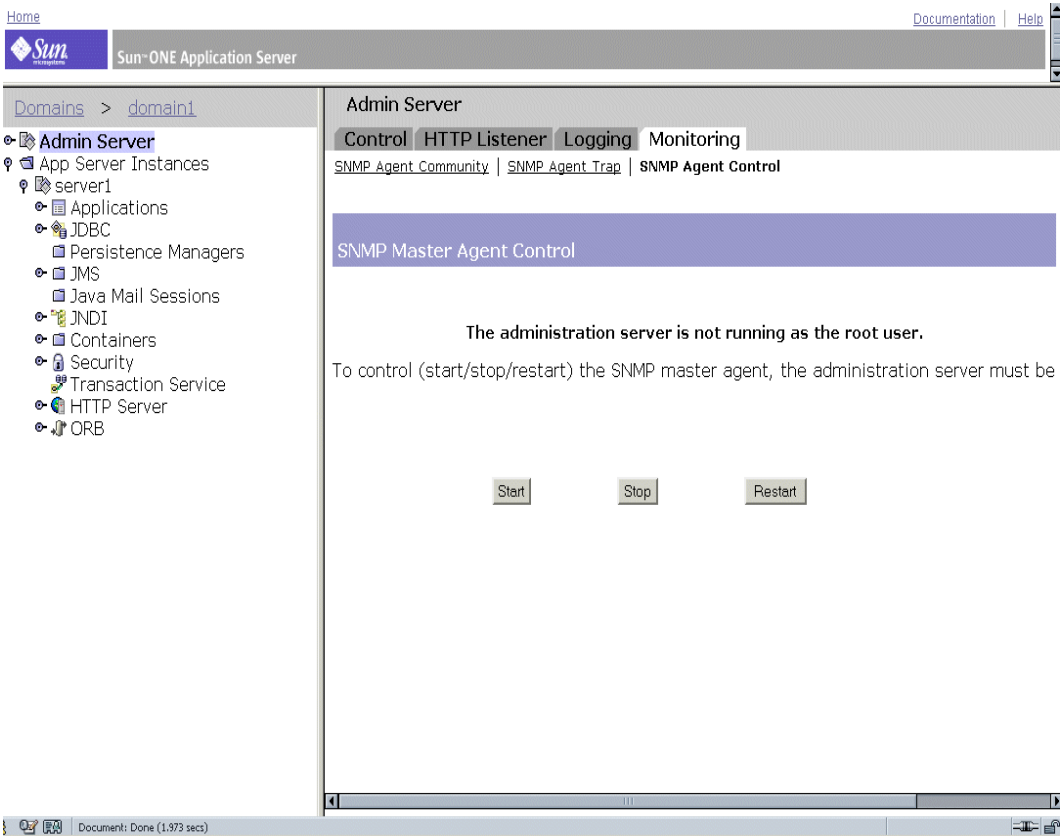
若要使用管理伺服器啓動 SNMP 主代理程式，請執行下列步驟：

注意 您必須以 root 身份登入 Sun ONE Application Server，以啓動 SNMP 主代理程式。

1. 登入管理伺服器。
2. 從左窗格中的 [Admin Server] 節點中，選擇 [Monitoring] 標籤。
3. 選擇右窗格頂端旁邊的 [SNMP Agent Control] 連結。

   螢幕上會顯示以下頁面。

圖 6-5 [SNMP Agent Control] 頁面



4. 按一下 [Start]。

   您也可以經由 SNMP 代理程式控制頁面，停止與重新啓動 SNMP 主代理程式。

啟用子代理程式

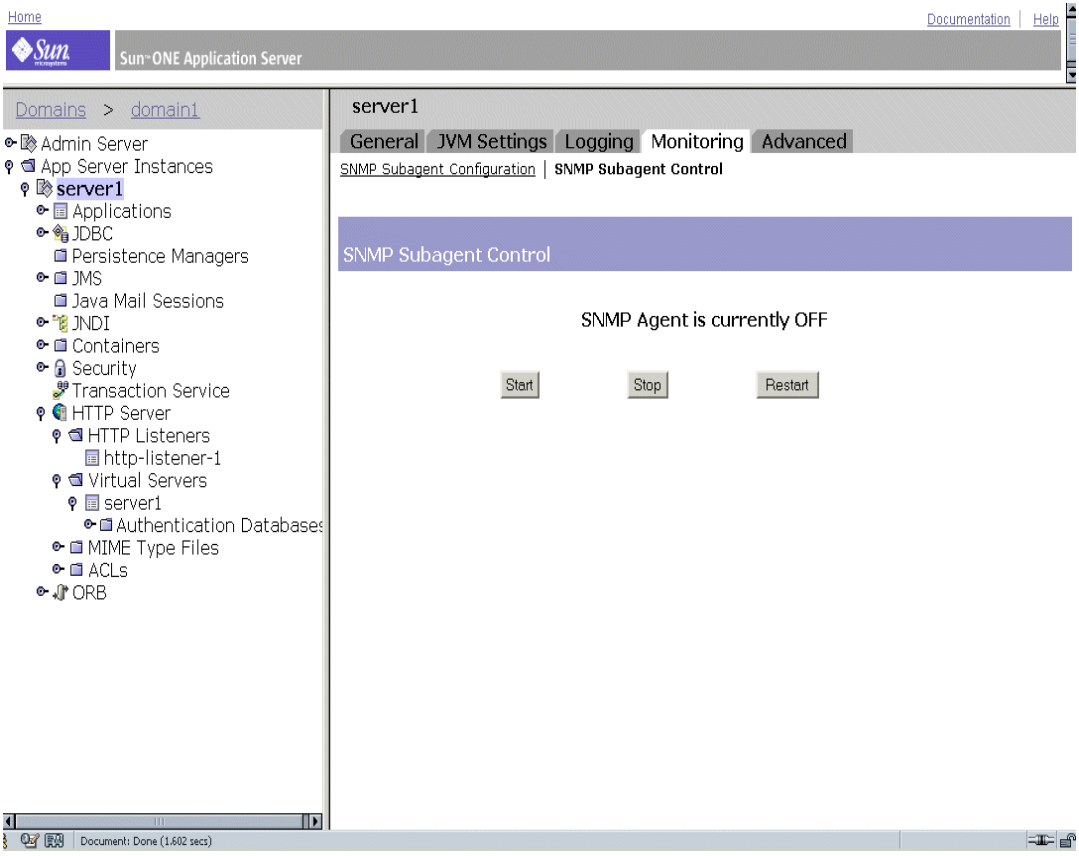
安裝了隨附管理伺服器的主代理程式之後，如果要嘗試啓動該主代理程式，您必須首先啓用伺服器實例的子代理程式。如需有關安裝主代理程式的資訊，請參閱第 154 頁的「安裝 SNMP 主代理程式」。

您可以使用子代理程式停止 UNIX/Linux 平台上的 SNMP 功能。您必須首先停止子代理程式，然後才能停止主代理程式。如果首先停止了主代理程式，便可能無法停止子代理程式。發生此情況後，請重新啓動主代理程式，然後停止子代理程式，接著停止主代理程式。

啓用 SNMP 子代理程式的步驟：

1. 請在左窗格中展開 [App Server Instances] 節點。
2. 選取伺服器實例，然後按一下 [Monitoring] 標籤。
3. 選取 [SNMP Subagent Control] 選項，以顯示下圖中展示的頁面。

圖 6-6 [SNMP Subagent Control] 頁面



經由此頁面，您可以啓動、停止或重新啓動 SNMP 子代理程式。控制按鈕上方會顯示子代理程式的狀況。

在 Windows 平台上，Windows SNMP 服務用於監視 Sun ONE Application Server；可以經由 [控制台]/[系統管理工具]/[服務] 選項來控制該服務。

注意 變更 SNMP 配置之後，您必須按一下 [OK]，然後經由 [SNMP Subagent Control] 頁面重新啓動 SNMP 子代理程式。

第 7 章

配置 Web 伺服器外掛程式

本章解釋 Sun ONE Application Server 如何處理超文件傳輸協定 (HTTP) 請求，以及如何藉由 Sun ONE Application Server 配置和使用 Web 伺服器外掛程式。還解釋如何藉由 Microsoft IIS 與 Apache Web 伺服器，配置和使用 Web 伺服器外掛程式。

本章包含以下主題：

- 關於 Web 伺服器外掛程式
- 處理用戶端請求
- Web 伺服器外掛程式配置
- Web 伺服器外掛程式 SAF 參考
- 使用 Web 伺服器外掛程式
- 配置 Microsoft IIS 以使用 Web 伺服器外掛程式
- 配置 Apache Web 伺服器

關於 Web 伺服器外掛程式

Web 伺服器外掛程式是 HTTP 反向代理外掛程式，可讓您指導 Sun ONE Web Server 或 Sun ONE Application Server 將特定的 HTTP 請求轉寄到另一伺服器中。例如，您可以配置連線至網際網路的 Web 伺服器，將針對特定 Web 應用程式的請求轉寄給受公司防火牆保護的應用程式伺服器。

在 Sun ONE Application Server 內，Web 伺服器外掛程式可讓一個伺服器實例將 HTTP (Web) 請求轉寄給另一個伺服器實例。

Web 伺服器外掛程式執行下列功能：

- 一旦可能，將重新使用代理伺服器的連線。這便無需開啓新連線以處理進來的請求。
- Web 伺服器外掛程式在開始接收請求時，便開始串流請求與回應。換句話說，外掛程式不會等到請求或回應全部收集完畢後，再將其轉寄給遠端伺服器。
- 在適當的時候，Web 伺服器外掛程式會維持與同一個遠端伺服器的多重外送 HTTP 連線。針對由 Web 伺服器外掛程式轉寄的請求而形成的連線稱爲外送 HTTP 連線。

若要瞭解 Web 伺服器外掛程式的工作方式，則有必要瞭解 HTTP 請求的基本原理，特別是 Sun ONE Application Server 處理 HTTP 請求的方法。

處理用戶端請求

Sun ONE Application Server 是可以直接接受和回應 HTTP 請求的應用程式伺服器。在本節中，我們將論述 HTTP 基本原理並查看 Sun ONE Application Server 處理請求的方式。本章節涵蓋以下主題：

- HTTP 基本原理
- 請求處理程序中的步驟

HTTP 基本原理

作爲快速摘要，HTTP/1.1 協定的工作方式如下：

- 用戶端 (通常爲瀏覽器) 開啓與伺服器的連線，並發送請求。
- 伺服器會處理請求，產生回應，如果發現 `Connection: Close` 標頭，則關閉連線。

請求包含指示一種方法 (如 `GET` 或 `POST`) 的行、指示所請求資源的一致性資源識別碼 (URI)，以及用空格分隔的 HTTP 協定版本。

其後面通常跟隨一些標頭、指示標頭結束的空白行，有時還有內文資料。標頭可能提供關於請求或用戶端內文資料的各種資訊。一般而言，僅針對 `POST` 與 `PUT` 方法發送標頭。

下面顯示的請求範例將由瀏覽器發送，以請求伺服器 `foo.com` 發回 */index.html* 中的資源。在該範例中，由於方法爲 `GET` (請求要點是取得某些資料而非發送資料)，因此不發送內文資料。

```
GET /index.html HTTP/1.0

User-agent: Mozilla

Accept: text/html, text/plain, image/jpeg, image/gif, */*

Host: foo.com
```

伺服器收到請求並進行處理。雖然伺服器可以同時處理多個請求，但其仍個別地處理每個請求。每個請求的處理都分爲連續的多個步驟，一起組成請求處理程序。

伺服器會產生回應，回應包括 HTTP 協定版本、HTTP 狀況碼以及用空格分隔的原因片語。回應後面，通常跟隨一些標頭。標頭結尾用空白行表示。回應的內文資料隨之出現。典型 HTTP 回應之外觀如下所示：

```
HTTP/1.0 200 OK
Server: Standard/7.0
Content-type: text/html
Content-length: 83

<HTML>
<HEAD><TITLE>Hello World</Title></HEAD>
<BODY>Hello World</BODY>
</HTML>
```

狀況碼和原因片語可告知用戶端伺服器是如何處理請求的。通常會傳回狀況碼 200，指示請求處理成功以及內文資料包含請求的項目。其他結果碼指示重新導向至另一個伺服器或瀏覽器的快取記憶體，或各種類型的 HTTP 錯誤 (如「404 未找到」)。

請求處理程序中的步驟

Sun ONE Application Server 初次啓動時，它會執行某些初始化工作，然後等待用戶端 (如瀏覽器) 發出 HTTP 請求。收到請求後，它首先會選取一個虛擬伺服器。

一旦選取了虛擬伺服器，虛擬伺服器的 `obj.conf` 檔案將指定以下列步驟處理請求的方式：

1. **AuthTrans** (授權轉換)

   確認請求中發送的任何授權資訊 (如名稱和密碼)。

2. **NameTrans** (名稱轉換)

   將邏輯 URI 轉換爲本機檔案系統路徑。

3. **PathCheck** (路徑檢查)

   檢查本機檔案系統路徑的有效性，並檢查請求者是否擁有存取檔案系統上所請求資源的權限。

4. **ObjectType** (物件類型)

   決定所請求資源的 MIME 類型 (多用途網際網路郵件編碼) (例如，`text/html`、`image/gif` 及其他)。

5. **Service** (產生回應)

   產生回應並將其傳回至用戶端。

6. **AddLog** (加入日誌項目)

   加入項目至日誌檔。

7. **Error** (服務)

   僅當前面的步驟中發生錯誤時，才會執行此步驟。如果發生錯誤，伺服器會記錄錯誤訊息並中斷處理程序。

Web 伺服器外掛程式配置

Web 伺服器外掛程式的配置和運作方式由一組配置檔案決定。Sun ONE Application Server 每次處理來自用戶端的請求時，都會查看在這些檔案中定義的配置。配置檔案命名爲 `obj.conf` 與 `init.conf`。`obj.conf` 檔案以虛擬伺服器名稱作爲字首，如 `server1-obj.conf`。如需更多資訊，請參閱第 341 頁的「obj.conf 檔案」。

Sun ONE Application Server 的每個實例都有自己的 `init.conf` 檔案，在該伺服器啓動時將參考此檔案。

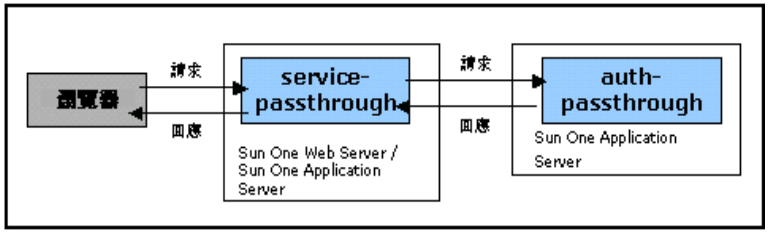
如前面主題所述，`obj.conf` 配置檔案包含一系列說明 (指令)，用以告知 Sun ONE Application Server 在用戶端請求與回應程序的每個階段應該做的工作。每個指令均可呼叫伺服器應用程式功能 (SAF)。

`obj.conf` 檔案對於 Sun ONE Application Server 的作業是不可缺少的。若透過管理介面變更伺服器，系統會自動更新 `obj.conf`。

`init.conf` 配置檔案可以設定於初始化期間用來配置伺服器的變數值。在伺服器啓動期間，伺服器將執行在該檔案中指定的配置參數。請參閱「*Sun ONE Application Server Configuration File Reference*」，以取得更多資訊。

下圖闡明 Web 瀏覽器、前端 Web 伺服器、後端應用程式伺服器與 Web 伺服器外掛程式的 `service-passthrough` 及 `auth-passthrough` SAF 之間的關係：

圖 7-1 Web 瀏覽器、Web 伺服器、應用程式伺服器與 Web 伺服器外掛程式 SAF 之間的關係



Web 伺服器外掛程式 SAF 參考

本節論述下列伺服器應用程式功能 (SAF) 的功能和行爲：

- init-passthrough
- auth-passthrough
- service-passthrough
- check-passthrough

init-passthrough

`init-passthrough` 功能初始化 Web 伺服器外掛程式。必須先呼叫該功能，然後才能使用 Web 伺服器外掛程式。

範例：

```
Init fn="load-modules" shlib="c:/plugins/passthrough.dll"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-
passthrough" NativeThread="no"

Init fn="init-passthrough"
```

auth-passthrough

`auth-passthrough` SAF 適用於 `AuthTrans-class` 指令。

auth-passthrough 功能可檢查進來的 HTTP (web) 請求，以取得以中間伺服器上執行的 service-passthrough 功能編碼的用戶端資訊。用戶端資訊包括：

- IP 位址請求來源
- 發源用戶端使用的 SSL 鍵值大小
- 發源用戶端提供的 SSL 用戶端證書

若 auth-passthrough 偵測到已編碼的用戶端資訊，它會把請求視爲來自發源用戶端的直接請求，而不視爲由執行 `service-passthrough` 的中間伺服器轉寄的請求。

這在兩層部署方案中很有用；

- 其中，Sun ONE Application Server 實例被公司防火牆後面的第二個防火牆所隱藏。
- 不允許任何用戶端直接連線至 S1AS 實例。

在這種網路架構中，用戶端永遠連線至執行代理外掛程式的前端 Web 伺服器。此 Web 伺服器是將請求轉寄至 Sun ONE Application Server 的伺服器。這表示 Sun ONE Application Server 僅能從代理主機 (在此情況下爲 Web 伺服器) 接收請求，而永遠不能直接從用戶端主機接收。這意味著如果應用程式 (部署於兩個防火牆後面的 Sun ONE Application Server 實例上) 查詢用戶端資訊 (如用戶端 IP 位址)，應用程式將取得代理主機 IP (由於它是針對中繼請求的實際發源主機)。`auth-passthrough` SAF 可用於修改此行爲，以便提供遠端 (代理的) 用戶端資訊。

由於 `auth-passthrough` 可能置換可以用於認證的資訊 (例如，產生請求的 IP 位址)，因此，必須僅使用可信賴的用戶端或伺服器連線至執行 `auth-passthrough` 的伺服器。作爲預防措施，建議僅使用公司防火牆後面的伺服器執行 `auth-passthrough`。可透過網際網路存取的伺服器不應該執行 `auth-passthrough` SAF。僅當需要關於發源用戶端的相關資訊時，才應該使用 `auth-passthrough` SAF。

請注意，在上述方案中，SSL 用戶端認證只能對 Web 伺服器 「開啓」，對應用程式伺服器永遠 「關閉」，如此才可讓配置正常執行。

指令範例：

```
AuthTrans fn="auth-passthrough"
```

service-passthrough

`service-passthrough SAF` 適用於 Service-class 指令。

`service-passthrough` SAF 可將請求從一個伺服器轉寄至另一個伺服器，以進行處理。可以配置 `service-passthrough` SAF 使用針對遠端伺服器的 SSL 或非 SSL (HTTPS 或 HTTP) 連線，而不管用來接收原始請求的連線類型如何。`service-passthrough` SAF 可以對發源用戶端的相關資訊進行編碼 (遠端伺服器上執行的 `auth-passthrough` 功能可能會對其解碼)。

`service-passthrough` 指令一般與 `obj.conf` 配置檔案中的其他指令結合起來使用，如下所述：

```
<Object name="passthrough">

ObjectType fn="force-type" type="magnus-internal/passthrough"

Error reason="Bad Gateway" fn="send-error"
uri="$docroot/badgateway.html"
```

```
</Object>

<Object name="default">

....

NameTrans fn="assign-name" from="(/webapp1|/webapp1/*)" name="passthrough"

...

</Object>
```

如果後端應用程式伺服器當機，使用者將看到本機 HTML 檔案 `badgateway.html`。假使執行 `service-passthrough` SAF 的伺服器需要提供其可存取的檔案，並且僅僅將拒絕的請求轉寄給後端應用程式伺服器，則 `ObjectType` 行會變更爲：

```
ObjectType fn="check-passthrough" type="magnus-internal/passthrough"
```

check-passthrough

`check-passthrough` SAF 適用於 `ObjectType-class` 指令。

check-passthrough 功能用於檢查在本機伺服器上可否取得所請求的資源 (例如 HTML 文件或 GIF 影像)。如果所請求的資源在本機不存在，`check-passthrough` SAF 將設定類型以指示應該將請求傳送至另一個伺服器，以便由 `service-passthrough` SAF 處理。

參數：

`type` - (可選) 當請求資源不存在時要設定的類型。預設值爲「`magnus-internal/passthrough`」。

範例

```
ObjectType fn="check-passthrough"
```

使用 Web 伺服器外掛程式

若要在 Sun ONE Web Server 上使用 Web 伺服器外掛程式，您必須變更 Sun ONE Application Server 和 Sun ONE Web Server 的配置檔案。請遵循本節中所列的程序配置並使用 Sun ONE Web Server 外掛程式：

- 對 Sun ONE Web Server 的變更
- 對 Sun ONE Application Server 的變更

對 Sun ONE Web Server 的變更

備份重要的配置檔案，例如 `magnus.conf` 和 `obj.conf`，然後再對其進行變更。

1. 在 Web 伺服器安裝區域中建立一個目錄，以包含通道外掛程式。 例如：

   `cd /`*`webserver_install_dir`*`/plugins`

   `mkdir -p passthrough/bin`

2. 將 Sun ONE Application Server 的通道外掛程式複製到這個新的 Web 伺服器目錄。例如：

   `cd` *`appserver_install_dir`*`/lib`

   `cp libpassthrough.so` *`webserver_install_dir`*`/plugins/passthrough/bin`

   對於 Windows，請複製 `passthrough.dll` 檔案。

3. 編輯 `magnus.conf` 檔案 (位於 *webserver_install_dir/https-host.domain*`/config` 下)，加入以下行：

```
Init fn="load-modules"
shlib="webserver_install_dir/plugins/passthrough/bin/libpassthrough.so"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough"
NativeThread="no"

Init fn="NSServletEarlyInit" EarlyInit=yes

Init fn="NSServletLateInit"  LateInit=yes

Init fn="init-passthrough"
```

4. 編輯 `obj.conf` 檔案 (位於 *webserver_install_dir/https-host.domain*`/config` 下)，並增加 `NameTrans` 指令，如下所示：

```
<Object name=default>
NameTrans fn="NSServletNameTrans" name="servlet"
NameTrans fn="assign-name" from="/*" name="passthrough"
</Object>
```

from="/*" URI 是在遠端伺服器上部署的 Web 應用程式的環境根，通道對應於 obj.conf 中 <Object> 的名稱。

例如：

```
<Object name="default">
...
NameTrans fn="assign-name" from="(/webapp1|/webapp1/*)"
name="passthrough"
...
</Object>
```

5. 在 obj.conf 檔案中增加以下行：

```
<Object name="passthrough">
ObjectType fn="force-type" type="magnus-internal/passthrough"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/passthrough" fn="service-passthrough"
servers="http://servername:port"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>
```

6. 重新啓動 Sun ONE Web Server 實例。

下一步，您必須配置 Sun ONE Web Server，以將請求路由至 Web 伺服器外掛程式。

對 Sun ONE Application Server 的變更

備份重要的配置檔案，例如 magnus.conf 和 obj.conf，然後再對其進行變更。

1. 編輯 *install_dir*/domains/domain1/server1/config/init.conf 檔案，加入以下行：

在 UNIX 上：

```
Init fn="load-modules"
shlib="webserver_install_dir/plugins/passthrough/bin/libpassthrough.so"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough"
NativeThread="no"
Init fn="init-passthrough"
```

在 Windows 上：

```
Init fn="load-modules" shlib="c:/install_dir/bin/passthrough.dll"
funcs="init-passthrough,auth-passthrough,check-passthrough, service-passthrough"
NativeThread="no"
Init fn="init-passthrough"
```

2. 編輯 *install_dir*/domains/domain1/server1/config/server1-obj.conf，並增加 AuthTrans 指令，如下所示：

```
<Object name="default">
AuthTrans fn="match-browser" browser="*MSIE*" ssl-unclean-shutdown="true"
AuthTrans fn="auth-passthrough"
fn="service-passthrough" servers="server"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>
```

伺服器是下列格式的 URL ：

http://*servername*:*port*

3. 重新啓動 Sun ONE Application Server 實例。

這將完成您需要對 Sun ONE Application Server 所作的變更。先前說明過的程序是配置應用程式伺服器的單一實例。若要啓動使用 Web 伺服器外掛程式的更多實例，您必須在這些應用程式伺服器實例的配置檔案中進行這些變更。

注意 對於 Solaris 和 Linux，外掛程式庫名稱將爲 libpassthrough.so。對於 Windows，外掛程式庫名稱將爲 passthrough.dll

配置 Microsoft IIS 以使用 Web 伺服器外掛程式

配置 Microsoft 網際網路資訊服務以使用 Web 伺服器外掛程式，包括配置 Web 伺服器外掛程式以用於 Microsoft IIS，以及配置 Microsoft IIS 以使用 Web 伺服器外掛程式。

您也可以配置伺服器儲存區以處理在不同伺服器上執行的多重應用程式。

本章節涵蓋以下主題：

- 配置用於 IIS 的 Web 伺服器外掛程式
- 配置 IIS 以使用 Web 伺服器外掛程式
- 配置多重伺服器儲存區
- sun-passthrough.properties 檔案範例

配置用於 IIS 的 Web 伺服器外掛程式

若要配置用於 IIS 的 Web 伺服器外掛程式，請執行下列工作：

1. 於 IIS `wwwroot` 目錄下，爲 Web 伺服器外掛程式建立一個目錄，方法爲從 C:\ 指令行提示處鍵入以下指令：

   ```
   md \Inetpub\wwwroot\sun-passthrough
   ```

2. 將外掛程式檔案複製到 `C:\Inetpub\wwwroot\sun-passthrough` 目錄下。

3. 使用文字編輯程式，將機器 (已安裝 Sun ONE Application Server) 的 URL 加入至 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 檔案。

   您需要藉由文字編輯程式加入以下資訊：

   `server=http://`*`appservername`*`:`*`port`*

   其中，*appservername* 是機器 (已安裝 Sun ONE Application Server) 的主機名稱或 IP 位址，`port` 是機器進行偵聽時所在連接埠的號碼 (該值通常設定爲 80)。

4. 在 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 檔案中，列示您想讓 Sun ONE Application Server 提供的環境根。

   這些環境根應該與 Sun ONE Application Server 上部署的應用程式環境根相符。對這些環境根的請求將由 Sun ONE Application Server 接受並處理，而其他請求由 IIS Web 伺服器處理。將請求傳送給 Web 應用程式的指令行爲：

   `passthrough=/`*`webapplication`*

   其中，*/webapplication* 是 Web 應用程式的環境根。若要將所有請求傳送至 Sun ONE Application Server，請增加以下行：

   ```
   passthrough=/
   ```

現在，您已在 Microsoft IIS 根目錄中配置了 Web 伺服器外掛程式。若要完成程序，您現在需要配置 Microsoft IIS 使用 Web 伺服器外掛程式。

配置 IIS 以使用 Web 伺服器外掛程式

若要配置 IIS 使用 Web 伺服器外掛程式，您需要開啓 Windows 網際網路服務管理程式。網際網路服務管理程式位於 [控制台] 資料夾內的 [管理工具] 資料夾中。

開啓網際網路服務管理程式，並執行下列工作：

1. 選取您要爲之啓用外掛程式的網站。該網站通常被命名爲「預設網站」。
2. 以滑鼠右鍵按一下網站，選取 [內容] 以開啓 [內容] 筆記本。
3. 開啓 [ISAPI 過濾器] 標籤，按一下 [新增] 按鈕，並執行以下步驟，以加入新的 ISAPI 過濾器：
   a. 在 [過濾器名稱] 欄位中，輸入 Sun ONE Application Server
   b. 在 [可執行] 欄位中，鍵入 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll
   c. 按一下 [確定]，關閉 [內容] 筆記本。
4. 您現在需要建立和配置新的虛擬目錄。執行以下步驟以建立和配置新的虛擬目錄：
   a. 以滑鼠右鍵按一下預設網站，選取 [新建]，然後選取 [虛擬目錄]。[虛擬目錄建立精靈] 將開啓。
   b. 在 [別名] 欄位中，鍵入 `sun-passthrough`。
   c. 在 [目錄] 欄位中，鍵入 `C:\Inetpub\wwwroot\sun-passthrough`。
   d. 請確定您核取了 [執行許可權] 核取方塊，並且所有其他許可權核取方塊均未核取。
   e. 按一下 [完成]。
5. 您需要停止並啓動 Web 伺服器，才能讓新設定生效。若要停止 Web 伺服器，請以滑鼠右鍵按一下網站並選取 [停止]。若要啓動 Web 伺服器，請以滑鼠右鍵按一下網站並選取 [啓動]。

   然後，在 Web 瀏覽器中鍵入以下內容，以存取 Web 應用程式環境根：

   `http://`*`webservername`*`/`*`webapplication`*

   其中，*webservername* 是 Web 伺服器的主機名稱或 IP 位址，*/webapplication* 是您在 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 檔案中列示的環境根，以確認 Web 伺服器、Web 伺服器外掛程式以及 Sun ONE Application Server 正常作業。

配置多重伺服器儲存區

在 sun-passthrough.properties 檔案中配置伺服器儲存區，便可能跨多重應用程式伺服器分割 Web 應用程式 (即，您在一組伺服器上執行某些應用程式，在另一組伺服器上執行其他應用程式)。對於每個伺服器儲存區，請選擇唯一名稱，包含字母和數字。一旦您完成爲 Microsoft IIS 安裝和配置 Web 伺服器外掛程式的步驟，如第 175 頁的「配置 Microsoft IIS 以使用 Web 伺服器外掛程式」一節所述，便可編輯 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties 檔案，並將您爲伺服器儲存區選擇的唯一名稱作爲相關伺服器和 passthrough 特性行的字首。在伺服器儲存區名稱後放置一個小數點 (.)。

例如，sun-passthrough.properties 檔案中的下列行可定義兩個伺服器儲存區。第一個伺服器儲存區包含 server-a 以及對環境根 /app1 的服務請求。第二個伺服器儲存區包含 server-b 以及對環境根 /app2 與 /app3 的服務請求。

```
server=http://server-a

passthrough=/app1

serverpool2.server=http://server-b

serverpool2.passthrough=/app2

serverpool2.passthrough=/app
```

sun-passthrough.properties 檔案範例

```
# Sun ONE Application Server web server plugin for IIS
#
# This file is used to configure the Sun ONE Application Server web server
# plugin for IIS.  Lines beginning with a '#' are ignored.
# server
#
# The server property specifies the URL of an application server.  If multiple
# server properties are given, the plugin will distribute load across the
# specified application servers.
#
server=http://localhost:8080
# passthrough
#
```

```
# The passthrough property specifies the context root (virtual directory) of a
# web application.  Requests for the given context root will be passed to the
# application server for processing.  If 'passthrough=/' is specified, all
# requests will be passed to the application server for processing.
#
# passthrough properties should be ordered from most to least specific.  For
# example, 'passthrough=/apps/app1' should appear before 'passthrough=/apps'.
#
# Multiple passthrough properties are allowed.
#
#passthrough=/webapp
#passthrough=/servlets
#passthrough=*.jsp
passthrough=/
# prefix
#
# The prefix property specifies the IIS virtual directory that contains the
# plugin DLL, sun-passthrough.dll.
#
prefix=/sun-passthrough
# error-url
#
# The error-url property specifies the URL of a page to redirect the client to
# when the application server is unavailable.
#
#error-url=/badgateway.htm
# It is possible to configure multiple server pools by prefixing the server
# and passthrough property names with a pool name followed by a period ('.').
# Pool names can be any sequence of letters and numbers.
#
# For example, the following properties define two server pools.  One server
```

```
# pool will service the web applications at '/app1' and the other will service
# the web applications at '/app2' and '/app3':
#
#serverpool1.server=http://server-a
#serverpool1.passthrough=/app1
#
#serverpool2.server=http://server-b
#serverpool2.passthrough=/app2
#serverpool2.passthrough=/app3
```

配置 Apache Web 伺服器

本節說明您如何才能編譯 Apache 來源碼以及配置 Apache Web 伺服器的安裝，以使用 Sun ONE Application Server。

若要啓動要傳送至 Sun ONE Application Server 的 HTTP 請求，您必須編譯 Apache 來源，以使用 `mod_proxy` 模組，然後修改 `httpd.conf` 檔案。

本節包含有關以下程序的資訊：

- 最低需求
- 使用 mod_proxy 模組編譯 Apache
- 修改 httpd.conf 檔案
- 啓動與停止 Apache

最低需求

您必須滿足以下需求，才能成功編譯 Apache Web 伺服器並使用 `mod_proxy` 外掛程式。

- `apache_1.3.27` (來源)。
- Sun 的 C 編譯器 (適用於 Solaris 8 和 9)。
- Sun ONE Application Server 7 可安裝程式。
- 在 Solaris 8 和 9 上，確定 `CC` 與 `make` 位於 `PATH` 中。

使用 mod_proxy 模組編譯 Apache

1. 從 www.apache.org 下載帶有內建 mod_proxy 模組的最新 Apache 來源分配。

   解壓縮來源分配。來源分配以壓縮的歸檔檔案形式提供。如果您安裝 Apache 1.3.27，來源分配歸檔檔案將讀取 apache_1.3.27.tar.gz。

2. 使用以下指令，對歸檔檔案進行解壓縮並解除磁帶存檔：

   ```
   $ tar -zxvf apache_1.3.27.tar.gz
   ```

   該指令將在目前的工作目錄下建立稱爲 apache_1.3.27 的目錄。

3. 您現在需要配置您的環境以編譯 Apache 來源碼。來源分配隨附一個稱爲 configure 的程序檔，該程序檔可檢查您的環境是否具有順利編譯 Apache 所必需的支援檔案 (如標頭、共用程式庫和公用程式)。

   若要配置您的環境，請移往 Apache 來源目錄，並繼續執行下列步驟：

   a. 確定下列路徑存在，同時在 Solaris 上安裝 Apache ：

      - CC=/opt/SUNWspro/bin/cc

        其中 /opt/SUNWspro/bin 是安裝 cc 的路徑。確定它位於您的 PATH 中。

      - 確定 /usr/ccs/bin 位於您的 PATH 中。

   b. 執行以下指令：

```
./configure --enable-module=proxy --prefix=/usr/local/apache
```

   prefix 引數中指定的路徑指出您要安裝 Apache 的位置。這是一個變數，您可以指定要安裝 Apache 的路徑。

   該指令將在螢幕上輸出數行。實質上，該指令會依據您的系統配置爲建構建立 Makefiles。如果 configure 發生錯誤，您可能遺漏了某些標頭檔案或公用程式 (必須首先安裝才能繼續執行)。

4. configure 程序檔成功執行後，您可以使用 make 指令編譯 Apache，如下所述：

   ```
   make
   ```

   該指令將在螢幕上輸出數行，指示程序正在編譯 Apache 來源碼並連結 Apache。此程序應該正常結束，無任何錯誤。不過，如果發生錯誤，請檢查是否已正確下載 Apache 的所有程式庫檔案和公用程式。

5. 您現在需要安裝 Apache。Apache 在 /usr/local/apache 目錄下 (或您指定的任何其他目錄) 安裝其本身。若要安裝 Apache，請執行以下指令：

   ```
   make install
   ```

   如果該指令執行成功，則您的系統現已安裝了 Apache。您應該查看以下目錄中的 Apache 安裝檔案：

   ```
   /usr/local/apache
   ```

   主配置檔案 (稱爲 httpd.conf) 將安裝於 /usr/local/apache/conf 目錄下。

注意 mod_proxy 缺少 Sun ONE Application Server 7 反向代理外掛程式的 SSL 傳輸和載入平衡功能。使用 mod_proxy 時，Web 應用程式也無法取得用戶端的 IP 位址或 SSL 用戶端憑證。

修改 httpd.conf 檔案

透過檔案 httpd.conf 配置 Apache。該檔案包含一些 Apache 指令，這些指令可決定 Apache 伺服器的各種作業參數。若要執行 Apache 的簡單安裝，您需要修改下列幾個指令：

```
ServerRoot "/usr/local/apache"
Port 5000
```

ServerRoot 是您安裝 Apache 的路徑。

現在已配置了 Apache 的預設行爲和 Web 服務。下一步，您必須將下列應用程式伺服器特定的指令加入至 httpd.conf 檔案以啓動 Apache，將 HTTP 請求轉寄給 Sun ONE Application Server ：

```
<IfModule mod_proxy.c>
ProxyPass / http://<s1as_server.some.domain>:<port>/
ProxyPassReverse  / http://<s1as_server.some.domain>:<port>/
</IfModule>
```

在這裡，應該將 *<s1as_server.some.domain>*:*<port>* 替代成您的 Sun ONE Application Server 之 URL 位址。複製這兩行以用於每個 Web 應用程式環境根，其中 */application* 是 Web 應用程式環境根，http://*server* 是 Sun ONE Application Server 的 URL。

這將完成 Apache Web 伺服器的配置。

啟動與停止 Apache

Apache 隨附一個標題爲 `apachectl` 的程序檔，該程序檔可讓啓動、停止和重新啓動 Apache 更容易。執行以下指令以啓動 Apache ：

```
$ /usr/local/apache/bin/apachectl start
```

若要停止 apache，請執行以下指令：

```
use /usr/local/apache/bin/apachectl stop
```

啓動後，您可以測試 Apache 的安裝。Apache 執行後，請在您的 Web 瀏覽器中鍵入以下位址： `http://localhost/`. 如果您的安裝成功並且 Apache 執行，您應該可以看到顯示有關成功訊息的測試頁面。

第 8 章

配置 J2EE 容器

Sun ONE Application Server 提供符合 J2EE 1.3 規格的各種 J2EE 容器。容器爲 J2EE 應用程式元件 (例如企業 Java Bean [EJB] 與訊息導引 Bean [MDB]) 提供執行期間支援。MDB 與 EJB 從不與其他 J2EE 應用程式元件直接互動。它們使用 EJB 容器的協定與方法相互互動，並且與平台服務互動，例如 Java 作業事件服務。容器插入在應用程式元件與 J2EE 服務之間。這允許容器透明地引入元件部署描述元定義的服務，例如宣告性作業事件管理、安全性檢查、資源匯集以及狀態管理。

Sun One Application Server 加入了 Web 容器與 EJB 容器。

本章包含以下主題：

- 關於 Web 容器
- 關於 EJB 容器

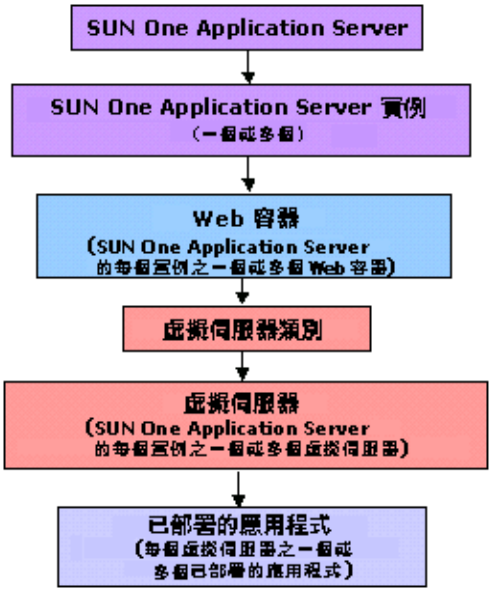
關於 Web 容器

Web 容器是容納 Web 應用程式的 J2EE 容器。Web 容器透過向開發人員提供執行 servlet 與 Java Server Page (JSP) 的環境，延伸 Web 伺服器功能性。Servlet 在沒有 CGI (共用閘道介面) 程式效能限制的情況下，提供以元件爲基礎，並且與平台無關的方法來建立基於 Web 的應用程式。JSP 技術是 servlet 技術的延伸，建立該技術是爲了支援 HTML 與 XML 頁面創作。包含在 Web 容器中的 Servlet 或 JSP 能夠呼叫企業 Java Beans (EJB) 容器中的 Bean 方法。可以使用物件請求代理程式 (ORB)，透過本機呼叫或遠端呼叫來呼叫 Bean 方法。

同時，Web 容器還提供 Web 應用程式存取本機 EJB，使用 JNDI (Java 命名目錄介面) 可以找到本機 EJB。

圖「在 Sun ONE Application Server 架構中的 Web 容器」解釋位於 Sun ONE Application Server 架構中的 Web 容器之角色與位置：

圖 8-1 在 Sun ONE Application Server 架構中的 Web 容器



本章節涵蓋以下主題：

- 瞭解 Web 容器的角色
- Web 應用程式配置
- Web 應用程式部署
- 單一登入功能
- 記錄 Web 容器

瞭解 Web 容器的角色

Web 容器的主要角色是爲 Web 應用程式提供執行環境，並向容器中的 Web 應用程式提供服務 (資料庫存取、安全性、多重執行緒等)。Web 應用程式是 Servlet、HTML 頁面、類別以及其他資源的集合，這些構成了 Sun ONE Application Server 中的完整應用程式。

以下爲 Web 應用程式的元素：

- Servlet
- JSP 頁
- 公用程式類別
- 靜態文件 (html 檔、影像檔、聲音檔等)
- 用戶端 Java applet、Bean 以及類別
- 將以上所有元素連接在一起的描述性複合資訊

Web 應用程式可以在 Sun ONE Application Server 中運行的 Web 容器中進行部署。

如需關於配置 Web 伺服器外掛程式並使之與 Sun ONE Application Server 協同作業的更多資訊，請參閱第 7 章「配置 Web 伺服器外掛程式」。

Web 應用程式配置

也可以將 Web 容器配置爲在虛擬伺服器中部署 Web 應用程式。可以將 Web 容器配置爲包含多個虛擬伺服器。並且可以將每個虛擬伺服器配置爲容納任何數目的 Web 應用程式。Web 應用程式在虛擬伺服器環境範圍內使用。如需關於虛擬伺服器的更多資訊，請參閱第 15 章「使用虛擬伺服器」。

本章節包含以下主題：

- 虛擬伺服器屬性
- Web 模組屬性

虛擬伺服器屬性

您可以爲虛擬伺服器指定特定可配置屬性的值。虛擬伺服器可以具有與之關聯的多個 Web 應用程式。需要使用者**登入**至 Web 應用程式。

如果此屬性爲單一登入，在 server.xml 檔案中，sso-enabled 設定爲預設值 true，使用者可以登入至與特定虛擬伺服器關聯的任何一個 Web 應用程式。因此，在同一個虛擬伺服器中運行的所有其他 Web 應用程式會識別識使用者身份。如果 sso-enabled 的值設定爲 false，則會停用存取此虛擬伺服器中所有 Web 應用程式的單一登入。

可以動態配置 sso-enabled 屬性，無需重新啓動伺服器便可使該屬性生效。

如需關於單一登入的更多資訊，請參閱第 190 頁的「單一登入功能」章節。

Web 模組屬性

在名爲 sun-web.xml 的檔案中指定了針對 Sun ONE Application Server 的部署描述元，可以在給定的 Web 應用程式之 WEB-INF 目錄中找到此檔案。

通常，每個 Web 應用程式均具有一個已配置的 sun-web.xml 檔案。但是，Web 容器不需要每個 Web 應用程式都具有一個 sun-web.xml 檔案。在沒有 sun-web.xml 檔案的情況下，Web 容器會爲 Sun ONE Application Server 特定的所有屬性假設預設值。

context-root 屬性

此屬性定義安裝 Web 應用程式所在的環境根。如果此屬性爲空字串，則會指定此 Web 應用程式爲虛擬伺服器的預設 Web 應用程式。虛擬伺服器的預設 Web 應用程式會回應無法解譯至其他 Web 應用程式 (部署至此虛擬伺服器) 的所有請求。每個虛擬伺服器具有一個預設 Web 應用程式。

對於預設 Web 應用程式，此欄位的值應該爲空字串 ""。

location 屬性

此屬性的輸入應該爲有效的目錄路徑，指出預設 Web 應用程式的位置。在安裝程序中，預設 Web 應用程式的位置設定爲 modules/*default-web-app*/ 目錄。

必須使用 location 屬性，此屬性可以爲已擷取 WAR (Web 歸檔) 檔案內容所在目錄之完整路徑或相對路徑。如果指定的路徑爲相對路徑，則需要爲虛擬伺服器層級中定義的**應用程式根目錄**之相對路徑。

例如：

location="*applications/<ear name>/<war-module name>/*"

location="*modules/<war-module name>*"

location="*/u/myapps/<war-module name>*"

location="*/u/myapps/<ear-name>/<war-module name>*"

enabled 屬性

此屬性的預設值爲 `true`，指示啓用 Web 應用程式爲請求提供服務。透過將 enabled 屬性的值設定爲 `false`，可以暫時停用 Web 應用程式，不再向請求提供服務。但是，不會移除 Web 應用程式的內容 (儲存於硬碟中)。

Web 應用程式部署

Web 容器經由 Web 歸檔 (WAR) 檔案或包含 WAR 檔案展開檢視 (`WEB-INF/lib`, `WEB-INF/classes` 等) 的目錄部署 Web 應用程式。無需重新啓動伺服器來部署應用程式。

Web 容器會對每個虛擬伺服器部署「預設」Web 應用程式。預設位置 (目錄) 在虛擬伺服器 `app root` 目錄的 `modules/`*`default-web-app`*`/` 子目錄中。此預設 Web 應用程式回應不能解譯至其他 Web 應用程式 (部署至此虛擬伺服器) 的所有請求。此 Web 應用程式由呼叫程式 servlet 與 JSP servlet 組成，呼叫程式 servlet 負責處理 `/servlet/*` 的請求；JSP servlet 負責爲 JSP 頁提供服務。只要使用者在 `web.xml` 與 `sun-web.xml` 檔案中指定 EJB 參考，預設 Web 應用程式便可以存取 EJB。

在虛擬伺服器的 `server.xml` 檔案中定義預設 Web 應用程式，表示如下：

```
<web-module context-root="" location="modules/default-web-app/">
```

動態重新部署與熱部署

動態重新部署是在不重新啓動伺服器的情況下，重新部署現有應用程式的功能。在應用程式配置 (其 `xml` 檔案的內容) 與特定類別變更時，才會發生動態重新部署。動態重新部署產生的行爲與動態重新載入整個應用程式類別的行爲相同。此外，動態重新部署還包括建立新的應用程式環境 (web 與 ejb) 並取消舊的應用程式環境。因此，動態重新部署會產生一種新的應用程式實例 (現有的階段作業資料除外)。同時，僅在開發模式下支援此功能，並且此功能會產生與動態重新載入相似的異常。同時，只有此重新啓動發生後，需要伺服器重新啓動的配置變更才會生效。只有應用程式與非共用獨立模組的中央配置指定了動態重新載入，才可以啓動動態重新載入。

重新載入 Web 應用程式時，無論階段作業管理程式是否配置了持續性機制，均會自動儲存並回復所有的現有階段作業資訊。

熱部署是在無需重新啓動伺服器的情況下，在伺服器執行期間部署應用程式的功能。此功能使用與動態重新部署相同的基礎架構。但是，由於沒有具體的狀態作參考，因此只有在生產時才支援此功能。

單一登入功能

只要使用者僅存取特定虛擬伺服器的任何 Web 應用程式中之未受保護資源，就不要求使用者認證其自身。

使用者存取與特定虛擬伺服器關聯的任何 Web 應用程式中之受保護資源時，會要求使用者使用目前存取的 Web 應用程式所定義的登入方法，認證其自身。

認證之後，便會使用與使用者關聯的角色，對所有關聯的 Web 應用程式執行存取控制決定。那樣，就不需要個別針對每個 Web 應用程式，認證使用者自身。

使用者登出 Web 應用程式之後，在所有 Web 應用程式中進行的使用者階段作業均會無效。其後存取任何應用程式之受保護資源的任何嘗試，均需要使用者再次認證其自身。

單一登入功能使用 HTTP cookie，傳輸將每個請求與已儲存的使用者身份關聯的記號，因此僅可以在支援 cookie 的用戶端環境中使用此功能。

記錄 Web 容器

您可以透過設定不同的**日誌層級**，控制 Web 容器與虛擬伺服器中任何應用程式的預設記錄行爲。請注意，記錄行爲不影響應用程式本身的記錄。

指定日誌層級可控制要記錄的訊息類型。例如，如果您指定僅記錄日誌層級爲 `FATAL` 的訊息，那麼比此值「更高」的日誌層級訊息將會靜謐忽略。只有使用明確日誌層級記錄的訊息，才與此值進行比對。

會無條件記錄沒有使用明確日誌層級記錄的訊息。預設行爲會記錄所有警告、錯誤與嚴重錯誤訊息。

若要爲 Web 容器設定日誌層級，請執行以下工作：

1. 在管理介面左窗格中，展開 Sun ONE Application Server 實例樹，以找到您要修改的 Web 容器配置。
2. 展開 [Containers] 標籤，然後從顯示的 J2EE 容器清單中選取 [Web Container]。您將在管理介面右窗格中看到在「記錄 Web 容器」中顯示的以下頁面。

圖 8-2 記錄 Web 容器

3. 從 [Log Level] 下拉式清單中，選取您要的日誌層級。如需關於所有日誌層級及其定義的清單，請參閱第 5 章「使用記錄功能」。
4. 按一下 [Save]，以儲存您的設定。

若要建立 Web 容器的附加特性，請按一下 [Properties] 按鈕。

關於 EJB 容器

企業 Java Bean 容器是控制企業 Bean，並爲其提供重要系統層級服務的執行環境。EJB 是在 EJB 容器內執行的元件，而 EJB 容器則在 EJB 伺服器內執行。以下是爲企業 Bean 提供的系統層級服務：

- 作業事件管理
- 安全性
- 生命週期管理
- 遠端連接性
- 資料庫連線匯集
- 命名服務

企業 Bean 是以 Java 撰寫的包含企業邏輯的伺服器元件。EJB 容器提供遠端存取企業 Bean。EJB 始終在容器環境中工作，此容器作爲 EJB 與容納 EJB 的伺服器之間的連結。EJB 容器啓用分散式應用程式建立 (使用您自己的元件與其他提供者提供的元件)。

使用 EJB 容器，Sun ONE Application Server 可以提供高階作業事件、狀態管理、多重執行緒與資源匯集包裝函式，因此使您無需瞭解低階 API 的詳細資訊。此容器提供 2.0 EJB 規格指定的所有標準容器服務，並且還提供針對 Sun ONE Application Server 的附加服務。

容器使用鈍化與啓動程序來管理 Bean 活動，以確保縮放性。

本章節涵蓋以下主題：

- 瞭解 EJB 容器的角色
- 配置 EJB 容器

瞭解 EJB 容器的角色

EJB 容器提供以下標準服務：

- 鈍化

  將 EJB 從記憶體傳輸至輔助儲存區的程序。鈍化允許在不損壞 Bean 的情況下，釋放 Bean 資源。使用這種方法，Bean 可以保持永久性，並且可在不耗用實例化時間的情況下重新被呼叫。

- 啓動

  將 EJB 從輔助儲存區傳輸至記憶體的程序。容器合約建立 EJB 及其容器之間的關係，並且對用戶端是完全透明的。這種關係包括：

  - 生命週期

    對於階段作業 Bean，包括 `javax.ejb.SessionBean` 與 `javax.ejb.SessionSynchronization` 介面執行。對於實體 Bean，包括 `javax.ejb.EntityBean` 介面執行。對於訊息導引 Bean，包括 `javax.ejb.MessageDriven` 介面執行。

  - 階段作業環境

    在建立 Bean 實例後，容器執行 `javax.ejb.SessionContext` 介面，將服務與資訊發送至階段作業 Bean 實例。

  - 實體環境

    在建立 Bean 實例後，容器執行 `javax.ejb.EntityContext` 介面，將服務與資訊發送至實體 Bean。

- 訊息環境

  在建立 Bean 實例後，容器執行 `javax.ejb.MDBContext` 介面，將服務與資訊發送至訊息導引 Bean。

- 環境

  容器執行 `java.util.Properties`，並使這些特性可用於其 EJB。

- 服務資訊

  容器可以爲其 EJB 提供服務。

Sun ONE Application Server 服務包括遠端存取、命名、安全性、並行處理、作業事件控制與資料庫存取。

EJB 容器負責：

- 建立允許遠端連接性的執行物件 (`EJBObject`)。
- 建立本地執行物件，該物件允許建立 `EJBObject`。
- 將本地執行物件與命名服務相連結，以便用戶端可以查看本地物件。
- 確保只有授權的用戶端才可以呼叫 Bean 方法 (使用 `EJBObject`)。
- 確保在適當的作業事件中呼叫企業方法。
- 管理 Bean 的生命週期。管理 Bean 生命週期包括：
  - 匯集 Bean
  - 呼叫適當的回呼方法 (例如 `ejbActivate/ejbPassivate`)
  - 管理資料庫連線區，以便應用程式可以更有效地使用與重複使用連線

實際執行詳細資訊是容器的一部分，基於容器與其 EJB 之間的標準規定介面。無需您瞭解或處理特定平台的執行詳細資訊。相反，可以建立一般的、以工作爲主的 EJB，使其與支援 EJB 標準的任何供應商產品協同使用。

瞭解 Sun ONE Application Server 使用的 EJB 類型很有用。

企業 Java Bean 類型

EJB 是表示以下其中一種項目的物件：

- 特定用戶端使用的階段作業，會自動維護呼叫多重用戶端方法的狀態。
- 永久性的實體物件，可能在多重用戶端中共用。
- 無狀態服務，例如訊息處理。

實體 Bean 主要用於處理資料存取 (使用 Java 資料庫連結性 [JDBC] API)，而階段作業 Bean 則提供暫時的應用程式物件並執行離散的企業工作。有三種 EJB，如以下主題所述：

- 關於階段作業 Bean
- 關於實體 Bean
- 關於訊息導引 Bean

關於階段作業 *Bean*

階段作業 Bean 爲特定用戶端請求執行企業規則或邏輯。

階段作業 Bean 用於表示暫時的物件與程序，例如單一資料庫記錄、用於編輯的文件複本或個別用戶端的專用企業物件。即，階段作業 Bean 是專用資源，僅由建立其的用戶端使用。由於只有單一用戶端才能使用這些物件，所以階段作業 Bean 可以維護特定用戶端的階段作業資訊 (稱爲**交談式**狀態)。

例如，可以建立 EJB 以模擬電子購物推車。每次使用者登入至應用程式後，應用程式會建立購物推車階段作業 Bean，以放置使用者購買的商品。使用者登出或完成購物之後，會釋放階段作業 Bean。

階段作業 Bean 具有以下特徵：

- 執行階段作業 Bean 與單一用戶端相關。
- 階段作業 Bean 使用期相對較短。
- 階段作業 Bean 並非在伺服器當機後仍能存在。
- 如果 EJB 容器當機，則會移除階段作業 Bean。
- 同時，階段作業 Bean 還可依據特性設定，處理作業事件管理。此爲選擇性的。
- 階段作業 Bean 會更新基礎資料庫中的共用資料。此爲選擇性的。
- 階段作業 Bean 可以爲無狀態或有狀態的 Bean。

無狀態階段作業 Bean。無狀態階段作業 Bean 會封裝特定用戶端在有限時間範圍內所需的暫時企業邏輯部分。無狀態階段作業 Bean 不維護交談式狀態。

有狀態階段作業 Bean。有狀態階段作業 Bean 是暫時的，但是維護交談式狀態，以保留關於用戶端之間呼叫的內容與值資訊。交談式狀態會啓用 Bean 容器，維護關於階段作業 Bean 狀態的資訊，並依需要，在程式執行的後一點中重新建立此狀態。

關於實體 *Bean*

實體 Bean 通常表示永久性資料 (直接在資料庫中進行維護，或作爲物件透過企業資訊系統 [EIS] 應用程式存取)。具有 EJB 與 EJB 容器的伺服器爲並行的作用中實體 EJB 提供可縮放執行環境。

實體 Bean 的一個簡單範例是定義一個實體 Bean 表示資料庫表格中的單一列，其中每個 Bean 實例表示一個特定列。較複雜的範例是用於表示資料庫中的結合表格之複雜檢視，例如，每個 Bean 實例表示一個購物推車中的商品內容。

實體 Bean 具有以下特徵：

- 實體 Bean 提供 EIS 資源 (通常是資料庫) 中資料的物件檢視。
- 所有使用者均可以存取實體 Bean。
- 實體 Bean 可以在伺服器當機後透明地存在。
- 實體 Bean 使用容器管理式的作業事件或 Bean 管理式的作業事件。

實體 Bean 表示永久性資料，可作爲容器管理式的持續性或 Bean 管理式的持續性。實體 Bean 持續性可以由 Bean 或容器進行管理。

Bean 管理式的持續性。實體 Bean 管理其自己的持續性。Bean 開發人員直接在 EJB 類別方法中執行持續性碼 (例如 JDBC 呼叫)。可能的不利因素是如果使用專用介面，則會遺失可攜性，並且在將 Bean 關聯至特定的資料庫時具有風險性。

容器管理式的持續性。實體 Bean 持續性由容器進行管理。由於容器透明地管理持續性狀態，因此無需在 Bean 方法中執行任何資料存取碼。此方法不僅易於執行，並且可使 Bean 在與特定資料庫沒有任何關係的情況下，完全可攜。

使用容器管理式持續性的實體 Bean，實際是使用 Bean 管理式持續性的實體 Bean 之自動產生 (透過容器) 版本。

如需關於建立與使用實體 Bean 的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to Enterprise JavaBeans Technology*」。

關於訊息導引 Bean

訊息導引 Bean 是允許 J2EE 應用程式非同步處理訊息的 EJB。訊息導引 Bean 在收到 Java Message Service 訊息時驅動。

從訊息導引 Bean 實例的建立至損毀，其始終在訊息導引 Bean 容器內。此容器爲訊息導引 Bean 提供安全性、作業事件、訊息並行處理、訊息導引 Bean 實例的生命週期管理與其他服務。具有 EJB 與 EJB 容器的伺服器爲並行的作用中訊息導引 EJB 提供可縮放執行環境。

J2EE 1.3 平台中的 Java Message Service API 指定了以下內容：

- 應用程式用戶端、EJB 元件、Web 元件可以發送或同步接收 Java Message Service 訊息。此外，應用程式用戶端可以非同步使用 Java Message Service 訊息。
- 訊息導引 Bean 啓用非同步使用訊息。Java Message Service 提供者可以選擇性地執行並行處理訊息導引 Bean 使用的訊息。

訊息導引 Bean 表示無狀態服務，實際上是非同步訊息使用者 (完全匿名並且沒有用戶端可見識別)。訊息導引 Bean 沒有本地介面或元件介面。用戶端將訊息發送至 Java Message Service 目標 (佇列或主題)，其中訊息導引 Bean 類別爲 MessageListener，透過 Java Message Service 存取訊息導引 Bean。

只有訊息導引 Bean 可以非同步接收訊息。不允許階段作業 Bean 或實體 Bean 爲 Java Message Service `MessageListener`。

訊息導引 Bean 具有以下特徵：

- 在收到單一用戶端訊息時執行。
- 非同步呼叫。
- 使用期相對較短。
- 不直接表示資料庫中的共用資料，但是可能存取並更新此資料。
- 會在 EJB 伺服器當機時移除。
- 爲無狀態。
- 選擇性地支持作業事件。

配置 EJB 容器

可以爲 EJB 容器配置日誌層級，也可以啓用監視。EJB 容器會處理 EJB 與 MDB。使用管理介面，可以配置此容器管理的 EJB 與 MDB 之設定。本章節包含以下主題：

- 執行一般配置
- 配置 EJB 設定
- 配置 MDB 儲存區設定

執行一般配置

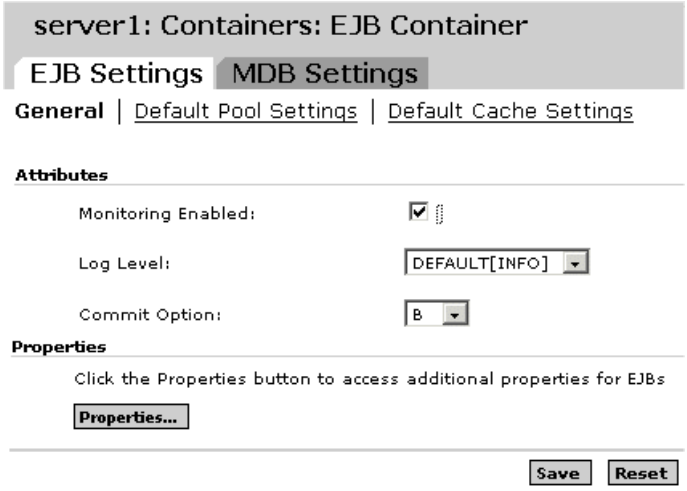
您可以配置 EJB 容器的以下方面：

- 記錄
- 監視
- 作業事件屬性

若要設定 EJB 容器的日誌層級，啓用監視並設定作業事件屬性，請執行以下工作：

1. 在管理介面的左窗格中，開啓 Sun ONE Application Server 實例樹，以找到您要修改的 EJB 容器配置。
2. 展開 [Containers] 標籤，然後從顯示的 J2EE 容器清單中選取 [EJB Container]。將在管理介面的右窗格中看到在「EJB 容器 - 一般配置」中顯示的視窗。

圖 8-3 EJB 容器 - 一般配置



3. 針對 [Monitoring Enabled] 來標示此核取方塊，以啓用 EJB 容器中的監視。現在已啓動此 Sun ONE Application Server 特定實例的 EJB 容器之監視。如需關於 EJB 容器可監視部分的清單，請參閱表格「EJB 容器的監視統計」。

4. 從 [Log Level] 下拉式清單中，選取您要的日誌層級。如需關於所有日誌層級及其定義的清單，請參閱第 5 章「使用記錄功能」指定日誌層級可控制要記錄的訊息類型。例如，如果您指定僅記錄日誌層級爲 FATAL 的訊息，那麼比此值「更高」的日誌層級訊息將會靜謐忽略。只有使用明確日誌層級記錄的訊息，才與此值進行比對。

   會無條件記錄沒有使用明確日誌層級記錄的訊息。預設行爲會記錄所有警告、錯誤與嚴重錯誤訊息。

5. 從 [Commit Option] 下拉式清單中，選取您要在 EJB 容器中使用的確定選項。

   作業事件可以用兩種方式結束：使用確定或回轉。作業事件確定後，會儲存其敘述產生的資料修改。您設計企業 Bean 時，會決定此確定是否爲容器管理式的作業事件或 Bean 管理式的作業事件。因此 UI 中的選項，B 表示 Bean 管理式的確定，C 表示容器管理式的確定。

6. 按一下 [Properties] 按鈕，以建立 EJB 容器的新特性。

7. 按一下 [OK]，以儲存您的設定。

下表顯示可以監視的 EJB 容器屬性：

表格 8-1 EJB 容器的監視統計

統計名稱	資料類型與單位	值範圍	注釋
minBeansInPool	整數	0-MAXINT	儲存區中 Bean 的最低偏好數目 (適用於無狀態階段作業 Bean)。
initialBeansInPool	整數	0-MAXINT	儲存區中 Bean 的初始數目 (適用於無狀態階段作業 Bean)。
maxBeansInPool	整數	0-MAXINT	儲存區中 Bean 的最大數目。(適用於無狀態階段作業 Bean)。
beanIdleTimeoutInSeconds	整數	0-MAXLONG	以秒表示的閒置逾時，超出此逾時，則會銷毀 Bean。
numBeansCreated	整數	0-MAXINT	到目前爲止建立的 Bean 數目。
numBeansDestroyed	整數	0-MAXINT	到目前爲止銷毀的 Bean 數目。
numThreadsWaitaing	整數	0-MAXINT	等待自由 Bean 的執行緒數
numBeansInPool	整數	0-MAXINT	儲存區內可用 Bean 的數目。(如果大於 0，則 numThreadsWaitaing 必須爲 0)。
maxBeansInCache	整數	0-MAXINT	快取記憶體中 Bean 的最大數目 (適用於實體 Bean 與有狀態 Bean)。

表格 8-1 EJB 容器的監視統計 (續)

統計名稱	資料類型與單位	值範圍	注釋
minBeansInCache	整數	0-MAXINT	快取記憶體中 Bean 的最小偏好數目 (適用於實體 Bean 與有狀態 Bean)。
cacheFaultsPercentage	倍數		導致從備份儲存中啓動的快取遺失數目。

配置 EJB 設定

使用管理介面，可以配置預設儲存區與 EJB 容器管理的 EJB 之 Bean 快取設定，如以下主題所述：

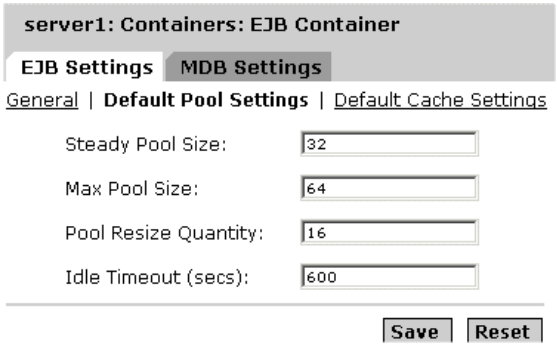
- 配置 EJB 儲存區設定
- 配置 EJB 快取設定

配置 *EJB* 儲存區設定

若要配置 EJB 儲存區設定，請執行以下工作：

1. 在管理介面的左窗格中，開啓您要修改其 EJB 設定的 Sun ONE Application Server 實例樹。
2. 展開 [Containers] 標籤，然後從顯示的 J2EE 容器清單中選取 [EJB Container]。將在管理介面的右窗格中看到在「配置 EJB 儲存區設定」中顯示的視窗。

圖 8-4 配置 EJB 儲存區設定



3. 在 [Steady Pool Size] 欄位中，指定儲存區中 Bean 的最小數目。這適用於無狀態階段作業 Bean。

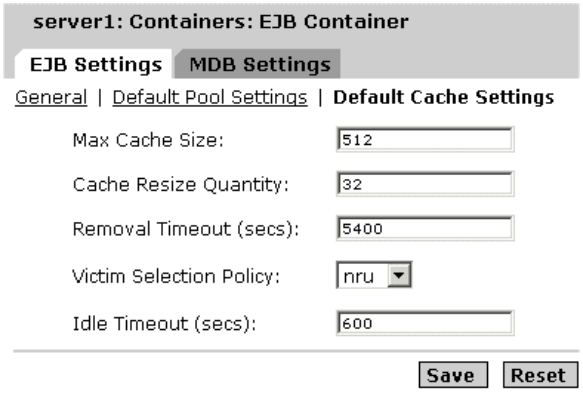
4. 在 [Max Pool Size] 下拉式清單中，及時在任何給定點中，指定在儲存區中需要的最多 Bean 數目。此設定適用於無狀態階段作業 Bean。
5. 在 [Pool Resize Quantity] 欄位中，若 Bean 的閒置時間超過了 idle-timeout-in-seconds 標籤中指定的時間，則請指定要從儲存區中移除的 Bean 數目。
6. 在 [Idle Timeout] (秒) 欄位中，指定 Bean 可以閒置的時間 (以秒表示)。閒置逾時期間過後，Bean 仍然閒置，則會銷毀 Bean。
7. 按一下 [Save]，以儲存您的變更。

配置 *EJB* 快取設定

若要配置 EJB 快取設定，請執行以下工作：

1. 在管理介面的左窗格中，開啓您要修改其 EJB 設定的 Sun ONE Application Server 實例樹。
2. 展開 [Containers] 標籤，然後從顯示的 J2EE 容器清單中選取 [EJB Container]。將在管理介面的右窗格中看到在「配置 EJB 儲存區設定」中顯示的視窗。

圖 8-5 配置 EJB 快取設定



3. 在 [Max Cache Size] 欄位中，指定在快取記憶體中維護的 Bean 最大數目。此屬性的預設值爲 idle-timeout-in-seconds 屬性中指定的值。
4. 在 [Cache Resize Quantity] 欄位中，如果在儲存區中的 Bean 數目超過了在 [Max Cache Size] 屬性中指定的數量，則請指定進行損毀選取的 Bean 數目。

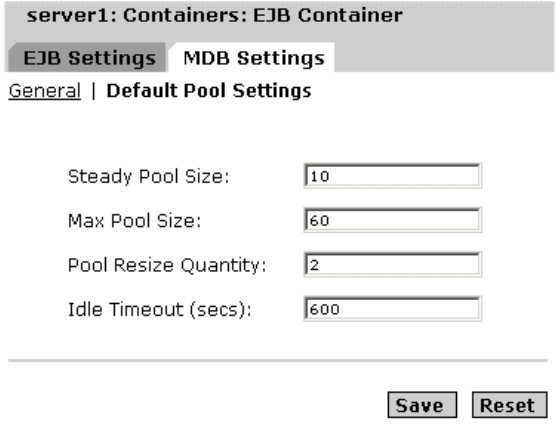
5. 在 [Removal Timeout] (以秒表示) 欄位中，指定可以持續使備份儲存中閒置的 Bean 鈍化的時間。如果用戶端未存取 Bean 的時間超過了 removal-timeout-in-seconds 屬性所指定的值，則 Bean 會從備份儲存中移除，從而用戶端將不能存取 Bean。
6. 從 [Victim Selection Policy] 下拉式清單中，選取犧牲者選取演算法，必須使用此演算法選取從儲存區中移除的犧牲者 Bean。
7. 在 [Idle Timeout] (以秒表示) 欄位中，指定允許 Bean 在快取記憶體中閒置的時間。此期限過後，Bean 會鈍化。Bean 鈍化 (在閒置備份儲存中) 的時間由 removal-timeout-in-seconds 參數控制。
8. 按一下 [Save]，以儲存您的變更。

配置 MDB 儲存區設定

使用管理介面，可以爲 EJB 容器管理的 MDB 配置預設儲存區設定。若要配置 MDB 的預設儲存區設定，請執行以下工作：

1. 在管理介面的左窗格中，開啓您要修改其 MDB 容器配置的 Sun ONE Application Server 實例樹。
2. 展開 [Containers] 標籤，然後從顯示的 J2EE 容器清單中選取 [EJB Container]。將在管理介面的右窗格中看到在「配置 MDB 儲存區設定」中顯示的視窗。

圖 8-6 配置 MDB 儲存區設定



3. 按一下 [MDB Settings]。在 [Steady Pool Size] 文字欄位中，指定儲存區中 Bean 的最小數目。這適用於無狀態階段作業 Bean。

4. 在 [Max Pool Size] 欄位中，請及時在任何給定點上，指定在儲存區中需要的最多 Bean 數目。
5. 在 [Pool Resize Quantity] 欄位中，若 Bean 的閒置時間超過了 idle-timeout-in-seconds 標籤中指定的時間，則請指定要從儲存區中移除的 Bean 數目。
6. 在 [Idle Timeout] (秒) 欄位中，指定 Bean 可以閒置的時間 (以秒表示)。閒置逾時期間過後，Bean 仍然閒置，則會銷毀 Bean。
7. 按一下 [Save]，以儲存您的設定。

第 9 章

使用作業事件服務

作業事件是企業不可或缺的一個組成部分。典型的企業作業事件包含雙方或多方之間的資產轉移。通常將這類轉移的精確記錄儲存於一個或多個資料庫中。由於這些資訊對於企業的運作至關重要，因此必須保持其有效性、現時性與可靠性。作業事件處理對於程式設計的初學者而言，可能有些困難。J2EE 平台提供數個抽象的概念來簡化可靠作業事件處理應用程式的開發程序。在本章中，我們將討論 Sun ONE Application Server 中的 J2EE 作業事件與作業事件支援。

本章對 Java 作業事件做一般的討論，而專門討論併入至 Sun ONE Application Server 的作業事件支援。

本章包含以下主題：

- 何爲作業事件？
- J2EE 中的作業事件
- 作業事件資源管理員
- 本機作業事件與分散式作業事件
- 容器管理作業事件
- Bean 管理式的作業事件
- 作業事件服務管理

何為作業事件？

若要模仿企業的作業事件，一個程式可能需要執行數個步驟。例如，一個財務程式需要透過執行以下以虛擬碼列示的步驟，才能將資金從活期存款帳戶轉入儲蓄存款帳戶：

```
begin transaction
debit checking account
credit savings account
update history log
commit transaction
```

在上述的虛擬碼中，`begin` 與 `commit` 敘述用於標記作業事件的界限。若要完成此作業事件，所有的三個步驟都必須成功地完成。如果未能成功地完成所有三個步驟，資料的完整性可能會受損。

這種保證被稱爲不可分性。作業事件可以用兩種方式結束：透過 `commit` 結束或透過 `rollback` 結束。作業事件確定之後，敘述在作業事件界限內所做的修改會被永久地儲存起來。這些變更具有**持久性**，也就是說，即使今後發生系統故障，它們也會保持不變。如果作業事件中的任何敘述失敗，該作業事件會回轉，並復原到目前爲止作業事件內由已執行敘述產生的所有影響。例如，在虛擬碼中，如果磁碟機在執行 credit 步驟期間當機，作業事件會回轉並復原由 debit 敘述所做的資料修改。

即使作業事件失敗，資料仍會保持完好無缺，因爲作業事件帳戶仍處於平衡狀態。該作業事件行爲被稱爲作業事件的**一致性**。

作業事件服務還提供**隔離**，表示在作業事件確定或回轉之前，其他的應用程式或執行緒不能觀察該作業事件的各個階段。一旦作業事件確定之後，其便能夠安全地被應用程式與執行緒觀察。

J2EE 中的作業事件

J2EE 中的作業事件處理包括以下五個參與者：作業事件管理員、應用程式伺服器、資源管理員、資源介面以及使用者應用程式。透過執行不同的 API 與功能，它們當中的每個項目都有助於作業事件處理順利地進行，如下所述：

- 作業事件管理員提供支援作業事件分隔、作業事件資源管理、同步化以及作業事件環境傳遞所需的服務與管理功能。
- 應用程式伺服器提供支援應用程式執行環境 (包含作業事件狀態管理) 所需的基礎架構。

- 資源管理員 (透過資源介面) 提供應用程式對資源的存取權。資源管理員參與分散式作業事件，其方法爲：執行由作業事件管理員使用的作業事件資源介面來傳遞作業事件關聯、作業事件完成以及恢復工作。關聯式資料庫伺服器便是這樣一個資源管理員。
- 資源介面是一個系統層級的軟體程式庫，應用程式伺服器或用戶端可使用該程式庫連線到資源管理員。資源介面通常專用於資料管理員。它可以作爲程式庫，在使用它的用戶端位址空間中使用。JDBC 驅動程式便是這樣一個資源介面。
- 作業事件使用者應用程式專門在 J2EE 應用程式伺服器環境中進行作業，使用 JNDI 查找作業事件資料來源，並選擇性地查找作業事件管理員。可以使用 EJB 的宣告性作業事件屬性設定值，或使用明確的程式化作業事件分隔。

術語資源管理員經常與資源介面相互替換使用，因爲兩個實體之間有著緊密的聯繫。

作業事件資源管理員

J2EE 作業事件中支援以下作業事件資源管理員。

- 資料庫
- JMS 提供者程式
- J2EE 連接器

資料庫

資料庫是 J2EED 應用程式中最常遇到的作業事件資源管理員。JDBC 是 J2EE 元件用來存取資料庫的 API。資料庫資源被配置爲 JDBC 資源。JDBC 資源由資源管理員或 JDBC 驅動程式進行管理。JDBC 驅動程式可以爲本機作業事件或全域作業事件提供支援，在某些情況下，可同時支援本機作業事件與全域作業事件。

Sun ONE Application Server 支援經由各種 J2EE 元件使用 JDBC 與作業事件。如需有關如何註冊與配置 JDBC 資源的更多詳細資訊，請參閱第 242 頁的「關於 JDBC 資源」。應用程式伺服器負責提供作業事件的連續性 (即，初始化作業事件，並經由多個應用程式元件存取資料庫)。例如，Servlet 可以啓動作業事件、存取資料庫、呼叫企業 Bean (作爲同一作業事件的一部分存取同一資料庫)，最後確定作業事件。

JMS 提供者程式

JMS 代表 Java Message Service。JMS 提供者是用於訊息代理程式服務的 J2EE 術語。JMS API 在應用程式之間提供可靠的作業事件訊息交換。對作業事件 JMS 資料來源的支援是 J2EE 中的一項必備功能。JMS 資源和 JDBC 資源可以參與同一項作業事件。

Sun ONE Application Server 與 Sun ONE Message Queue 整合使用，後者是一個功能齊全的 JMS 提供者，並且是相應的作業事件資源管理員。透過這種方式，Sun ONE Application Server 可以經由 Servlet、JSP 頁面以及企業 Bean 啓動作業事件 JMS 存取。也可以將協力廠商 JMS 提供者與 Sun ONE Application Server 結合使用。如需更多詳細資訊，請參閱第 11 章「使用 JMS 服務」。

J2EE 連接器

Sun ONE Application Server 支援將 XATransaction 模式作爲作業事件資源管理員使用的資源介面。平台必須經由 Servlet、JSP 頁面、以及企業 Bean 啓用對資源介面的作業事件存取。也可以經由單一作業事件內的多個應用程式元件存取資源介面。例如，Servlet 可能要啓動作業事件、存取資源介面、呼叫企業 Bean (也作爲同一作業事件的一部分存取資源介面)，最後確定作業事件。

本機作業事件與分散式作業事件

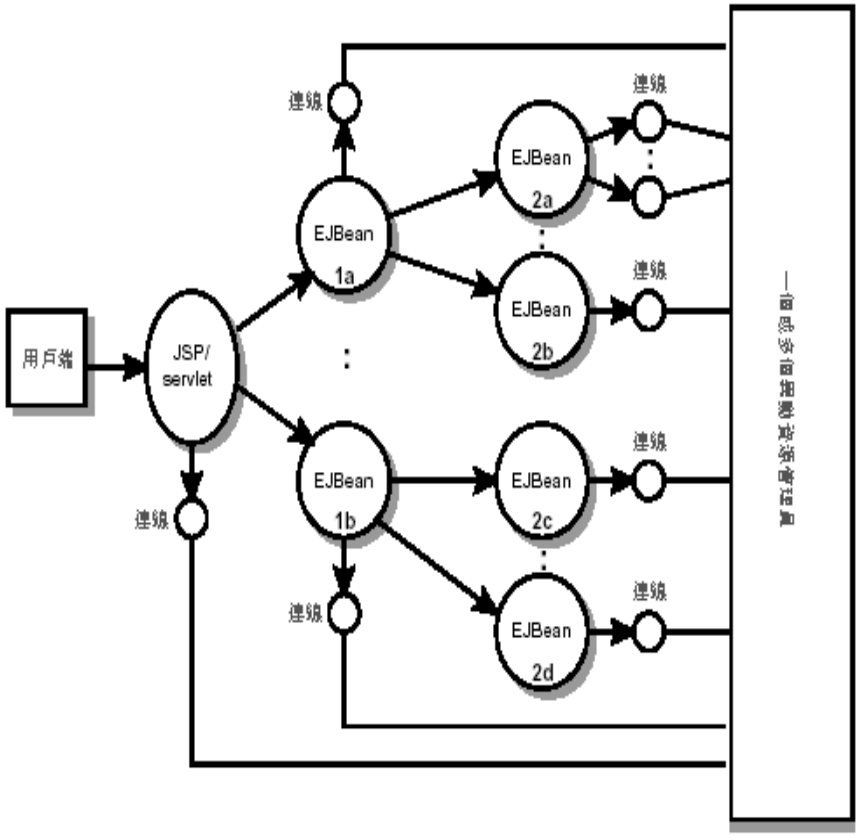
僅包含一種資源的作業事件可以使用本機作業事件完成。本機作業事件也需要所有參與的應用程式元件在一個程序中執行。包含多種資源，或多個參與程序的作業事件會變爲分散式作業事件或全域作業事件。本機作業事件最佳化使用特定的資源管理員最佳化，可以使 J2EE 應用程式對其進行觀察。

作業事件類型在很大程度上由相關資源管理員執行的介面所決定。例如，執行 `javax.sql.DataSource` 介面的 JDBC 資料來源可以參與本機作業事件。執行 `javax.sql.XADataSource` 介面的資料來源能夠參與全域作業事件。某些 JDBC 資源執行兩種介面，當這樣的 JDBC 資源透過 Sun ONE Application Server 註冊時，可能需要在 Sun ONE Application Server 配置中提供附加配置資訊，以指出該資源優先使用的功能。

與全域作業事件相比，本機作業事件更爲簡單，並且自然更爲有效。當需要轉變的資料在多個資料來源中展開的時候，本機作業事件便無法勝任該項作業。有時，不可能預知作業事件內需要利用的資料來源數目。因此，我們在工作中經常會遇到全域作業事件。全域作業事件可以執行某些效能增強的最佳化作業。

J2EE 支援作業事件應用程式結合使用 Servlet 或 JSP，在一項作業事件中存取多個企業 Bean。每個元件可以使用一個或多個連線來存取一個或多個作業事件資源管理員。在下圖中，呼叫樹經由存取多個企業 Bean 的 Servlet 或 JSP page 啓動，這樣反過來可能會存取其他企業 Bean。元件通過連線存取資源管理員。

作業事件中的 J2EE 元件存取資源



例如，應用程式可能要求上圖中的所有元件作爲單一作業事件的一部分存取資源。應用程式伺服器提供者必須提供作業事件功能來支援這樣的方案。

J2EE 作業事件管理支援相同層級的作業事件。相同層級的作業事件不能包含任何子作業事件 (嵌套式)。

作業事件恢復是分散式作業事件的一個重要方面。如果資源在關鍵時刻不可存取，或者發生其他不可恢復的錯誤，則分散式作業事件的狀況可能有問題。自動與手動恢復中斷的或不完全的作業事件是 Sun ONE Application Server 中的一項重要功能。您可以使用管理介面啓用自動作業事件恢復功能。如需有關如何控制作業事件恢復的更多資訊，請參閱第 216 頁的「作業事件服務管理」。

連線 (在此處作爲資源的同義詞) 可被標示爲可共用或不可共用。一個 J2EE 應用程式元件若採用非共用方式的連線，就必須爲連線作業提供其部署資訊，以避免容器共用該連線。在已變更的安全屬性、隔離層、字元設定值，以及本土化配置的情形下需要該資訊。

容器不應該嘗試共用標示爲不可共用的連線。如果未將連線標示爲不可共用，則必須讓應用程式知曉實際上是否共用這些連線。

J2EE 應用程式元件可以使用選擇性的部署描述元元素 `res-sharing-scope`，指示是否可以共用與資源管理員的連線。如果尚未提供部署提示，容器應該假定連線是可共用的。J2EE 應用程式元件可以快取連線物件，並且可以在多個作業事件中重複使用這些連線物件。提供連線共用的容器應該透通地切換此類快取連線物件 (在派送時間)，以在正確的作業事件範圍內指向適當的共用連線。

設計企業 Bean 應用程式時，開發人員必須確定指定界限的方式。

容器管理作業事件

在具有容器管理作業事件的企業 Bean 中，EJB 容器設定作業事件的界限。您可以使用任何企業 Bean 類型的容器管理作業事件：階段作業 Bean、實體 Bean 或訊息導引 Bean。容器管理作業事件簡化了開發程序，因爲企業 Bean 程式碼不會明確地標示作業事件的界限。程式碼不包括開始與結束作業事件的敘述。

通常，容器會在企業 Bean 方法啓動前的瞬間開始一項作業事件。並在方法結束前的瞬間確定作業事件。每一種方法均可以與一項單一的作業事件關聯。在一種方法中不允許有嵌套式作業事件或多項作業事件。

容器管理作業事件不要求所有方法均與作業事件相關聯。部署 Bean 時，您可以經由設定作業事件的屬性來指定與作業事件相關聯的 Bean 方法。

本章節包含以下主題：

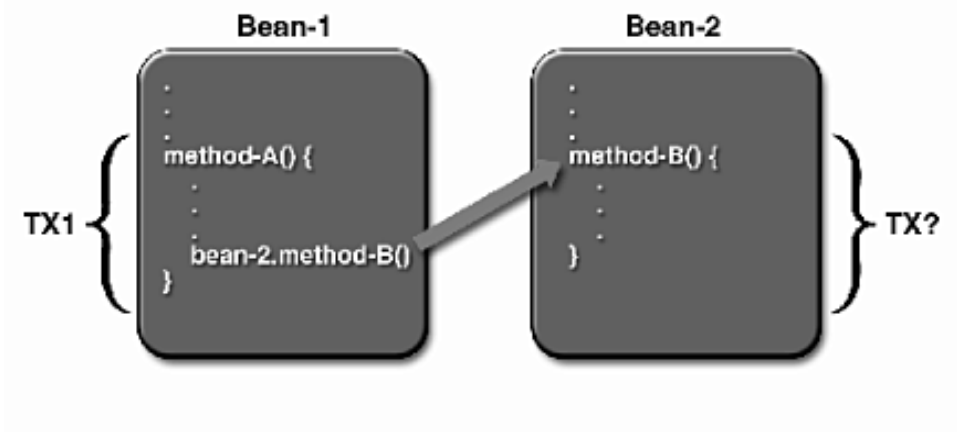
- 作業事件屬性
- 設定作業事件屬性
- 回轉容器管理作業事件
- 同步化階段作業 Bean 的實例變數

- 容器管理作業事件中禁用的方法

作業事件屬性

作業事件屬性控制作業事件的範圍。下圖展示控制範圍重要的原因。在圖中，方法 A 開始一項作業事件，然後呼叫 Bean 2 的方法 B，當方法 B 執行時，其會在由方法 A 啓動的作業事件範圍內執行，還是在一個新的作業事件範圍內執行？答案取決於方法 B 的作業事件屬性。

圖 9-1 作業事件屬性



作業事件屬性可以含有以下其中一個值：

- Required
- RequiresNew
- Mandatory
- NotSupported
- Supports
- Never

Required

如果用戶端正在作業事件中執行，並呼叫企業 Bean 方法，則方法會在用戶端的作業事件中執行。如果用戶端與作業事件無關聯，容器會在執行方法之前啓動新的作業事件。

Required 屬性適用於大多數作業事件。因此，您可能想要將該屬性作爲預設值使用，至少在開發的早期階段這樣做。由於作業事件屬性具有宣告性，因此，以後您可以輕鬆地對其進行變更。

RequiresNew

如果用戶端正在作業事件中執行，並呼叫企業 Bean 方法，則容器會採取以下步驟：

- 暫停用戶端的作業事件
- 啓動新的作業事件
- 將呼叫授權給方法
- 方法完成之後，繼續用戶端的作業事件

如果用戶端與作業事件無關聯，容器會在執行方法之前啓動新的作業事件。

如果要保證方法總在新的作業事件中執行，請使用 RequiresNew 屬性。

Mandatory

如果用戶端正在作業事件中執行，並呼叫企業 Bean 方法，則方法會在用戶端的作業事件中執行。如果用戶端與作業事件無關聯，容器會拋出 TransactionRequiredException。

如果企業 Bean 方法必須使用用戶端的作業事件，則使用 Mandatory 屬性。

NotSupported

如果用戶端正在作業事件中執行，並呼叫企業 Bean 方法，則容器在呼叫方法之前會暫停用戶端的作業事件。方法完成之後，容器會繼續用戶端的作業事件。

如果用戶端與作業事件無關聯，容器在執行方法之前不會啓動新的作業事件。

對於不需要作業事件的方法，請使用 NotSupported 屬性。由於作業事件包含耗用時間，因此該屬性可能會提昇效能。

Supports

如果用戶端正在作業事件中執行，並呼叫企業 Bean 方法，則方法會在用戶端的作業事件中執行。如果用戶端與作業事件無關聯，容器在執行方法之前不會啓動新的作業事件。

由於方法的作業事件行爲可能變化，因此您應該慎用 Supports 屬性。

Never

如果用戶端正在作業事件中執行，並呼叫企業 Bean 方法，則容器會拋出 `RemoteException`。如果用戶端與作業事件無關聯，容器在執行方法之前不會啓動新的作業事件。

屬性摘要

下表概括作業事件屬性的作用。T1 與 T2 作業事件均由容器控制。T1 作業事件與在企業 Bean 中呼叫方法的用戶端相關聯。在多數情況下，用戶端是另一個企業 Bean。T2 作業事件由容器在方法執行前的瞬間啓動。

在最後一欄中，術語「無」表示企業方法不在由容器控制的作業事件中執行。但是，這種企業方法中的資料庫呼叫可能由 DBMS 的作業事件管理員控制。

表格 9-1 作業事件屬性

作業事件屬性	用戶端的作業事件	企業方法的作業事件
Required	無	T2
	T1	T1
RequiresNew	無	T2
	T1	T2
Mandatory	無	錯誤
	T1	T1
NotSupported	無	無
	T1	無
Supports	無	無
	T1	T1

設定作業事件屬性

由於作業事件屬性儲存在部署描述元中，因此可以在 J2EE 應用程式開發的以下幾個階段中進行變更：企業 Bean 的建立階段、應用程式的組譯階段以及部署階段。但是，開發人員還應該在建立 Bean 時負責指定屬性。只有將各元件組譯到大型應用程式的應用程式開發人員才能修改屬性。部署 J2EE 應用程式的人員不負責指定作業事件屬性。

您可以爲整個企業 Bean 或個別方法指定作業事件屬性。如果您爲方法指定了一個屬性，而爲 Bean 指定了另一個屬性，則方法屬性優先。爲個別方法指定屬性時，Bean 的類型與需求不一致。階段作業 Bean 需要爲企業方法定義的屬性，但是禁用 create 方法的屬性。實體 Bean 需要企業方法、create 方法、remove 方法以及 finder 方法的作業事件屬性。訊息導引 Bean 需要 `onMessage` 方法的作業事件屬性 (`Required` 或 `NotSupported`)。

回轉容器管理作業事件

有兩種方法可以回轉容器管理作業事件。第一種，如果拋出一個系統異常，容器將會自動回轉作業事件。第二種，透過呼叫 `EJBContext` 介面中的 `setRollbackOnly` 方法，Bean 方法指導容器回轉作業事件。如果 Bean 拋出一個應用程式異常，則回轉不能自動進行，但可以呼叫 `setRollbackOnly` 來引發回轉。

在以下範例中，`BankEJB` 範例的 `transferToSaving` 方法展示 `setRollbackOnly` 方法。如果發生活期存款帳戶結欠，`transferToSaving` 會呼叫 `setRollBackOnly` 並拋出一個應用程式異常 (`InsufficientBalanceException`)。`updateChecking` 與 `updateSaving` 方法會更新資料庫表格。如果更新失敗，這些方法會拋出一個 `SQLException`，而 `transferToSaving` 方法會拋出一個 `EJBException`。由於 `EJBException` 是系統異常，因此，其會導致容器自動回轉作業事件。以下爲 `transferToSaving` 方法的程式碼：

```
public void transferToSaving(double amount) throws
    InsufficientBalanceException  {

    checkingBalance -= amount;
    savingBalance += amount;

if (checkingBalance < 0.00) {
context.setRollbackOnly();

throw new InsufficientBalanceException();
}
    try {
        updateChecking(checkingBalance);

    updateSaving(savingBalance);
    } catch (SQLException ex) {
        throw new EJBException
            ("Transaction failed due to SQLException: "
            + ex.getMessage());
    }
}
```

當容器回轉作業事件的時候，總會復原 SQL 呼叫在作業事件內對資料所做的變更。但是，只有在實體 Bean 中，容器才會復原對實例變數所做的變更。(透過自動呼叫實體 Bean `ejbLoad` 方法 [經由資料庫載入實例變數]，可以執行該項作業。) 發生回轉的時候，階段作業 Bean 必須明確地重設在作業事件內變更的所有實例變數。若要重設階段作業 Bean 的實例變數，最簡單的方法是執行 `SessionSynchronization` 介面。

您也可以透過指令行介面傳送作業事件 ID，來回轉作業事件。如需更多詳細資訊，請參閱第 218 頁的「使用指令行介面管理作業事件」。

同步化階段作業 Bean 的實例變數

`SessionSynchronization` 介面 (選擇性) 可讓您同步化實例變數及其在資料庫中相應的值。容器在作業事件的每個主要階段都會呼叫 `SessionSynchronization` 方法 — `afterBegin`、`beforeCompletion` 以及 `afterCompletion`。

`afterBegin` 方法會在新的作業事件開始後通知實例。容器首先呼叫 `afterBegin`，然後才呼叫作業事件中的第一個企業方法。`afterBegin` 方法是經由資料庫載入實例變數的絕佳位置。例如，`BankBean` 類別使用 `afterBegin` 方法載入 `checkingBalance` 與
`savingBalance` 變數：

```
public void afterBegin() {

    System.out.println("afterBegin()");
    try {
        checkingBalance = selectChecking();
        savingBalance = selectSaving();
    } catch (SQLException ex) {
        throw new EJBException("afterBegin Exception: " +
            ex.getMessage());
    }
}
```

在企業方法完成之後、而作業事件確定之前的瞬間，容器呼叫 `beforeCompletion` 方法。`beforeCompletion` 方法是階段作業 Bean 回轉作業事件的最後機會 (透過呼叫 `setRollbackOnly`)。如果還沒有透過實例變數的值更新資料庫，則階段作業 Bean 會使用 `beforeCompletion` 方法執行這項作業。

`afterCompletion` 方法表示作業事件已經完成。它只有一個單一的布林參數，如果作業事件被確定，其值爲 true，如果作業事件被回轉，則其值爲 false。如果發生回轉，階段作業 Bean 便可以使用 `afterCompletion` 方法，重新顯示其資料庫中的實例變數：

```
public void afterCompletion(boolean committed) {

    System.out.println("afterCompletion:" + committed);
    if (committed == false) {
        try {
            checkingBalance = selectChecking();
            savingBalance = selectSaving();
} catch (SQLException ex) {
            throw new EJBException("afterCompletion SQLException:
            " + ex.getMessage());
}
}
}
```

容器管理作業事件中禁用的方法

您不應該呼叫任何可能干擾作業事件界限 (由容器設定) 的方法。禁用的方法列示如下：

- `java.sql.Connection` 的 commit、`setAutoCommit` 以及 rollback 方法
- `javax.ejb.EJBContext` 的 `getUserTransaction` 方法
- `javax.transaction.UserTransaction` 的任何方法

但是，您可以使用這些方法在 Bean 管理式的作業事件中設定界限。

Bean 管理式的作業事件

在 Bean 管理式的作業事件中，階段作業或訊息導引 Bean 中的程式碼明確地標示了作業事件的界限。實體 Bean 不能具有 Bean 管理式的作業事件，其必須使用容器管理作業事件作爲替代。儘管具有容器管理作業事件的 Bean 需要較少的編碼，但是它們有一個限制條件：方法在執行時可以與單一的作業事件關聯，也可以根本不與任何作業事件關聯。如果該限制條件令您在編碼 Bean 時感到困難，應該考慮使用 Bean 管理式的作業事件。

以下虛擬碼展示您可以透過 Bean 管理式的作業事件取得的細紋控制。透過檢查各種情況，虛擬碼決定是否啓動或停止企業方法內的作業事件。

```
begin transaction
...
update table-a
...
if (condition-x)
```

```
    commit transaction
else if (condition-y)
    update table-b
    commit transaction
else
    rollback transaction
    begin transaction
    update table-c
    commit transaction
```

作業事件服務管理

您可以使用管理介面或指令行介面管理作業事件。

本章節包含以下主題：

- 使用管理介面管理作業事件
- 使用指令行介面管理作業事件

使用管理介面管理作業事件

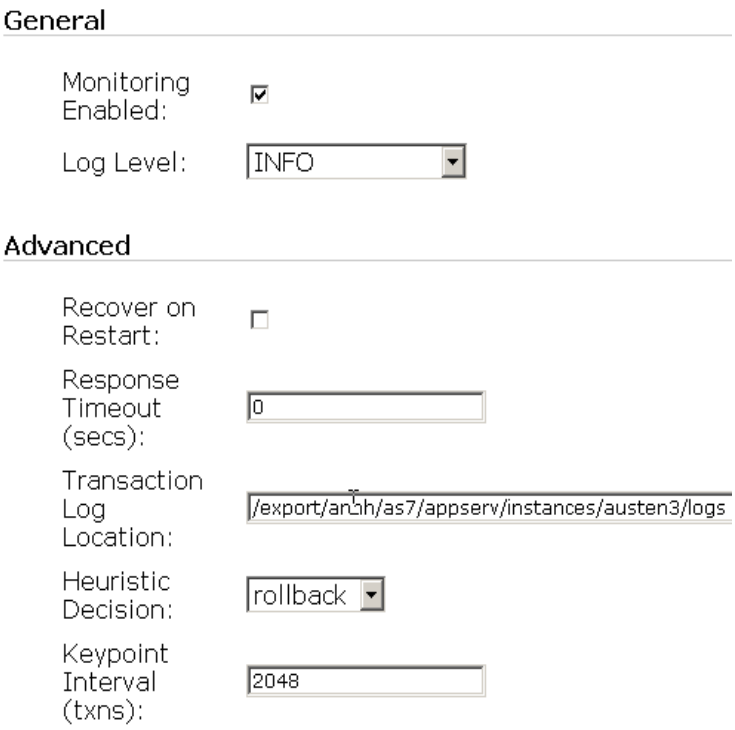
使用管理介面，您可以啓用對作業事件的監視、設定其日誌層級以及指定其進階選項。

您可以控制實例的作業事件服務屬性，例如恢復策略和逾時。您在此處指定的特性與配置儲存在 `server.xml` 檔案中。

若要配置作業事件服務選項，請執行以下工作：

1. 在管理介面的左窗格中，開啓您要修改其作業事件配置的 Sun ONE Application Server 實例樹。
2. 從顯示的 J2EE 服務清單中選取 [Transaction Service]。在管理介面右窗格的視圖「配置作業事件服務選項」中，您將看到以下視窗：

圖 9-2 配置作業事件服務選項



3. 若要啓用對作業事件的監視，請標示 [Monitoring Enabled] 核取方塊。下表列示 Java 作業事件服務中可被監視的功能：

表格 9-2 Java 作業事件服務的可監視屬性

特性	類型	描述
transactionsCompleted	int	啓用監視後所完成的作業事件數目
transactionsRolledBack	int	啓用監視後已回轉的作業事件數目
transactionsRecovered	int	啓用監視後已恢復的作業事件數目
transactionsInFlight	int	正在處理的作業事件數目
timeStamp	long	以毫秒表示，記錄產生統計資料的時間。這將是由 `System.getCurrentTimeInMillis()` 報告的任何內容。

4. 從 [Log Level] 下拉式清單中，選取您要爲作業事件設定的日誌層級。如需有關日誌層級及其併入方式的更多資訊，請參閱第 5 章 「使用記錄功能」。

5. 標示 [Recover on Restart] 核取方塊，以在伺服器重新啓動時自動恢復失敗的作業事件。當資源在作業事件確定協定的關鍵時刻不可存取時，作業事件也許不會完成，並繼續駐留在作業事件日誌檔中。如果已經標示了該核取方塊，伺服器會在重新啓動時嘗試恢復中斷的作業事件。如果有關資源仍然不可存取，也許會延遲伺服器的重新啓動。依預設，不標示該核取方塊。

6. 對於具有容器管理作業事件的企業 Bean，您可以透過設定作業事件逾時 (秒) 特性的值來控制作業事件逾時間隔時間。

   如果將該特性的值設定爲 0，則作業事件便不會逾時。

   在 [Transaction Timeout] (以秒表示) 欄位中，指定作業事件逾時間隔時間。如果作業事件沒有在指定的時間內完成，其便會被回轉。如果該屬性的值設定爲 0，則作業事件便不會逾時。

7. 在 [Transaction Log Location] 欄位中，指定要儲存日誌檔的絕對目錄路徑。您需要重新啓動伺服器，以使新的作業事件日誌目錄生效。

8. 從 [Heuristic Decision] 下拉式方塊中，選取您要套用至作業事件的啓發式決策。從指示的選項中選取 [Commit] 或 [Rollback]，以便在不能明確地確定作業事件結果時，指定應用程式伺服器在恢復期間應該如何確定一個不確定作業事件的結果。如果將 [Heuristic Decision] 設定爲 [Rollback]，其便會將作業事件回轉。在某些情況下，也可以確定這樣的作業事件。

9. 在 [Keypoint Interval (transactions)] 欄位中，指定日誌中要點事件作業之間作業事件的數目。透過移除已完成作業事件的項目，並壓縮檔案，要點事件作業可以縮小作業事件日誌檔的大小。如果該屬性值較大，則作業事件日誌檔也會較大，但是減少要點事件作業意味著更高的潛在效能。如果屬性值較小 (例如，100)，則日誌檔也會較小，但是由於要點事件作業的頻率較大，會稍微降低效能。

使用指令行介面管理作業事件

您可以使用指令行介面 (CLI) 管理並監視資料庫作業事件，將在以下章節中加以解釋：

- 列示執行中的作業事件
- 管理作業事件
- 凍結作業事件服務
- 監視作業事件

這些章節解釋如何使用指令行介面管理並監視作業事件。

列示執行中的作業事件

以下指令應該用於取得執行中的作業事件資料 (假定您處於多模式下，並且已設定使用者名稱與密碼)：

```
- asadmin> get --monitor
<instanceName>.transaction-service.inflight-tx
```

多行輸出的形式爲：

```
Transaction Id State Elapsed Time (ms)

txnid1 Prepared 20

txnid2 Active 100

txnid3 Active 120

  ... ... ...
```

管理作業事件

在列示執行中的作業事件中給定的範例內，讓我們假定您要透過以下作業事件 Id 來回轉作業事件：`txn-ids`、`txnid2` 以及 `txnid3`。回轉所選作業事件的指令範例如下：

```
asadmin> set --monitor
<instanceName>.transaction-service.rollback-list=txnid2,txnid3
```

凍結作業事件服務

若要凍結作業事件服務，請執行以下指令：

```
asadmin> set --monitor
<instanceName>.transaction-service.freeze=true
```

凍結作業事件服務期間，應用程式伺服器中的作業事件管理員會暫停執行中的所有作業事件。在生產部署系統中，不建議使用凍結指令。

若要取消凍結作業事件服務，請執行以下指令：

```
asadmin> set --monitor
<instanceName>.transaction-service.freeze=false
```

當作業事件服務重新設定爲進行動作時，系統將在停止處開始繼續作業。如果活性系統保持的凍結狀態時間太長，某些資料庫連線可能逾時，導致作業事件被回轉。

監視作業事件

若要取得作業事件的監視資料 (包括執行中作業事件的資料)，請執行以下指令：

```
asadmin> get --monitor <instanceName>.transaction-service.*
```

當您執行該指令時，如果沒有作用中的作業事件，將會得到以下輸出結果：

```
total-tx-completed = 5
total-tx-rolledback = 2
total-tx-inflight = 0
isFrozen = false
tx-inflight = No active transactions found.
```

當您執行該指令時，如果存在作用中的作業事件，將會得到以下輸出結果：

```
total-tx-completed = 5
total-tx-rolledback = 2
total-tx-inflight = 2
isFrozen = false
tx-inflight =
Transaction Id State Elapsed Time (ms)
txnid1 Prepared 500
txnid2 Active 360
```

第 10 章

配置命名與資源

本章描述 Sun ONE Application Server 使用的 J2EE 資源，並論述建立與管理這些資源所使用的方法。

本章包含以下主題：

- 關於 J2EE 命名服務與資源
- 關於 Java 命名與目錄介面 (JNDI)
- 關於持續性管理程式資源
- 關於 JDBC 資源
- 關於 Java 郵件資源

關於 J2EE 命名服務與資源

包含 EJB、Web 應用程式元件與應用程式用戶端的 J2EE 應用程式可以存取廣泛的資源，例如，資源管理程式、資料來源 (例如 SQL 資料來源)、連線 Factory、郵件階段作業、Java Message Service (JMS) 目標物件以及 URL 連線 Factory。J2EE 平台透過 Java 命名與目錄 (JNDI) 命名服務向應用程式展示此類資源。

Sun ONE Application Server 可讓您建立並管理以下 J2EE 資源：

- JDBC 資料來源
- Java 郵件階段作業
- JMS 目標

JDBC 資料來源

JDBC 資料來源是 J2EE 資源，您可以使用 Sun ONE Application Server 來建立並管理該資源。

JDBC API 是與關係資料庫系統具有連接性的 API。JDBC API 具有兩個元件：

- 應用程式元件存取資料庫使用的應用程式層級介面。
- 將 JDBC 驅動程式貼附於 J2EE 平台的服務提供者介面。

JDBC `DataSource` 物件表示以 Java 程式設計語言撰寫的資料來源。從基本上來說，資料來源是儲存資料的設備。可以如大公司複雜的資料庫一樣複雜，也可以如具有欄和列的檔案一樣簡單。JDBC 資料來源是可以透過 Sun ONE Application Server 建立並管理的 J2EE 資源。

如需關於 JDBC 資料來源的更多資訊，請參閱第 242 頁的「關於 JDBC 資源」。

Java 郵件階段作業

JMS 目標是可以透過 Sun ONE Application Server 建立並管理的 J2EE 資源。

許多網際網路應用程式需要發送電子郵件通知的功能。因此，J2EE 平台包含 JavaMail API 及其 JavaMail 服務提供者，JavaMail API 可讓應用程式元件傳送網際網路郵件。JavaMail API 具有兩個元件：

- 用於傳送郵件的應用程式元件所使用的應用程式層級介面。
- 在 J2EE SPI 層級上使用的服務提供者介面。

Java 郵件階段作業是可以透過 Sun ONE Application Server 建立並管理的 J2EE 資源。如需關於 Java 郵件階段作業的更多資訊，請參閱第 261 頁的「關於 Java 郵件資源」。

JMS 目標

Java Messaging Service (JMS) 是標準的訊息傳送 API，支援可靠的點對點訊息傳送以及出版訂閱模型。此規格需要可以執行點對點訊息傳送與出版訂閱訊息傳送的 JMS 提供者。

JMS 提供兩種一般類型的管理物件：連線 Factory 與目標。雖然這兩者均封裝特定的提供者資訊，但是其在 JMS 用戶端內的功能大不相同。連線 Factory 用於建立訊息伺服器的連線，而目標物件用於識別 JMS 訊息傳送服務使用的實體目標。

關於 Java 命名與目錄介面 (JNDI)

本章節論述 Java 命名與目錄介面 (JNDI)。Java 命名與目錄介面 (JNDI) 是存取各種命名與目錄服務的應用程式設計介面 (API)。J2EE 元件透過呼叫 JNDI 查找方法來尋找物件。

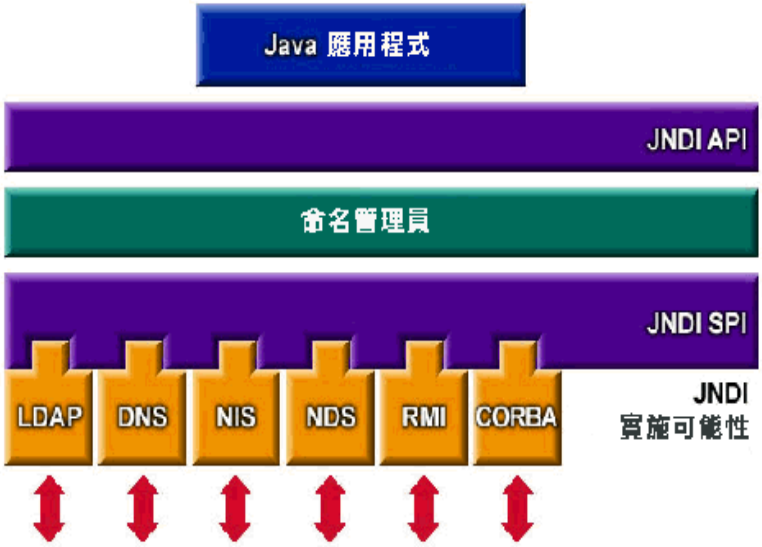
本章節涵蓋以下主題：

- JNDI 架構
- J2EE 命名服務
- 命名參考與連結資訊
- J2EE 標準部署描述元中的命名參考
- JNDI 連線 Factory

JNDI 架構

JNDI 架構由應用程式設計介面 (API) 與服務提供者介面 (SPI) 組成。Java 應用程式使用 JNDI API 存取各種命名與目錄服務。SPI 可讓各種命名與目錄服務以透明方式被插接，從而允許使用 JNDI API 的 Java 應用程式存取其服務。下圖「JNDI 架構簡介」列舉可透過 JNDI API 存取的服務：

圖 10-1 JNDI 架構簡介



J2EE 命名服務

JNDI 名稱是一個易懂的物件名稱。透過 J2EE 伺服器提供的命名與目錄服務，這些名稱與其物件相對應。因爲 J2EE 元件透過 JNDI API 存取此服務，通常物件易懂名稱指的是物件 JNDI 名稱。Pointbase 資料庫的 JNDI 名稱是 `jdbc/Pointbase`。啓動該資料庫後，Sun ONE Application Server 會從配置檔案中讀取資訊，並自動在名稱空間中加入 JNDI 資料庫名稱。

需要 J2EE 應用程式用戶端、企業 Bean 與 Web 元件來存取 JNDI 命名環境。

此應用程式元件命名環境是在部署或組合時允許自訂應用程式元件企業邏輯的一種機制。使用此應用程式元件環境可自訂應用程式元件，而無需存取或變更應用程式元件源代碼。

J2EE 容器執行應用程式元件環境，並將其作爲 JNDI 命名環境提供給應用程式元件實例。應用程式元件環境使用如下：

- 應用程式元件企業方法使用 JNDI 介面存取此環境。應用程式元件提供者在部署描述元中宣告所有環境項目，這些項目是應用程式元件期望在運行時間在其環境中提供的項目。
- 此容器提供儲存應用程式元件環境的 JNDI 命名環境實現，並且還提供讓部署程式建立並管理每個應用程式元件環境的工具。
- 部署程式使用容器提供的工具初始化在應用程式元件部署描述元中宣告的環境項目。此部署程式可以設定並修改環境項目的值。
- 在運行時間，此容器可讓應用程式元件實例使用環境命名環境。此應用程式元件實例使用 JNDI 介面取得環境項目值。

每個應用程式元件均定義其自身的環境項目集。在同一容器中應用程式元件的所有實例共用同樣的環境項目。不允許應用程式元件實例在運行時間修改此環境。如需關於 J2EE 容器 (例如 Web 容器與 EJB 容器) 使用 JNDI 命名服務來查看物件的更多資訊，請參閱第 185 頁的「配置 J2EE 容器」。

命名參考與連結資訊

資源參考是部署描述元中的元素，用於識別資源的元件編碼名稱。更具體地來說，編碼名稱參考資源的連線 Factory。在下面一節所提供的範例中，資源參考名稱爲 `jdbc/SavingsAccountDB`。

資源的 JNDI 名稱與資源參考名稱不相同。此命名方法需要您在部署之前對映這兩個名稱，但是也會分離資源中的元件。因爲此分離作用，以後元件需要存取另一個資源時，您不必變更程式碼中的名稱。這種靈活性也可讓您更容易從預先存在的元件中組合 J2EE 應用程式。

下表「JNDI 查找及其關聯的參考」列示 JNDI 查找以及與其關聯的、由 Sun ONE Application Server 使用的 J2EE 資源之參考。

表格 10-1 JNDI 查找及其關聯的參考

JNDI 查找名稱	關聯的參考
`java:comp/env`	應用程式環境項目
`java:comp/env/jdbc`	JDBC DataSource 資源管理程式連線 Factory
`java:comp/env/ejb`	EJB 參考
`java:comp/UserTransaction`	UserTransaction 參考
`java:comp/env/mail`	JavaMail 階段作業連線 Factory
`java:comp/env/url`	URL 連線 Factory
`java:comp/env/jms`	JMS 連線 Factory 與目標
`java:comp/ORB`	應用程式元件共用的 ORB 實例

J2EE 標準部署描述元中的命名參考

命名參考是應用程式使用的字串，用於查找給定的命名環境中之物件。對於每個 J2EE 應用程式來說，均具有命名環境，並且在標準元件部署描述元中配置參考。本章節描述在 Sun ONE Application Server 中使用的標準部署描述元功能。本節包含以下主題：

- 應用程式環境項目
- EJB 參考
- 資源管理程式連線 Factory 的參考
- 資源環境參考
- UserTransaction 參考
- COSNaming 服務

應用程式環境項目

使用 <env-entry> 定義的環境項目提供指定 J2EE 應用程式部署時間參數的一種方法。請注意，在 Web 應用程式中，可以使用 <context-param> 定義 Servlet 環境初始化參數，但是使用 <env-entry> 是優選的方法，因爲應用程式部署程式可以透過明確地指定這些應用程式參數的名稱、類型與值來配置它們。

以下範例描述在 J2EE 標準部署描述元中指定的 <env-entry> 語法：

```
<env-entry>
<description> Send pincode by mail </description>
<env-entry-name> mailPincode </env-entry-name>
<env-entry-value> false </env-entry-value>
<env-entry-type> java.lang.Boolean </env-entry-type>
</env-entry>
```

<env-entry-type> 標籤指定此項目的完整類別名稱。此處爲 snippet 碼，使用應用程式元件中的 JNDI 查找 <env-entry> (snippet 術語指的是 Servlet/JSP 或實體 Bean 或 IIOP 應用程式用戶端)：

```
Context initContext = new InitialContext();
Boolean mailPincode = (Boolean)
initContext.lookup("java:comp/env/mailPincode");

// 使用者可以在子環境中使用相對名稱
Context envContext = initContext.lookup("java:comp/env");
Boolean mailPincode = (Boolean)
envContext.lookup("mailPincode");
```

EJB 參考

除部署描述元支援外，JNDI 命名服務可讓應用程式使用「邏輯」名稱 (稱爲 EJB 參考) 對映至企業 Bean 的本地介面，如以下範例所述：

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
<ejb-link> EmployeeEJB </ejb-link>
</ejb-ref>
```

應用程式元件 (例如 JSP) 可以使用 JNDI 存取 EJB 本地物件，如以下範例所述：

```
Context initContext = new InitialContext();
Context envContext = initContext.lookup("java:comp/env");
Object result = envContext.lookup("ejb/EmplRecord");
EmployeeRecordHome emplRecordHome = (EmployeeRecordHome)
javax.rmi.PortableRemoteObject.narrow(result,
EmployeeRecordHome.class);
```

ejb-ref-name 元素定義在應用程式碼中使用的字串 (如以上提供的範例)。ejb-link 元素將此參考連結至使用在 ejb-jar.xml 中定義的實體 Bean 之 ejb-name 元素定義的目標企業 Bean。在不修改應用程式部署描述元或企業 Bean 描述元的情況下，也可能提供此連結。

資源管理程式連線 Factory 的參考

Factory 是應需要建立其他物件的物件。資源 Factory 可以建立資源物件 (例如資料庫連線或訊息服務連線)。透過標準部署描述元中的 <resource-ref> 元素來配置這些連線。

以下範例描述使用 Factory 的資訊：

範例 A：

宣告對 JDBC 連線 Factory 的參考，此 Factory 會傳回 javax.sql.DataSource 物件類型：

```
<resource-ref>
<description> Primary database </description>
<res-ref-name> jdbc/primaryDB </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

範例 B：

以下是 JavaMail 階段作業資源 Factory 的範例參考：

```
<resource-ref>
<description> mail Session </description>
<res-ref-name> mail/Session </res-ref-name>
<res-type> javax.mail.Session </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

<res-type> 是資源 Factory 的完全合格類別名稱。<res-auth> 變數的值可以指定爲 Container 或 Application。若要瞭解有關配置 Java 郵件階段作業資源 Factory 的更多資訊，請參閱第 261 頁的「關於 Java 郵件資源」。

如果指定 Container，則 Web 容器在將資源 Factory 連結至 JNDI 查找登錄之前會處理此認證。如果指定 Application，則 Servlet 必須以程式化方式處理認證。在描述資源類型的單獨子環境下，查找的不同資源 Factory 如下所示：

- jdbc/ 表示 JDBC javax.sql.DataSource Factory
- jms/ 表示 JMS javax.jms.QueueConnectionFactory 或 javax.jms.TopicConnectionFactory

- `mail/` 表示 `JavaMail javax.mail.Session factory`
- `url/` 表示 `java.net.URL factory`

以下爲 snippet 碼，用於取得自應用程式元件與處理認證的容器之間的 JDBC 連線：

```
InitialContext initContext = new InitialContext();
DataSource source =
(DataSource) initContext.lookup("java:comp/env/jdbc/primaryDB");
Connection conn = source.getConnection();
```

請注意，爲了確保這些資源參考可以使用，在執行期間，`res-ref-name` 必須對映至有效的資源 Factory。

資源環境參考

資源環境參考提供透過 JNDI 查找，來存取與資源關聯的管理物件之方法。例如，應用程式可能需要存取 JMS Destination 物件。在標準部署描述元中定義的 `<resource-env-ref>` 元素可讓應用程式宣告資源需求。

`<resource-env-ref>` 與 `<resource-ref>` 元素之間的主要區別在於不存在特定的資源認證需求，資源 Factory 描述元必須備份這兩個元素。

範例：

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

以下程式碼部分可讓您存取 JMS Topic 物件：

```
InitialContext initContext = new InitialContext();
javax.jms.Topic myTopic =
(javax.jms.Topic) initContext.lookup("java:comp/env/jms/MyTopic");
```

請注意，爲了可以使用這些 `resource-env-ref` 變數，管理員必須在運行時間使目標資源 Factory 可用。如需關於存取 JMS 主題與佇列目標的更多資訊，請參閱第 11 章「使用 JMS 服務」。

UserTransaction 參考

J2EE 需要容器提供在 JNDI 名稱 `java:comp/UserTransaction` 下的 `UserTransaction` 物件執行。`UserTransaction` 物件可讓應用程式啓動、確定並中斷作業事件。

若要以程式化方式啓動並執行作業事件，元件會執行 JNDI 查找 `java:comp/UserTransaction`，取得容器預設作業事件協調者的參考。傳回的物件實施 `javax.transaction.UserTransaction` 介面，並可以用於在此程式中開始、確定、回轉作業事件與查詢作業事件狀態。Sun ONE Application Server 中的 JNDI 執行支援作業事件協調者的查找。如需關於 `javax.transaction.UserTransaction` 介面的更多資訊，請參閱第 203 頁的「使用作業事件服務」。

初始命名環境

Sun ONE Application Server 中的命名支援主要以 J2EE 1.3 (具有一些加入的增強功能) 爲基礎。應用程式元件透過 `InitialContext()` 建立初始環境時，Sun ONE Application Server 會傳回作爲應用程式命名環境控點的物件。進而此物件爲 `java:comp/env namespace` 提供子環境。每個應用程式取得自己的名稱空間，即每個應用程式具有 `java:comp/env name` 空間，並且在一個應用程式名稱空間中連結的物件與其他應用程式中連結的物件不相衝突。

COSNaming 服務

EJB 互用性協定需要使用 COSNaming 協定透過 JNDI API 查找 EJB 物件。

需要 EJB 容器能夠在 CORBA `CosNaming` 服務中出版 `EJBHome` 物件參考。`CosNaming` 服務必須在定義的 `CosNaming` 模組中執行 IDL 介面，並且必須允許用戶端透過 IIOP 進行解譯與列示作業。

`CosNaming` 服務必須滿足 CORBA 互用名稱服務規格中的需求，以向主機、連接埠與物件鍵值提供其根 `NamingContext` 物件。`CosNaming` 服務必須能夠對在廣告主機、連接埠與物件鍵值中的根 `NamingContext` 提供 IIOP 呼叫服務。

需要用戶端容器 (即 EJB、Web 或應用程式用戶端容器) 包含 JNDI `CosNaming` 服務提供者，該程式使用在互用名稱服務規格中定義的機制與伺服器 `CosNaming` 服務進行聯絡，並使用標準 `CosNaming` API 解譯 `EJBHome` 物件。JNDI `CosNaming` 服務提供者可以或不能使用 JNDI SPI 架構。JNDI `CosNaming` 服務提供者必須透過從以下 URL 建立物件參考，來存取伺服器 `CosNaming` 服務的根 `NamingContext`：

`corbaloc:iiop:1.2@<host>:<port>/<objectkey>` (其中 `<host>`、`<port>` 與 `<objectkey>` 是伺服器 `CosNaming` 服務宣傳的根 `NamingContext` 之對應值)，或使用對等的機制。

在部署期間，用戶端容器的開發人員應該取得伺服器 CosNaming 服務的主機、連接埠與物件鍵值以及用戶端元件部署描述元中的每個 ejb-ref 元素之伺服器 EJBHome 物件的 CosNaming 名稱 (例如，透過瀏覽伺服器名稱空間)。ejb-ref-name (JNDI 查找呼叫中的用戶端碼使用) 應該連接至 EJBHome 物件的 CosNaming 名稱。在運行時間，用戶端元件 JNDI 查找呼叫使用 CosNaming 服務提供者，此提供者與伺服器 CosNaming 服務進行聯絡，解譯 CosNaming 名稱，並向用戶端元件傳回 EJBHome 物件參考。

由於 EJBHome 物件名稱的範圍在提供的主機與連接埠中存取的 CosNaming 服務之名稱空間內，所以不必結合使用用戶端名稱空間與伺服器容器。

使用 CosNaming 的優點是可以更好地與互用性所需的 IIOP 基礎架構整合，並與非 J2EE CORBA 用戶端與伺服器互用。由於 CosNaming 僅儲存 CORBA 物件，因此，供應商可能將使用其他企業目錄服務存取其他資源。

Sun ONE Application Server 結合了基於 J2EE 1.3 規格的 JNDI 之所有命名資源。

CosNaming 提供者　若要支援全域 JNDI 名稱空間 (IIOP 應用程式用戶端可以存取)，Sun ONE Application Server 包含基於 J2EE 的 CosNaming 提供者，它支援 CORBA 參考 (遠端 EJB 參考) 的連結。傳回至 IIOP 用戶端的 InitialContext 是 CosNaming 提供者。Sun ONE Application Server 伺服器實例註冊 IIOP 用戶端查找與連結的實體 Bean。

請注意，Sun ONE Application Server 將儲存在 CosNaming 與本機 JNDI 命名環境中的物件視爲暫時物件：即，在每個伺服器啓動以及應用程式重新載入時，所有相關的物件將再次與名稱空間連結。若要瞭解配置 CORBA/IIOP 用戶端支援的更多資訊，請參閱第 297 頁的「配置 CORBA/IIOP 用戶端伺服器」。

JNDI 連線 Factory

對於 J2EE Web 應用程式，web.xml 檔案中的部署描述元是定義應用程式環境項目參考、資源管理程式 (例如 SQL 資料來源) 連線 Factory 參考或 EJB 參考的版面配置區。應用程式使用 J2EE 容器提供的 JNDI InitialNamingContext 查找這類參考。僅對部署描述元進行變更，即不需存取或修改應用程式來源碼，不同的應用程式伺服器環境就可攜帶這些應用程式。同樣，J2EE 需要實體 Bean (ejb-jar.xml) 的部署描述元與 IIOP 應用程式用戶端 (application-client.xml) 成爲這些 JNDI 命名參考的主要設備。

連線 Factory 是產生連線物件的物件，其中連線物件可讓 J2EE 元件存取資源。資料庫的連線 Factory 是 javax.sql.DataSource 物件，該物件可建立 java.sql.Connection 物件。

在 Sun ONE Application Server 中，您可以配置存取以下資源與資源 Factory 的方法：

- JDBC 連線 Factory
- 基於 MQ 的 JMS 連線 Factory
- JavaMail 階段作業連線 Factory
- JCA 連接器 Factory
- 一般的、自訂使用者寫入的資源物件 Factory
- 支援外部資源儲存庫，例如 LDAP

`server.xml` 的 `<resources> </resources>` 標籤中指定所有 Sun ONE Application Server 資源 Factory，並且這些 Factory 具有使用 `jndi-name` 屬性指定的 JNDI 名稱。此屬性用於註冊伺服器範圍的名稱空間中的 Factory。部署程式可以使用 `resource-ref-mapping` 元素，將使用者指定的應用程式特定資源參考名稱 (在 `resource-ref` 或 `resource-env-ref` 元素中宣告) 對映至這些伺服器範圍的資源 Factory。這樣可作出關於在給定的應用程式中使用哪個 JDBC 驅動程式 (與其他資源 Factory) 的部署時間決定。

自訂資源存取本機 JNDI 儲存庫，而外部資源存取外部 JNDI 儲存庫。兩種類型的資源均需要使用者指定的 Factory 類別元素、JNDI 名稱屬性等。在本章節中，我們將論述如何配置 J2EE 資源的 JNDI 連線 Factory 資源以及如何存取這些資源。

本章節涵蓋以下主題：

- 建立自訂資源的步驟
- 建立外部 JNDI 資源
- 存取外部 JNDI 儲存庫
- 對映應用程式資源參考
- 關於 URL 連線 Factory 資源
- 對映應用程式資源環境參考
- 對映 EJB 參考

建立自訂資源的步驟

在 `server.xml` 中定義的 `custom-resource` 元素提供指定自訂伺服器範圍資源物件 Factory 的方式。這類物件 Factory 實施 `javax.naming.spi.ObjectFactory` 介面。此元素與伺服器範圍名稱空間中使用的 JNDI 名稱 (與其他 Sun ONE Application Server 資源一樣透過 jndi-name 子元素指定)、JNDI 類型、資源 Factory 類別名稱以及用於實例化同一資源 Factory 類別的標準特性集關聯。

以下範例闡明 `javax.naming.spi.ObjectFactory` 介面的執行：

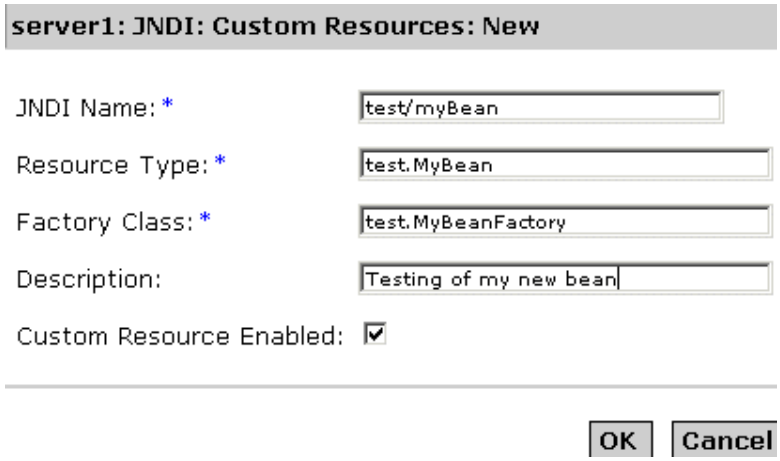
```
<resources> <custom-resource jndi-name="test/myBean"
res-type="test.MyBean"factory-class="test.MyBeanFactory"
enabled="true">

<property name="foo" value="test custom bean prop" />
</custom-resource>
</resources>
```

您需要確保資源參考的環境參考與 EJB 參考連結至使用 `server.xml` 中的 `custom-resource` 與 `external-jndi-resource` 標籤定義的配置伺服器範圍資源。動態重新部署應用程式元件在 JNDI 命名環境中存在問題。Sun ONE Application Server 將釋放所有特定應用程式的參考，並將所有的新參考重新連結至新安裝的應用程式命名環境中。

使用管理介面建立自訂資源的步驟：

1. 在管理介面的左窗格中，開啓您要修改其 JNDI 配置的 Sun ONE Application Server 實例。
2. 開啓 [JNDI] 標籤，然後按一下 [Custom Resources]。如果已建立了任何自訂資源，將在右窗格中列示這些資源。若要建立新自訂資源，請按一下 [New]。您會在管理介面的右窗格中看到 「[JNDI Custom Resources] 頁面」：

圖 10-2 [JNDI Custom Resources] 頁面



3. 在 [JNDI Name] 欄位中，輸入用於存取資源的名稱。該名稱將在 JNDI 命名服務中註冊。
4. 在 [Resource Type] 欄位中，輸入完全合格的類型定義，如以上範例所示。[Resource Type] 定義格式應該如下所示：`xxx.xxx`。
5. 在 [Factory Class] 欄位中，輸入您要建立的自訂資源之 Factory 類別名稱。[Factory Class] 是使用者指定的 Factory 類別名稱。此類別執行 `javax.naming.spi.ObjectFactory` 介面。
6. 在 [Description] 欄位中，輸入要建立的資源之描述。此描述是字串值，最多可以爲 250 個字元。
7. 標示 [Custom Resource Enabled] 核取方塊，以啓用自訂資源。
8. 按一下 [OK]，以儲存自訂資源。

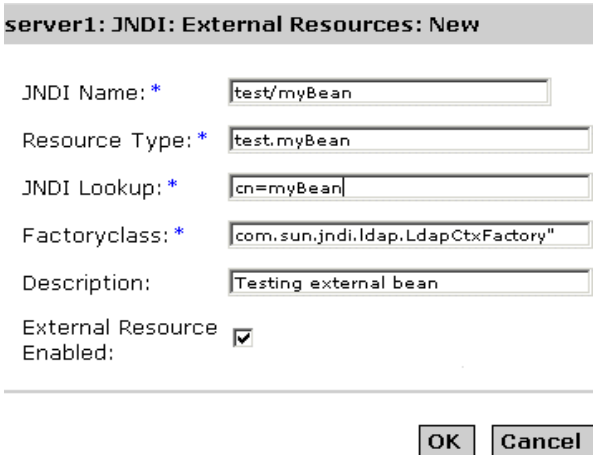
建立外部 JNDI 資源

使用管理介面建立外部資源的步驟：

1. 在管理介面的左窗格中，開啓您要修改其 JNDI 配置的 Sun ONE Application Server 實例。
2. 開啓 [JNDI]，並選取 [External Resources]。如果已建立了任何外部資源，會在右窗格中列示這些資源。若要建立新的外部資源，請按一下 [New]。

您會看到以下視窗，如管理介面右窗格中的「[JNDI External Resources] 頁面」所示：

圖 10-3 [JNDI External Resources] 頁面



3. 在 [JNDI Name] 欄位中，輸入用於存取資源的名稱。該名稱將在 JNDI 命名服務中註冊。
4. 在 [Resource Type] 欄位中，輸入完全合格的類型定義，如以上範例所示。[Resource Type] 定義格式應該如下所示：`xxx.xxx`。
5. 在 [JNDI Lookup] 欄位中，輸入要在外部儲存庫中查找的 JNDI 值。例如，如果您要建立連線至外部儲存庫，測試 Bean 類別的外部資源，[JNDI Lookup] 可以讀取 `cn=`*testmybean*。
6. 在 [Factory Class] 欄位中，輸入 JNDI Factory 類別外部儲存庫。例如，`com.sun.jndi.ldap`。此類別執行 javax.naming.spi.`ObjectFactory` 介面。
7. 在 [Description] 欄位中，輸入要建立的資源之描述。此描述是字串值，最多可以爲 250 個字元。
8. 標示 [External Resource Enable] 核取方塊，以啓用外部資源。
9. 按一下 [OK]，以儲存自訂資源。

存取外部 JNDI 儲存庫

通常，在 Sun ONE Application Server 中運行的應用程式需要對外部 JNDI 儲存庫中儲存的資源之存取權。例如，一般 Java 物件可以作爲每個 Java 網目儲存在 LDAP 伺服器中。外部 JNDI 資源元素可讓使用者配置這類外部資源儲存庫。外部 JNDI Factory 必須實施 `javax.naming.spi.InitialContextFactory` 介面。

範例：

```
< 資源 >
<!-- external-jndi-resource 元素指定存取儲存在外部 JNDI 儲存庫中 J2EE 資源的方式。
-- 以下範例闡明存取儲存在 LDAP 中 java 物件的方式。
-- factory-class 元素指定用來存取資源 Factory 所需的 JNDI InitialContext Factory。
-- 特性元素對應於外部 JNDI 環境可用的環境並且 jndi-lookup-name 指的是擷取指定的（在此例中
-- 爲 java）物件所要查找的 JNDI 名稱。
-->
<external-jndi-resource jndi-name="test/myBean"
jndi-lookup-name="cn=myBean"
res-type="test.myBean"
factory-class="com.sun.jndi.ldap.LdapCtxFactory">

<property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
<property name="SECURITY_AUTHENTICATION" value="simple" />
<property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
<property name="SECURITY_CREDENTIALS" value="changeit" />
</external-jndi-resource>
</resources>
```

對映應用程式資源參考

應用程式特定的資源參考必須對映至預先定義的伺服器範圍資源 Factory。Sun ONE Application Server 特定的資源參考對映元素用於此對映。

在以下範例中，我們將查看 Web 應用程式部署描述元 `web.xml`，其中資源參考指定給 JDBC DataSource。

```
<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

所需的 `res-ref-name` 也可以對映至容器範圍的 Orcale JDBC 連線資源 Factory，如下所示：

```
<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </resource-ref-name>
<jndi-name> jdbc/estore/InventoryDB </jndi-name>
</resource-ref>
```

關於 URL 連線 Factory 資源

URL 連線 Factory 不需要在 server.xml 中定義的任何資源。對應的 Sun ONE Application Server 應用程式 (Web 或 ejb) 部署描述元的 jndi-name 元素指定目標 URL。

例如，讓我們假設 Web 應用程式部署描述元 `web.xml` 指定 `java.net.URL` 資源參考，並且此參考對應於 `sun-web.xml` 中的 URL `http://www.`*sun.com/index.html*：

將以下列方式對映：

```
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<res-type>java.net.URL</res-type>
<res-auth>Container </res-auth>
</resource-ref>
```

```
<sun-web-app>
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<jndi-name> http://www.sun.com/index.html </jndi-name>
</resource-ref>
</sun-web-app>
```

對映應用程式資源環境參考

應用程式特定的資源環境參考宣告必須對映至應用程式伺服器運行時間環境中可用的目標資源物件。Sun ONE Application Server 特定的配置檔案中定義的資源環境對映元素讓部署程式如下對映：

範例：

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

此參考對映至 `server.xml` 中定義的 `jms/iMQ/Topics/Stocks/SUNW` 主題。請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」，以取得更多資訊。

```
<resource-env-ref-mapping>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<jndi-name> jms/iMQ/Topics/Stocks/SUNW </jndi-name>
</resource-env-ref-mapping>
```

對映 EJB 參考

也可以從 `ejb-name` 中分離在應用程式碼中使用的實際 ejb-name，使其用於目標企業 Bean。在不想修改 Web 應用程式部署描述元 `web.xml` 並使用企業 Bean 部署描述元的 `ejb-name` 時，會特別有用。Sun ONE Application Server 特定的配置在不使用 Sun ONE Application Server 特定部署描述元中的 `ejb-ref-mapping` 元素的情況下，可讓您將 `ejb-ref-name` 元素對映至目標 Bean 的 `ejb-name`。

範例：

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
</ejb-ref>
```

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<jndi-name> AccountEJB </jndi-name>
</ejb-ref-mapping>
```

關於持續性管理程式資源

此模組描述持續性，並建立使用 Sun ONE Application Server 支援的可插接式持續性管理程式的框架。

此模組包含以下主題：

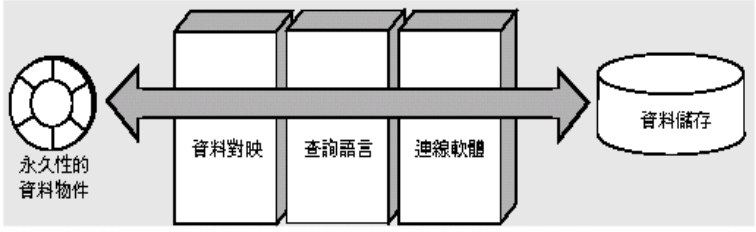
- 什麼是持續性？
- 持續性管理程式的角色
- 預先部署 Bean 配置
- 建立新的持續性管理程式

什麼是持續性？

大部分企業應用程式的主要功能是程式化處理永久性資料 (在應用程式之外的長期儲存資料)。雖然可以從暫存記憶體中讀取永久性資料，使用並修改它，但是也可以寫出至關係資料庫或平面檔系統進行長期儲存。

在物件導向程式設計系統中，永久性資料在記憶體中表示爲由應用程式碼處理的一個或多個資料物件。一般地，可以透過下圖「基本持續性機制」所示的多個軟體層，進行資料儲存中的永久性資料與其作爲記憶體中永久性資料物件的表示法之間的通訊：

圖 10-4 基本持續性機制



每個資料儲存透過用於設定並維護資料儲存與應用程式之間的連線之驅動程式軟體，具有與外部世界相連接的介面。使用所建立的連線，查詢語言用於擷取資料儲存中的資訊，並將其讀入應用程式，或反之，將應用程式中的資料寫入資料儲存。另一層提供記憶體中資料物件與資料儲存中資訊之間的對映。

透過這種一般機制，程式設計者可以將應用程式使用並處理的永久性資料表示爲運行時間物件。此機制支援所有基本的持續性作業，常縮寫爲 CRUD：

- C - 建立永久性資料 (在資料儲存中插入)
- R - 擷取永久性資料 (從資料儲存中選取)
- U - 更新永久性資料
- D - 刪除永久性資料

持續性管理程式的角色

持續性管理程式 (PM) 使用 EJB 容器中的容器管理式持續性負責實體 Bean 的持續性。實體 Bean 提供者負責提供作爲抽象類別的實體 Bean 類別。持續性管理程式提供者的工具負責提供具體實施。這些軟體可以透過對抽象實體 Bean 與相關的類別進行子分類，並提供具體實施或利用封裝與授權來實現其功能。

持續性管理程式工具提供的類別負責管理實體 Bean 之間的關係，並管理對其永久性狀態的存取。PM 工具同時還負責提供 `java.util.Collection` 類別的實施，這些類別用於維護容器管理式的關係 (CMR)。

預先部署 Bean 配置

企業 Java Bean 標準提供兩種類型的實體 Bean 持續性。它們爲容器管理式的持續性 (CMP) 與 Bean 管理式的持續性 (BMP)。EJB 2.0 規格不會定義 EJB 伺服器與持續性管理程式之間的標準 API。

本節描述在部署與產生程式碼時的整合需求。可以使用部署平均分配多個項目。一般來說，部署程序可以分爲三個主要步驟：配置、產生程式碼與安裝。

必須爲 Bean 指定多項特性，包括所使用的持續性機制、持續性供應商與使用中的版本以及持續性機制需要的附加資訊。大多數持續性供應商具有專案的概念，表示所有關聯的 Bean 及其作爲單一單元部署的相依類別。在每個專案中可以具有一個特定供應商的 xml 檔案。

支援部署的三個標準檔案包括 `ejb-jar.xml`、`sun-ejb-jar.xml` 與 `sun-cmp-mappings.xml`。在 `sun-ejb-jar.xml` 中的每個具有 CMP Bean 之 EJB 模組必須具有一個 `<pm-descriptors>`，並且至少其具有的一個 `<pm-descriptor>` 元素指定了五個屬性。這五個屬性爲 `pm-identifier`、`pm-version`、`pm-config`、`pm-class-generator` 與 `pm-mapping-factory`。

Sun ONE Application Server 特定的描述元 (如 `sunEjb_jar_2_0.DTD` 中的描述元) 定義持續性管理程式相關的標籤。範例 CMP 描述元可能與 Sun ONE Application Server DTD 中定義的描述元類似：

`PM` 描述元包含一個或多個 `pm` 描述元，但是必須在任何給定的時間僅使用一個描述元

```
<!ELEMENT pm-descriptors ( pm-descriptor+, pm-inuse)>
<!--
pm-descriptor 描述與實體 Bean 關聯的持續性管理程式的特性
-->
```

```
<!ELEMENT pm-descriptor ( pm-identifier, pm-version, pm-config?,
pm-class-generator?,

pm-mapping-factory?)>

<!--
```

此元素描述提供 PM 實施的供應商，例如，這可能是 Sun ONE Application Server Transparent Persistence、TopLink、Versant 或 CocoBase：

```
-->

<!ELEMENT pm-identifier (#PCDATA)>

<!--

pm-version 進一步指定使用的 PM 供應商產品版本

-->

<!ELEMENT pm-version (#PCDATA)>

<!--

pm-config 指定要使用的供應商特定配置檔案

-->

<!ELEMENT pm-config (#PCDATA)>

<!--

pm-class-generator 指定供應商特定的具體類別產生器

此爲供應商特定的類別名稱：

-->

<!ELEMENT pm-class-generator (#PCDATA)>

<!--

pm-mapping-factory 指定供應商特定的對映 Factory

此爲供應商特定的類別名稱：

-->

<!ELEMENT pm-mapping-factory (#PCDATA)>
```

建立新的持續性管理程式

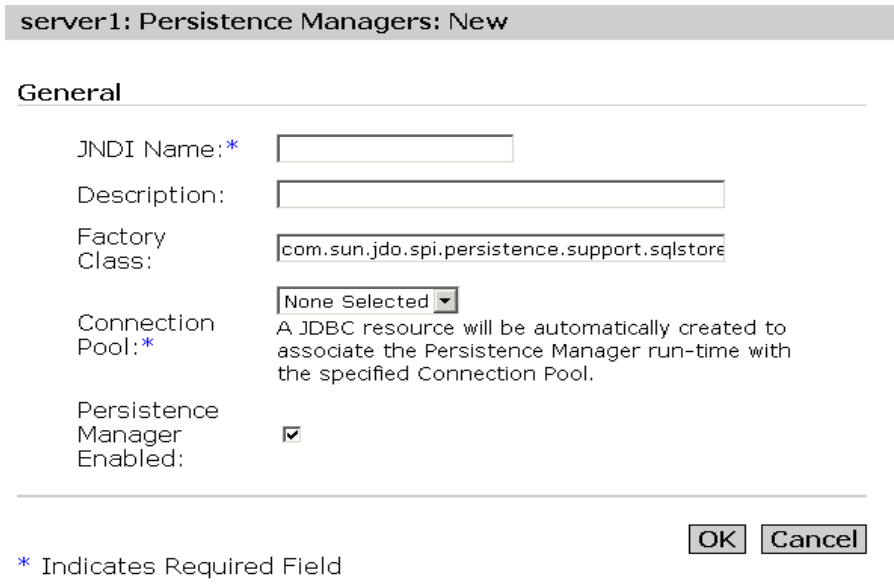
使用管理介面，您可以建立新的持續性管理程式實例。建立新持續性管理程式實例的步驟：

1. 從管理介面的左窗格中，開啓您要建立新的持續性管理程式之 Sun ONE Application Server 實例，然後從顯示的伺服器元件清單中按一下 [Persistence Manager]。

   如果已爲 Sun ONE Application Server 的該特定實例建立了任何持續性管理程式，將會看到在管理介面右窗格中顯示的清單。

2. 若要建立新的持續性管理程式，請按一下 [New]。將會看到圖「建立新的持續性管理程式」中顯示的以下視窗：

圖 10-5 建立新的持續性管理程式



3. 此爲應用程式伺服器執行期間使用的 JNDI 名稱，用於尋找代表應用程式的特定持續性管理程式。此名稱必須與 Sun 特定部署描述元的實體 Bean cmp-resource 元素中定義的名稱相同。

4. 在 [Description] 欄位中，提供新持續性管理程式的說明。此欄位的值爲字串，最多可以包含 250 個字元。

5. 在 [Factory Class] 欄位中，提供持續性管理程式的 Factory 類別連線。`setEntityContext` 透過 JNDI 名稱查找來查找此連線 Factory。此 Factory 類別名稱是建立持續性管理程式實例的持續性管理程式 Factory 類別名稱。依預設，將此設定爲 Sun ONE Application Server 內部持續性管理程式 Factory 類別。如果您使用替代的實施，必須確保此類別在伺服器類別路徑中可用。

6. 從 [Connection Pool] 下拉式清單中，選取匯集新的持續性管理程式所在的資料庫連線區。使用連線匯集，實體 Bean 可以請求單一連線，並使用它執行多個用戶端執行緒的並行處理描述。與任何其他資料庫存取類似，持續性管理程式將使用連線匯集來改進效能與可延伸性。如果您尚未建立連線區，請選擇現有的連線區或「無選取」。

   備註：將自動建立 JDBC 資源以讓 PM 運行時間連結至使用 JNDI 的連線區 - JDBC 資源的 JNDI 名稱將與 PM JNDI 名稱相同，均使用「PM」作爲前綴。刪除持續性管理程式也會刪除關聯的 JDBC 資源。

7. 若要啓用持續性管理程式，請標示 [Persistence Manager Enabled] 核取方塊。會立即爲指定的連線 Factory 啓用持續性管理程式。

8. 按一下 [OK]，以儲存變更。

關於 JDBC 資源

此模組一般說明有關 JDBC API 的資訊，而具體說明 JDBC 資源及其實施，以及在 Sun ONE Application Server 中的特定用法。

此模組由以下章節組成：

- 關於 JDBC API
- 關於資料庫存取模型
- 關於 JDBC 資料來源
- 關於 JDBC 連線
- 關於 JDBC 作業事件

關於 JDBC API

JDBC API 是 Java API，用以虛擬存取任何類型的表格資料。(JDBC 是商標名稱，並不是縮寫，然而，JDBC 常常被認爲表示「Java 資料庫連線」。) JDBC API 由類別集與以 Java 程式設計語言撰寫的介面 (提供工具 / 資料庫開發人員所使用的標準 API) 組成，並且 JDBC API 使得使用 all-Java API 寫入資料庫應用程式成爲可能。

JDBC API 使 SQL 敘述很容易發送至關係資料庫系統，並支援所有的 SQL 方言。但是 JDBC 3.0 API 優於 SQL，同時，其可以與其他各種資料來源進行互動，例如資料庫之外的檔案。

JDBC API 的值是應用程式可以虛擬存取任何資料來源並且在任何使用 Java 虛擬機器的平台中執行此應用程式。換句話說，使用 JDBC API，無需寫入一個程式來存取 Sybase 資料庫，無需寫入另一個程式存取 Oracle 資料庫，無需另一個程式存取 IBM DB2 資料庫等。您可以使用 JDBC API 寫入單一程式，並且此程式能夠將 SQL 或其他敘述發送至適當的資料來源。並且，使用 Java 程式設計語言撰寫的應用程式，不必擔心寫入要在不同平台中執行的不同應用程式。Java 平台與 JDBC API 的組合可讓程式設計師只寫入一次程式碼，然後在任何地方執行此程式碼。

JDBC API 的功能？

基於 JDBC 技術的驅動程式 (JDBC 驅動程式) 可以執行以下三個動作：

- 建立與資料來源的連線
- 向資料來源發送查詢，並更新敘述
- 處理結果

以下程式代碼段提供這三個步驟的簡單範例：

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/AcmeDB");
Connection con = ds.getConnection("myLogin", "myPassword");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");
while (rs.next()) {
          int x = rs.getInt("a");
          String s = rs.getString("b");
          float f = rs.getFloat("c");
   }
```

關於資料庫存取模型

JDBC API 支援資料庫存取的兩層與三層模型。Sun ONE Application Server 結合了更通用的兩層資料庫存取模型。

本章節涵蓋以下主題：

- 兩層資料庫存取模型
- 三層資料庫存取模型

兩層資料庫存取模型

在兩層資料庫存取模型中，Java applet 或應用程式使用 DBMS 專用協定直接與資料來源對話。此存取模型需要 JDBC 驅動程式，該驅動程式可以與正在存取的特定資料來源通訊。使用者指令將傳送至資料庫或其他資料來源，並且將那些敘述結果發回至使用者。資料來源可以位於使用者透過網路連線的另一台機器上。此配置指的是用戶端/伺服器配置。將使用者機器作爲用戶端，儲存資料來源的機器作爲伺服器。網路可以是企業內部網路，例如，在公司內部連線員工機器的網路，或者是網際網路。

三層資料庫存取模型

在三層資料庫存取模型中，Java Applet 或應用程式將指令傳送至服務的「中間層」，該層再將指令發送至資料來源。用戶端應用程式透過 HTTP、RM、CORBA 或其他呼叫與中間層通訊。中間層透過 DBMS 專用協定與資料來源進行通訊。資料來源處理指令，並將結果傳回中間層，該層再將其傳回使用者。MIS 指示器發現三層模型具有很大吸引力，因爲中間層可以維護對存取的控制，以及維護對公司資料所作的各種更新。另一個優點是其簡化了應用程式的部署，最後，在許多情況下，三層架構可以提供效能優勢。

關於 JDBC 資料來源

`DataSource` 物件是表示以 Java 程式設計語言撰寫的資料來源。從基本上來說，資料來源是儲存資料的設備。其精密程度可以類似於大公司的複雜資料庫，也可以與具有列與欄的簡單檔案一樣。資料來源可以駐留在遠端伺服器上，或可以在本機桌面機器上。應用程式使用連線存取資料來源，`DataSource` 物件可以作爲 `DataSource` 實例表示的特定資料來源連線之 Factory。DataSource 介面提供兩種建立與資料來源連線的方法。

`DataSource` 物件具有識別與描述其所表示的資料來源之特性。同時，`DataSource` 物件使用 JNDI 命名服務，並且在獨立於使用其的應用程式之外建立、部署並管理此物件。驅動程式供應商將提供作爲其 JDBC 2.0 或 3.0 驅動程式產品部分的 `DataSource` 介面基本執行之類別。

本章節涵蓋以下主題：

- DataSource 物件的特性
- 註冊 JDBC 資源

DataSource 物件的特性

DataSource 物件具有識別與描述其表示的真實資料來源之特性集，這些特性包括資料庫伺服器位置、資料庫名稱、用於與伺服器通訊的網路協定等資訊。DataSource 特性遵循 JavaBeans 設計型樣，通常在部署 DataSource 物件時設定這些特性。

爲了支援不同供應商之 DataSource 執行的一致性，JDBC 2.0 API 指定標準的特性集與每個特性的標準名稱。

執行 DataSource 介面的類別實例表示一個特別的資料來源。此實例提供的每個連線將參考同一資料來源。在基本的 DataSource 執行中，對 DataSource.getConnection 方法的呼叫會傳回連線物件，與 DriverManager 功能傳回的連線物件類似，是資料來源的實體連線。

JNDI 提供一致的方法，用於應用程式在網路上查找與存取遠端服務。遠端服務可以是企業服務，包括訊息傳送服務或應用程式特定的服務，但是，JDBC 應用程式主要對資料庫服務感興趣。一旦建立了 DataSource 物件並使用 JNDI 命名服務註冊了該物件，應用程式便可以使用 JNDI API 存取 DataSource 物件，然後該物件用於連線至其所表示的資料來源。

執行連線匯集的 DataSource 物件也會產生 DataSource 類別表示的特定資料來源連線。但是，方法 DataSource.getConnection 傳回的連線物件是對 PooledConnection 物件的處理，而不是實體連線。應用程式僅以通常的方式使用連線物件，但不瞭解使用的是另一種方式。除了匯集的連線應該明確地關閉之外 (適用於所有連線)，連線匯集對應用程式碼無影響。當應用程式關閉匯集的連線時，此連線會結合可重新使用的連線區。下一次在呼叫 DataSource.getConnection 時，如果可以使用一個匯集的連線，則會傳回其中一個連線的處理。因爲連線匯集避免在每次請求時建立一個新的實體連線，這有助於使應用程式的執行效能顯著加快。

也可以實施 DataSource 類別來使用分散式作業事件環境。例如，EJB 伺服器支援分散式作業事件，並且需要與此伺服器互動而實施的 DataSource 類別。在這種情況下，DataSource.getConnection 方法傳回可以在分散式作業事件中使用的 Connection 物件。作爲規則，EJB 伺服器提供對連線匯集與分散式作業事件的支援。與連線匯集類似，可以內部處理作業事件管理，因此使用分散式作業事件很容易。唯一的需求是在分散作業事件 (涉及兩個或多個資料來源) 時，應用程式無法呼叫作業事件方法 (commit 或 rollback)，也無法在 auto-commit 模式下進行連線。這些限制的原因是作業事件管理程式開始並結束的分散式作業事件在其包含的範圍之內，因此，應用程式無法對作業事件開始或結束的時間進行任何影響。若要瞭解關於 Java 作業事件的更多資訊，請參閱第 9 章「使用作業事件服務」。

註冊 JDBC 資源

您可以使用管理介面或指令行介面在 Sun ONE Application Server 中註冊 JDBC 資源。

本章節涵蓋以下主題：

- 使用指令行註冊資源
- 使用管理介面註冊資源

使用指令行註冊資源

若要使用指令行介面註冊 JDBC 資源，請執行以下指令：

```
./asadmin create-jdbc-resource
```

用於註冊 JDBC 資源的 XML snippet 必須指定一些屬性，如下所示 (引用自 `sun-server_7_0.dtd`)。

```
<!-- JDBC javax.sql.DataSource resource definition -->

<!ELEMENT jdbc-resource (description?, property*)>

<!ATTLIST jdbc-resource  jndi-name CDATA     #REQUIRED

pool-name CDATA     #REQUIRED

enabled %boolean; 'true'>
```

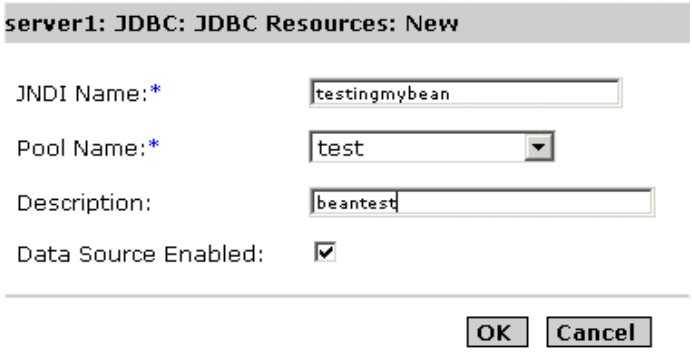
請注意，所有指定的是應用程式參考的 J2EE 應用程式內部資料來源的符號名稱。`pool-name` 屬性指向具名的儲存區定義，此定義指定資料庫連接性的各個方面。管理員可以使用此啓用的屬性來關閉某些資源。

使用管理介面註冊資源

使用管理介面註冊資料來源的步驟：

1. 在管理介面的左窗格中，開啓您要註冊 JDBC 資源的 Sun ONE Application Server 實例。
2. 開啓 JDBC。
3. 在 JDBC 下，按一下 [JDBC Resource]。
4. 在右窗格中，按一下 [New]。如圖「建立新的 JDBC 資源」所示的建立新 JDBC 資源的頁面顯示在右窗格中。

圖 10-6 建立新的 JDBC 資源



5. 提供用於建立資源的 JNDI 名稱。

   JDBC 資源儲存於 JNDI 儲存庫中，並使用 JNDI 名稱存取。JNDI 名稱在 `Java:comp:env/` 下具有明確的根，因此無需指定名稱部分。建議 JDBC 資源 (DataSource) 儲存在 '`jdbc`' 子環境下，這樣 JNDI 名稱就類似於 `jdbc/EmployeeDB_DS`。

6. 從 [Pool Name] 下拉式清單的儲存區名稱清單中，選取新資料來源的儲存區名稱。所有註冊的連線區將出現在此下拉式清單中。您選取的儲存區名稱指向具名的儲存區定義，此定義指定資料庫連接性的各個方面。更多的 JDBC 資源可以使用單一的儲存區定義。若要瞭解關於如何配置 JDBC 連線區的更多資訊，請參閱第 250 頁的「使用管理介面建立新的 JDBC 連線區」。

7. 在 [Description] 欄位中，提供描述資料來源用途的簡要說明。此說明不能超過 250 個字元。

8. 標示 [Enabled] 核取方塊，以啓用或停用資料來源。只有啓用了資料來源，才能使用它連線至資料庫。

9. 按一下 [OK]，以註冊新的資料來源，或按一下 [Cancel]，以取消新的資料來源。您按一下取消時，會返回至 JDBC 資源主頁面，從中您可以再次建立新的資料來源。

關於 JDBC 連線

Connection 物件表示與資料庫的連線。連線階段作業包括執行的 SQL 敘述以及透過該連線傳回的結果。單一應用程式可以與單一資料庫具有一個或多個連線，或可以與多個不同的資料庫具有多個連線。

使用者可以透過呼叫 `Connection.getMetaData` 方法，取得關於 Connection 物件資料庫的資訊。此方法傳回 `DatabaseMetaData` 物件，該物件包含關於資料庫表格、其支援的 SQL 語法、其儲存的程序以及此連線功能等方面的資訊。

應用程式使用 DataSource 物件產生的 Connection 物件。情況總是這樣，應用程式應該包含「finally」區塊，以確定連線已關閉 (即使已拋出異常)。如果 Connection 物件是匯集的連線，這樣做尤爲重要，因爲其會確保有效的連線會返回至可用連線的儲存區。以下程式碼分段 (其中 con 是 Connection 物件) 是關閉連線 (如果爲有效連線) 的 finally 區塊範例。

```
finally{
        if (con != null) con.close();
}
```

請注意，`finally` 區塊顯示在 try/catch 區塊之後，如下例所示。其中 `ds` 爲 DataSource 物件。

```
try {
        Connection con = ds.getConnection("user", "secret");
        // . . . 執行應用程式工作的程式碼
} catch {
        // . . . 處理 SQLException 的程式碼
} finally {
        if (con != null) con.close();
}
```

本章節涵蓋以下主題：

- 關於 JDBC URL
- 配置 JDBC 連線區
- 關於連線匯集
- 監視 JDBC 連線匯集
- 關於連線共用

關於 JDBC URL

URL (統一資源位址) 提供尋找網際網路上資源的資訊。可以作爲位址。

JDBC URL 提供識別資料來源的方法，以便適當的驅動程式會識別它，然後建立與資料來源的連線。驅動程式撰寫者是實際決定識別特定驅動程式的 JDBC URL 內容的人。使用者無需擔心如何形成 JDBC URL，只需使用 URL (隨使用者要使用的驅動程式一同提供)。JDBC 的任務是爲驅動程式撰寫者提供某些慣例，讓其在結構化 JDBC URL 時，遵循這些慣例。

因爲各種驅動程式均使用 JDBC URL，因此這些慣例有必要非常靈活。首先，可讓不同的驅動程式使用不同的機制命名資料庫。例如，ODBC 子協定，讓 URL 包含屬性值 (但不是必需的)。

第二，JDBC URL 可讓驅動程式撰寫者在 URL 內對所有需要的連線資訊進行編碼。例如，這可讓要與指定的資料庫進行通訊的 applet 開啓資料庫連線，而不需要使用者執行任何系統管理作業。

第三，JDBC URL 允許使用間接層級。意思是 JDBC URL 可以參考由網路命名系統動態轉換爲實際名稱的邏輯主機名稱或資料庫名稱。這可讓系統管理員避免指定特定主機作爲 JDBC 名稱一部分。其中有許多不同的網路名稱服務，並且對可以使用哪個服務沒有限制。

此處顯示 JDBC URL 的標準語法。其具有三個部分，用冒號分隔。

`jdbc:<`*`subprotocol`*`>:<`*`subname`*`>`

JDBC URL 的三個部分按如下方式分爲：

- jdbc - 協定：

  JDBC URL 中的協定始終爲 jdbc。

- `<subprotocol>`

  驅動程式的名稱或資料庫連接性機制的名稱，可能受一個或多個驅動程式支援。Sub-protocol 名稱的一個重要範例爲 ODBC (爲指定 ODBC 樣式的資料來源名稱之 URL 所保留)。例如，若要透過 JDBC-ODBC 橋接存取資料庫，可以使用 URL，例如 `jdbc:odbc:fred`。

  在此範例中，subprotocol 爲 ODBC，subname fred 爲本機 ODBC 資料來源。

  如果要使用網路名稱服務 (這樣，JDBC URL 中的資料庫名稱不必爲其實際名稱)，命名服務可以爲子協定。例如，URL 可以爲：

  ```
  jdbc:dcenaming:accounts-payable
  ```

  在此範例中，URL 指定本機 DCE 命名服務應該將 accounts-payable 資料庫名稱解譯爲一個更特定的名稱 (可用來連線至實際的資料庫)。

- `<subname>`：

  識別資料來源的方式。子名稱可以因子協定而有所不同。並且它可以具有驅動程式撰寫者選擇的任何內部語法，包括 `sub-subname`。使用 `subname` 爲尋找資料來源提供充足的資訊。在上例中，因爲 ODBC 提供了資訊的剩餘項目，所以使用 fred 已經足夠。但是遠端伺服器中的資料來源需要更多的資訊。例如，如果在網際網路上存取資料來源，網際網路位址應該作爲 `subname` 一部分包含在 JDBC URL 中，並且應該遵循以下標準的 URL 命名慣例：

  //hostname:*port*/*subsubname*

  假如 `dbnet` 是用於連線至網際網路上主機的協定，則 JDBC URL 可能會如下所示：

  `jdbc:`*dbnet*`://`*wombat:356/fred*

配置 JDBC 連線區

Sun ONE Application Server 可讓使用者建立具名的 JDBC 連線區。JDBC 連線區定義用於建立連線區的特性。命名儲存區定義後，可以重新使用定義來配置多重 JDBC 資源。每個具名的儲存區定義會在伺服器啓動時產生實體儲存區創設。如果兩個或多個 JDBC 資源指向同一個儲存區定義，在運行時，其會使用同一個連線的儲存區。

您可以使用管理介面與指令行介面建立並配置 JDBC 連線區，如以下章節詳細論述：

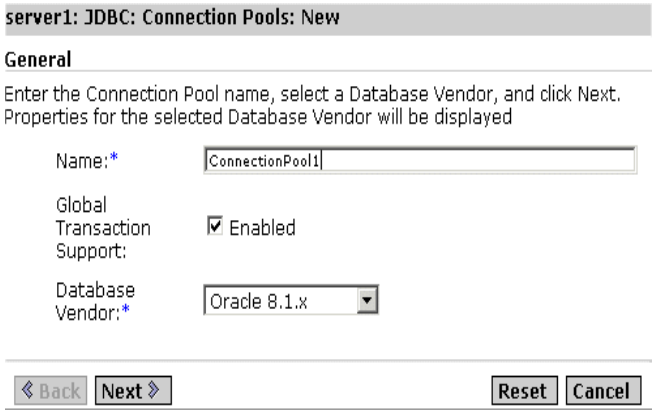
- 使用管理介面建立新的 JDBC 連線區
- 使用指令行介面建立新的 JDBC 連線區
- 使用指令行介面管理 JDBC 連線區

使用管理介面建立新的 *JDBC* 連線區

若要使用管理介面建立新的 JDBC 連線區，請執行以下工作：

1. 在管理介面的左窗格中，開啓您要爲其建立 JDBC 連線區的 Sun ONE Application Server 實例。
2. 選取 Sun ONE Application Server 下列示的 J2EE 服務清單中之 [JDBC]，然後開啓其下面的 `[ConnectionPools]` 標籤，會在管理介面的右窗格中看到圖「建立新的 JDBC 連線區」。

圖 10-7 建立新的 JDBC 連線區



3. 在 [Name] 欄位中，提供您要建立的連線區之 JNDI 名稱。
4. 標示 [Global Transaction Support Enabled] 核取方塊，以啓用新連線區的全域作業事件支援。能夠加入全域作業事件的連線區指的是 XA 可用的連線區。
5. 從 [Database Vendor] 下拉式清單中選取資料庫供應商，然後按一下 [Next]。需要在顯示的下一個畫面中配置連線區設定。

配置連線區設定

若要配置連線區設定，請執行如第 250 頁的「使用管理介面建立新的 JDBC 連線區」中給定的步驟 1 至步驟 5。在您按一下 [Next] 時，如步驟 5 中所述，新頁面會出現在管理介面的右窗格中。它包含以下區段：

- General
- Properties
- Pool Settings
- Connection Validation
- Transaction Isolation

在此頁面的 [General] 區段內，指定基於下表指導原則所提供的參數之值：

表格 10-2 一般設定

參數	描述
Name	連線區名稱。
DataSource ClassName	供應商特定的類別名稱，執行 DataSource 和/或 XADataSource API。
Description	連線區說明。

在此頁面的 [Properties] 區段中，您可以指定標準特性與專用的 JDBC 連線區特性；許多特性是選擇性的。依預設，會提供所有的標準特性名稱。需要查閱資料庫供應商說明文件，以決定需要哪些標準特性與供應商特定特性。

在此視窗的 [Pool Settings] 區段內，指定基於下表指導原則所提供的參數之值：

表格 10-3 連線區設定

參數	描述
Steady Pool Size	請指定必須在儲存區中維護的最小連線數目。在連線指定給請求的執行緒時，會從儲存區中移除該連線，從而減少目前儲存區的大小。同時，固定的儲存區大小也指啓動伺服器時，加入至儲存區的項目數。
Max Pool Size	指定某時某點上儲存區中所允許的最大連線數目。
Pool Resize Quantity	儲存區向固定儲存區大小縮小時，是以批次來重調大小的。此值決定批次的大小。值越大，會延遲連線的再循環；值越小，效率越低。請注意，儲存區容量僅在一段時間內的一次連線時增加，因此此欄位不會影響連線容量的增加。
Idle Timeout (以秒表示)	連線可以在儲存區中閒置的最大時間 (以秒表示)。這一時間過後，儲存區執行可以關閉此連線。
Max Wait tim	呼叫者在取得連線逾時之前等待的時間。預設等待時間很長，意思是呼叫者可以等待很長一段時間。

在此視窗的 [Connection Validation] 與 [Transaction Isolation] 區段中，基於下表中給定的指導原則，選取連線區的驗證方法以及作業事件隔離方法：

表格 10-4 連線驗證與作業事件隔離

參數	描述
Connection Validation Required	如果核取此欄位，則連線在傳送至應用程式之前會得到驗證。這可讓應用程式伺服器自動重新建立資料庫連線 (由於網路故障或資料庫伺服器當機，資料庫不可用時)。對連線進行驗證，會耗用額外的時間，並且會略微降低效能。
Validation Method	爲了驗證資料庫連線，應用程式伺服器可以使用三種方法。需要瞭解資料庫的功能，以決定使用適當的一種方法。三種驗證方法爲： • `auto-commit, meta-data` - con.getAutoCommit() 與 con.getMetaData() 方法通常用於驗證連線，但是不幸的是，許多 JDBC 驅動程式快取這些呼叫結果，因此無法始終提供可靠的驗證。您應該洽詢您的供應商，以決定是否快取這些呼叫。 • `table`：此方法需要應用程式伺服器對使用者指定的表格執行查詢。實際查詢爲「select (count *) from <table-name>」。雖然此表格不需要任何列，但是其必須存在並且可以存取。您不應該使用具有大量列的表格或已經常存取的表格。
Table Name	如果您選取了最後的驗證選項，從 [Validation Method] 下拉式清單中選取的 `table` 會在此處指定表格名稱。
Fail All Connections	如果決定單一連線無效，請核取此方塊，使儲存區中的所有連線無效，然後重新建立連線。如果未核取此方塊，僅有在使用這些連線時，才個別重新建立這些連線。
Transaction Isolation	允許您選取此連線的作業事件隔離層級。如果未指定，儲存區會使用 JDBC 驅動程式提供的預設隔離層級作業。
Guarantee Isolation Level	只有指定了隔離層級，此參數才可用。這確保了從儲存區中取得的任何連線均具有相同的隔離層級。例如，如果在上次使用隔離層級時，程式化變更了連線的隔離層級 (例如，`con.setTransactionIsolation`)，這種機制會將其變回至指定的隔離層級。

使用指令行介面建立新的 *JDBC* 連線區

本章節透過使用範例描述使用指令行介面建立 JDBC 連線區的方式。

下表列示需要建立連線區的所有選項，例如伺服器名稱、密碼。範例值已在下表中使用。建議您準備安裝 Sun ONE Application Server 特定的參數，然後再運行本節中解釋的指令。

表格 10-5 使用指令行介面建立 JDBC 連線區所需的選項

所需選項的說明	範例值
應用程式伺服器管理使用者名稱	***admin***
應用程式伺服器管理密碼	*adminadmin*
應用程式伺服器管理連接埠	*8888*
應用程式伺服器機器名稱	*sas.sun.com*
應用程式伺服器實例名稱	*server1*
連線區的資料來源類別名稱	*oracle.jdbc.xa.client.OracleXADataSource* 注意：使用您要爲其建立連線區的資料庫之資料來源類別名稱。此範例使用的資料庫爲 Oracle。
Jdbc 資源說明範例	*Jdbc Resource*
連線區說明範例	*Jdbc Connection Pool*
Jdbc 資源名稱	*jdbc/SampleJdbcResource*
連線區名稱	*SampleJdbcConnectionPool*
資料庫使用者名稱	*oracle*
資料庫密碼	*oracle*
Jdbc 連線 URL	*jdbc:oracle:thin:@oracleserver.sun.com:1521:ORA*

以下範例利用「使用指令行介面建立 JDBC 連線區所需的選項」表中列示的變數。

範例 1：

此範例建立名爲 `SampleJdbcConnectionPool` 的 JDBC 連線區。在此範例中，使用兩個步驟程序，可以建立 JDBC 連線區，如下所示：

- 步驟 1 - 建立連線區
- 步驟 2 - 將變更套用至實例

步驟 1 - 建立連線區

以下是建立 JDBC 連線區的指令行介面語法：

```
asadmin create-jdbc-connection-pool --user admin_user [--password
admin_password] [--host localhost] [--port 4848] [--secure | -s]
[--instance instancename] --datasourceclassname classname [--restype
res_type] [--steadypoolsize 8] [--maxpoolsize 32] [--maxwait 60000]
[--poolresize 2] [--idletimeout 300] [--isolationlevel isolation_level]
[--isisolationguaranteed] [--isconnectvalidatereq=false]
[--validationmethod auto-commit] [--validationtable tablename]
[--failconnection=false] [--description text] [--property
(name=value)[:name=value]*] connectionpool_id
```

例如，以下指令將建立名爲 SampleJdbcConnectionPool 的連線區。

```
asadmin create-jdbc-connection-pool --user admin --password
adminadmin --host sas.sun.com --port 8888 --instance server1
--restype javax.sql.XADataSource --datasourceclassname
oracle.jdbc.xa.client.OracleXADataSource --description "Sample Jdbc
Connection Pool" --property
User="oracle":Password="oracle":URL="jdbc\:oracle\:thin\:@oracleser
ver.sun.com\:1521\:ORA" SampleJdbcConnectionPool
```

注意 如果您要爲新的連線區啓動「全域作業事件支援」，請設定 `--restype javax.sql.XADataSource`。在 URL 特性中，使用 (\:) 取代冒號 (:)。

成功建立 JDBC 連線區後，您便會看到以下訊息：

```
建立了 id = SampleJdbcConnectionPool 的 JDBC 連線區資源
```

步驟 2 - 將變更套用至實例

既然已成功建立了 JDBC 連線區，您需要將變更套用至 Sun ONE Application Server 的目前實例。

以下是將變更套用至 Sun ONE Application Server 實例的語法。

```
asadmin reconfig --user admin_user [--password admin_password] [--host
localhost] [--port adminport] [--secure | -s]
[--discardmanualchanges=false|--keepmanualchanges=false] instancename
```

例如，以下指令將變更套用至 *server1* (Sun ONE Application Server 的實例)。

```
asadmin reconfig --user admin --password adminadmin --host sas.sun.com
--port 8888 server1
```

將變更套用至 Sun ONE Application Server 實例後，您便會看到以下訊息。

```
重新配置已成功
```

使用指令行介面管理 *JDBC* 連線區

您可以使用指令行介面管理 JDBC 連線區及其特性，如本章節中所述：

列示連線區。以下指令列示爲 *server1* (步驟 2 中使用的 Sun ONE Application Server 實例) 建立的所有連線區。

```
asadmin list-jdbc-connection-pools --user admin --password adminadmin
--host sas.sun.com --port 8888 server1
```

變更 JDBC 連線區特性。您可以依如下步驟變更 JDBC 連線區特性 (例如，`maxPoolSize` 特性)：

1. 執行以下指令，以取得爲 JDBC 連線區屬性 `maxPoolSize` 指定的值。

   ```
   asadmin get -u admin -w adminadmin -H sas.sun.com -p 8888
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize
   ```

   您執行此指令時，會看到以下結果：

   ```
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSiz
   e = 32
   ```

   透過執行以下指令，將 MaxPoolSize 值變更爲 80：

   ```
   asadmin set -u admin -w adminadmin -H sas.sun.com -p 8888
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize="80"
   ```

   依所示指定值後，您便會看到以下訊息：

   ```
   屬性 maxPoolSize 設定爲 80
   ```

2. 使用以下指令，將變更套用至 Sun ONE Application Server 實例：

   ```
   asadmin reconfig --user admin --password adminadmin --host
   sas.sun.com --port 8888 server1
   ```

變更 User 特性。在以下部分範例碼中，您可以將 "`User`" 特性從 `oracle` 變更爲 `System`。

```
asadmin create-jdbc-connection-pool --user admin --password adminadmin
--host sas.sun.com --port 8888 --instance server1 --restype
javax.sql.XADataSource --datasourceclassname oracle.jdbc.xa.client.OracleXADataSource
--description "Sample Jdbc Connection Pool" --property
User="oracle":Password="oracle":URL="jdbc\:oracle\:thin\:@oracleserver.sun.com\:1521\
:ORA" SampleJdbcConnectionPool
```

1. 執行以下指令，以變更 User 特性。

   ```
   asadmin set -u admin -w adminadmin -H sas.sun.com -p 8888
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.property.User="System"
   ```

   使用者的名稱從 *Oracle* 變更爲 *System*。

2. 變更使用者名稱之後，執行以下指令來套用變更：

   ```
   asadmin reconfig --user admin --password adminadmin --host
   sas.sun.com --port 8888 server1
   ```

建立名為 SampleJdbcResource 的 JDBC 資源。您可以建立 JDBC 資源，如下詳述。以下是建立 JDBC 資源的語法：

```
asadmin create-jdbc-resource --user admin_user [--password
admin_password] [--host localhost] [--port 4848] [--secure | -s]
[--instance instancename] --connectionpoolid id [--enabled=true]
[--description text] [--property (name=value)[:name=value]*] jndiname
```

1. 請執行以下指令，以建立名爲 `SampleJdbcResource.` 的 JDBC 資源。

   ```
   asadmin create-jdbc-resource --user admin --password adminadmin
   --host sas.sun.com --port 8888 --instance server1 --description
   "Sample Jdbc Resource" --connectionpoolid SampleJdbcConnectionPool
   jdbc/SampleJdbcResource
   ```

   您執行此指令時，會建立 JDBC 資源，然後將看到以下訊息：

   使用 `jndiname = jdbc/SampleJdbcResource` 建立了外部 `JDBC` 資源

2. 下一步，您需要運行以下指令，將變更套用至 Sun ONE Application Server 實例。

   ```
   asadmin reconfig --user admin --password adminadmin --host
   sas.sun.com --port 8888 server1
   ```

3. 請執行以下指令，以列示實例 *server1* 中的所有 JDBC 資源。

   ```
   asadmin list-jdbc-resources --user admin --password adminadmin
   --host sas.sun.com --port 8888 server1
   ```

關於連線匯集

應用程式可以透過 JNDI 首先查找 DataSource，以取得連線。完成此作業的範例碼分段顯示如下：

```
InitialContext ctx  = new InitialContext();

DataSource ds = (DataSource)
ctx.lookup("java:comp/env/jdbc/employee_ds");
```

取得 DataSource 之後，應用程式元件可以用兩種方法取得連線，這要取決於對 J2EE 部署描述元中 <res-auth> 元素的值設定。如果此元素的值爲 Container，則應用程式可以使用 ds.getConnection() 方法 (即，不指定任何登入資訊) 取得連線。否則，應用程式必須提供登入資訊，以取得資源管理程式中的連線，如 ds.getConnecion.(*userName, password*)。

儲存區將提供對 getConnection() 所有請求的服務。將基於 server.xml 中描述的參數集建立 JDBC 連線區。建立連線區後，連線區包含最初可用的連線數目。因此，使用連線區中目前可用的連線，可以滿足 ds.getConnection() 請求。下一個請求 (如果未向連線區傳回先前的連線) 將發現連線區爲空，並會建立增量的連線，前提是不超過連線區中指定的最大連線數限制。連線區執行會追蹤建立的連線數。如果 getConnection() 請求發現連線區爲空，或者建立的連線數等於儲存區中的最大連線數，將會阻止目前的請求。只有在連線共用不可能的情況下，才發生此類問題，並且繼續，直到連線傳回到連線區爲止。

在伺服器仍在執行的情況下，即使資料庫當機後又恢復，連線區仍會繼續正常工作，條件是必須啓用連線驗證，如第 251 頁的「配置連線區設定」中所述。

依您從 [Validation Type] 下拉式清單中選取的值而定，儲存區執行程式執行以下參數：

- 如果您選擇 auto-commit 作爲連線驗證類型，系統會透過執行 conn.getAutoCommit() 方法，查看連線是否有效。如果此方法未拋出 SQLException，那麼會假設 Connection 爲有效。auto-commit 爲此參數的預設選項。
- 如果您選擇 meta-data 作爲連線驗證類型，那麼將會執行 conn.getMetaData() 方法，以檢查連線的複合資料；如果此方法未拋出 SQLException，那麼會假設 Connection 爲有效。
- 如果您選擇 table 作爲連線驗證類型，那麼將執行 "Select * From <table-name>" 查詢。如果此呼叫未拋出 SQLException，那麼會假設連線有效。

如果您啓用 fail-all-connections 特性，發現儲存區中的任何連線無效時，所有連線將關閉，並重新建立。否則，無效與重新建立是惰性的，在使用個別連線時才發生。

儲存區執行也提供再循環儲存區中所有可用連線的功能。因此，如果連線在超過指定的閒置期間仍被閒置，可能會關閉，並使儲存區的大小成爲固定儲存區大小。如果儲存區最終仍爲閒置，可能導致容器必須重新建立先前的連線，並且使儲存區永遠具有可用連線的固定儲存區。在決定是否設定相對於最大儲存區大小的固定儲存區大小時，您必須記住這一點。

監視 JDBC 連線匯集

您可能需要定期監視儲存區作業，以決定儲存區大小配置是否有效地工作。下表列示可以監視的所有 JDBC 連線匯集參數。

請注意，啓用監視技術與可以監視的屬性有可能在以後的版本中得到改進。

表格 10-6 用於監視的 JDBC 連線區參數

屬性名稱	資料類型	描述
`total-threads-waiting`	整數	等待 JDBC 連線的執行緒總數。
`total-outbound-connections`	整數	JDBC 連線驗證的故障總數
`total-connections-timed-out`	整數	逾時的連線請求總數。

關於連線共用

J2EE 應用程式取得的多個連線使用同一資源管理程式時，儲存區執行將在同一作業事件範圍內提供連線共用。可以從以下範例中瞭解作業事件範圍術語：

Bean_A 啓動作業事件 (Tx1)，並且取得連線。然後 Bean_A 呼叫同一作業事件 (Tx1) 中的
Bean_B 之方法。現在，如果 Bean_B 從同一 `DataSource` 中取得連線，並且使用同一登入資訊，很顯然，只有在 Bean_A 完成此作業事件時，才可以共用同一連線。同時請注意，只有資源共用範圍在 J2EE 部署描述元中設定爲 `Shareable` 時，才能共用連線。如果不需要連線共用，則資源共用範圍必須在部署描述元中設定爲 `Unshareable`。Sun ONE Application Server 提供連線共用，因爲這會提供更佳的效能。

關於 JDBC 作業事件

作業事件由已經執行、完成，然後確定或回轉的一個或多個敘述組成。當呼叫方法 `commit` 或 `rollback` 時，目前的作業事件會結束，另一個作業事件會開始。

通常依預設，新的 Connection 物件在 auto-commit 模式中，意思是在完成敘述時，會自動對此敘述呼叫方法 `commit`。在這種情況下，由於會個別確定每個敘述，所以作業事件僅由一個敘述組成。如果已停用 auto-commit 模式，只有明確地呼叫 `commit` 或 `rollback` 方法時，作業事件才會終止，因此此作業事件將包含上次呼叫 commit 或 rollback 後執行的所有敘述。在第二種情況下，作業事件中的所有敘述已確定或回轉爲群組。

方法 `commit` 使得 SQL 敘述對資料庫所作的任何變更成爲永久變更，並且此方法還釋放作業事件具有的任何鎖定。方法 `rollback` 將放棄那些變更。

在一個作業事件中的兩個更新中，有時您可能不想讓一個變更在一個更新中生效，除非另一個更新也受到影響。透過停用 auto-commit，並且將兩個更新群組爲一個作業事件，可以完成這一動作。如果兩個更新均成功，那麼會呼叫 `commit` 方法，使這兩個更新爲永久更新，如果一個或兩個更新失敗，那麼會呼叫 `rollback`，在執行更新之前，會回復已存在的值。大多數 JDBC 驅動程式支援作業事件。

在 `javax.sql` 套裝軟體中的類別與介面使 Connection 物件可以成爲分散式作業事件部分，即爲與多個 DBMS 伺服器連線的作業事件。爲了能夠加入分散式作業事件，Connection 物件必須由 `DataSource` 物件產生 (已執行此物件，以在中間層伺服器分散式作業事件基礎架構中使用)。與 `DriverManager` 產生的 Connection 物件不同，由 `DataSource` 物件產生的 Connection 物件依預設，將停用 auto-commit 模式。從另一方面來說，對 `DataSource` 物件的標準執行，將產生與 `DriverManager` 類別產生的物件相同的 Connection 物件。

Connection 物件爲分散式作業事件部分時，作業事件管理程式會決定何時對此物件呼叫方法 commit 或 rollback。這樣，當 Connection 物件加入分散式作業事件時，應用程式不應該影響連線開始或結束，例如呼叫方法 `Connection.commit` 或 `Connection.rollback`，或開啓連線的 auto-commit 模式。這些會干涉作業事件管理程式處理分散式作業事件。

關於 Java 郵件資源

JavaMail API 允許存取包含在訊息儲存中的電子郵件訊息，並且允許使用訊息傳輸建立與發送電子郵件訊息。網際網路標準 MIME 訊息包含特定的支援。可以透過支援特定儲存與傳輸協定的協定提供者來存取訊息儲存與傳輸。JavaMail API 規格不需要任何特定的協定提供者，但是，JavaMail 包含 IMAP 訊息儲存提供者與 SMTP 訊息傳輸提供者。

JavaMail API 提供抽象類別集，用於定義組成郵件系統的物件。API 定義如 Message、Store 與 Transport 之類的類別。可以延伸 API，並且可以將其分爲子類別，以提供新的協定，並在必要時新增功能。此外，API 提供抽象類別的具體子類別。這些子類別，包括 `MimeMessage` 與 `MimeBodyPart`，執行廣泛使用的網際網路郵件協定。

JavaMail API 主要由 IMAP、MAPI、CMC、c 用戶端與其他電子郵件訊息傳送系統 API 產生。JavaMail API 支援許多不同的訊息傳送系統執行，如不同的訊息儲存、不同的訊息格式、不同的訊息傳輸。JavaMail API 提供基本類別集與定義用戶端應用程式 API 的介面，開發人員可以將 JavaMail 類別分爲子類別，以提供特定訊息傳送系統的執行，如 IMAP、POP3 與 SMTP。

本章節涵蓋以下主題：

- 關於 JavaMail 訊息處理程序
- 關於 JavaMail 的架構元件
- 關於 JavaBean 啓動框架 (JAF)
- 關於 JavaMail 配置參數
- JavaMail 階段作業參考的 J2EE 部署描述元
- Sun ONE Application Server 部署描述元中的項目
- 建立新的 JavaMail 階段作業
- 配置進階資源特性

關於 JavaMail 訊息處理程序

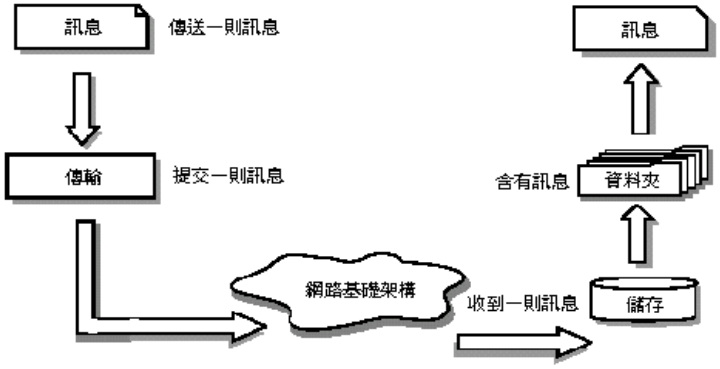
JavaMail API 執行以下功能，組成了典型用戶端應用程式的標準郵件處理程序：

- 建立了由標頭屬性集合組成的郵件訊息以及如在 Content-Type 標頭欄位中指定的某些已知資料類型的資料區塊。JavaMail 使用 Part 介面與 Message 類別來定義郵件訊息。它使用 JAF 定義的 DataHandler 物件包含在訊息中的資料。
- 建立 Session 物件，用於認證使用者，並且控制存取訊息儲存與傳輸。
- 將訊息發送至其接收者清單。
- 從訊息儲存中擷取訊息。
- 對擷取的訊息執行高層指令。高層指令 (如 view 與 print) 預定透過 JAF-Aware JavaBean 執行。

注意 目前，JavaMail 框架不定義支援訊息傳送、安全性、取消連線作業、目錄服務或過濾功能的機制。

下圖列舉 JavaMail API 處理訊息處理程序的方式：

圖 10-8 Java Mail API 的訊息處理程序



透過靜態 Factory 方法配置 JavaMail API 來建立 `javax.mail.Session`。Sun ONE Application Server 使用 JNDI 請求 Session 物件，並且使用 `resource-ref` 元素列示其對部署描述元中的 Session 物件的需要。JavaMail API Session 物件被認爲是資源 Factory。

提供的訊息傳輸能夠處理 `javax.mail.internet.InternetAddress` 類型的位址與 `javax.mail.internet.MimeMessage` 類型的訊息。必須正確配置預設的訊息傳輸，才能使用 `javax.mail.Transport` 類別的發送方法發送此類訊息。

JavaMail API 的抽象層宣告預定支援所有郵件系統支援的郵件處理功能之類別、介面與抽象方法。構成抽象層的 API 元素將被分爲子類別，並在必要時延伸，以支援標準資料類型，並在必要時與訊息存取協定與訊息傳輸協定相接。

網際網路執行層使用網際網路標準 - `RFC822` 與 `MIME` 執行抽象層部分。

關於 JavaMail 的架構元件

本章節描述組成 JavaMail 架構的主要元件，如以下主題所述：

- Message 類別
- 訊息儲存與擷取
- 訊息構成與傳輸

Message 類別

Message 類別是抽象的類別，定義屬性集與郵件訊息內容。Message 類別中的屬性指定尋址資訊，並定義內容結構，包括內容類型。內容表示爲涉及實際資料的 DataHandler 物件。

Message 類別執行 Part 介面。Part 介面定義需要使用的屬性，用於定義並格式化 Message 物件具有的資料內容，並且與郵件系統成功相接。Message 類別加入透過訊息傳輸系統進行訊息路由所需的 From、To、Subject、Reply-To 與其他屬性。當 Message 類別包含在資料夾中時，Message 物件具有與之關聯的旗標集。JavaMail 提供支援特定訊息傳送執行的 Message 子類別。

訊息內容是位元組集合，或是位元組集合的參考，封裝在 Message 物件中。JavaMail 並不瞭解訊息內容的資料類型或格式。Message 物件透過中間層 — JavaBean 啓動框架 (JAF) 與其內容進行互動。這種分隔可讓 Message 物件處理任何隨意內容，並透過呼叫同一種 API 方法以任何適當的傳輸協定來傳輸它。訊息接收者通常瞭解內容資料類型與格式，並且了解如何處理此內容。

同時，JavaMail API 支援多部分 Message 物件，其中每個 Bodypart 定義其自己的屬性集與內容。

訊息儲存與擷取

訊息儲存在 Folder 物件中。Folder 物件可以包含子資料夾以及訊息，因此提供了樹狀資料夾階層。Folder 類別宣告擷取、附加、複製與刪除訊息的方法。Folder 物件也可以將事件發送至作爲事件偵聽程式註冊的元件。

Store 類別

Store 類別定義具有資料夾階層及其訊息的資料庫。Store 類別還指定存取資料夾與擷取儲存在資料夾中的訊息之存取協定。Store 類別還提供建立資料庫連線、擷取資料夾與關閉連線的方法。執行訊息存取協定 (IMAP、POP3 等) 的服務提供者透過將 Store 類別進行子分類來開始。通常，使用者透過連線至特定的儲存執行，啓動郵件系統中的階段作業。

訊息構成與傳輸

用戶端透過實例化適當的 Message 子類別來建立新的訊息。用戶端會設定如接收者位址與主題等屬性，並且在 Message 物件中插入內容。最後，透過呼叫 Transport.send 方法發送此訊息。Transport 類別以將訊息路由到其目標位址的傳輸代理程式爲模型。此類別提供將訊息發送至接收者清單的方法。呼叫 Message 物件中的 Transport.send 方法，會識別基於其目標位址的適當傳輸。

Session 類別

Session 類別定義全域特性與每個使用者郵件相關的特性，這些特性定義郵件啓用用戶端與網路之間的介面。

JavaMail 系統元件使用 Session 物件設定並取得特定特性。Session 類別同時還提供桌面應用程式可以共用的預設認證工作時段物件。Session 類別是最終的具體類別，不能進行子類別分類。Session 類別還可作爲執行特定存取協定與傳輸協定的 Store 與 Transport 物件 Factory。透過對 Session 物件呼叫適當的 Factory 方法，用戶端可以取得支援特定協定的 Store 與 Transport 物件。

關於 JavaBean 啓動框架 (JAF)

JavaMail 使用 JavaBean 啓動框架 (JAF) 以便封裝訊息資料，並且處理要與資料互動的指令。與訊息資料互動應該透過不是由 JavaMail API 提供的支援 JAF 之 JavaBean 進行。

使用 JavaBean 啓動框架標準延伸，使用 Java 技術的開發人員可以利用標準服務決定資料任一部分的類型，封裝對它的存取，發現該資料中可用的作業，並且創設適當的 Bean 以執行上述作業。例如，如果瀏覽器取得 JPEG 影像，此框架會啓動瀏覽器將資料流識別爲 JPEG 影像，從此類型，瀏覽器會找到並創設可以處理或檢視影像的物件。

JavaBean 啓動框架 API 支援各種 MIME 資料類型。JavaMail API 必須包含 `javax.activation.DataContentHandlers`，用於下表中指出的 Java 程式設計語言類型對應之以下 MIME 資料類型。

表格 10-7 Java 類型對映的 JavaMail API MIME 資料類型

MIME 類型	Java 類型
Text/Plain	java.lang.String
Multipart/	javax.mail.internet.MIME.Multipart
Message/rfc822	javax.mail.internet.MIME.Message

JavaBean 啓動框架將 MIME 資料類型的支援整合至 Java 平台。MIME 位元組串流可以使用 avax.activation.DataContentHandlerobjects，從 Java 程式設計語言物件轉換或轉換爲 Java 程式設計語言物件。可以指定 JavaBean 元件對 MIME 資料作業，例如檢視或編輯資料。JavaMail API 使用 JavaBean 啓動框架來處理電子郵件訊息中包含的資料。雖然使用複雜電子郵件的應用程式可能需要此框架，但典型的 J2EE 應用程式不需直接使用 JavaBean 啓動框架。

關於 JavaMail 配置參數

Sun ONE Application Server 中的 JavaMail 資源使用以下配置參數。這些配置參數是將從 `server.xml` 檔案的 `mail-resource` 元素中讀取的名稱、值對。

- JNDI Name

  JNDI 名稱指定從 J2EE 應用程式參考的此郵件資源之名稱。

- Enabled

  `enabled` 配置參數指定是否將在 JNDI 樹中出版此郵件資源並且參考它。如果 J2EE 應用程式參考了已停用的資源，其將接收到 `NameNotFoundException` 異常。

- store-protocol

  指定預設訊息存取協定。`Session.getStore()` 方法傳回執行此協定的 Store 物件。用戶端可以置換此特性，並且透過 `Session.getStore(String protocol)` 方法明確指定此協定。

- store-protocol class

  指定執行以上指定的儲存協定之類別名稱。此類別的預設值爲 `com.sun.mail.imap.IMAPStore`。

- transport-protocol

  指定預設傳輸協定。`Session.getTransport()` 方法傳回執行此協定的 Transport 物件。用戶端可以置換此特性，並且透過 `Session.getTransport(String protocol)` 方法明確指定此協定。

- transport-protocol class

  指定執行以上指定的傳輸協定之類別名稱。此類別的預設值爲 `com.sun.mail.smtp.SMTPTransport`。

- host

  指定預設的郵件伺服器。如果此協定特定的主機特性不存在，則 Store 與 Transport 物件的連線方法使用此特性尋找目標主機。

- user

  指定連線至郵件伺服器時提供的使用者名稱。如果此協定特定的使用者名稱特性不存在，則 Store 與 Transport 物件的連線方法使用此特性以取得此使用者名稱。

- from

  指定目前使用者的傳回位址。透過 `InternetAddress.getLocalAddress` 方法來指定目前使用者的電子郵件位址。

- debug

  指定初始除錯模式。將此特性設定爲 True，將開啓除錯模式，而將其設定爲 False，則會關閉除錯模式。

- mail-<protocol>-host

  指定協定特定的預設郵件伺服器。這會置換 `mail.host` 特性。可以取決於 store-protocol 屬性的值設定此特性。如需 store-protocol 的 imap 或 POP 值，需要分別使用名稱 `mail.imap.host` 或 `mail.pop3.host` 加入特性。應該在您的郵件系統配置中適當地設定此特定特性的值。例如，如果 store-protocol 設定爲 IMAP，特性名稱 `mail-imap-host` 的值將爲 `spaceduck.acme.com`。

- mail-<protocol>-user

  指定連線至郵件伺服器的協定特定之預設使用者名稱。這會置換 `mail.user` 特性。因此該特性可以爲 `mail.imap.user` 或 `mail.pop3.user`，具體取決於 store-protocol 屬性的選取。例如，如果 store-protocol 設定爲 IMAP，特性名稱 `mail-imap-user` 的值將爲 `fredbloggs`。

JavaMail 階段作業參考的 J2EE 部署描述元

使用此伺服器註冊 JavaMail 資源後，可以使用 JNDI 查找在任何 J2EE 應用程式元件中參考此資源。爲了部署參考資源管理程式連線 Factory 的應用程式，元件提供者必須在標準的 J2EE 1.3 部署描述元中宣告所有的資源管理程式連線 Factory 參考。

JavaMail 參考的完整 J2EE1.3 描述元元素如下所述：

```
<resource-ref>
    <description>
        用於發送我的郵件之 JavaMail 資源
     </description>
     <res-ref-name>mail/MyMailSession</res-ref-name>
     <res-type>javax.mail.Session</res-type>
     <res-auth>Container</res-auth>
     <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

Sun ONE Application Server 部署描述元中的項目

如需參考郵件資源的每個已部署元件，部署程式必須將元件中使用的資源名稱對映於使用命名服務註冊的 DataSource 之實際 jndi 名稱。部署工具可以協助部署程式輕鬆進行此對映。此對映在 Sun ONE Application Server specific xml 中註冊。包含此對映的 Sun ONE Application Server 特定的 XML 分段如下所示：

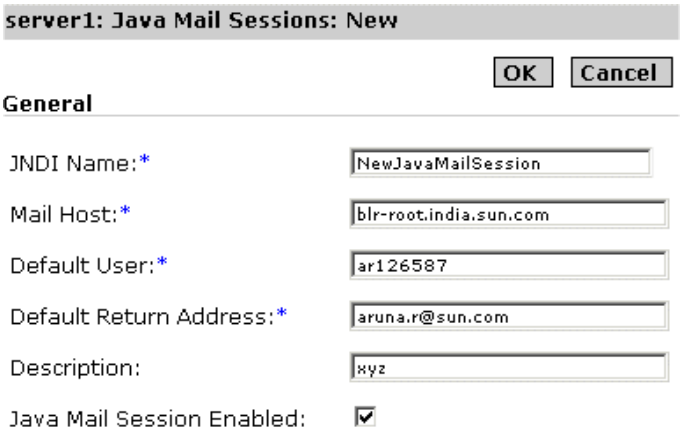
```
<resource-ref>
   <res-ref-name>mail/MyMailSession</res-ref-name>
   <jndi-name>mail/Session</jndi-name>
</resource-ref>
```

建立新的 JavaMail 階段作業

您可以使用管理介面配置 JavaMail 階段作業。若要建立與配置新的 JavaMail 階段作業，請執行以下工作：

1. 在管理介面的左窗格中，展開您要爲其建立新的 JavaMail 階段作業的 Sun ONE Application Server 實例。
2. 按一下 [JavaMail Sessions]。您會在管理介面右窗格中看到以下視窗，如圖「配置 JavaMail 階段作業」所示：

圖 10-9 配置 JavaMail 階段作業



3. 在 [JNDI Name] 文字欄位中，提供您要建立的 Java 郵件階段作業之 JNDI 名稱。使用此伺服器註冊 Java 郵件資源之後，便可以使用 JNDI 查找在任何 J2EE 應用程式元件中參考此資源。
4. 在 [Mail Host] 文字欄位中，指定預設郵件伺服器的 DNS 名稱。如果協定特定的主機特性不存在，則 Store 與 Transport 物件的連線方法會使用此特性，來尋找目標主機。
5. 在 [Default User] 文字欄位中，指定連線至郵件伺服器時提供的使用者名稱。如果協定特定的使用者名稱特性不存在，則 Store 與 Transport 物件的連線方法會使用此特性，以取得使用者名稱。
6. 在 [Default Return Address] 欄位中，指定目前使用者的預設傳回位址。預設位址應該以下列形式表示：username@host。
7. 在 [Description] 欄位中，提供此 Java 郵件階段作業的說明。

8. 標示 [Java Mail Session Enabled] 核取方塊，以啓用您建立的 Java 郵件階段作業。
9. 按一下 [OK]，以儲存您已經配置的新 JavaMail 階段作業。

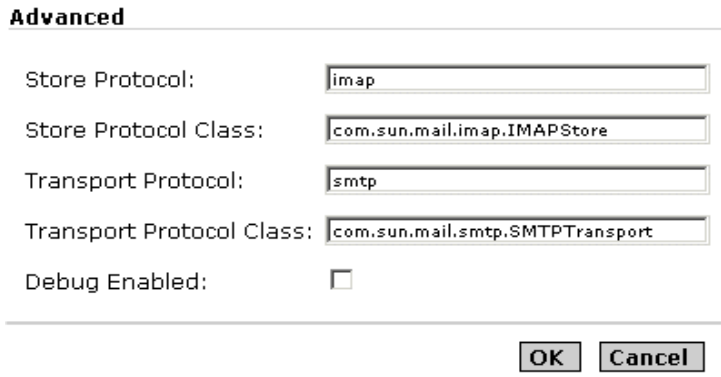
配置進階資源特性

您可以使用管理介面爲新的 JavaMail 階段作業配置一些附加特性。特性名稱與值對取決於使用中的郵件協定。也可以直接在 server.xml 檔案中指定這些特性。

若要配置附加特性，請執行以下工作：

1. 在管理介面的左窗格中，展開您要修改其 JavaMail 階段作業配置的 Sun ONE Application Server 實例。
2. 按一下 [JavaMail Sessions]。您會在管理介面右窗格 (在「建立新的 JavaMail 階段作業」中解釋的主要配置區段下方) 中看到以下視窗，如圖「配置 JavaMail 階段作業的附加資源」所示：

圖 10-10 配置 JavaMail 階段作業的附加資源



3. 在 [Store Protocol] 文字欄位中，指定您要在此特定 JavaMail 階段作業中使用的儲存協定，例如 POP3 或 IMAP。
4. 在 [Store Protocol Class] 文字欄位中，指定您已指出的儲存協定之類別名稱，如給定的範例所示。
5. 在 [Transport Protocol] 文字欄位中，指定您要在此特定 JavaMail 階段作業中使用的傳輸協定，例如 SMTP。

6. 在 [Transport Protocol Class] 文字欄位中，指定您已爲此階段作業指出的傳輸協定之類別名稱，如以上範例所示。
7. 標示 [Debug Enabled] 核取方塊，以啓用對此 JavaMail 特定階段作業的除錯。啓用此核取方塊會開啓除錯模式。
8. 按一下 [OK]，以儲存附加特性配置。

完整的郵件資源配置範例如下所示：

```
<mail-resource
    jndi-name = "mail/Session"
    enabled = "true"
    store-protocol = "imap"
    store-protocol-class = "com.sun.mail.imap.IMAPStore"
    transport-protocol = "smtp"
    transport-protocol-class = "com.sun.mail.smtp.SMTPTransport"
    host = "gopostal.acme.com"
    user = "kingkong"
    from = "kingkong@acme.com"
    debug = "false">
    <property name = "mail-imap-host" value = "spaceduck.acme.com"/>
    <property name = "mail-imap-user" value = "fredbloggs"/>
</mail-resource>
```

第 11 章

使用 JMS 服務

Sun ONE Application Server 提供使用 Java Message Service (JMS) 應用程式設計介面 (API) 進行訊息傳送作業的應用程式支援。JMS 是程式設計介面集，提供 Java 應用程式在分散式環境中建立、傳送、接收並讀取訊息的一般方法。

特別強調的是，JMS 是 Java 2 Enterprise Edition (J2EE) 應用程式執行非同步訊息傳送服務的標準方法。因此，J2EE 元件 (Web 元件或企業 JavaBeans [EJB] 元件) 可以使用 JMS API 傳送由專用 EJB (名爲訊息驅動 Bean [MDB]) 非同步使用的訊息。

Sun ONE Application Server 一般對 JMS 訊息傳送的支援，特別是對 MDB 的支援，需要執行 JMS 規格的訊息傳送介體 (JMS 提供者)。Sun ONE Application Server 使用 Sun ONE Message Queue (MQ) 版本 3.01 作爲其原生 JMS 提供者。

MQ 緊密整合至 Sun ONE Application Server 中，提供透明的 JMS 訊息傳送支援。此支援 (在 Sun ONE Application Server 中稱爲 *JMS* **服務**) 僅需要最少的管理。

本章提供瞭解並管理內建 JMS 服務 (該服務透過 Sun ONE Message Queue 提供) 所需的資訊。其包括以下主題：

- 關於 JMS
- 內建 JMS 服務
- 內建 JMS 服務的管理

關於 JMS

JMS 規格描述支援分散式、企業訊息傳送的程式設計介面集。企業訊息傳送系統可讓獨立分散式元件或應用程式在訊息之間互動。這些元件無論是否在同一系統中、同一網路中或透過網際網路鬆散連線，均使用訊息傳送來傳輸資料，並且與其各自的功能相互作用。

若要支援企業級訊息傳送，JMS 會提供可靠的非同步訊息發送。

可靠的發送。從一個元件向另一個元件傳送的訊息不能因網路或系統故障而遺失。這意味著系統必須能夠保證訊息成功發送。

非同步發送。若要大量的元件能夠同時交換訊息，並且支援高密度的訊息流量，則傳送訊息不能取決於使用者是否願意立即接收此訊息。如果使用者很忙或離線，系統必須允許發送訊息，並且在隨後使用者就緒時能夠收到此訊息。這稱爲非同步訊息發送，通稱爲儲存再前送訊息傳送。

本主題簡要概述 JMS 概念與術語：

- 基本訊息傳送系統概念
- JMS 規格
- 訊息導引 Bean

如需 JMS 的完整說明，請參閱 JMS 1.0.2 規格，可以在以下位置找到：

```
http://java.sun.com/products/jms/docs.html
```

基本訊息傳送系統概念

一些基本的概念通常構成企業訊息傳送系統的基礎，特別是 JMS 的基礎。本主題將論述以下內容。

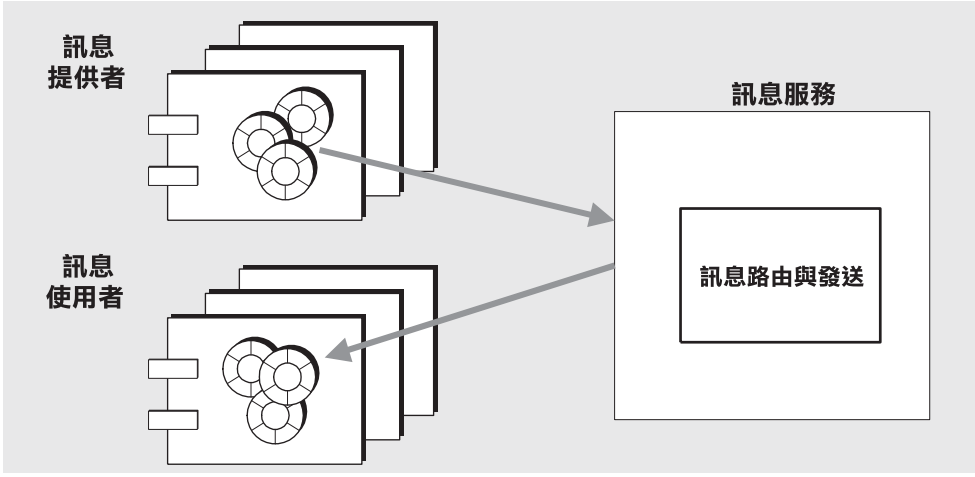
訊息

訊息由某種格式 (訊息內文) 的資料與描述訊息 (訊息標頭) 特徵或特性的複合資料組成，例如訊息目標、使用期或由訊息傳送系統決定的其他特徵。

訊息服務架構

在圖「訊息服務架構」中列舉了訊息傳送系統的基本架構。它由訊息提供者與訊息使用者組成，兩者之間透過一般的訊息服務進行訊息交換。一般來說，任何數目的訊息提供者與使用者均可以駐留在同一訊息傳送元件中。訊息提供者將訊息傳送至訊息服務。訊息服務反過來使用訊息路由與發送元件，將訊息發送至已在此訊息中註冊興趣的一個或多個訊息使用者。訊息路由與發送元件負責保證訊息發送至所有適當的使用者。

圖 11-1 訊息服務架構



訊息發送模型

訊息提供者與使用者之間有許多關係：一對一、一對多與多對多關係。例如，您可以使訊息從：

- 一個提供者向一個使用者發送
- 一個提供者向多個使用者發送
- 多個提供者向一個使用者發送
- 多個提供者向多個使用者發送

這些關係通常會成爲兩個訊息發送模型：**點對點**與**發佈/訂購**訊息傳送。點對點發送模型主要集中於由特定提供者發送並由特定使用者接收的訊息。發佈/訂購發送模型主要集中於由一些提供者中的任何使用者發送的訊息與任意多個使用者接收的訊息。這些訊息發送模型可以重疊。

從歷程來看，訊息傳送系統支援這兩種發送模型的各種組合。JMS API 試圖建立一般程式設計方法，用以支援點對點與發佈/訂購發送模型。

JMS 規格

JMS 指定一個訊息結構、一個程式設計模型以及一組控制訊息傳送作業的規則與語義。

JMS 訊息結構

依據 JMS 規格，訊息由三個部分組成：標頭、特性 (可以作爲標頭的延伸) 與內文。

標頭。此標頭指定訊息的 JMS 特徵：訊息目標、是否具有持續性、使用期及其優先權。這些特徵決定了訊息傳送系統發送訊息的方式。

特性。特性是選擇性的 — 它們提供應用程式根據不同的選取條件篩選訊息可使用的值。特性是選擇性的。

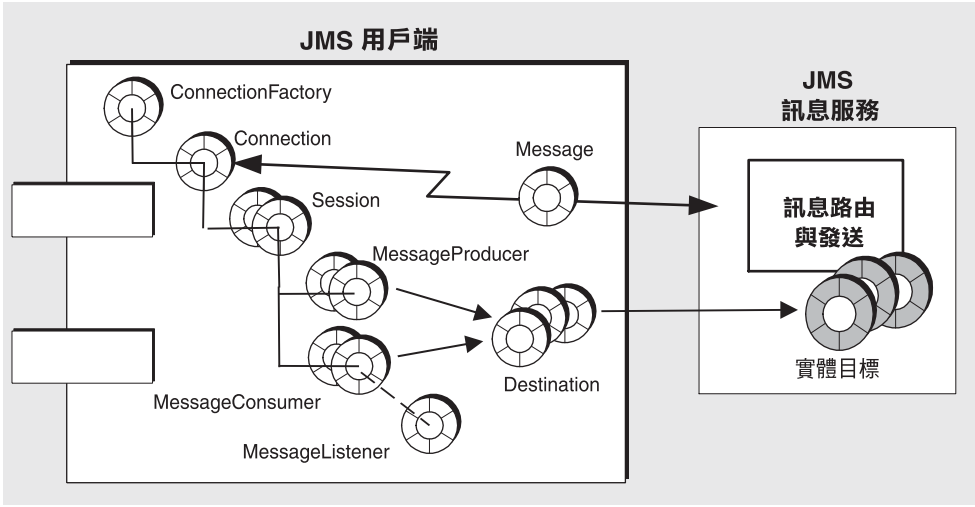
訊息內文。訊息內文包含要交換的實際資料。JMS 支援六種內文類型。

JMS 程式設計模型

在 JMS 程式設計模型中，JMS 用戶端 (元件或應用程式) 透過 JMS 訊息服務方式交換訊息。訊息提供者將訊息傳送至訊息服務，而訊息使用者透過訊息服務接收這些訊息。使用執行 JMS API 的物件集 (由 JMS 提供者提供)，執行這些訊息傳送作業。圖「JMS 程式設計物件」顯示用於程式設計訊息發送的 JMS 物件。

在 JMS 程式設計模型中，JMS 用戶端使用 `ConnectionFactory` 物件，建立將訊息傳送至 JMS 訊息服務並從該服務接收訊息要使用的連線。`Connection` 是訊息服務的 JMS 用戶端使用中連線。通訊資源的分配與用戶端的認證均在連線建立時發生。

圖 11-2 JMS 程式設計物件



此連線用於建立階段作業。`Session` 是提供與使用訊息的單執行緒環境。其用於建立傳送訊息的訊息提供者與接收訊息的訊息使用者。階段作業透過一系列的確認選項或作業事件 (可以由分散式作業事件管理程式管理) 支援可靠的發送。

JMS 用戶端使用 `MessageProducer`，將訊息傳送至指定的實體目標，在 API 中表示爲 destination 物件。訊息提供者可以指定預設發送模式 (永久性訊息對非永久性訊息)、優先權與使用期，這些會控制提供者發送至實體目標的所有訊息。

類似的是，JMS 用戶端使用 `MessageConsumer`，接收指定實體目標中的訊息，在 API 中表示爲 destination 物件。訊息使用者可以支援訊息的同步或非同步使用。非同步使用可以透過使用者註冊 MessageListener 來實現。用戶端在階段作業執行緒呼叫 `MessageListener` 物件的 `onMessage()` 方法時，使用訊息。

管理物件：提供者獨立性

在圖第 274 頁的「JMS 程式設計模型」中描述的兩個物件取決於 JMS 提供者執行 JMS 訊息服務的詳細資訊。連線 Factory 物件取決於基礎協定與提供者發送訊息所使用的機制，而 destination 物件取決於特定命名慣例以及提供者使用的實體目標功能。

通常，這些提供者特定的特徵會使 JMS 用戶端碼與 JMS API 執行的詳細資訊相依。不過，若要使 JMS 用戶端碼與提供者無關，JMS 規格需要以標準化方式存取提供者特定的物件 (稱爲管理物件)，而不是直接在用戶端碼中創設。

管理物件封裝提供者特定的執行以及配置資訊。這些物件由管理員建立與配置，儲存在名稱服務中，由用戶端應用程式透過標準 JNDI 查找碼進行存取。以這種方式使用管理物件，可以使 JMS 用戶端碼與提供者無關。

JMS 提供兩種一般類型的管理物件：連線 Factory 與目標。兩種類型均封裝提供者特定的資訊，但是在 JMS 用戶端內卻具有完全不同的用法。連線 Factory 用於建立訊息伺服器的連線，而 destination 物件用於識別 JMS 訊息服務使用的實體目標。

注意 在 Sun ONE Application Server 的環境中，JMS 管理物件被視爲 JMS 資源，類似於其他 Application Server 資源。

訊息導引 Bean

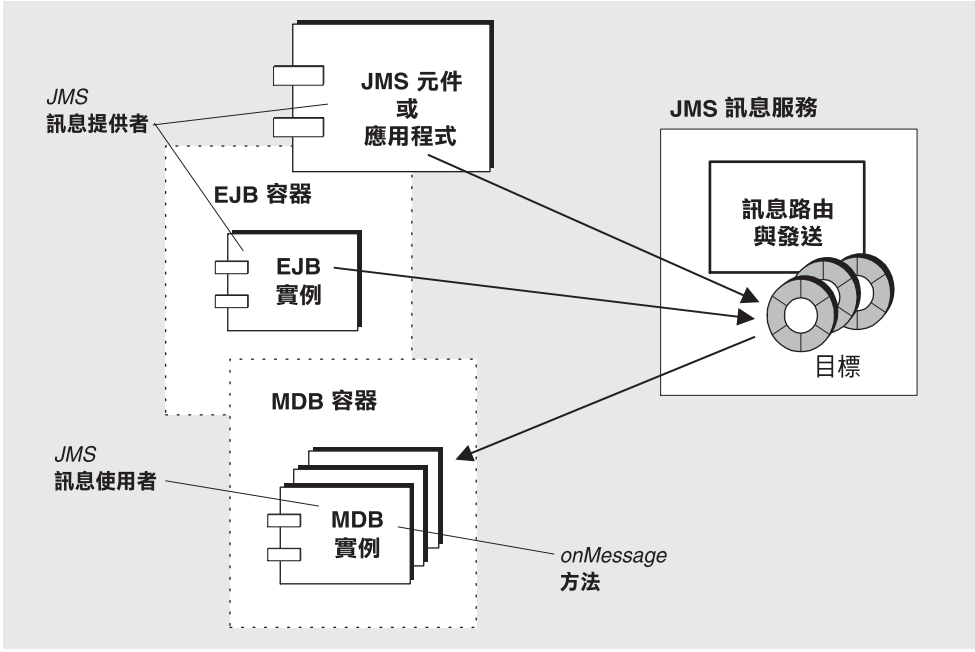
除了在圖第 274 頁的「JMS 程式設計模型」中介紹的一般 JMS 用戶端程式設計模型外，還有一個在 J2EE 應用程式環境中使用的更爲專用的 JMS API。這一專用的 JMS 用戶端稱爲**訊息導引** *Bean*，是在 EJB 2.0 規格中指定的家族 EJB 元件之一 (位於 `http://java.sun.com/products/ejb/docs.html`)。

需要使用訊息導引 Bean 的原因在於只可以同步呼叫其他 EJB 元件 (階段作業 Bean 與實體 Bean)。因此，在您呼叫這些 Bean 的方法時，會阻止使用這些資源，直至這些方法已完成。這些 EJB 元件沒有非同步接收訊息的機制，因爲這些元件僅透過標準的 EJB 介面存取。

但是，非同步訊息傳送是許多企業應用程式的需求。因此，需要可以在沒有緊密與訊息提供者結合的情況下，接收訊息並使用訊息的 EJB 元件。

MDB 是專用的 EJB 容器支援的專用 EJB 元件 (爲此環境支援的元件提供分散式服務的軟體環境)。

圖 11-3 MDB 訊息使用者



訊息導引 Bean。MDB 是執行 JMS `MessageListener` 介面的 JMS 訊息使用者。其 `onMessage` 方法 (由 MDB 開發人員撰寫) 在 MDB 容器收到訊息時呼叫。`onMessage` 方法使用訊息的方式與任何 JMS MessageListene 物件的 `onMessage` 方法使用訊息的方式一樣。MDB 可以使用從單一目標發送的訊息。這些訊息可以由獨立的 JMS 用戶端應用程式、Web 元件或其他 EJB 元件提供，如圖「MDB 訊息使用者」所示。

MDB 容器。專用的 EJB 容器支援 MDB，MDB 負責建立 MDB 實例，並且爲非同步使用訊息設定這些實例。在階段作業儲存區與關聯的 MDB 實例中接收訊息時，會設定與訊息服務 (包括認證) 的連線、建立與給定目標關聯的階段作業儲存區以及管理訊息分配。由於容器可控制 MDB 實例的生命週期，因此其會管理 MDB 實例儲存區，以便容納進來的訊息負載。

與 MDB 關聯的是部署描述元，此描述元指定了容器在設定訊息使用時所使用的管理物件之 JNDI 查找名稱。連線 Factory 與目標。部署描述元可能也包括可由部署工具用於配置容器的其他資訊。每個此類容器僅支援單一 MDB 實例。

如需關於配置 Sun ONE Application Server 的 EJB 容器 MDB 設定的資訊，請參閱第 195 頁的「關於訊息導引 Bean」。

內建 JMS 服務

一般對 JMS 訊息傳送的支援，特別是對 MDB 的支援，內建於 Sun ONE Application Server 中。這種支援可以透過 Sun ONE Message Queue 與 Sun ONE Application Server 的緊密整合來實現，提供原生的內建 JMS 服務。

本主題包含瞭解此內建 JMS 服務所必需的以下主題：

- 關於 Sun ONE Message Queue (MQ)
- 整合 MQ 與 Sun ONE Application Server

如需關於管理內建 JMS 服務的資訊，請參閱第 284 頁的「內建 JMS 服務的管理」。

關於 Sun ONE Message Queue (MQ)

Sun ONE Message Queue (MQ) 是執行 JMS 開放標準的企業訊息傳送系統：其爲 JMS 提供者。

MQ 產品的某些功能超越了可靠的、非同步訊息傳送的 JMS 規格之最低需求。某些功能 (中央管理、可調效能、多訊息傳送傳輸支援、使用者認證與授權) 可以在整合於 Sun ONE Application Server 中的 MQ Platform Edition 中取得。其他功能 (可縮放訊息伺服器與安全訊息傳送) 可透過升級至 MQ Enterprise Edition 取得。

如圖第 279 頁的「MQ 系統架構」所示，MQ 訊息傳送系統由多個部分組成，這些部分可以協同作業以提供可靠的訊息發送。

MQ 訊息傳送系統的主要部分如下：

- MQ 訊息伺服器
- MQ 用戶端執行期間
- MQ 管理物件
- MQ 管理工具

以下主題簡要介紹這些內容。如需關於 MQ 訊息傳送系統的更完整論述，請參閱 MQ「**管理員指南**」，可以在以下位置找到：

```
http://docs.sun.com/
```

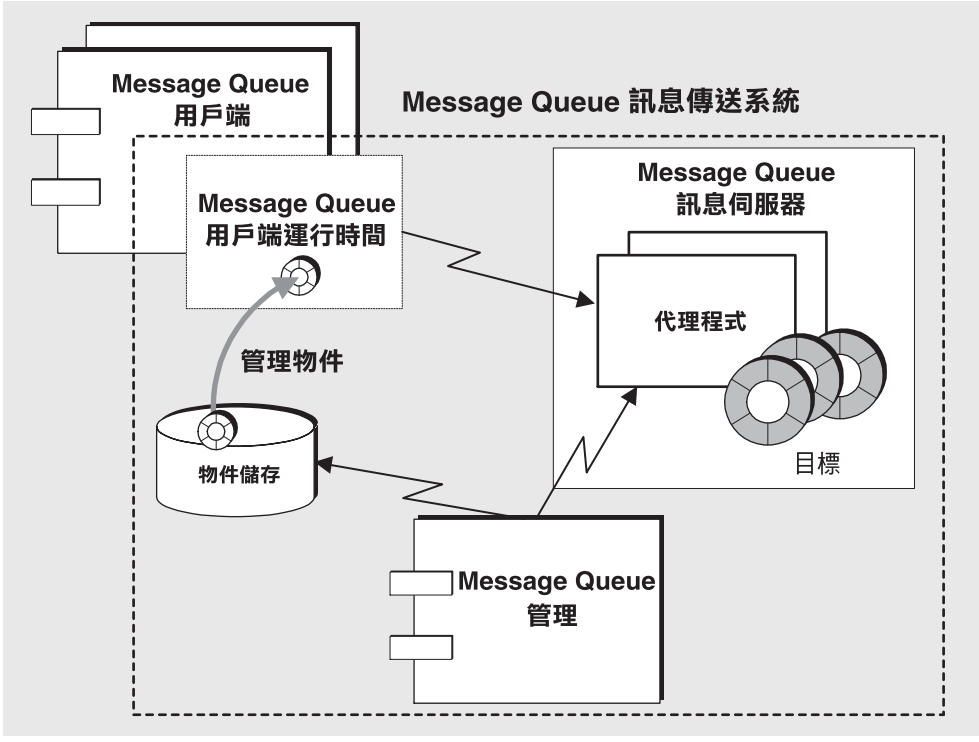
MQ 訊息伺服器

如圖第 279 頁的 「MQ 系統架構」所示，MQ 訊息伺服器的主要部分爲代理程式與實體目標。

代理程式。代理程式爲 MQ 訊息傳送系統提供發送服務。訊息發送依賴於處理連線服務、訊息路由與發送、持續性、安全性以及記錄的一些支援元件。訊息伺服器可以使用一個或多個代理程式實現縮放性。

實體目標。訊息發送是兩階段的程序，即從提供訊息的用戶端發送至代理程式維護的實體目標，然後從該目標發送至一個或多個使用訊息的用戶端。實體目標表示代理程式實體記憶體和/或永久性儲存體中的位置 (請參閱第 280 頁的 「實體目標」，以取得更多資訊)。

圖 11-4 MQ 系統架構



代理程式

在 MQ 訊息傳送系統中的訊息發送 (即從提供訊息的用戶端發送至目標，然後從目標發送至一個或多個使用訊息的用戶端)，由代理程式 (在 MQ 3.01 Enterprise Edition 中，則爲在串聯中使用的代理程式叢集) 執行。若要執行訊息發送，代理程式必須設定與用戶端的通訊通道；執行認證與授權；正確路由訊息；保證可靠的發送以及提供監視系統效能的資料。

若要執行這一複雜的功能集，代理程式使用各種不同的元件，每個元件在發送程序中具有一個特定的角色。您可以配置這些內部元件，以最佳化代理程式的效能，這要取決於載入狀況、應用程式的複雜性等。請參閱 MQ「**管理員指南**」，以取得更多資訊。

實體目標

MQ 訊息傳送以兩階段訊息發送爲前提：第一個是從提供者用戶端至代理程式目標的訊息發送，第二個是從代理程式目標至一個或多個使用者用戶端的訊息發送。有兩種類型的目標：佇列 (點對點發送模型) 與主題 (發佈/訂購發送模型)。這些目標表示代理程式實體記憶體中的位置，從中對內送的訊息編組，然後再路由至使用者用戶端。

通常使用管理工具建立實體目標，不過，也可以在收到訊息時由代理程式自動建立目標。

佇列目標。佇列目標用於點對點訊息傳送，這意味著訊息最終僅傳送至在目標中註冊興趣的其中一個使用者。當訊息從提供者用戶端發出時，這些訊息會佇列，然後發送至使用者用戶端。

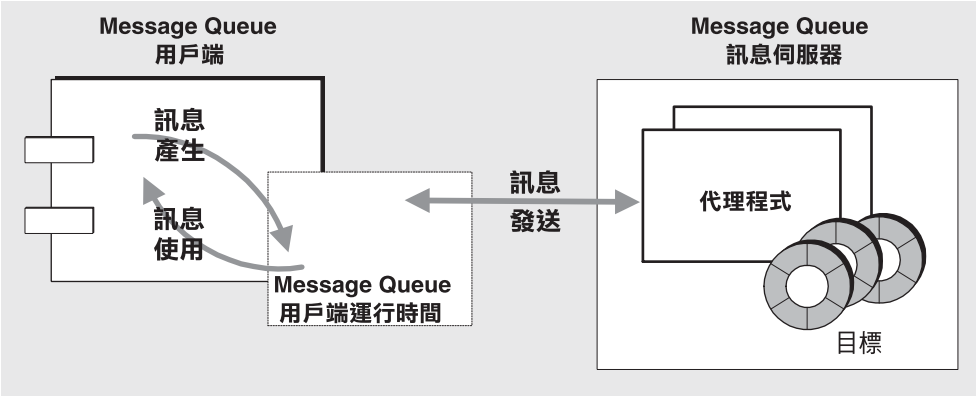
主題目標。主題目標用於發佈/訂購訊息傳送，這意味著訊息最終發送至已在目標中註冊興趣的所有使用者。當訊息從提供者發出時，其會路由至訂購此主題的所有使用者。如果使用者已註冊長期訂閱此主題，則他們不必在訊息發送至此主題時處於使用中，因爲代理程式將儲存此訊息，直至使用者再次處於使用中，然後再將此訊息發送至使用者。

MQ 用戶端執行期間

MQ 用戶端運行時間向 JMS 用戶端 (獨立應用程式、Web 元件或 EJB 元件) 提供 MQ 訊息伺服器的介面 —它向 JMS 用戶端提供用戶端將訊息傳送至目標，然後從此類目標接收訊息所需的所有程式設計介面的實現。

圖第 281 頁的「訊息傳送作業」列舉在 JMS 用戶端與 MQ 用戶端執行之間的訊息產生與使用的互動方式，而在 MQ 用戶端執行期間與 MQ 訊息伺服器之間具有訊息發送互動。

圖 11-5 訊息傳送作業



MQ 管理物件

MQ 管理物件允許 JMS 用戶端碼成爲獨立的提供者 (請參閱第 275 頁的「管理物件：提供者獨立性」)。可以透過封裝提供者特定的執行與物件中的配置資訊 (然後可以採用提供者無關的方式由用戶端應用程式使用)，執行此動作。MQ 管理物件由管理員建立與配置，儲存在名稱服務中，由 JMS 用戶端透過標準 JNDI 查找碼進行存取。

連線 Factory 管理物件。連線 Factory 物件用於在 JMS 用戶端 (獨立的應用程式、Web 元件或 EJB 元件) 與 MQ 訊息伺服器之間建立實體連線。連線 Factory 物件在代理程式中沒有實體表示 — 該物件僅用於啓動 JMS 用戶端，以建立與代理程式的連線。也可用來指定連線行爲與使用連線存取代理程式的用戶端執行期間行爲：因此，MQ 連線 Factory 具有很多可配置屬性，可讓您調整 MQ 系統效能。

目標管理物件。目標管理物件 (佇列或主題) 在代理程式中表示公共具名的目標管理物件對應的實體目標 (實體佇列或實體主題)。透過建立目標管理物件，可讓 JMS 用戶端 (訊息使用者和/或訊息提供者) 存取相應的實體目標。

MQ 管理工具

MQ 管理工具分爲兩類：指令行公用程式與圖形使用者介面 (GUI) 管理主控台。

管理主控台。您可以使用管理主控台，連線至代理程式並管理它；建立關於代理程式的實體目標；連線至物件儲存，然後加入、更新管理物件或從物件儲存中刪除它。某些工作是您無法使用管理主控台執行的；主要工作包括啓動代理程式、建立代理程式叢集、配置更多代理程式專用特性以及管理使用者資料庫。

指令行公用程式。使用 MQ 公用程式執行使用管理主控台可以執行的所有工作，此外，啓動並管理代理程式，配置更多代理程式專用特性並管理 MQ 使用者資料庫。

整合 MQ 與 Sun ONE Application Server

MQ Platform Edition 將作爲 Sun ONE Application Server 安裝程序的一部分自動安裝。如需更多資訊，請參閱「*Sun ONE Application Server* 安裝指南」。

此安裝向 Sun ONE Application Server 提供支援任意多個 Sun ONE Application Server 實例的 JMS 訊息傳送系統。依預設，每個伺服器實例均具有關聯的內建 JMS 服務，支援在此實例中執行的所有 JMS 用戶端。

本主題包括下列主題：

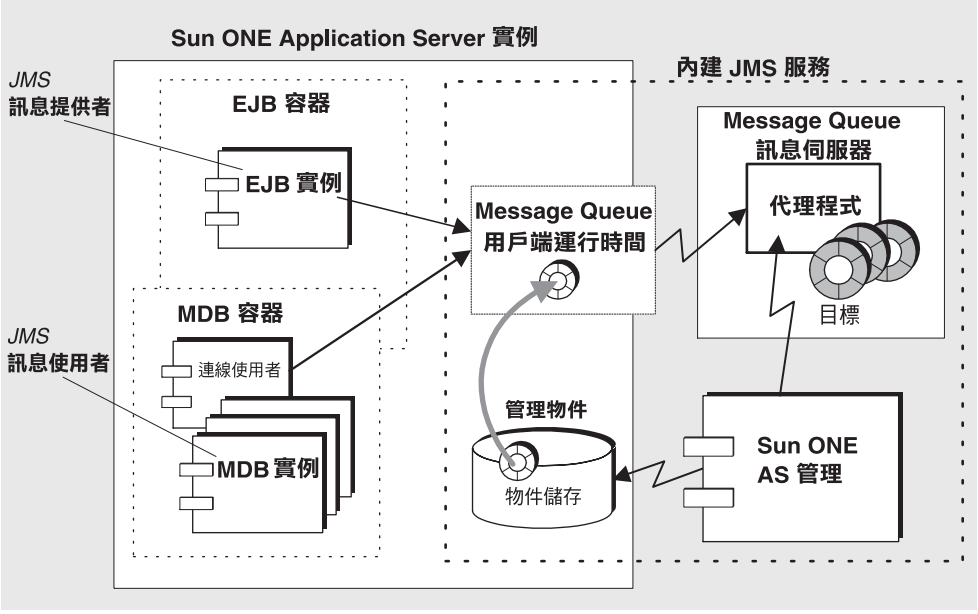
- 內建 JMS 服務架構
- 停用內建 JMS 服務

可以使用 Sun ONE Application Server 管理工具，管理內建的 JMS 服務 (請參閱第 284 頁的「內建 JMS 服務的管理」)。

內建 JMS 服務架構

除以下所述的一些限制外，內建 JMS 服務 (在圖第 282 頁的「內建的 MQ 訊息傳送系統」中進行說明) 類似於一般的 MQ 訊息傳送系統 (如圖第 279 頁的「MQ 系統架構」所示)。

圖 11-6 內建的 MQ 訊息傳送系統



MQ 訊息伺服器。每個 Sun ONE Application Server 實例均與其自身的內建 JMS 服務關聯。內建 JMS 服務使用單代理程式訊息伺服器。此代理程式在 Sun ONE Application Server 實例之外的單獨程序中運行，如圖「內建的 MQ 訊息傳送系統」所示。依預設，代理程式實例 (內建 JMS 服務) 在啓動其關聯伺服器實例時啓動，並在關閉該伺服器實例時關閉。伺服器實例的內建 JMS 服務之配置資訊記錄在 Sun ONE Application Server 配置儲存區中 (`server.xml` 檔案)，並且可以進行修改，如第 285 頁的「配置 JMS 服務」所述。

MQ 用戶端執行期間。JMS 服務的用戶端執行期間部分是支援 JMS API 的程式庫集。任何在伺服器實例中執行的 JMS 用戶端 (JMS 用戶端元件，包括 MDB)，均可存取這些程式庫。

MQ 管理物件。內建 JMS 服務使用 Sun ONE Application Server 提供的物件儲存區。每個伺服器實例均具有其自己的物件儲存區。JMS 服務在物件儲存區中儲存管理物件 (連線 Factory 與目標資源)。可以建立這些管理物件資源，如第 290 頁的「管理受管理物件資源」中所述，並且透過 JMS 用戶端使用 JNDI 查找碼存取這些資源。

Sun ONE Application Server 管理。Sun ONE Application Server 管理介面與指令行公用程式實施 MQ 管理功能的有限子集。管理介面與指令行可讓您配置內建的 JMS 服務，建立與刪除實體目標，以及建立與刪除 JMS 用戶端執行 JMS 訊息傳送作業所需的管理物件資源。但是，這些管理工具不允許 (或不協助) 執行更複雜的管理工作，例如設定代理程式特性、調整 MQ 用戶端執行期間、修改 MQ 使用者儲存庫、管理 MQ 安全性等。如果您要爲內建的 JMS 服務執行這些管理工作，必須使用隨 MQ 安裝並在 MQ「**管理員指南**」中描述的管理工具。如需 MQ 與 Sun ONE Application Server 管理功能的比對，請參閱表格第 285 頁的「Sun ONE Message Queue 與 Sun ONE Application Server 管理功能的比對」。

停用內建 JMS 服務

依預設，內建 JMS 服務在啓動關聯的 Sun ONE Application Server 實例時啓動 (即，啓動 MQ 代理程式)。但是，可能需要在啓動伺服器實例時，停用 JMS 服務的自動啓動功能，這是因爲伺服器實例不需要支援 JMS 訊息傳送，或是因爲伺服器實例使用**外部** *JMS* **服務**。(若要停用內建的 JMS 服務，請參閱第 285 頁的「配置 JMS 服務」。)

外部 JMS 服務是不在 Sun ONE Application Server 內控制的訊息傳送系統。對於 MQ 而言 (**原生** JMS 提供者)，這意味著僅僅啓動並獨立管理 MQ 訊息伺服器，使用 MQ 管理工具。在各種伺服器實例中執行的 JMS 用戶端仍然可以使用 MQ 管理物件存取 MQ 訊息伺服器。這些管理物件可以儲存在與每個應用程式伺服器實例關聯的物件儲存區中，也可儲存在 MQ 管理工具管理的單獨的獨立物件儲存區中 (如有必要，此儲存區可以由多個伺服器實例共用)。

有這樣一些方案，伺服器實例可以在其中使用外部 JMS 服務。最有可能的是當不同伺服器實例中的 JMS 用戶端需要存取同一個實體目標時。在這種情況下，伺服器實例必須均存取同一個訊息伺服器。若要實現此作業，必須停用所有伺服器實例的內建 JMS 服務，並配置所有 JMS 用戶端，以執行存取外部 JMS 服務的適當 JNDI 查找。此外，必須使用外部 JMS 服務提供者管理工具獨立管理外部 JMS 服務 (管理訊息伺服器、建立實體目標、建立所需的所有管理物件)。

配置多個應用程式伺服器實例以共用單一 MQ 代理程式實例的步驟：

1. 對所有伺服器實例停用 JMS 服務。
2. 管理獨立於任何伺服器實例的共用 MQ 代理程式，這就意味著您必須使用管理外部服務的管理工具啓動並關閉此代理程式。同時，必須管理獨立於 Sun ONE Application Server 的實體目標。
3. 在每個伺服器實例中配置連線 Factory JMS 資源，以便該資源會指向此外部 MQ 代理程式 (必須適當設定 `imqBrokerHostName` 與 `imqBrokerHostPort` 特性)。
4. 在 Sun ONE Application Server 中部署 JMS 應用程式時，使用此連線 Factory 資源。

可以在同時執行外部與內建 JMS 服務。伺服器實例中的 JMS 用戶端可以存取其所需的任何 JMS 服務。

不建議使用的一種可能性是數個伺服器實例共用同一個內建 JMS 服務 (啓用一個服務，則會停用其他服務)。不建議使用的原因是已啓用的 JMS 服務僅在其關聯的伺服器實例執行時才執行，從而很難管理這種情況。

您停用內建 JMS 服務時，同時也會停用執行與此內建 JMS 服務關聯的管理工作功能。必須使用管理外部服務的管理工具，執行支援外部 JMS 服務所需的所有管理工作。

內建 JMS 服務的管理

此主題主要論述內建 JMS 服務的管理。**會對伺服器實例逐個執行管理**。

內建 JMS 服務管理需要以下工作：

- 配置 JMS 服務
- 管理實體目標
- 管理受管理物件資源
- 使用指令行介面管理內建 JMS 服務

可以使用 Sun ONE Application Server 的管理介面或指令行公用程式執行管理。這些管理工具與 MQ 管理工具的比對顯示在表格「Sun ONE Message Queue 與 Sun ONE Application Server 管理功能的比對」中。

表格 11-1 Sun ONE Message Queue 與 Sun ONE Application Server 管理功能的比對

功能	Sun ONE MQ 管理工具	Sun ONE AS 管理介面	Sun ONE AS 管理指令行
管理 MQ 代理程式狀態	是	啓動/停止	啓動/停止
配置 MQ 代理程式	是	否	否
管理 MQ 代理程式使用者儲存庫	是	否	否
多代理程式叢集	是	否	否
管理安全性	是	否	否
管理實體目標	是	建立/刪除	建立/刪除
管理持久訂購與作業事件	是	否	否
管理受管理物件資源	是	建立/刪除/配置	是

以下小節解釋使用 Sun ONE Application Server 管理介面執行 JMS 服務管理工作的方式。

配置 JMS 服務

在安裝期間，爲內建 JMS 服務設定了一些 JMS 服務特性。可以透過配置 JMS 服務，變更這些特性的預設值。

在表格「JMS 服務特性」中描述 JMS 服務特性。

表格 11-2 JMS 服務特性

特性	描述	預設值
日誌層級	您要寫入至 Sun ONE Application Server 日誌檔的記錄資訊層級。如需更多資訊，請參閱第 5 章「使用記錄功能」。	`DEBUG_HIGH`

表格 11-2 JMS 服務特性 (**續**)

特性	描述	預設值
連接埠	提供內建 JMS 服務的代理程式實例的主連接埠號。依預設，內建 JMS 服務使用預設的主連接埠號。但是，如果此連接埠與其他軟體衝突，或者您要啓動多個 Sun ONE Application Server 實例，則需要爲每個實例指定唯一的主連接埠號。 請注意，如果在安裝期間指定給 JMS 服務的連接埠號後來被另一個服務使用，則有可能發生連接埠衝突。在這種情況下，必須爲 JMS 服務指定另一個連接埠號。	`7676`
管理員使用者名稱/密碼	執行代理程式管理工作 (例如管理實體目標，請參閱第 287 頁的「管理實體目標」) 所需要的使用者名稱/密碼。如果您要管理員對代理程式實例的存取具有安全性 (依預設，任何使用者均具有存取權)，需要首先在代理程式使用者儲存庫中建立適當的項目 (如 MQ「管理員指南」所述)，然後在此爲管理員使用者名稱與密碼輸入相應的值。	`admin/admin`
啓動逾時	以秒爲單位指定伺服器實例等待 JMS 服務啓動的時間。如果超出了此逾時時間，則伺服器實例啓動會中斷。	`60`
啓動引數	指定在啓動 JMS 服務中使用的任何引數。`imqbroker` 指令的啓動引數以及指定這些引數的方法，可以在 MQ「**管理員指南**」中找到。(如果提供 `-name` 與 `-port` 引數，則會忽略。)	
啓用啓動	指定在伺服器實例啓動時是否啓動內建 JMS 服務。如果您不支援 JMS 訊息傳送或使用外部 JMS 訊息服務，則將此特性設定爲 FALSE。	`TRUE`

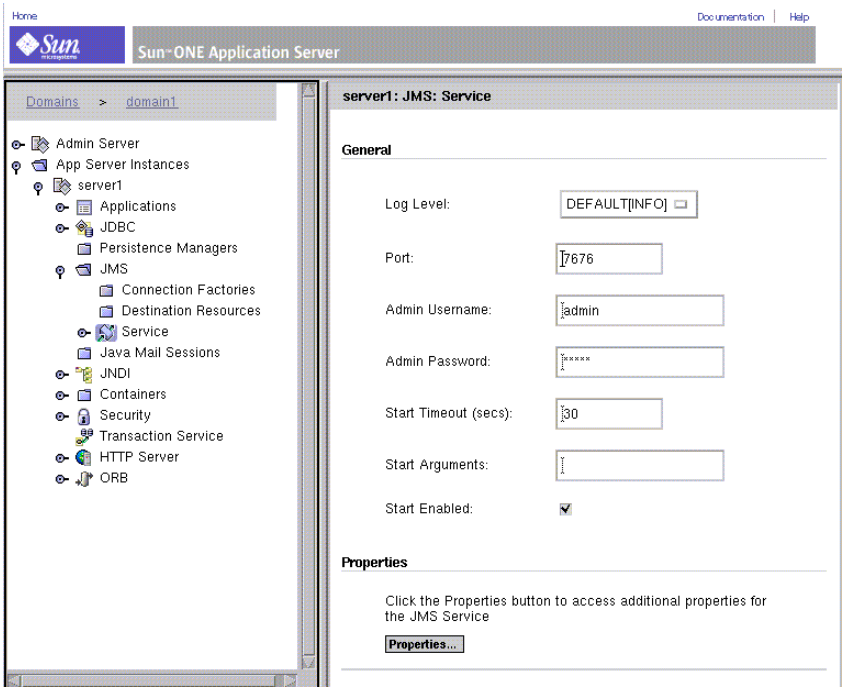
可以在啓動內建 JMS 服務的對應服務實例之前，配置此服務。但是，如果伺服器實例已經執行，配置變更僅在伺服器實例停止、隨後重新啓動之後才生效。

配置內建 JMS 服務的步驟：

1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。
4. 選取 [Service] 連結。

   [JMS Service Configuration] 螢幕會顯示在右窗格中。

圖 11-7 [JMS Service Configuration] 螢幕



5. 修改您需要的任何特性 (請參閱表格第 285 頁的 「JMS 服務特性」)。
6. 按一下 [Save] 按鈕。

   會重新顯示 [JMS Service] 螢幕。

管理實體目標

在 JMS 訊息傳送中，JMS 提供者將訊息傳送至訊息服務的實體目標中，然後這些訊息透過實體目標派送至 JMS 使用者。

對於內建 JMS 服務，您可以明確地建立這些實體目標，或透過在 JMS 服務 (MQ 代理程式) 接收到訊息時，自動建立實體目標。通常，透過明確建立訊息傳送應用程式所需的實體目標，可以更嚴密地控制訊息傳送系統及其資源。不再需要這些目標時，可以刪除它們。

爲了建立或刪除內建 JMS 服務的實體目標，JMS 服務必須在運行，並且管理員使用者名稱與密碼 (在配置內建 JMS 服務時指定) 必須對應於代理程式使用者儲存庫中的有效項目 (請參閱表格第 285 頁的 「JMS 服務特性」)。

使用管理介面，您可以爲內建 JMS 服務中的實體目標，執行以下管理工作：

- 建立佇列或主題目標
- 列示實體目標
- 刪除實體目標

建立佇列或主題目標

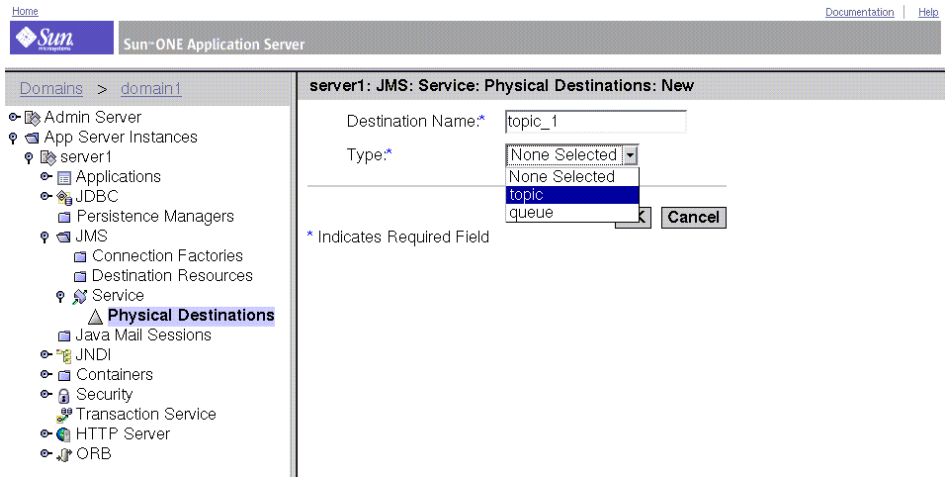
建立佇列或主題目標的步驟：

1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。
4. 開啓 [Service] 連結。
5. 選取 [Physical Destinations] 連結。

   [Physical Destinations] 螢幕會顯示在右窗格中。
6. 按一下 [New] 按鈕。

   [Physical Destinations:New] 螢幕會顯示在右窗格中。

圖 11-8 [New JMS Physical Destination] 螢幕



7. 輸入實體目標的名稱。
8. 從 [Type] 下拉式清單中選取 [queue] 或 [topic]。
9. 按一下 [OK] 按鈕。

   會重新顯示右窗格，並在現有佇列與主題目標清單中顯示新的佇列或主題目標。

列示實體目標

列示現有佇列與主題目標的步驟：

1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。
4. 開啓 [Service] 連結。
5. 選取 [Physical Destinations] 連結。

   右窗格中會顯示目前的實體目標。

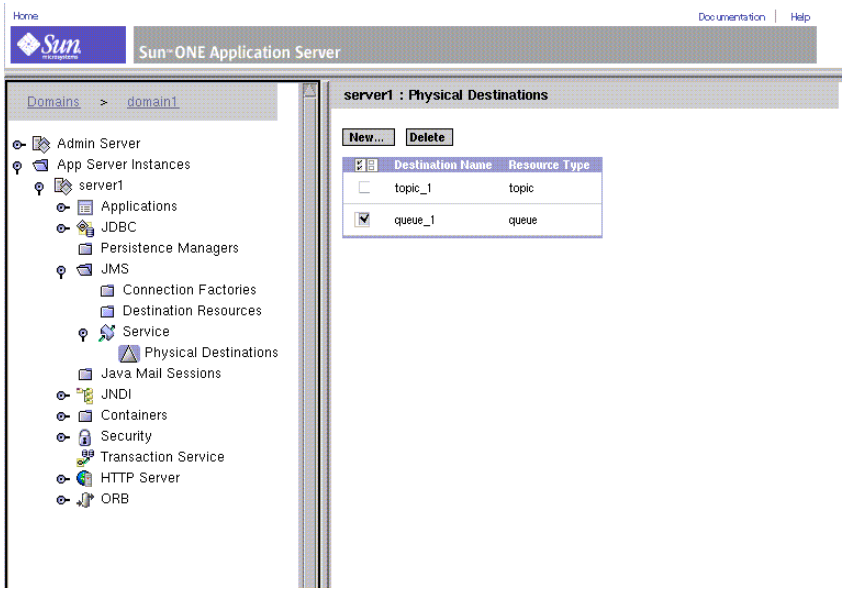
刪除實體目標

您可以依需要刪除佇列或主題目標。

刪除實體目標的步驟：

1. 列示現有的目標 (請參閱第 289 頁的「列示實體目標」)。
2. 在 [Select] 方塊內，按一下希望刪除的每個目標。

圖 11-9 [JMS Physical Destinations] 螢幕



3. 按一下 [Delete] 按鈕，移除選取的目標。

   清單會重新顯示，顯示其餘的目標。

管理受管理物件資源

JMS 用戶端使用 MQ 管理物件 (Sun ONE 應用程式服務將其作爲 *JMS* **資源**)，來存取 JMS 服務 (爲內建服務或外部服務)。

J2EE 元件使用兩種管理物件資源 (連線 Factory 與目標) 來取得 JMS 服務的連線，然後將訊息發送至訊息服務的實體目標並從其傳出 (請參閱第 281 頁的「MQ 管理物件」)。

由於建立管理物件資源不直接涉及 JMS 服務，因此無需啓用 JMS 服務，並且無需有效的使用者名稱與密碼 (請參閱表格第 285 頁的「JMS 服務特性」)，便可爲伺服器實例建立管理物件資源。

管理物件屬性

若要支援 JMS 訊息傳送，必須建立在伺服器實例中執行的所有 JMS 用戶端所需的管理物件資源。至少需要爲每個管理物件資源指定一個 JNDI 查找名稱、類型 (連線 Factory、佇列或主題)、說明 (可選)，以及是否啓用了此資源。在以下章節中會論述其他屬性。

目標 (佇列或主題)

對於佇列或主題管理物件而言，也需要指定對應實體目標的名稱。

連線 *Factory*

對於連線 Factory 管理物件而言，依預設，管理介面建立指向內建 JMS 服務的連線 Factory，即指向代理程式實例，其主機名稱爲本機主機，其連接埠號是在配置 JMS 服務時設定的號碼 (請參閱表格第 285 頁的「JMS 服務特性」)。

但是，如果您停用特定伺服器實例的 JMS 服務，則任此伺服器實例支援的所有 JMS 用戶端均必須使用外部 JMS 服務。您建立用於建立外部 JMS 服務連線的連線 Factory 時，需要設定指定適當代理程式實例的主機名稱與連接埠號的屬性。

連線 Factory 管理物件具有用來調整伺服器實例 MQ 用戶端運行時間的其他屬性。在 MQ「*Developer's Guide*」中論述了這些內容。

使用管理介面建立的連線 Factory 管理物件支援分散式作業事件管理程式。

管理物件資源管理工作

經由管理介面，您可以爲管理物件資源執行以下管理工作：

- 建立佇列或主題管理物件 (目標資源)
- 建立 ConnectionFactory 管理物件
- 列示管理物件資源
- 刪除管理物件資源

建立佇列或主題管理物件 (目標資源)

建立佇列或主題管理物件的步驟：

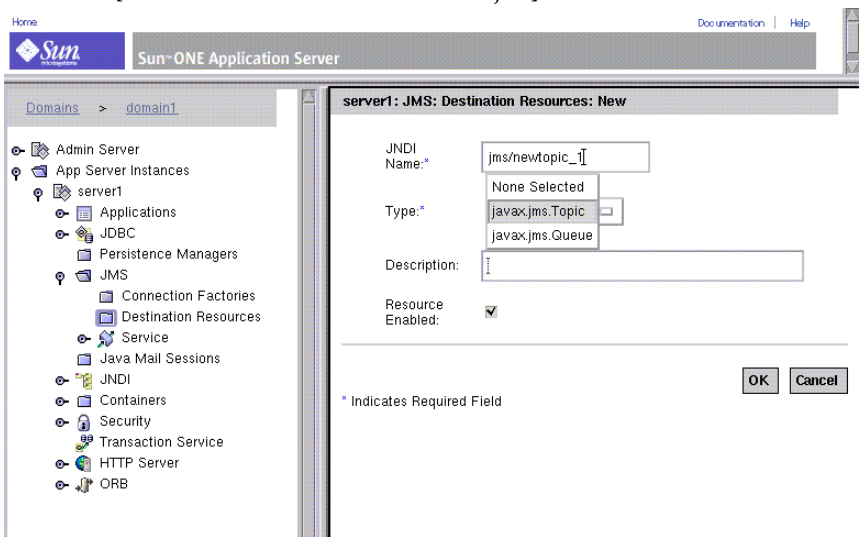
1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。
4. 選取 [Destination Resources] 連結。

   [Destination Resources] 螢幕會顯示在右窗格中。

5. 按一下 [New] 按鈕。

   會顯示 [Destination Resources:New] 螢幕。

圖 11-10 [New Destination Administered Object] 螢幕



6. 輸入與此目標管理物件關聯的 JNDI 查找名稱。
7. 從下拉式清單中選取 「queue」或 「topic」物件類型。
8. 按一下 [OK] 按鈕。

   右窗格中會重新顯示 [Destination Resource: New] 螢幕。

此外，必須透過指定 `imqDestinationName` 特性，爲此物件指定目標名稱。此特性的值應該與實體目標名稱相符。

變更此特性值的步驟：

1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。
4. 開啓 [Destination Resources] 資料夾。
5. 選取您要編輯的 [Destination Resource]。

   右窗格中會顯示 [Destination Resource] 螢幕。

6. 在右窗格中，按一下 [Properties]。

   會出現 [Edit Properties] 螢幕。

7. 在 [Name] 欄位中，輸入 imqDestinationName。
8. 在 [Value] 欄位中，輸入實體目標名稱。
9. 按一下 [OK]。

   右窗格中會重新顯示 [Destination Resources] 螢幕。

建立 *ConnectionFactory* 管理物件

建立佇列連線 Factory 或主題連線 Factory 管理物件的步驟：

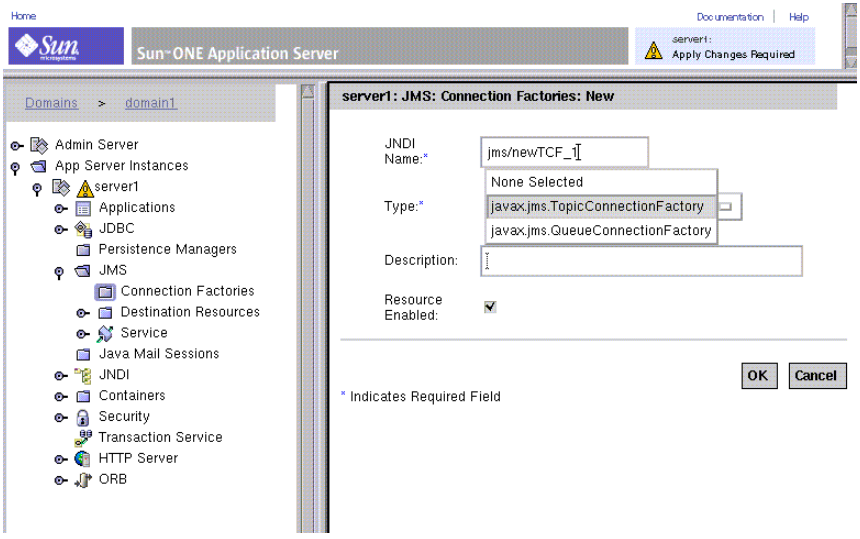
1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。
4. 選取 [Connection Factories] 連結。

   [Connection Factory Resources] 螢幕會顯示在右窗格中。

5. 按一下 [New] 按鈕。

   會顯示 [Connection Factory Resources: New] 螢幕。

圖 11-11 [New ConnectionFactory Administered Object] 螢幕



6. 輸入與此連線 Factory 管理物件關聯的 JNDI 查找名稱。
7. 從下拉式清單中選取連線 Factory 物件類型。
8. 按一下 [OK] 按鈕。

   右窗格中會重新顯示 [Connection Factory Resources] 螢幕，其中包括清單中新建立的連線 Factory 物件。

如果此連線 Factory 建立與代理程式的連線，而不是內建 JMS 服務的代理程式連線，則必須設定 `imqBrokerHostName` 與 `imqBrokerHostPort` 特性的值。

變更這些特性值的步驟：

1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。
4. 開啓 [Connection Factories] 資料夾。
5. 選取您要編輯的 [Connection Factory] 資源。

   右窗格中會顯示 [Connection Factory] 螢幕。
6. 在右窗格中，按一下 [Properties]。

   會出現 [Edit Properties] 螢幕。
7. 在 [Name] 欄位中，輸入 `imqBrokerHostName`。
8. 在 [Value] 欄位中，輸入此特性的值。
9. 在 [Name] 欄位中，輸入 `imqBrokerHostPort`。
10. 在 [Value] 欄位中，輸入此特性的值。
11. 按一下 [OK]。

    右窗格中會重新顯示 [Connection Factory] 螢幕。

列示管理物件資源

列示現有管理物件的步驟：

1. 開啓管理介面。
2. 開啓左窗格中的伺服器實例。
3. 開啓 [JMS] 資料夾。

4. 選取 [Destination Resources] 或 [Connection Factory Resources] 連結。

   右窗格中會顯示目前的目標或連線 Factory 管理物件。

刪除管理物件資源

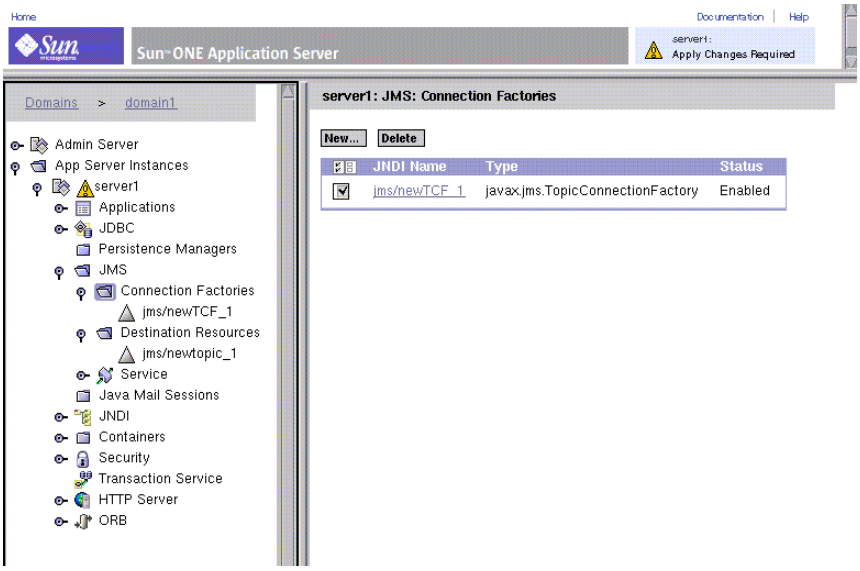
刪除管理物件資源的步驟：

1. 列示現有的管理物件資源 (請參閱第 294 頁的 「列示管理物件資源」)。

   右窗格中會顯示現有的管理物件資源。

2. 在 [Select] 方塊內，按一下您希望刪除的每個物件。

圖 11-12 [JMS Connection Factory] 螢幕，重新顯示



3. 按一下 [Delete] 按鈕，移除選取的每個物件。

   螢幕會重新顯示此清單，顯示其餘的管理物件資源。

使用指令行介面管理內建 JMS 服務

Sun ONE Application Server 具有指令行公用程式 `asadmin`，您可以使用此公用程式執行您使用管理介面可以執行的所有相同的工作。

使用以下 asadmin 指令，可配置並管理內建 JMS 服務。

管理內建 JMS 服務的 asadmin 指令

指令	使用
add-resources	加入類型爲 jdbc、jms 或 javamail 的一個或更多資源。
create-jmsdest	建立 JMS 實體目標。
create-jms-resource	建立 JMS 資源。
delete-jmsdest	刪除 JMS 實體目標。
delete-jms-resource	刪除 JMS 資源。
jms-ping	ping JMS 提供者，以查看其是否在執行。
list-jmsdest	列示伺服器實例的 JMS 實體目標。
list-jms-resources	列示伺服器實例的 JMS 資源。
get and set jms-service	取得並設定 JMS 服務的屬性。
get and set jms-resource	取得並設定 JMS 資源的屬性。

如需關於這些指令語法的資訊，請參閱 asadmin 線上輔助說明。如需關於 asadmin 的更多資訊，以及 jms-service 與 jms-resource 的屬性清單，請參閱附錄 A「使用指令行介面」。

第 12 章

配置 CORBA/IIOP 用戶端伺服器

本章解釋在 Sun ONE Application Server 環境中使用 RMI/IIOP 協定，配置 CORBA/IIOP 用戶端支援的方式。

本章包含以下主題：

- 關於 CORBA/IIOP 用戶端支援
- 配置 ORB

關於 CORBA/IIOP 用戶端支援

J2EE 平台透過其互用性需求，間接支援各種類型的用戶端、不同硬體平台以及許多軟體應用程式。作爲與 J2EE 相容的產品，Sun ONE Application Server 支援可以確保互通功能的標準協定集與格式。

CORBA (共用物件需求代理程式架構) 模型以請求分散式物件或伺服器服務的用戶端爲基礎，透過明確定義的介面，以遠端方法請求形式發送物件請求。遠端方法請求傳送關於需要執行的作業之資訊，包括服務提供者的物件名稱 (稱爲物件參考) 與實際參數 (如果有任何實際參數的話)。CORBA 動態處理許多網路程式設計工作，如物件註冊、物件位置、物件啓動、請求非多工、錯誤處理、排列與作業派送。

本章節包含以下主題：

- 關於互用性
- 關於 ORB
- 關於 RMI/IIOP 功能性
- 關於認證程序

關於互用性

實際上，互用性就是將以各種語言撰寫的應用程式整合至企業環境中的功能。在這些現有的應用程式中，其中一個或多個應用程式可以在個人電腦平台中執行，而另一些則可以在 UNIX 中執行。此外，這些企業環境也可以支援獨立的 Java 技術 (基於 J2EE 平台間接支援的應用程式)。

J2EE 授權支援 CORBA IIOP (網際網路 Orb 交換協定) 協定。CORBA 以一種對於使用者來說透明的方式，定義指定網路中分散式物件之間互用性的模型。CORBA 以一種與執行無關的方式，透過定義指定分散式物件外部可見性特性的方法，來定義模型。

關於 ORB

物件請求代理程式 (縮寫爲 ORB) 是 CORBA 的中央元件。ORB 提供所需的基礎架構來識別並尋找物件、處理連線管理、發送資料並請求通訊。

一個 CORBA 物件從不與另一個 CORBA 物件直接通訊。該物件是透過遠端存根向在本機機器中執行的 ORB 發出請求。然後，本機 ORB 將請求發送至使用網際網路 Orb 交換協定 (縮寫爲 IIOP) 的另一台機器中的 ORB。然後，遠端 ORB 找到適當的物件 (servant)、處理請求並傳回結果。可以將 IIOP 作爲 JAVA 應用程式或物件 (使用 RMI-IIOP 技術) 使用的遠端方法調用 (縮寫爲 RMI) 協定。

關於 RMI/IIOP 功能性

CORBA 可指定允許任何位置應用程式相互通訊的 ORB。這種互用性透過 IIOP 提供，並且通常在內部網路設定中可以找到。透過 IIOP，由 RMI 取得的某些功能性如下：

- 與以其他語言撰寫的物件具有的互用性。
- 傳輸作業事件與安全性環境的能力。
- 爲 ORB 服務提供的隨插即用環境。
- 與 EJB 的互用性。
- 使用 COSNaming 服務 (基於 IIOP 的命名服務)。EJB 互用性協定需要使用 COSNaming 透過 Java 命名目錄介面 (縮寫爲 JNDI) API 查找 EJB 物件。

Sun ONE Application Server 附帶的 JAVA ORB 支援以下功能性：

- CSIv2 (共用安全互用性版本 2) 的相符層級爲 0。
- 完全與 COSNaming 相容的服務執行 IDL 介面，並協助 EJB 容器發佈 EJBHome 參考。
- IIOP/GIOP Ver 1.2。CORBA 可指定允許任何位置應用程式相互通訊的 ORB。這種互用性透過 IIOP 提供。

關於認證程序

認證是確認識別的程序。在網路互動環境中，認證是一方對另一方的確信識別。證書是支援認證的一種方式。

以下兩種認證是可套用的：

伺服器認證。伺服器認證是指用戶端對伺服器的確信識別，即，對假設負責管理特定網路位址的伺服器之組織進行識別。

用戶端認證。用戶端認證是指伺服器對用戶端的確信識別，即，對假設使用用戶端軟體的使用者進行識別。

用戶端可以擁有多個證書，就象使用者可以具有數個不同部分的識別一樣。

配置 ORB

您可以爲每個 Sun ONE Application Server 實例配置多重 IIOP 偵聽程式。依預設，會配置一個 IIOP 偵聽程式。您可以爲 ORB 配置 IIOP 偵聽程式特性，並加入附加偵聽程式。

也可以啓用 ORB 監視、指定記錄訊息所用的日誌層級、指定執行緒儲存區設定、配置 IIOP 偵聽程式連接埠與 IIOP 路徑的 SSL 配置。在本小節中，將論述配置 Sun ONE Application Server 實例 ORB 支援的方式。

本章節包含以下主題：

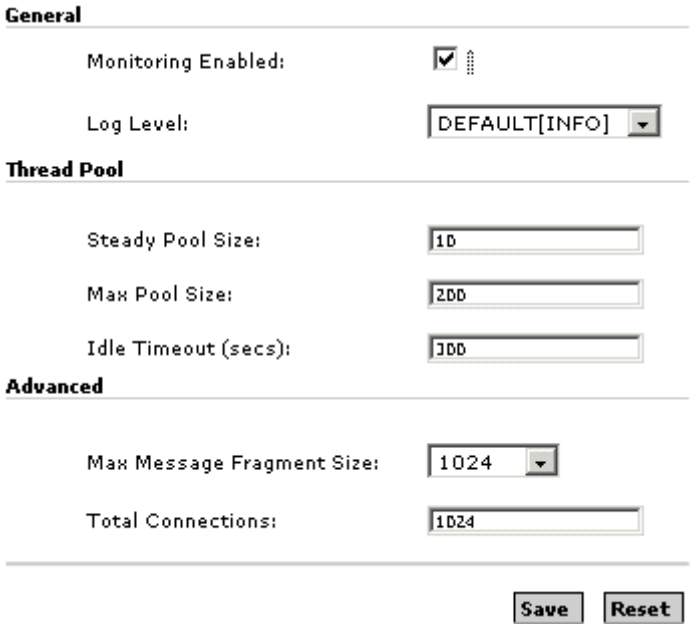
- 執行一般 ORB 配置
- 配置 ORB 的 IIOP 偵聽程式

執行一般 ORB 配置

使用管理介面，可以啓用監視、設定日誌層級、配置執行緒儲存區的儲存區設定。若要執行一般 ORB 配置，請執行以下工作：

1. 在管理介面的左窗格中，展開您要爲其配置 ORB 設定的 Sun ONE Application Server 實例。
2. 按一下 [ORB] 標籤。將在管理介面右窗格中看到圖 「一般 ORB 配置」：

圖 12-1 一般 ORB 配置



3. 在此視窗的 [General] 區段內，可以對 ORB 啓用監視，並設定日誌層級。
   a. 若要啓用 ORB 監視，請標示 [Monitoring Enabled] 核取方塊。
   b. 從 [Log Level] 下拉式清單中，選擇您要的日誌層級。此伺服器的預設日誌層級通常設定爲 INFO。ORB 的預設層級將使用此伺服器的預設值。因此，日誌層級將在下拉式清單中顯示爲 [Default (INFO)]。

提供日誌層級，記錄一系列嚴重級別的訊息 (從「FINEST」至「FATAL」)。設定日誌層級可讓您選取顯示在日誌中的訊息之細緻程度。「WARNING」的細緻程度將顯示 「WARNING」、「ALERT」、「SEVERE」與 「FATAL」訊息。通常，必須在伺服器層級設定顆粒性，但是可以使用此設定來控制 Sun ONE Application Server ORB 中顯示的訊息。

4. 在此視窗的 [Thread Pool] 區段中，可以指定 ORB 使用的請求執行緒之儲存區設定。

   請求執行緒處理使用者對應用程式元件的請求。Sun ONE Application Server 收到請求時，會將此請求指定給執行緒儲存區中的可用執行緒。執行緒會執行用戶端請求，並傳回結果。例如，如果請求需要使用目前被佔用的系統資源，則此執行緒將等待，直至此資源可用時，才允許請求使用此資源。

   可以指定最小與最大數目的執行緒，用於處理應用程式的請求。可以在這兩個值之間，動態調整執行緒儲存區。您指定的最小執行緒儲存區大小向 ORB 表明，在處理應用程式請求時至少分配的執行緒數目。此數目可以增加到您指定的最大執行緒儲存區大小。

   增加程序可用的執行緒數目，可讓程序同時回應更多的應用程式請求。

   a. 在 [Steady Pool Size] 欄位中，指定儲存區中執行緒的最小數目。同時，儲存區將縮小至執行緒閒置 (在 [Idle Timeout] (以秒表示) 欄位中指定的閒置時間) 之後的這一數目。

   b. 在 [Max Pool Size] 欄位中，指定執行緒儲存區可以增加到的最大執行緒數目。

   c. 在 [Idle Timeout] (以秒表示) 欄位中，指定要清除的執行緒儲存區中閒置執行緒之逾時。

5. 在此視窗的 [Advanced] 區段中，可以配置 ORB 的進階選項，如下所示：

   a. 在 [Max Message Fragment Size] 欄位中，指定最大的 GIOP 1.2 訊息大小，以便支援分段程序。預設分段大小爲 1024。

   b. 在 [Total Connections] 欄位中，指定 ORB 伺服器程序允許的進來遠端 IIOP 連線的最大數目。

6. 按一下 [Save]，以儲存您的設定。如果您要復原至先前設定而不儲存最近變更，請按一下 [Revert]。

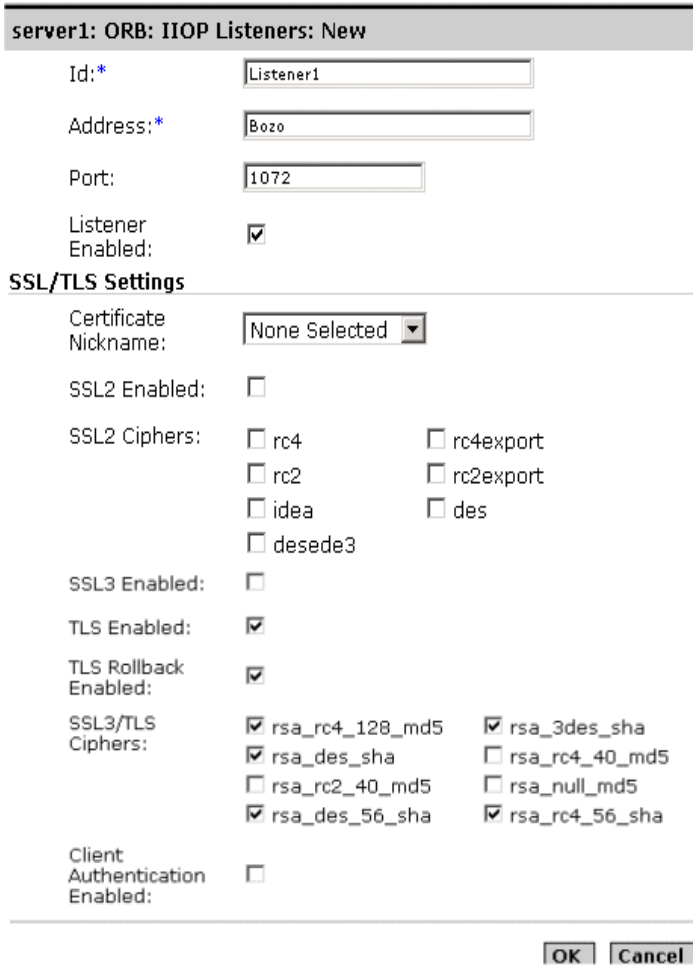
配置 ORB 的 IIOP 偵聽程式

每個新的 Sun ONE Application Server 實例均具有預設 ORB 配置，其中包括預先配置的 IIOP 偵聽程式。IIOP 偵聽程式是偵聽套接字 (偵聽指定的連接埠，並接受從基於 CORBA 的用戶端應用程式進來的連線)。您可以爲單一 Sun ONE Application Server 實例配置任意數目的 IIOP 偵聽程式。

若要建立新的 IIOP 偵聽程式或配置 IIOP 偵聽程式特性，請執行以下工作：

1. 在管理介面的左窗格中，展開您要爲其配置 ORB 特性的 Sun ONE Application Server 實例。
2. 按一下 [ORB]，並開啓其下方的 [IIOP Listener] 標籤。您將看到已爲 Sun ONE Application Server 特定實例配置的所有 IIOP 偵聽程式之清單。
3. 若要建立新的 IIOP 偵聽程式，請按一下 [New] (如果正在編輯現有的 IIOP 偵聽程式，只要開啓此偵聽程式，然後執行以下步驟中列示的工作即可)。按一下 [New] 或開啓現有的 IIOP 偵聽程式時，將看到圖 「建立新的 IIOP 偵聽程式」：

圖 12-2 建立新的 IIOP 偵聽程式



4. 您可以配置 IIOP 偵聽程式的一般參數，如下所示：

   a. 在 [Id] 文字欄位中，提供識別此偵聽程式的名稱。您可以使用任何識別碼，例如 *ORB_Listener1*、*ORB_Listener2* 等。

   b. 在 [Address] 文字欄位中，鍵入已安裝了 Sun ONE Application Server 的機器之位址。您可以用 `machinename.domainname` 格式指定此機器位址，如給定的範例所示，也可以提供此機器的 IP 位址。

c. 在 [Port] 文字欄位中，鍵入新的 IIOP 偵聽程式之唯一連接埠號。預設 IIOP 偵聽程式隨附有預設連接埠號。可以變更此連接埠號。但是，在變更此連接埠號之前，請確定您指定的新連接埠號未被任何其他現有的軟體應用程式或程序使用。

d. 若要啓用此偵聽程式，請標示 [Listener Enabled] 核取方塊。

5. 在此頁的 [SSL/TLS Settings] 區段中，可以設定 IIOP 偵聽程式的安全性。請核取與安全套接字層 (SSL) 和傳輸層安全性 (TLS) (包括所有密碼) 關聯的適當方塊。您可以選取 SSL2 或 SSL3/TLS 套接字。可以配置偵聽程式的 SSL/TLS 設定，如下所示：

a. 在 [Certificate Nickname] 欄位中，提供在 SSL 交握過程中，伺服器向用戶端展示的證書小名。必須先前已安裝了證書，才能在此清單中看到此證書小名。

b. 標示 [SSL2 Enabled] 欄位，爲偵聽程式路徑啓用 SSL2 安全性選項。

c. 選取您要使用的 SSL2 密碼，以提供 SSL2 安全性。針對所需的密碼，標示此核取方塊。必須具有令人信服的原因，才能不使用特定的密碼組，否則應該選取全部密碼。

d. 標示 [SSL3 Enabled] 欄位，爲偵聽程式路徑啓用 SSL3 安全性選項。

e. 標示 [TLS Enabled] 欄位，以啓用 TLS。必須在尋求存取伺服器的瀏覽器中啓用 TLS。在 Netscape Navigator 6.0 中核取 TLS 與 SSL3。

f. 標示 [TLS Rollback Enabled] 欄位。爲了啓用 TLS 回轉，需要首先啓用 TLS。同時要確定在您啓用此選項時，已停用 SSL3 與 SSL2。在 Microsoft Internet Explorer 5.0 與 5.5 中，使用 [TLS Rollback] 選項。

g. 選取您要用於 SSL3 與 TLS 的 SSL3/TLS 密碼。僅有已啓用 SSL3 或 TLS 後，才選取這些密碼。必須具有令人信服的原因，才能不使用特定的密碼組，否則應該選取全部密碼。

h. 標示 [Client Authentication Enabled] 核取方塊，以指出是否啓用了 ORB 偵聽程式連接埠 (執行用戶端認證的 SSL IIOP 連線)。用戶端認證是認證用戶端證書的程序，透過加密確認證書簽名與證書鏈 (直達可信任的 CA 清單中的 CA) 來實現。

6. 按一下 [OK]，以儲存 IIOP 偵聽程式設定。

注意

- 您安裝 Sun ONE Application Server 時，會爲預設的伺服器實例建立 IIOP 偵聽程式。預設 IIOP 偵聽程式連接埠的預設連接埠號爲 3700。
- 請注意，每個 IIOP 偵聽程式必須具有一個不同的連接埠號。同時請注意，您在 [Address] 文字欄位中提供的機器位址必須是安裝了 Sun ONE Application Server 的機器之位址。
- 如需關於偵聽程式路徑的 SSL 設定之更多資訊，以及 Sun ONE Application Server 安全性的其他詳細資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。

第 13 章

部署應用程式

本章描述部署各種 Sun ONE Application Server 模組與應用程式的方式。

Sun ONE Application Server 模組與應用程式包含 J2EE 標準元素和 Sun ONE Application Server 特定元素。本章僅詳細描述 Sun ONE Application Server 特定元素。

若要瞭解有關封裝、組譯模組與應用程式以進行部署的資訊，請參閱「*Sun ONE Application Server Developer's Guide*」。

本章包含以下主題：

- 關於 J2EE 模組
- 關於 J2EE 應用程式
- J2EE 標準描述元
- Sun ONE Application Server 描述元
- 命名標準
- 部署目錄結構
- 運行時間環境
- 關於類別載入器
- 部署模組與應用程式
- 應用程式部署描述元檔案

關於 J2EE 模組

J2EE 模組是一個或多個 J2EE 元件的集合，這些元件屬於同一種容器類型，並具有該類型的部署描述元。其中一個描述元是 J2EE 標準描述元，另一個則是 Sun ONE Application Server 特定描述元。J2EE 模組的類型有以下幾種：

- **Web 應用程式歸檔檔案 (WAR)：**Web 應用程式是可以附帶並部署於數個 J2EE 應用程式伺服器的 Servlet、HTML 頁、類別以及其他資源的集合。WAR 檔案可能包含下列項目：Servlet、JSP、JSP 標記庫、公用程式類別、靜態頁面、用戶端 applet、Bean、Bean 類別以及部署描述元 (`web.xml` 與可選的 `sun-web.xml`)。
- **EJB JAR 檔案：**EJB JAR 檔案是組譯企業 Bean 的標準格式。該檔案包含 Bean 類別 (本地、遠端、本機以及執行)、所有的公用程式類別以及部署描述元 (`ejb-jar.xml` 與可選的 `sun-ejb-jar.xml`)。如果 EJB 是一個具有容器管理持續性的實體 Bean，則也可能包括 CMP 部署描述元 `sun-cmp-mapping.xml`。
- **應用程式 (RMI/IIOP) 用戶端 JAR 檔案：**RMI/IIOP 用戶端是 Sun ONE Application Server 特定 J2EE 用戶端類型。RMI/IIOP 用戶端支援標準的 J2EE 應用程式用戶端規格，此外，還支援直接存取 Sun ONE Application Server。其部署描述元爲 `application-client.xml` 與可選的 `sun-application-client.xml`。
- **資源 RAR 檔案：**RAR 檔案適用於 J2EE CA 連接器。連接器模組類似於裝置驅動程式。其爲一種可攜式方法，可讓 EJB 存取外部企業系統。每個 Sun ONE Application Server 連接器都具有一個 J2EE XML 檔案 (`ra.xml`)。連接器還必須具有一個 Sun ONE Application Server 部署描述元 (`sun-ra.xml`)。

必須以全部模組的來源碼使用套裝軟體定義，這樣，類別載入器便可以在部署模組之後正確地尋找類別。

由於部署描述元中的資訊具有宣告性，因此可以變更該資訊，而不需要修改來源碼。在執行期間，J2EE 伺服器會讀取此資訊，並依此資訊進行作業。

EJB JAR 與 Web 模組也可以被組譯爲獨立的 `.jar` 或 `.war` 檔案，並被單獨部署，獨立於任何應用程式，如下圖所示。

關於 J2EE 應用程式

J2EE 應用程式是透過應用程式部署描述元連接在一起的一個或多個 J2EE 模組的邏輯集合。元件可以在模組層級或應用程式層級上進行組譯。元件還可以在模組層級或應用程式層級上進行部署。

元件首先組譯到模組中，然後組譯到 Sun ONE Application Server 應用程式 `.ear` 檔案中，以備部署。

每個模組具有一個 Sun ONE Application Server 部署描述元和一個 J2EE 部署描述元。Sun ONE Application Server 管理介面使用這些部署描述元來部署應用程式元件，並透過 Sun ONE Application Server 註冊資源。

每個應用程式包含一個或多個模組、一個選擇性 Sun ONE Application Server 部署描述元以及一個必備的 J2EE 應用程式部署描述元。所有項目以 Java 歸檔 (`.jar`) 檔案格式組譯到一個副檔名爲 `.ear` 的檔案中。

J2EE 標準描述元

J2EE 平台提供組譯與部署工具。這些工具將 JAR 檔案作爲標準的元件、應用程式套裝軟體使用，並將基於 XML 的部署描述元作爲自訂參數使用。如需有關 J2EE 組譯與部署程序的更多資訊，請參閱**透過 *J2EE* 開發企業應用程式**，v 1.0，第 7 章。

J2EE 規格，v1.3 中對 J2EE 標準部署描述元進行描述。

若要在部署之前檢查這些部署描述元的正確性，請參閱「*Sun ONE Application Server Developer's Guide*」中有關部署描述元檢驗器的資訊。

下表「J2EE 標準描述元」顯示尋找關於 J2EE 標準部署描述元更多資訊的位置。左欄列示部署描述元，右欄列示尋找關於這些描述元更多資訊的位置。

表格 13-1 J2EE 標準描述元

部署描述元	尋找更多資訊的位置
`application.xml`	Java 2 平台企業版規格，v1.3，第 8 章「應用程式組譯與部署 - J2EE: 應用程式 XML DTD」。
`web.xml`	Java Servlet 規格，v2.3，第 13 章「部署描述元」，以及 JavaServer 頁面規格，v1.2，第 7 章「作爲 XML 文件的 JSP 頁面」和第 5 章「標記延伸」
`ejb-jar.xml`	企業 JavaBeans 規格，v2.0，第 16 章「部署描述元」。
`application-client.xml`	Java 2 平台企業版規格，v1.3，第 9 章「應用程式用戶端 - J2EE: 應用程式用戶端 XML DTD」。

表格 13-1 J2EE 標準描述元

部署描述元	尋找更多資訊的位置
`ra.xml`	Java 2 企業版、J2EE 連接器架構規格，v1.0，第 10 章「封裝與部署」。

您可以在以下網站中找到規格：

`http://java.sun.com/products/`

Sun ONE Application Server 描述元

Sun ONE Application Server 使用附加的部署描述元配置 Sun ONE Application Server 特定功能。除了連接器模組所需的 `sun-ra.xml` 檔案之外，這些功能均爲選擇性的。

若要在部署之前檢查這些部署描述元的正確性，請參閱「*Sun ONE Application Server Developer's Guide*」中關於部署描述元檢驗器的資訊。

下表「Sun ONE Application Server 描述元」顯示尋找關於 Sun ONE Application Server 部署描述元更多資訊的位置。左欄列示部署描述元，右欄列示尋找關於這些描述元更多資訊的位置。

表格 13-2 Sun ONE Application Server 描述元

部署描述元	尋找更多資訊的位置
`sun-application.xml`	第 325 頁的「應用程式部署描述元檔案」
`sun-web.xml`	*Sun ONE Application Server Developer's Guide to Web Applications*
`sun-ejb-jar.xml` 與 `sun-cmp-mapping.xml`	*Sun ONE Application Server Developer's Guide to Enterprise Java Beans*
`sun-application-client.xml` 與 `sun-acc.xml`	*Sun ONE Application Server Developer's Guide to Clients*
`sun-ra.xml`	*Sun ONE J2EE CA Service Provider Implementation Administrator's Guide*

注意 Sun ONE Application Server 部署描述元在 UNIX 系統上必須具有 600 級存取權限。

所有 Sun ONE Application Server 部署描述元的 DTD 綱目檔均位於目錄 *install_dir*`/appserv/lib/dtds` 中。

命名標準

應用程式與個別部署的 EJB JAR、WAR 以及連接器 RAR 模組 (由 `server.xml` 檔案中的 `name` 屬性指定) 的名稱在 Sun ONE Application Server 中必須是唯一的。如果您沒有明確地指定名稱，則會將預設名稱作爲檔案名稱的第一部份 (沒有副檔名 `.war` 或 `.jar`)。如需關於 `server.xml` 的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

在同一個應用程式內，不同類型的模組可以具有相同的名稱，因爲部署應用程式時，包含各類型模組的目錄使用 `_jar`、`_war` 或 `_rar` 字尾命名。一個應用程式內，類型相同的模組必須具有唯一的名稱。此外，在一個應用程式內，資料庫綱目檔的名稱必須是唯一的。

建議將類似於 Java 套裝軟體的命名機制用於在 `ejb-jar.xml` 檔案的 `<module-name>` 部分中找到的模組檔名、EAR 檔名、模組名稱，以及在 `ejb-jar.xml` 檔案的 `<ejb-name>` 部分找到的 EJB 名稱。使用這種類似於套裝軟體的命名機制可以確保不會發生名稱衝突。該命名慣例的優勢不僅適用於 Sun ONE Application Server，也適用於其他 J2EE 應用程式伺服器。

EJB 的 JNDI 查找名稱也必須是唯一的。在這裡，建立連續的命名慣例也會非常有用。例如，將應用程式名稱與模組名稱附加到 EJB 是保證名稱唯一性的一種方式。在這種情況下，`mycompany.pkging.pkgingEJB.MyEJB` 就是模組 `pkgingEJB.jar` 內 EJB 的 JNDI 名稱，該模組封裝於應用程式 `pkging.ear` 中。

請確定您的套裝軟體和檔案的名稱不含有空格或作業系統不支援的非法字元。

部署目錄結構

部署應用程式時，包含各類型模組的目錄會以字尾 _jar、_war 和 _rar 命名。如果您使用 asadmin deploydir 指令部署目錄，而非部署 EAR 檔案，則您的目錄結構必須遵循此慣例。

模組與應用程式目錄結構遵循 J2EE 規格中概述的結構。

以下是一個包含 Web 模組、EJB 模組以及用戶端模組的簡單應用程式的目錄結構範例。

```
+ converter_1/
|--- converterClient.jar
|--+ META-INF/
|  |--- MANIFEST.MF
|  |--- application.xml
|  '--- sun-application.xml
|--+ war-ic_war/
|  |--- index.jsp
|  |--+ META-INF/
|  |  |--- MANIFEST.MF
|  '--+ WEB-INF/
|     |--- web.xml
|     '--- sun-web.xml
|--+ ejb-jar-ic_jar/
|  |--- Converter.class
|  |--- ConverterBean.class
|  |--- ConverterHome.class
|  '--+ META-INF/
|     |--- MANIFEST.MF
|     |--- ejb-jar.xml
|     '--- sun-ejb-jar.xml
'--+ app-client-ic_jar/
   |--- ConverterClient.class
   '--+ META-INF/
      |--- MANIFEST.MF
      |--- application-client.xml
      '--- sun-application-client.xml
```

以下是一個個別部署的連接器模組的目錄結構範例。

```
+ MyConnector/
|--- readme.html
|--- ra.jar
|--- client.jar
|--- win.dll
|--- solaris.so
'--+ META-INF/
   |--- MANIFEST.MF
   |--- ra.xml
   '--- sun-ra.xml
```

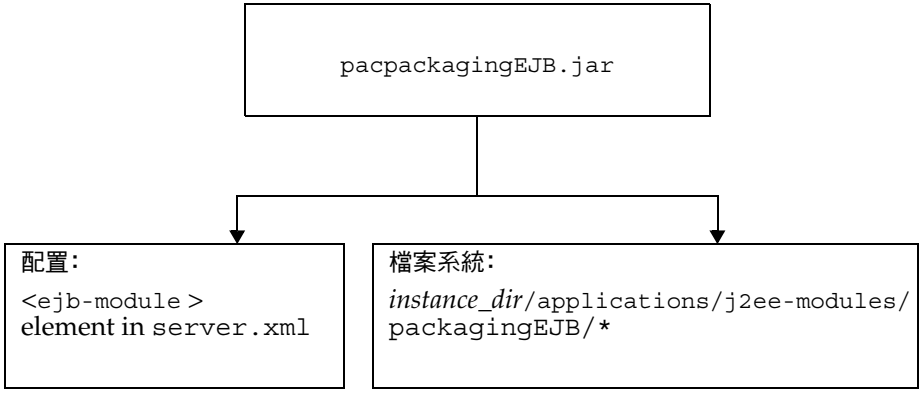
運行時間環境

無論您將元件作爲個別部署的模組，還是作爲應用程式進行部署，都會影響檔案系統與伺服器配置。請參閱圖「模組運行時間環境」與「應用程式運行時間環境」。

模組執行環境

下圖「模組運行時間環境」展示個別部署基於模組之部署的環境。

圖 13-1 模組運行時間環境



對於檔案系統項目，依如下所示擷取模組：

instance_dir/applications/j2ee-modules/*module_name*
instance_dir/generated/ejb/j2ee-modules/*module_name*
instance_dir/generated/jsp/j2ee-modules/*module_name*

generated/ejb 目錄包含存根與連接；generated/jsp 目錄包含已編譯的 JSP。

生命週期模組依如下方式擷取：

instance_dir/applications/lifecycle-modules/*module_name*

在 server.xml 中依如下所示加入配置項目：

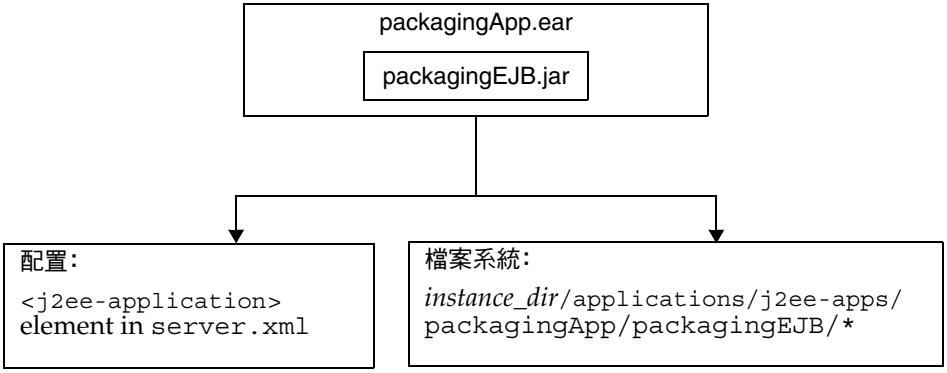
```
<server>
    <applications>
        <type-module>
            ...module configuration...
        </type-module>
    </applications>
</server>
```

在 server.xml 內，模組的 *type* 可以是 lifecycle、ejb、web 或 connector。如需關於 server.xml 的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

應用程式執行環境

下圖「應用程式運行時間環境」圖解基於應用程式的部署之環境。

圖 13-2 應用程式運行時間環境



對於檔案系統項目，依如下所示擷取應用程式：

instance_dir/applications/j2ee-apps/*app_name*
instance_dir/generated/ejb/j2ee-apps/*app_name*
instance_dir/generated/jsp/j2ee-apps/*app_name*

generated/ejb 目錄包含存根與連接；generated/jsp 目錄包含已編譯的 JSP。

在 server.xml 中依如下所示加入配置項目：

```
<server>
    <applications>
        <j2ee-application>
            ...application configuration...
        </j2ee-application>
    </applications>
</server>
```

如需關於 server.xml 的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

配置 server.xml 以使用 FastJavac 編譯器

依預設，Sun ONE Application Server 使用內建 JDK 編譯器在部署期間編譯應用程式。您也可以在部署期間使用 Sun One Studio's FastJavac 編譯器，其編譯速率更快。

在隨附式 Solaris 安裝版本中，FastJavac 編譯器的位置是不透明的。爲了使用 FastJavac 編譯器，您需要透過編譯器的路徑來配置管理伺服器的 server.xml，如下所示：

在 server.xml 中的 java-config 元素內加入以下 jvm-option：

```
<java-config java-home="/<install-dir>/jdk" server-classpath="....." >

jvm-options>-Dcom.sun.aas.deployment.java.compiler=/<install-dir>/s
tudio4/bin/fastjavac/fastjavac.sun</jvm-options>

<property name="com.sun.aas.deployment.java.compiler.options"

value="-jdk /<install-dir>/jdk" />

</java-config>
```

關於類別載入器

瞭解 Sun ONE Application Server 類別載入器可協助您確定如何爲模組與應用程式放置支援的 JAR 和資源檔，並確定放置的位置。

在 Java Virtual Machine (JVM) 中，類別載入器動態地載入解決相依性所需的特定 java 類別檔案。例如，若要建立一個 `java.util.Enumeration` 實例，某個類別載入器會將相關的類別載入到環境中。如需有關類別載入器的更詳細論述，請參閱「*Sun ONE Application Server Developer's Guide*」。

部署模組與應用程式

本節描述將 J2EE 應用程式與模組部署到 Sun ONE Application Server 中的各種方法。它涵蓋下列主題：

- 部署名稱與錯誤
- 部署生命週期
- 部署模組或應用程式
- 部署 WAR 模組
- 部署 EJB JAR 模組
- 部署生命週期模組
- 部署 RMI/IIOP 用戶端
- 部署 J2EE CA 資源介面
- 部署靜態內容
- 存取共用框架

部署名稱與錯誤

當您部署應用程式或模組時，server.xml 檔案中會產生一個唯一的名稱。請勿變更此名稱。部署期間，伺服器會偵測所有名稱衝突，並且不會載入具有非唯一名稱的應用程式或模組。發生這種情況後，訊息便會發送至伺服器日誌。如需有關命名的更多資訊，請參閱第 311 頁的「命名標準」。

如果在部署期間發生錯誤，將無法部署應用程式或模組。如果應用程式中的某個模組包含一個錯誤，將無法部署整個應用程式。

如需關於 server.xml 的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

部署生命週期

應用程式在初始部署完畢之後，可能會被修改、重新載入、重新部署、停用、重新啓用，並最終被取消部署 (從伺服器中移除)。本節涵蓋下列與部署生命週期相關的主題：

- 動態部署
- 停用已部署的應用程式或模組
- 動態重新載入

動態部署

您可以在不重新啓動伺服器的情況下部署、重新部署或取消部署應用程式或模組。這稱爲動態部署。

儘管動態部署主要針對開發人員，但是可以在不重新啓動伺服器的情況下，在作業環境中使用動態部署建立新的線上應用程式和模組。一旦重新部署完成之後，瞬時的階段作業便會無效。用戶端必須重新啓動階段作業。

停用已部署的應用程式或模組

您可以停用已部署的應用程式或模組，而不必將其從伺服器中移除。每個應用程式與模組在 server.xml 檔案中都具有一個 enabled 屬性，並在管理介面中有一個相應的選項，您可以變更該選項。如需關於 server.xml 的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

動態重新載入

如果啓用了動態重新載入，您便無須在變更應用程式或模組的程式碼時將其重新部署。您所要做的全部工作就是將已變更的類別檔案複製到應用程式或模組的部署目錄下。伺服器會定期檢查變更，然後自動地動態重新部署應用程式，並套用變更。

這在開發環境中非常有用，因爲它能快速測試程式碼變更。但是，不建議在生產環境中使用動態重新載入，因爲這樣做可能會降低效能。此外，一旦重新載入完成之後，瞬時的階段作業便會無效。用戶端必須重新啓動階段作業。

若要啓用動態重新載入，您可執行下列其中一個動作：

- 使用管理介面：
  - **a.** 開啓伺服器實例之下的應用程式元件。
  - **b.** 移往 [Applications] 頁面。
  - **c.** 核取 [Reload Enabled] 方塊以啓用動態重新載入。
  - **d.** 在 [Reload Poll Interval] 欄位中，輸入一定的秒數以設定檢查 (查看程式碼是否變更) 與動態重新載入應用程式和模組的間隔時間。
  - **e.** 按一下 [Save] 按鈕。
  - **f.** 移往伺服器實例頁面，並選取 [Apply Changes] 按鈕。
- 編輯 `server.xml` 檔案的 `applications` 元素之下列屬性：
  - `dynamic-reload-enabled="true"` 啓用動態重新載入。
  - `dynamic-reload-poll-interval-in-seconds` 設定檢查 (查看程式碼是否變更) 與動態載入應用程式與模組的間隔時間。

  如需關於 `server.xml` 的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

此外，若要載入新 Servlet 檔案，重新載入 EJB 相關的變更，或重新載入部署描述元的變更，您必須執行下列動作：

1. 在已部署應用程式的根處建立名爲 `.reload` 的空檔案：

   instance_dir`/applications/j2ee-apps/`*app_name*`/.reload`

   或在個別部署模組的根處建立空檔案：

   instance_dir`/applications/j2ee-modules/`*module_name*`/.reload`

2. 每次進行上述變更時，請明確更新 `.reload` 檔案的時間戳記 (在 UNIX 中爲 `touch .reload`)。

對於 JSP，變更會依在 `sun-web.xml` 檔案內 `jsp-config` 元素的 `reload-interval` 特性中設定的頻率自動重新載入。若要停用 JSP 動態重新載入，請設定 `reload-interval="-1"`。

部署工具

此節討論可用於部署模組與應用程式的各種工具。部署工具包括：

- asadmin 公用程式
- 管理介面
- Sun ONE Studio

asadmin 公用程式

您可以使用 `asadmin` 公用程式，在本端伺服器上部署或取消部署應用程式與個別部署的模組。不支援在多台機器上同時進行部署。此節僅簡要描述 `asadmin` 公用程式。

若要部署生命週期模組，請參閱第 322 頁的「部署生命週期模組」。

asadmin deploy

`asadmin deploy` 指令用於部署 WAR、JAR、RAR 或 EAR 檔案。若要部署應用程式，請在指令中指定 `--type application`。若要部署個別模組，請指定 `--type ejb`、`web`、`connector`。語法如下，其中顯示選擇性參數的預設值：

```
asadmin deploy --user admin_user [--password admin_password] [--host
localhost] [-port 4848] [--secure | -s] [--virtualservers
virtual_servers] [--type application|ejb|web|connector] [--contextroot
contextroot] [--force=true] [--precompilejsp=false] [--name
component_name] [--upload=true] [--retrieve local_dirpath] [--instance
instance_name] filepath
```

例如，下列指令用於部署個別 EJB 模組：

```
asadmin deploy --user jadams --password secret --host localhost
--port 4848 --type ejb --instance server1 packagingEJB.jar
```

asadmin deploydir

`asadmin deploydir` 指令用於在開放式目錄結構中部署應用程式或模組。結構必須爲第 312 頁的「部署目錄結構」中指定的結構。*dirpath* 位於 *instance_dir*`/applications/j2ee-apps` 之下，還是位於 *instance_dir*`/applications/j2ee-modules` 之下，決定了其爲應用程式還是個別部署的模組。語法如下，其中顯示選擇性參數的預設值：

```
asadmin deploydir --user admin_user [--password admin_password] [--host
localhost] [-port 4848] [--secure | -s] [--virtualservers
virtual_servers] [--type application|ejb|web|connector] [--contextroot
contextroot] [--force=true] [--precompilejsp=false] [--name
component_name] [--instance instance_name] dirpath
```

例如，下列指令用於部署個別 EJB 模組：

```
asadmin deploydir --user jadams --password secret --host localhost
--port 4848 --type ejb --instance server1 packagingEJB
```

asadmin undeploy

`asadmin undeploy` 指令用於取消部署應用程式或模組。若要取消部署應用程式，請在指令中指定 `--type app`。若要取消部署模組，請指定 `--type ejb`、`web` 或 `connector`。語法如下，其中顯示選擇性參數的預設值：

```
asadmin undeploy --user admin_user [--password admin_password] [--host
localhost] [-port 4848] [--secure | -s] [--type
application|ejb|web|connector] [--instance instance_name] component_name
```

例如，下列指令用於取消部署個別 EJB 模組：

```
asadmin undeploy --user jadams --password secret --host localhost
--port 4848 --type ejb --instance server1 packagingEJB
```

管理介面

您可以使用管理介面將模組與應用程式部署到本機或遠端 Sun ONE Application Server 網站。若要使用此工具，請採取下列步驟：

1. 開啓伺服器實例之下的應用程式元件。
2. 移往 [Enterprise Applications]、[Web Applications]、[Connector Modules] 或 [EJB Modules] 頁面。
3. 按一下 [Deploy] 按鈕。
4. 輸入模組或應用程式的完整路徑 (或按一下 [Browse] 來查找該路徑)，然後按一下 [OK] 按鈕。

5. 輸入模組或應用程式名稱。

   如果模組或應用程式已經透過核取適當的方塊而存在，則您也可以重新部署它。此項是選擇性的。

6. 透過核取虛擬伺服器名稱旁邊的方塊，可以將應用程式或模組指定給一個或多個虛擬伺服器。

7. 按一下 [OK] 按鈕。

若要部署生命週期模組，請參閱第 322 頁的 「部署生命週期模組」。

Sun ONE Studio

您可以使用 Sun ONE Studio 4 部署 J2EE 應用程式與模組。如需有關使用 Sun ONE Studio 的更多資訊，請參閱 「*Sun ONE Studio 4, Enterprise Edition Tutorial*」。

注意 在 Sun ONE Studio 中，部署模組或應用程式是指**執行**該模組或應用程式。執行也包括確定伺服器正在執行並顯示正確的 URL，以啓動模組或應用程式。

部署模組或應用程式

您可以部署應用程式或獨立於應用程式的個別模組。第 313 頁的 「運行時間環境」描述在基於應用程式或個別模組的部署中，執行期間與檔案系統的含意。

當元件需要由下列項目存取時，更應該使用基於個別模組的部署：

- 其他模組
- J2EE 應用程式
- RMI/IIOP 用戶端 (基於模組的部署允許經由 RMI/IIOP 用戶端、Servlet 或 EJB 共同存取 Bean。)

所有模組可以結合爲一個 EAR 檔案，然後作爲單一模組進行部署。這類似於獨立部署 EAR 檔案中的模組。

部署 WAR 模組

您可以使用第 319 頁的 「部署工具」中描述的任何一種方法部署 WAR 模組。

您可以透過將 `-keepgenerated` 特性加入至 `sun-web.xml` 內的 `jsp-config` 元素中，以保留產生的 JSP 來源。如果您在部署 WAR 模組時納入了此特性，並且該特性位於應用程式中，則產生的來源會保留在 *instance_dir*`/generated/jsp/j2ee-apps/`*app_name*`/`*module_name* 下；或者，如果該特性位於個別部署的 web 模組中，則產生的來源會保留在 *instance_dir*`/generated/jsp/j2ee-modules/`*module_name* 下。如需關於 `-keepgenerated` 特性的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to Web Applications*」。

部署 EJB JAR 模組

您可以使用第 319 頁的 「部署工具」中描述的任何一種方法部署 EJB JAR 模組。

您可以透過將 `-keepgenerated` 旗標加入至 `server.xml` 檔案內 `java-config` 元素的 `rmic-options` 屬性中，以保留產生的存根與連接來源。如果您在部署 EJB JAR 模組時納入了此旗標，並且該旗標位於應用程式中，則產生的來源會保留在 *instance_dir*`/generated/ejb/j2ee-apps/`*app_name*`/`*module_name* 下；或者，如果該旗標位於個別部署的 EJB JAR 模組中，則產生的來源便保留在 *instance_dir*`/generated/ejb/j2ee-modules/`*module_name* 下。如需關於 `-keepgenerated` 標幟的更多資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

部署生命週期模組

如需關於生命週期模組的一般資訊，請參閱「*Sun ONE Application Server Developer's Guide*」。

您可以使用下列工具部署生命週期模組：

- asadmin 公用程式
- 管理介面

asadmin 公用程式

若要部署生命週期模組，請使用 `asadmin create-lifecycle-module` 指令。語法如下，其中顯示選擇性參數的預設值：

```
asadmin create-lifecycle-module --user admin_user [--password
admin_password] [--host localhost] [-port 4848] [--secure | -s]
[--instance instance_name] --classname classname [--classpath classpath]
[--loadorder load_order_number] [--failurefatal=false] [--enabled=true]
[--description text_description] [--property (name=value)[:name=value]*]
modulename
```

例如：

```
asadmin create-lifecycle-module --user jadams --password secret
--host localhost --port 4848 --instance server1 --classname
RMIServer MyRMIServer
```

若要取消部署生命週期模組，請使用 `asadmin delete-lifecycle-module` 指令。語法如下，其中顯示選擇性參數的預設值：

```
asadmin delete-lifecycle-module --user admin_user [--password
admin_password] [--host localhost] [-port 4848] [--secure | -s]
[--instance instance_name] module_name
```

例如：

```
asadmin delete-lifecycle-module --user jadams --password secret
--host localhost --port 4848 --instance server1 MyRMIServer
```

若要列示部署在伺服器實例上的生命週期模組，請使用 `asadmin list-lifecycle-modules` 指令。語法如下，其中顯示選擇性參數的預設值：

```
asadmin list-lifecycle-modules --user admin_user [--password
admin_password] [--host localhost] [-port 4848] instance_name
```

例如：

```
asadmin list-lifecycle-module --user jadams --password secret --host
localhost --port 4848 server1
```

管理介面

您也可以使用管理介面部署生命週期模組。請依照下列步驟執行：

1. 開啓伺服器實例之下的應用程式元件。
2. 移往 [Life Cycle Modules] 頁面。
3. 按一下 [Deploy] 按鈕。

4. 輸入下列資訊：
   - Name (必填) - 生命週期模組的名稱。
   - Class Name (必填) - 完全合格的生命週期模組類別檔案名稱。
   - Classpath (可選) - 生命週期模組的類別路徑。指定模組的位置。預設位置在應用程式的根目錄下。
   - Load Order (可選) - 決定生命週期模組在啓動時載入的次序。具有較小整數值的模組可以更快地載入。值的範圍從 `101` 到作業系統的 `MAXINT`。保留從 `1` 到 `100` 的值。
   - Failure Fatal (可選) - 決定在生命週期模組失敗時，是否關閉伺服器。預設值爲 `false`。
   - Enable (可選) - 決定是否啓用生命週期模組。預設值爲 `true`。
5. 按一下 [OK] 按鈕。

部署 RMI/IIOP 用戶端

只有與 EJB 通訊的用戶端才有必要部署。部署 RMI/IIOP 用戶端需要三個步驟：

1. 部署 RMI/IIOP 用戶端將要存取的 EAR 或 EJB JAR。
2. 組譯必要的用戶端檔案，並部署用戶端。
3. 部署完畢之後，將會在下列位置建立用於應用程式的用戶端 JAR 檔案：

   instance_dir`/applications/j2ee-apps/`*app_name*`/`*app_name*`Client.jar`

   或在下列位置建立用於個別部署模組的用戶端 JAR 檔案：

   instance_dir`/applications/j2ee-modules/`*module_name*`/`*module_name*`Client.jar`

   用戶端 JAR 檔案包含連接以及 RMI/IIOP 用戶端必需的類別。將該檔案複製到用戶端機器上，並在用戶端上設定環境變數 APPCPATH，以指向該 JAR 檔案。

現在，您準備執行用戶端。如需更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to Clients*」。

部署 J2EE CA 資源介面

您可以使用第 319 頁的 「部署工具」中描述的任何一種方法部署連接器模組。

部署靜態內容

靜態內容 (HTML、影像等) 可以駐留在 Web 伺服器以及 Sun ONE Application Server 之上。然而，註冊 WAR 檔案之後，會在應用程式伺服器上部署靜態內容。Sun ONE Application Server 隨附的全部範例之靜態內容均駐留在應用程式伺服器上。

例如，若要存取應用程式伺服器上的靜態檔案 `index.html`，請使用：

`http://`*`server`*`:`*`port`*`/NASApp/&lt`*`context_root`*`/index.html`

存取共用框架

當 J2EE 應用程式與模組使用共用框架類別 (例如元件與程式庫) 時，可將類別置入系統類別載入器或共用類別載入器的路徑中，而不是載入應用程式或模組中。如果您將一個大型的共用程式庫組譯到每個使用該程式庫的模組中，會導致檔案過大，透過伺服器註冊要佔用太長時間。此外，不同類別載入器中可能包含同一類別的數個版本，這會造成資源浪費。

如需關於系統類別載入器的更多資訊，請參閱第 316 頁的 「關於類別載入器」。

應用程式部署描述元檔案

Sun ONE Application Server 應用程式包括兩個部署描述元檔案：

- 一個是 J2EE 標準檔案 (`application.xml`)，在 Java Servlet 規格，v2.3，第 13 章「部署描述元」中對其進行了描述。
- 一個是選擇性的 Sun ONE Application Server 特定檔案 (`sun-application.xml`)，在本節中對其進行了描述。

如需有關應用程式部署描述元檔案的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide*」。

管理 HTTP 伺服器功能與虛擬伺服器

第 14 章「配置 HTTP 功能」

第 15 章「使用虛擬伺服器」

第 16 章「管理虛擬伺服器內容」

第 14 章

配置 HTTP 功能

本章描述配置 Sun ONE Application Server 中 HTTP 相關功能個人喜好的方式。如需虛擬伺服器與 HTTP 偵聽程式的相關個人喜好，請參閱第 15 章「使用虛擬伺服器」。

本章包含以下主題：

- 關於 HTTP 功能
- 配置檔案快取記憶體
- 微調伺服器以提昇效能
- 配置 HTTP 服務品質
- 加入與使用執行緒儲存區
- 配置檔案快取記憶體
- 編輯進階設定
- 配置 MIME 類型

關於 HTTP 功能

Sun ONE Application Server HTTP 功能包括設定應用程式伺服器實例的效能層級、設定效能微調的相關參數以及使用檔案快取記憶體提昇效能。這些設定儲存在下列兩個檔案中：`init.conf` 與 `server.xml`。您可以在 [Advanced Settings] 頁面上編輯 `init.conf` 設定。如需更多資訊，請參閱第 333 頁的「編輯進階設定」。

其他需要編輯的特性儲存在 `http-service` 元素內的 `server.xml` 檔案中。如需有關 `init.conf` 檔案與 `server.xml` 檔案的更多資訊，請參閱 「*Sun ONE Application Server Administrator's Configuration File Reference*」。

配置檔案快取記憶體

Sun ONE Application Server 使用檔案快取記憶體以更快地提供靜態資訊。檔案快取記憶體包含有關檔案與靜態檔案內容的資訊。同時，它還快取用於加速處理伺服器剖析 HTML 的資訊。

依預設，檔案快取記憶體總處於開啓狀態。檔案快取記憶體的設定包含在名爲 `nsfc.conf` 的檔案中。只有在檔案快取記憶體參數已經變更，不再爲預設值之後，才可以使用該檔案。如需有關 `nsfc.conf` 的更多資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

配置檔案快取記憶體的步驟：

1. 在左窗格中，按一下 [HTTP Server]。
2. 按一下 [File Caching] 標籤。
3. 在欄位中輸入所需的值。
4. 按一下 [OK]。

如需有關使用檔案快取記憶體提昇效能的更多資訊，請參閱「 *Sun ONE Application Server Performance Tuning and Sizing Guide*」。

微調伺服器以提昇效能

在 [Performance Tuning] 頁面上，您可以配置控制 Sun ONE Application Server 效能的設定值，方法爲控制以下項目：該伺服器可以處理的請求數目、逾時前請求在沒有活動的情況下保持開啓狀態的時間以及您是否使用 DNS 執行用戶端 IP 的反向查找。同時，如果您使用 DNS，便可以將與效能相關的功能設定爲是否正在使用非同步 DNS 和 DNS 快取設定。

如需有關微調的更多資訊，請參閱「*Sun ONE Application Server Performance Tuning Guide*」。

設定效能微調設定的步驟：

1. 在左窗格中，按一下 [HTTP Server]
2. 按一下 [Tuning] 標籤
3. 在欄位中輸入所需的值。
4. 按一下 [OK]。

如需有關可經由管理介面微調的設定之附加資訊，請參閱線上輔助說明。

配置 HTTP 服務品質

服務品質是指您爲伺服器設定的效能限制。例如，一個 ISP 可能需要依據所提供的頻寬，針對虛擬伺服器收取不同數額的費用。

在您可以使用特定虛擬伺服器的服務品質之前，必須首先爲伺服器實例啓用該服務品質，並設定一些值。

爲伺服器實例配置服務品質設定的步驟：

1. 在左窗格中，按一下 [HTTP Server]。
2. 按一下 [QOS] 標籤。
3. 如要啓用整個服務品質，請按一下 [Enable]。

   依預設，服務品質處於停用狀態。啓用服務品質會稍微增加伺服器的耗用時間。
4. 選擇 [Recompute Interval]。

   重新計算間隔時間是指每次計算頻寬之間間隔的毫秒數。其預設值爲 100 毫秒。
5. 選擇 [Metric Interval]。

   公制間隔時間是指計算流量所用的間隔時間 (以秒爲單位)。其預設值爲 30 秒。在此期間計算的所有頻寬都會被平均分配，以每秒所通過的位元組表示。

   如果您的網站需要傳輸大量的大型檔案，請使用較大的值 (幾分鐘甚至更長) 或該欄位。較大的檔案傳輸可能會在較短的公制間隔時間內佔用所有可用的頻寬，如果您已經執行了最大的頻寬設定，還可能導致連線被拒絕。由於會用頻寬除以公制間隔時間來取得平均頻寬，因此，較長的間隔時間會消除由大型檔案引起的尖波。

   如果頻寬限制遠小於可用頻寬 (例如，頻寬限制爲 1 MB/秒，但是連線到主網站的頻寬爲 1 GB/秒)，則應該縮短公制間隔時間。

   請注意，如果您要傳輸大型的靜態檔案，而頻寬限制遠小於可用頻寬，則您必須決定要微調檔案的大小還是公制間隔時間，因爲必須要有相對的方案解決此問題。
6. 設定伺服器的頻寬限制 (以位元組/秒爲單位)。
7. 選擇是否執行頻寬限制設定。

   如果選擇執行頻寬限制，一旦伺服器達到其頻寬限制，附加的連線便會遭拒。

   如果不執行頻寬限制，則當伺服器的頻寬超過限制的頻寬後，伺服器便會在錯誤日誌中記錄一則訊息。

8. 選擇允許伺服器使用的最大連線數。

   該數目是指並行處理的請求數目。

9. 選擇是否執行連線限制設定。

   如果選擇執行連線限制，一旦伺服器達到其連線限制，附加的連線便會遭拒。

   如果不執行連線限制，則當伺服器的連線數超過限制的連線後，伺服器便會在錯誤日誌中記錄一則訊息。

10. 若要指定附加的名稱/值對，請按一下 [Properties] 按鈕。

11. 按一下 [OK]。

若要使用指令行介面的 `asadmin` 公用程式配置服務品質，請使用以下指令：

- `create-http-qos`
- `delete-http-qos`

這些指令使用下列語法：

`asadmin create-http-qos --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`] [--virtualserver` *virtual_server_id*`] [--bwlimit` *bandwidth_limit*`] [--enforcebwlimit` *enforce_bandwidth_limit*`] [--connlimit` *connection_limit*`] [--enforceconnlimit` *enforce_connection_limit*`]` *instancename*

`asadmin delete-http-qos --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`] [--virtualserver` *virtual_server_id*`]` *instancename*

如果您指定了一個虛擬伺服器，便可使用這些指令建立或刪除該虛擬伺服器的服務品質資訊。如果您沒有指定虛擬伺服器，則指令會影響伺服器實例。

如需有關指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

如需有關對服務品質功能限制的更多資訊，請參閱第 138 頁的「使用 CLI 管理作業事件服務」。

加入與使用執行緒儲存區

您可以使用執行緒儲存區將一定數目的執行緒配置給特定的服務，這樣，該服務使用的執行緒數目便不會超過您要其使用的數目。執行緒儲存區的另一個作用就是用於執行 thread-unsafe 外掛程式。透過將儲存區定義爲最多只能有一個執行緒，指定的服務功能便只能處理一項請求。

當您加入一個執行緒儲存區時，需要指定的資訊包括執行緒的最小與最大數目、堆疊大小以及佇列大小。

加入執行緒儲存區的步驟：

1. 在左窗格中，按一下 [HTTP Server]。
2. 按一下 [Thread Pool]。
3. 在欄位中輸入所需的值。
4. 按一下 [OK]。

執行緒儲存區便會顯示在頁面的底端。若要編輯或刪除執行緒儲存區，請按一下儲存區旁邊的 [Edit] 或 [Delete] 按鈕。

執行緒儲存區設定完畢之後，便可將其指定爲某項特定服務的執行緒儲存區來使用。

如需關於使用執行緒儲存區提昇效能的更多資訊，請參閱「*Performance Tuning and Sizing Guide*」。

編輯進階設定

Sun ONE Application Server 在啓動時會查看 *instance_dir*`/config/` 目錄中名爲 `init.conf` 的檔案，以建立影響伺服器行爲與配置的全域變數設定集。Sun ONE Application Server 執行 `init.conf` 中定義的全部指令。

[Advanced Settings] 頁面上會顯示這些設定。您可以編輯 `init.conf` 檔案中影響下列區域的特定設定：

- DNS
- SSL
- 效能
- CGI
- Keep-alive
- 記錄

如需 `init.conf` 檔案的完整描述，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

編輯進階設定的步驟：

1. 在左窗格中，按一下 [HTTP Server]。
2. 按一下 [Advanced] 標籤。
3. 按一下您要變更的設定類型 (DNS、SSL 等等)。
4. 依需要變更設定，然後按一下 [OK]。

如需關於每種設定的更多資訊，請參閱線上輔助說明。

配置 MIME 類型

[Mime Types] 頁面可讓您編輯伺服器的 MIME 檔案。MIME (多用途網際網路郵件延伸標準) 類型控制系統所支援的多媒體檔案類型。MIME 類型還指定特定伺服器檔案類型的檔案副檔名，例如，指定可以作爲 CGI 程式的檔案。

您可以依需要建立任意數目的 MIME 類型檔案，並將其與應用程式伺服器實例或虛擬伺服器相關聯。依預設，伺服器上存在一個名爲 `mime.types` 的 MIME 類型檔案，該檔案不能被刪除。

建立新 MIME 類型檔案的步驟：

1. 在左窗格中，[HTTP Server] 之下，按一下 [MIME Type File]。
2. 在右窗格中，按一下 [New]。
3. 輸入 MIME 檔案的識別碼以及檔案名稱。
4. 按一下 [OK]。

編輯 MIME 檔案中定義的步驟：

1. 在左窗格中，[HTTP Server] 之下，按一下 [MIME Type File] 旁邊的圖示以展開該檢視。
2. 按一下您要編輯的 MIME 檔案之 ID。
3. 在該頁面上，編輯與 ID 關聯的 MIME 檔案名稱。
4. 若要編輯 MIME 檔案的副檔名，請按一下 [Edit MIME file]。
5. 若要編輯現有的項目，請按一下該項目旁邊的 [Edit]。
6. 在接下來顯示的頁面上進行變更，並按一下 [Change MIME Type]。

7. 若要刪除 MIME 類型，請按一下其旁邊的 [Remove]。
8. 若要加入一個新的 MIME 類型，請在欄位中輸入種類、內容類型、檔案字尾，然後按一下 [New Type]。

若要使用指令行介面的 asadmin 公用程式配置 MIME 類型，請使用下列指令：

- `create-mime`
- `delete-mime`
- `list-mimes`

這些指令使用下列語法：

`asadmin create-mime --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`] [--instance` *instancename*`] --mimefile` *filename mime_id*

`asadmin delete-mime --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`] [--instance` *instancename*`]` *mime_id*

`asadmin list-mimes --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`]` *instancename*

如需有關指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

如需有關結合使用 MIME 類型檔案與虛擬伺服器的資訊，請參閱線上說明和第 15 章「使用虛擬伺服器」。

第 15 章

使用虛擬伺服器

本章解釋如何使用 Sun ONE Application Server 設定和管理虛擬伺服器。如需關於配置虛擬伺服器內容設定的資訊，請參閱第 16 章「管理虛擬伺服器內容」。

本章包含以下主題：

- 虛擬伺服器簡介
- 以虛擬伺服器使用 Sun ONE Application Server 功能
- 建立和配置 HTTP 偵聽程式
- 建立和配置虛擬伺服器
- 部署虛擬伺服器

虛擬伺服器簡介

使用虛擬伺服器時，您可以藉由單一安裝的伺服器爲公司或個人提供領域名稱、IP 位址以及某些伺服器監視功能。對於使用者而言，雖然您提供了硬體並維護虛擬伺服器，但看起來使用者好像使用的是自己的 Web 伺服器。

安裝非隨附式版本的 Sun ONE Application Server 時，將建立應用程式伺服器實例的預設虛擬伺服器。亦即，對於預設應用程式伺服器實例 `server1` 而言，也會同時建立名爲 `server1` 的虛擬伺服器。如果您使用的是隨附式 Solaris 9 版本，則需要建立伺服器實例。建立伺服器實例時，也會建立同名的虛擬伺服器。對於您建立的每個附加應用程式伺服器實例，仍會建立虛擬伺服器。如需關於建立和配置虛擬伺服器的更多資訊，請參閱第 347 頁的「建立和配置虛擬伺服器」。如需關於部署虛擬伺服器的更多資訊，請參閱「部署虛擬伺服器」。

該虛擬伺服器可控制 Sun ONE Application Server 的 HTTP 功能，此功能在每個虛擬伺服器基礎之上可用。您也許不想使用多重虛擬伺服器，但仍需配置與您的應用程式伺服器實例一起建立的預設虛擬伺服器，以爲該應用程式伺服器實例配置某些特性。

虛擬伺服器的設定儲存在目錄 *instance_dir*/config 下 server.xml 檔案中的 virtual-server 元素內。如需關於該檔案的更多資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

有關虛擬伺服器的某些資訊還儲存在其 obj.conf 檔案中。每個虛擬伺服器都有單獨的 obj.conf 檔案。

本章節包括下列主題：

- HTTP 偵聽程式
- 虛擬伺服器
- obj.conf 檔案
- 用於請求處理的虛擬伺服器選取
- 文件根
- 使用存取日誌檔和伺服器日誌檔

HTTP 偵聽程式

伺服器與用戶端的連線發生於 HTTP 偵聽程式 (也稱爲偵聽套接字) 上。您建立的每個 HTTP 偵聽程式都有 IP 位址、連接埠號、傳回伺服器名稱以及預設虛擬伺服器。如果您想讓 HTTP 偵聽程式在機器給定連接埠上的所有已配置 IP 位址上進行偵聽，請爲 IP 位址使用 0.0.0.0、any、ANY 或 INADDR_ANY。如需關於建立和配置 HTTP 偵聽程式的更多資訊，請參閱第 344 頁的「建立和配置 HTTP 偵聽程式」。

安裝非隨附式版本的 Sun ONE Application Server 時，將自動建立一個 HTTP 偵聽程式 http-listener-1。安裝期間，此 HTTP 偵聽程式將使用 IP 位址 0.0.0.0 及您指定的作爲 HTTP 伺服器連接埠號的連接埠號 (預設值爲 80，或者如果您未以 root 使用者身份安裝，則在 UNIX 上預設值爲 1024)。您無法刪除預設的 HTTP 偵聽程式。如果您使用多重虛擬伺服器，則您可將預設 HTTP 偵聽程式用於所有虛擬伺服器，或是建立多重 HTTP 偵聽程式。

若使用 Solaris 9 隨附的 Sun ONE Application Server，則在建立伺服器實例時將建立您的 HTTP 偵聽程式。該 HTTP 偵聽程式使用 IP 位址 0.0.0.0 以及您在建立實例時所指定的連接埠號。

由於 HTTP 偵聽程式爲 IP 位址與連接埠號的組合，所以您的多重 HTTP 偵聽程式可以使用相同的 IP 位址和不同的連接埠號，或者使用不同的 IP 位址和相同的連接埠號。例如，您可以使用 1.1.1.1:81 和 1.1.1.1:82。另外，只要將您的機器配置爲回應兩個位址的狀態，您便可使用 1.1.1.1:81 和 1.2.3.4:81。不過，如果使用 0.0.0.0 IP 位址 (在一個連接埠上的所有 IP 位址上偵聽)，您便無法設定其他 IP 位址的 HTTP 偵聽程式 (在用於特定 IP 位址的同一個連接埠上偵聽)。例如，如果讓 HTTP 偵聽程式使用 0.0.0.0:80 (連接埠 80 上的所有 IP 位址)，您便也無法建立使用 1.2.3.4:80 的 HTTP 偵聽程式。

每個 HTTP 偵聽程式還具有一個預設的虛擬伺服器，如果偵聽程式無法連線至請求中指定的虛擬伺服器，將向該預設虛擬伺服器發送請求。

另外，您可以指定 HTTP 偵聽程式中接受者執行緒 (有時稱爲接受執行緒) 的數目。接受執行緒是等待連線的執行緒。執行緒接受連線並將其放入佇列中，在佇列中將由工作者執行緒接受這些連線。理想的做法爲，您想擁有足夠多的接受執行緒，以便在發生新的請求時總有一個可用，但又需要數目相當少，以免給系統造成太重負擔。預設值爲 1。最佳規則是讓系統上的每個 CPU 有一個接受執行緒。如果您發現效能受到損害，可以調整此值。

您還可以指定是否爲 HTTP 偵聽程式啓用安全性，以及您將使用哪種安全性 (例如，哪種 SSL 及哪些密碼)。

虛擬伺服器

若要建立虛擬伺服器，必須首先決定您想要哪種虛擬伺服器。您可以使用基於 IP 位址的虛擬伺服器，或基於 URL 主機的虛擬伺服器。若要建立虛擬伺服器，您需指定的僅僅爲虛擬伺服器 ID、一個或多個 HTTP 偵聽程式、一台或多台 URL 主機。

本章節包括下列主題：

- 虛擬伺服器類型
- 基於 IP 位址的虛擬伺服器
- 基於 URL 主機的虛擬伺服器
- 預設虛擬伺服器

虛擬伺服器類型

所有虛擬伺服器都有指定的 URL 主機。不過，虛擬伺服器也可能與基於 HTTP 偵聽程式的 IP 位址相關聯。如果虛擬伺服器的 HTTP 偵聽程式在特定 IP 位址上偵聽，此虛擬伺服器將稱爲基於 IP 位址的虛擬伺服器。

如果數個虛擬伺服器在同一個 IP 位址上偵聽，將由 URL 主機識別它們，因而其成爲基於 URL 主機的虛擬伺服器。

發生新的請求時，伺服器會根據 IP 位址或主機標頭中的值，決定將請求發送至哪個虛擬伺服器。它會首先評估 IP 位址。如需更多資訊，請參閱第 342 頁的「用於請求處理的虛擬伺服器選取」。

基於 IP 位址的虛擬伺服器

若要讓單個電腦具有多重 IP 位址，您必須透過作業系統對映這些位址，或者提供其他卡。若要透過作業系統設定多重 IP 位址，請使用網路控制台 (Windows) 或 `ifconfig` 公用程式 (UNIX)。請注意，對於不同的平台，使用 `ifconfig` 的指示也不同。請參考作業系統說明文件以取得更多資訊。

透過建立一個於特定 IP 位址上偵聽的 HTTP 偵聽程式，便可建立基於 IP 位址的虛擬伺服器。然後便可關聯作爲 HTTP 偵聽程式預設虛擬伺服器的虛擬伺服器。如需關於虛擬伺服器部署方法的更多資訊，請參閱第 353 頁的「部署虛擬伺服器」。

基於 URL 主機的虛擬伺服器

透過提供唯一的 URL 主機，便可設定基於 URL 主機的虛擬伺服器。主機請求標頭的內容用於將伺服器導向至正確的虛擬伺服器。

例如，如果您想爲客戶 (*aaa*、*bbb* 和 *ccc*) 設定虛擬伺服器，以使每位客戶擁有個別的網域名稱，則您首先要配置 DNS 以確認每位客戶的 URL (`www.aaa.com`、`www.bbb.com`、`www.ccc.com`) 都可解析爲您要使用的 HTTP 偵聽程式之 IP 位址。然後，您可以將每個虛擬伺服器的 URL 主機設定爲正確設定 (例如，`www.aaa.com`)。請注意，您可將主機對映至 `/etc/hosts` 檔案中的 IP 位址。

您可以擁有任意多個與 HTTP 偵聽程式相關聯的基於 URL 主機之虛擬伺服器。

由於基於 URL 主機的虛擬伺服器使用主機請求標頭將使用者導向正確頁面，所以並非所有的用戶端軟體都與其共同執行。不支援 HTTP 主機標頭的舊用戶端軟體不起作用。這些用戶端將收到用於 HTTP 偵聽程式的預設虛擬伺服器。

預設虛擬伺服器

使用主機請求標頭選取基於 URL 主機的虛擬伺服器。如果一般使用者的瀏覽器不傳送主機標頭，或伺服器找不到指定的主機標頭，則 HTTP 偵聽程式的預設虛擬伺服器將處理請求。

並且，對於基於 IP 位址的虛擬伺服器，如果 Sun ONE Application Server 找不到指定的 IP 位址，HTTP 偵聽程式的預設虛擬伺服器會處理請求。您可以配置預設虛擬伺服器，以從特殊文件根目錄下發送錯誤訊息或伺服器頁面。

注意 請勿混淆用於 HTTP 偵聽程式的預設虛擬伺服器與安裝伺服器時所建立的預設虛擬伺服器。預設虛擬伺服器是用於預設應用程式伺服器實例的虛擬伺服器。用於 HTTP 偵聽程式的預設虛擬伺服器是您指定的任何預設虛擬伺服器。

建立 HTTP 偵聽程式時，請指定預設虛擬伺服器。您隨時可以變更預設虛擬伺服器。

obj.conf 檔案

依預設，每個虛擬伺服器都有儲存虛擬伺服器設定的單獨 `obj.conf` 檔案。若您透過管理介面或指令行介面變更設定，這些變更將在配置檔案內 (包含虛擬伺服器的 `obj.conf` 檔案) 自動執行。所有 `obj.conf` 檔案都位於 *instance_dir*/config 目錄之下。每當本指南提及「obj.conf 檔案」時，均指所有 `obj.conf` 檔案或所述虛擬伺服器的 `obj.conf` 檔案。

缺少字首的名爲 `obj.conf` 的檔案是 Sun ONE Application Server 用來爲每個虛擬伺服器建立 `obj.conf` 檔案的範本。編輯該檔案不會影響任何現有的虛擬伺服器，但會影響隨後建立的所有虛擬伺服器。如需關於直接編輯 `obj.conf` 檔案的更多資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

依預設，每個作用中的 `obj.conf` 檔案都將命名爲 *virtual_server_name*`-obj.conf`。由於伺服器實例的預設虛擬伺服器均以該實例命名，所以，當您首次建立伺服器實例時，其 `obj.conf` 檔案將命名爲 *instance_name*`-obj.conf`。透過管理介面編輯或直接編輯這些檔案之一，會變更虛擬伺服器的配置。

用於請求處理的虛擬伺服器選取

在伺服器能夠處理請求之前，其必須經由 HTTP 偵聽程式接受請求，再將請求導向至正確的虛擬伺服器。本章節論述決定虛擬伺服器的方法。

- 如果將 HTTP 偵聽程式配置給僅一個預設虛擬伺服器，則會選取該虛擬伺服器。
- 如果爲 HTTP 偵聽程式配置了多個虛擬伺服器，則請求 `Host` 標頭將符合虛擬伺服器的 `hosts` 屬性。如果未提供 `Host` 標頭或無符合的 `hosts` 屬性，將選取 HTTP 偵聽程式的預設虛擬伺服器。

如果將虛擬伺服器配置給 SSL HTTP 偵聽程式，在伺服器啓動時，將針對憑證主旨式樣核取其 `hosts` 屬性，如果它們不相符，將產生一條警告並寫入至伺服器日誌。

決定虛擬伺服器後，Sun ONE Application Server 會執行虛擬伺服器的 `obj.conf` 檔案。如需關於伺服器如何決定在 `obj.conf` 中執行哪些指令的詳細資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

文件根

文件根 (有時稱爲主文件目錄) 是一個中央目錄，包含要用於遠端用戶端的所有虛擬伺服器檔案。

文件根目錄可提供限制存取虛擬伺服器檔案的簡單方法。它也會使得很容易移動文件至新目錄 (也許在不同磁碟上)，而不會變更任何 URL，這是因爲 URL 中指定的路徑與主文件目錄相應。

例如，如果文件目錄爲 *install_dir*`/docs`，則諸如 `http://www.sun.com/products/info.html` 的請求會讓伺服器在 *install_dir*`/docs/info.html` 中尋找檔案。如果您變更文件根 (亦即移動所有檔案與子目錄)，則您只需變更虛擬伺服器使用的文件根，而不要將所有 URL 對映至新目錄或以某種方式通知用戶端在新目錄中查找。

預設 Sun ONE Application Server 實例 (server1) 的文件根將成爲在 server1 應用程式伺服器實例內建立的虛擬伺服器文件根。可以爲您建立的每個虛擬伺服器置換該目錄。

以虛擬伺服器使用 Sun ONE Application Server 功能

Sun ONE Application Server 有許多功能，如 SSL 和存取控制，您可以配合虛擬伺服器來使用。下列各節描述功能並提供在哪裡尋找更多資訊的資訊。

本章節包括下列主題：

- 配合虛擬伺服器使用 SSL
- 使用存取日誌檔和伺服器日誌檔
- 配合虛擬伺服器使用存取控制
- 配合虛擬伺服器使用 CGI

配合虛擬伺服器使用 SSL

如果您要在虛擬伺服器上使用 SSL，大多數情況下您可以使用基於 IP 位址的虛擬伺服器。通常採用的連接埠爲 443。在基於 URL 主機的虛擬伺服器上很難使用 SSL，這是因爲 Sun ONE Application Server 必須先讀取請求，然後才可決定將請求發送至哪個 URL 主機。伺服器讀取請求後，交換安全資訊的初始交握已經發生。

唯一的異常是當所有基於 URL 主機的虛擬伺服器具有同一個 SSL 配置，包括使用「萬用字元憑證」的同一個伺服器憑證。如需更多資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。

配合虛擬伺服器執行 SSL 的一種方法是具備兩個 HTTP 偵聽程式，一個使用 SSL 並偵聽連接埠 443，另一個不使用 SSL。使用者通常透過非 SSL HTTP 偵聽程式存取虛擬伺服器。若需要使用安全作業事件，使用者可以按一下網頁上的按鈕以啓動安全作業事件。然後，請求將通過安全的 HTTP 偵聽程式。

由於 SSL 作業事件大大慢於非 SSL 作業事件，所以此設計將 SSL 作業事件限制爲僅必要的作業事件。在其餘時間使用較快的非 SSL 連線。

如需藉由 Sun ONE Application Server 和虛擬伺服器設定和使用安全性的更多資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。如需配合虛擬伺服器執行 SSL 配置範例的圖解，請參閱第 355 頁的「範例 2 ：安全伺服器」。

使用存取日誌檔和伺服器日誌檔

存取日誌檔是記錄 HTTP 存取虛擬伺服器的檔案。建立新的虛擬伺服器時，依預設，存取日誌檔是與應用程式伺服器實例相同的日誌檔。許多情況下，您會希望每個個別的虛擬伺服器都有自己的日誌檔。若要如此設定，您可以變更每個虛擬伺服器的日誌路徑。如果要將所有虛擬伺服器存取作業記錄到同一個存取日誌檔中，您可以變更伺服器實例的記錄設定，以便將虛擬伺服器 ID 納入日誌檔中。如需關於變更應用程式伺服器實例記錄的更多資訊，請參閱第 5 章「使用記錄功能」。

伺服器日誌檔是記錄資訊訊息和錯誤的檔案。建立新的虛擬伺服器時，依預設，其日誌檔與應用程式伺服器實例日誌檔相同。您可以變更每個虛擬伺服器的日誌檔。

配合虛擬伺服器使用存取控制

藉由虛擬伺服器，您可以在每個虛擬伺服器基礎上設定存取控制。您甚至可以配置它，從而讓每個虛擬伺服器可透過 LDAP 資料庫使用使用者認證和群組認證。如需更多資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。

配合虛擬伺服器使用 CGI

您可以在虛擬伺服器上使用 CGI。您需要在每個虛擬伺服器上設定將儲存 CGI 的目錄，並設定用於 CGI 的檔案類型。如需關於 CGI 的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to Web Applications*」。

建立和配置 HTTP 偵聽程式

在伺服器能夠處理請求之前，其必須經由 HTTP 偵聽程式接受請求，然後將請求導向至正確的虛擬伺服器。建立伺服器實例時 (在安裝期間或以後)，將自動建立一個 HTTP 偵聽程式 `http-listener-1`。該 HTTP 偵聽程式使用 IP 位址 0.0.0.0 以及您指定的作爲應用程式伺服器連接埠號的連接埠號。您無法刪除預設的 HTTP 偵聽程式。

本章節包含以下主題：

- 建立 HTTP 偵聽程式
- 編輯 HTTP 偵聽程式設定
- 刪除 HTTP 偵聽程式

建立 HTTP 偵聽程式

使用管理介面建立 HTTP 偵聽程式的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 按一下 [HTTP Listeners]。
3. 按一下 [New]。
4. 填寫欄位。

   HTTP 偵聽程式必須具備唯一的連接埠號和 IP 位址組合。您可以使用 IPV4 或 IPV6 位址。如果您要爲基於 IP 位址的虛擬伺服器建立 HTTP 偵聽程式，請爲 HTTP 偵聽程式指定特定的 IP 位址。

   [Return Server Name] 欄位指定伺服器發送至用戶端的 URL 中的主機名稱。這會影響伺服器自動產生的 URL；但不會影響儲存在伺服器中目錄和檔案的 URL。如果您的伺服器使用一個別名，則該名稱應爲此別名。

   如果未首先找到其他虛擬伺服器，則預設虛擬伺服器便是將回應針對 HTTP 偵聽程式之請求的虛擬伺服器。如需更多資訊，請參閱第 342 頁的「用於請求處理的虛擬伺服器選取」。

   必須先啓用 HTTP 偵聽程式，其才能接受請求。

   您也可以啓用安全性並配置該 HTTP 偵聽程式的進階特性。若要指定 IPV6，請在 [Family] 欄位內使用值 inet6。如果該值爲 inet6，IPv4 位址在伺服器日誌中將含有字首 ::ffff:。
5. 按一下 [OK]。

請注意，建立 HTTP 偵聽程式時，您必須在預設虛擬伺服器欄位內輸入現有的虛擬伺服器。可以使用透過伺服器實例建立的虛擬伺服器，然後在建立其他虛擬伺服器後，返回並變更它 (如果願意)。

若要藉由指令行介面建立 HTTP 偵聽程式，請使用 `asadmin` 公用程式的 `create-http-listener` 指令。若要取得所有已建立 HTTP 偵聽程式的清單，請使用 `list-http-listeners` 指令。

若要建立 HTTP 偵聽程式，請使用下列語法：

```
asadmin create-http-listener --user username [--password password]
[--host hostname] [--port adminport] [--secure | -s] [--passwordfile
file_name] --address address [--instance instancename] --listenerport
listener_port --defaultvs virtual_server --servername server_name [--family
family] [--acceptorthreads acceptor_threads] [--blockingenabled
blocking_enabled] [--securityenabled security_enabled] [--enabled enabled]
listener_id
```

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 asadmin 的更多資訊，請參閱附錄 A「使用指令行介面」。

編輯 HTTP 偵聽程式設定

使用 管理介面 編輯 HTTP 偵聽程式設定的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [HTTP Listeners]。
3. 按一下您要編輯的 HTTP 偵聽程式。
4. 進行所需的變更，然後按一下 [Save]。

如需更多資訊，請參閱線上輔助說明。

您也可以使用指令行介面中的 asadmin 公用程式，來編輯 HTTP 偵聽程式。使用 get 指令可取得目前設定，使用 set 指令可將其設定爲新值。

若要取得 HTTP 偵聽程式所有屬性的值，請執行：

```
asadmin> get server_instance.http-listener.http_listener_name.*
```

例如，若要取得預設 HTTP 偵聽程式的值，請執行：

```
asadmin> get server1.http-listener.http-listener-1.*
```

若要設定任何屬性的值，請執行：

```
asadmin> set server_instance.http-listener.http_listener_name.attribute_name=value
```

例如，若要將屬性 defaultVirtualServer 設定爲用於 http-listener-1 的 server2，請執行：

```
asadmin> set
server1.http-listener.http-listener-1.defaultVirtualServer=server2
```

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 asadmin 的更多資訊，請參閱附錄 A「使用指令行介面」。

刪除 HTTP 偵聽程式

使用管理介面刪除 HTTP 偵聽程式的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 按一下 [HTTP Listeners]。
3. 按一下您要刪除的 HTTP 偵聽程式旁邊的核取方塊。
4. 按一下 [Delete]。

若要藉由指令行介面刪除 HTTP 偵聽程式，請使用 `asadmin` 公用程式的 `delete-http-listener` 指令，使用下列語法：

`asadmin delete-http-listener ---user` *username* `[--password` *password*`] [--host` *hostname*`] [--port` *adminport*`] [--secure | -s] [--passwordfile` *file_name*`] --instance` *instance httplistener_id*

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

建立和配置虛擬伺服器

設定 HTTP 偵聽程式後，便可建立和使用虛擬伺服器。

本章節包含以下主題：

- 建立虛擬伺服器
- 編輯虛擬伺服器設定
- 刪除虛擬伺服器

建立虛擬伺服器

使用管理介面建立虛擬伺服器的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 按一下 [Virtual Servers]。
3. 按一下 [New]。
4. 填寫必需欄位及選擇性欄位。
5. 按一下 [Save]。

若要藉由指令行介面建立虛擬伺服器，請使用 asadmin 公用程式的 create-virtual-server 指令，使用下列語法：

```
asadmin create-virtual-server --user username ---user username
[--password password] [--host hostname] [--port adminport] [--secure |
-s] [--passwordfile file_name] [--instance instancename] --hosts hosts
--mime mime_types_file [--httplisteners http-listeners] [--defaultwebmodule
default_web_module] [--configfile config_file] [--defaultobj default_object]
[--state state] [--acls acls] [--acceptlang accept_language] [--logfile
logfile] [--property (name=value)[:name=value]*] virtual_server_id
```

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 asadmin 的更多資訊，請參閱附錄 A「使用指令行介面」。

建立虛擬伺服器時，可以輸入下列類型的設定：

- 必需的設定
- 選擇性一般設定
- Web 應用程式設定
- CGI 設定
- HTTP 服務品質設定

必需的設定

虛擬伺服器的必需設定包括名稱 (ID) 以及 URL 主機。

您也必須指定 MIME 類型檔案。MIME 類型檔案包含檔案副檔名至檔案類型的對映。例如，透過 MIME 類型檔案，您可以指定將所有以 .cgi 結尾的檔案視爲 CGI 檔案。

您無需爲每個虛擬伺服器建立單獨的 MIME 類型檔案。相反，您可以依需要建立任意數量的 MIME 類型檔案，並將其與虛擬伺服器關聯起來。預設 MIME 類型檔案稱爲 mime1，檔名爲 mime.types。

如需關於 MIME 類型檔案的更多資訊，請參閱第 334 頁的「配置 MIME 類型」。

選擇性一般設定

除了必需欄位，您還可以設定選擇性欄位。

HTTP 偵聽程式

HTTP 偵聽程式可處理與虛擬伺服器的連線。您必須指定一個連線，以便讓遠端用戶端存取虛擬伺服器。

ACL

套用於虛擬伺服器的存取控制清單 (ACL)。如需更多資訊，請參閱 「*Sun ONE Application Server Administrator's Guide to Security*」。

接受語言標頭

當用戶端使用 HTTP 1.1 與伺服器聯絡時，其可以發送描述用戶端接受的語言之標頭資訊。您可以配置伺服器來剖析該語言資訊。

例如，如果以日文和英文儲存文件，您可以選擇剖析接受語言標頭。當使用日文作爲接受語言標頭的用戶端聯絡伺服器時，其會收到日文版頁面。當使用英文作爲接受語言標頭的用戶端聯絡伺服器時，其會收到英文版頁面。

如果您不支援多種語言，則不應該剖析接受語言標頭。

狀態

此狀態是虛擬伺服器的狀態，與應用程式伺服器實例是「開啓」還是「關閉」無關。如果該頁面上顯示的虛擬伺服器狀態爲 「開啓」，且應用程式伺服器實例也是 「開啓」，則虛擬伺服器僅能接受請求。

這也適合用於預設應用程式伺服器實例的預設虛擬伺服器。如果關閉應用程式伺服器實例，您的預設虛擬伺服器仍將設定爲 「開啓」，但不會接受連線。

有效狀態爲 「開啓」、「關閉」或 「停用」。如果將虛擬伺服器設定爲 「開啓」，其將能夠接受連線。

您無法關閉或停用用於應用程式伺服器實例的預設虛擬伺服器。

日誌檔

日誌檔 (亦稱爲伺服器日誌檔) 是記錄資訊訊息和錯誤的檔案。存取日誌檔是記錄 HTTP 存取虛擬伺服器的檔案。

文件根

文件根 (有時稱爲主文件目錄) 是一個中央目錄，包含要用於遠端用戶端的所有虛擬伺服器檔案。如需更多資訊，請參閱第 342 頁的 「文件根」。

Web 應用程式設定

Web 應用程式集合了 Servlet、JavaServer 頁面、HTML 文件，以及其他可能包含影像檔、壓縮歸檔檔案與其他資料的 Web 資源。Web 應用程式可以封裝到歸檔檔案中 (WAR 檔案)，或存在於開放式目錄結構中。

Sun ONE Application Server 7 支援 Servlet 2.3 API 規格，允許 Servlet 和 JSP 納入到 Web 應用程式中。另外，Sun ONE Application Server 7 還支援非 J2EE 應用程式元件 SHTML 和 CGI。

建立虛擬伺服器時，可以爲虛擬伺服器指定預設 Web 模組。預設 Web 模組將回應無法解析至該虛擬伺服器部署的其他 Web 模組的所有請求。如果您不指定預設 Web 模組，將使用環境根爲空的 Web 模組。如果沒有環境根爲空的 Web 模組，將建立並使用系統的預設 Web 模組。

部署 Web 應用程式時，請指定虛擬伺服器。一旦部署了 Web 應用程式，其便會出現於可用 Web 模組清單中，可選擇作爲虛擬伺服器的預設 Web 模組。若將 Web 模組指定爲虛擬伺服器的預設 Web 模組，則虛擬伺服器會自動加入至 Web 應用程式的虛擬伺服器清單。

CGI 設定

在您建立虛擬伺服器時設定的 CGI 設定可監管使用者及群組 CGI 程式執行爲，CGI 執行開始前要變更爲的目錄 (chroot) 以及 chroot 之後要變更爲的目錄。

在 UNIX 上，您還可以設定 nice，其爲用於決定 CGI 程式相對於伺服器之優先權的增量。通常，以 nice 值 0 執行伺服器並且 nice 增量處於 0 (CGI 程式以和伺服器相同的優先權執行) 與 19 (CGI 程式以大大低於伺服器的優先權執行) 之間。

HTTP 服務品質設定

服務品質指的是您爲虛擬伺服器設定的效能限制。例如，ISP 可能會依據允許虛擬伺服器使用的頻寬大小，對虛擬伺服器收取不同數額的費用。可以執行 (即僅允許指定的頻寬和最大連線數) 或不執行這些設定。如果未執行設定，當超出限制時，訊息將記錄到日誌檔中。如需更多資訊，請參閱第 138 頁的「使用 CLI 管理作業事件服務」。

除了透過管理介面變更這些設定外，您還可以使用指令行介面的 `asadmin` 公用程式。若要使用指令行介面的 `asadmin` 公用程式配置服務品質，請使用以下指令：

- `create-http-qos`
- `delete-http-qos`

這些指令使用以下語法：

```
asadmin create-http-qos --user username [--password password] [--host
hostname] [--port adminport] [--secure | -s] [--passwordfile file_name]
[--virtualserver virtual_server_id] [--bwlimit bandwidth_limit]
[--enforcebwlimit enforce_bandwidth_limit] [--connlimit connection_limit]
[--enforceconnlimit enforce_connection_limit] instance_name
```

```
asadmin delete-http-qos --user username [--password password] [--host
hostname] [--port adminport] [--secure | -s] [--passwordfile file_name]
[--virtualserver virtual_server_id] instance_name
```

如果您指定虛擬伺服器，則這些指令可爲該虛擬伺服器建立或刪除服務品質資訊。如果您不指定虛擬伺服器，則指令會影響伺服器實例。

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 asadmin 的更多資訊，請參閱附錄 A 「使用指令行介面」。

編輯虛擬伺服器設定

設定虛擬伺服器後，您便可對其進行編輯。如需關於編輯虛擬伺服器設定的資訊，請參閱下列主題：

- 使用管理介面編輯一般設定
- 使用指令行介面編輯一般設定
- 編輯 CGI 設定
- 編輯文件處理設定、文件目錄設定以及 HTTP/HTML 設定

使用管理介面編輯一般設定

虛擬伺服器的一般設定指建立虛擬伺服器後即可以設定的設定。若要變更設定，請執行以下步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器。
4. 進行需要的變更。

   可以變更的區域包括服務品質設定、加入 ACL、與內容相關的設定 (如文件根和接受語言標頭)、CGI 相關設定 (如使用者、群組、nice 及 chroot 設定) 以及預設 Web 模組。
5. 按一下 [Save]。

如需關於以上某些設定的更多資訊，請參閱第 347 頁的「建立和配置虛擬伺服器」。另請參閱線上輔助說明。

使用指令行介面編輯一般設定

您也可以使用指令行介面中的 asadmin 公用程式，來編輯這些設定。使用 get 指令可取得目前設定，使用 set 指令可將其設定爲新值。

若要經由虛擬伺服器取得所有屬性，請使用以下語法：

```
asadmin> get instance_name.virtual-server.vserver_id.*
```

例如：

```
asadmin> get server1.virtual-server.vs1.*
```

如果您想取得應用程式伺服器實例 `server1` 的所有屬性，請使用以下語法：

```
asadmin> get server1.virtual-server.server1.*
```

若要設定一個屬性，如接受語言標頭，請使用以下語法：

```
asadmin> set
server1.virtual-server.server1.virtualserver.acceptLanguage=false
```

注意 您可以使用指令行介面，爲 [General] 頁面上的所有欄位設定值。不過，您不能使用指令行介面爲其他標籤頁面上的欄位設定值，如 CGI 標籤上的頁面。

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

編輯 CGI 設定

如需關於編輯 CGI 的資訊，請參閱「*Sun ONE Application Server Developers Guide to Web Applications*」。

編輯文件處理設定、文件目錄設定以及 HTTP/HTML 設定

如需關於變更這些設定的資訊，請參閱第 16 章「管理虛擬伺服器內容」。

刪除虛擬伺服器

刪除虛擬伺服器的步驟：

1. 在管理介面左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 按一下 [Virtual Servers]。
3. 按一下您要刪除的虛擬伺服器旁邊的核取方塊。
4. 按一下 [Delete]。

   使用管理介面，無法刪除所有虛擬伺服器。

若要藉由指令行介面刪除虛擬伺服器，請使用 `asadmin` 公用程式的 `delete-virtual-server` 指令。

用法如下：

```
asadmin delete-virtual-server --user username [--password password]
[--host hostname] [--port adminport] [--secure | -s] [--passwordfile
file_name] --instance instance virtualserver_id
```

如需關於指令語法的更多資訊，請參閱指令行介面輔助說明。如需關於使用 `asadmin` 的更多資訊，請參閱附錄 A「使用指令行介面」。

部署虛擬伺服器

Sun ONE Application Server 的虛擬伺服器架構非常靈活。一個應用程式伺服器實例可以具有任意多個安全與非安全 HTTP 偵聽程式。您可以將任意多個虛擬伺服器與這些 HTTP 偵聽程式關聯起來。可以使用基於 IP 位址的與基於 URL 主機的兩種虛擬伺服器。

每個虛擬伺服器都可以 (並非必須) 具有自己的 ACL 清單、自己的 `mime.types` 檔案以及自己的 Java Web 應用程式集。

此設計提供了爲各種應用程式配置伺服器的最大靈活性。下列範例論述 Sun ONE Application Server 可用的某些可能配置。

- 範例 1 ：預設配置
- 範例 2 ：安全伺服器
- 範例 3 ：企業網路主機作業
- 範例 4 ：大量主機作業

範例 1 ：預設配置

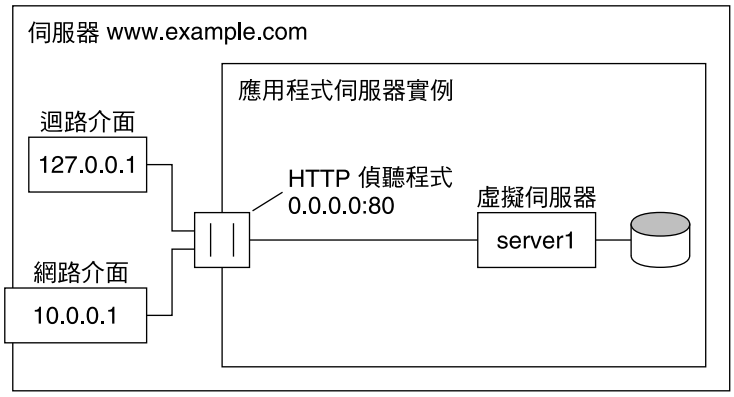
預設配置是一個應用程式伺服器實例。該應用程式伺服器實例只有一個 HTTP 偵聽程式，其在爲您電腦配置的任何 IP 位址上的連接埠 80、1024 或您選取的任何其他連接埠上偵聽。

您本機網路中的某個機制可以爲您電腦所配置的每個位址建立名稱至位址的對映。在下面的範例中，電腦有兩個網路介面：位址 127.0.0.1 上的回送介面 (即使沒有網路卡也存在該介面) 以及位址 10.0.0.1 上的乙太網路介面。

經由 DNS，名稱 example.com 將對映至 10.0.0.1。HTTP 偵聽程式配置爲在該機器配置的任何位址之連接埠 80 上 ("0.0.0.0:80") 偵聽。

由於預設配置中沒有基於 IP 位址的虛擬伺服器，所以唯一的 HTTP 偵聽程式便是預設的 HTTP 偵聽程式。所有連線均連接至虛擬伺服器 server1。

圖 15-1 預設配置



DNS

www.example.com	10.0.0.1

在此配置中，下列連線到達伺服器並由虛擬伺服器 VS1 提供服務。

- http://127.0.0.1/ (於 example.com 上啓動)
- http://localhost/ (於 example.com 上啓動)
- http://example.com/
- http://10.0.0.1/

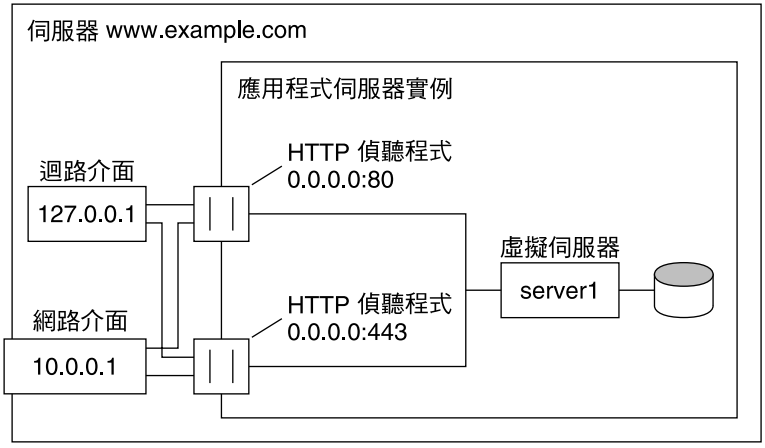
將此配置用於傳統 HTTP 伺服器。您無需加入附加虛擬伺服器或 HTTP 偵聽程式。透過變更 server1 的設定，可以配置伺服器的設定。

範例 2 ：安全伺服器

如果您要在預設配置中使用 SSL，只需將 HTTP 偵聽程式變更爲安全模式即可。

也可以加入配置爲 0.0.0.0:443 的新的安全 HTTP 偵聽程式，並將 server1 與此新的 HTTP 偵聽程式關聯起來。虛擬伺服器現在具有兩個 HTTP 偵聽程式，一個使用安全 HTTP 偵聽程式，一個不使用。此時，您的伺服器將使用和不使用 SSL 提供相同的內容，即 http://example.com/ 與 https://example.com/ deliver the same content。

圖 15-2 安全伺服器



DNS

www.example.com	10.0.0.1

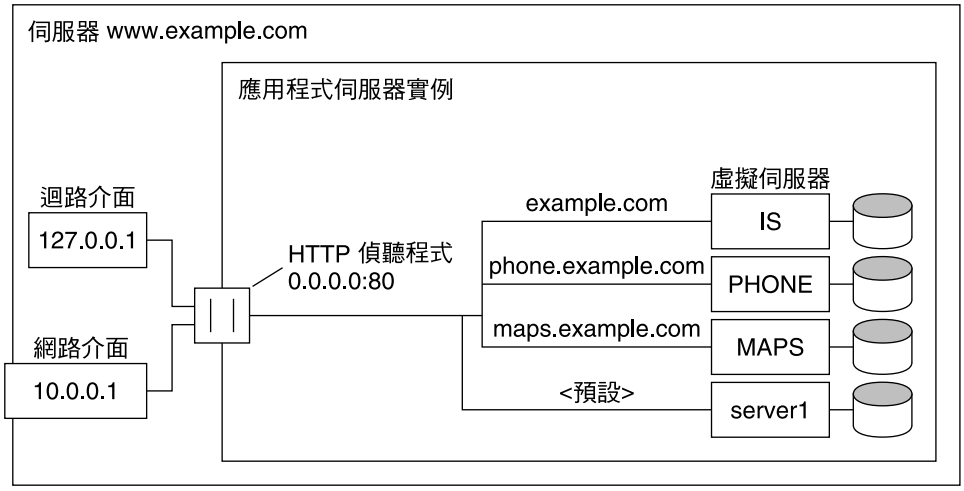
請注意，SSL 參數將附加至 HTTP 偵聽程式。

範例 3 ：企業網路主機作業

Sun ONE Application Server 的更複雜配置是該伺服器針對企業網路的部署來接待一些虛擬伺服器。例如，您有三個內部網站，員工可以從中查看其他使用者的電話號碼、查看校園地圖並向資訊服務部門追蹤其請求狀況。先前 (在此範例中)，這些網站部署在三台不同的電腦上，網站名稱 phone.example.com、maps.example.com、is.example.com 分別對映至這三台電腦。

爲使硬體和管理耗用時間降到最低，您要將這三個網站合併爲一個應用程式伺服器，部署在機器 example.com 上。您可以採用以下兩種方法進行設定：使用基於 URL 主機或基於 IP 位址的虛擬伺服器。兩種伺服器均有明顯的優勢和劣勢。

圖 15-3 使用基於 URL 主機之虛擬伺服器的企業網路主機作業



DNS

www.example.com	10.0.0.1
is.example.com	10.0.0.1
phone.example.com	10.0.0.1
maps.example.com	10.0.0.1

雖然基於 URL 主機的虛擬伺服器容易設定，但其具有以下劣勢：

- 在此配置中支援 SSL，需要使用萬用字元證書的非標準設定。如需更多資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。
- 基於 URL 主機的虛擬伺服器無法與繼承的 HTTP 用戶端協同工作。

基於 IP 位址的虛擬伺服器優勢如下：

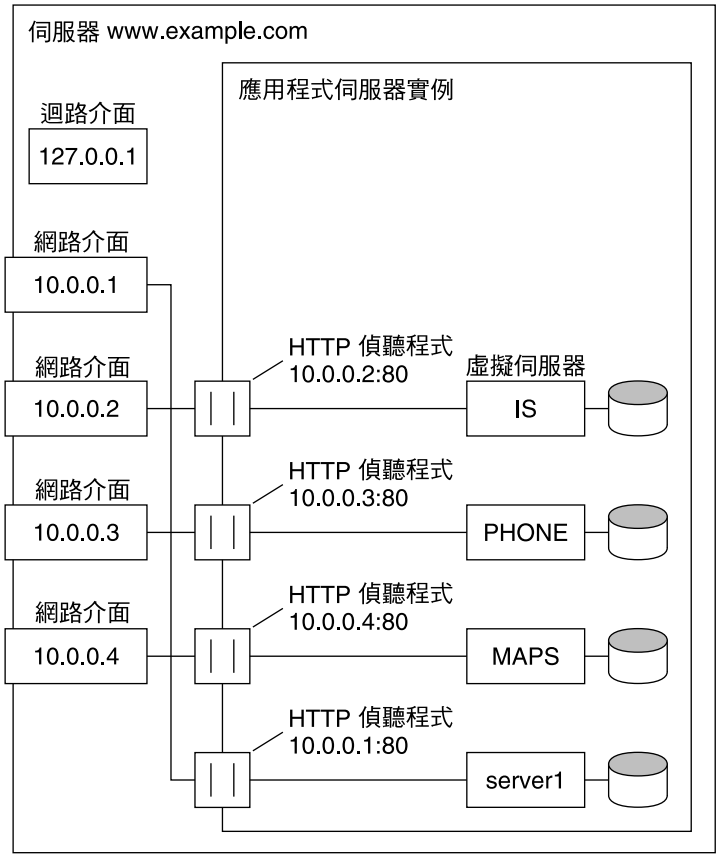
- 其可與不支援 HTTP/1.1 主機標頭的舊用戶端協同工作。
- 可直接提供 SSL 支援。

劣勢有：

- 其需要在主機電腦上變更配置 (實際或虛擬網路介面的配置)
- 其不與成千上萬的虛擬伺服器調節配置

兩種配置均需要爲三個名稱設定名稱至位址的對映。在基於 IP 位址的配置中，每個名稱都對映至不同位址。必須設定主機，才能收到所有這些位址上的連線。在基於 URL 主機的配置中，所有名稱都可以對映至機器原本具有的同一個位址。

圖 15-4 使用基於 IP 位址之虛擬伺服器的企業網路主機作業



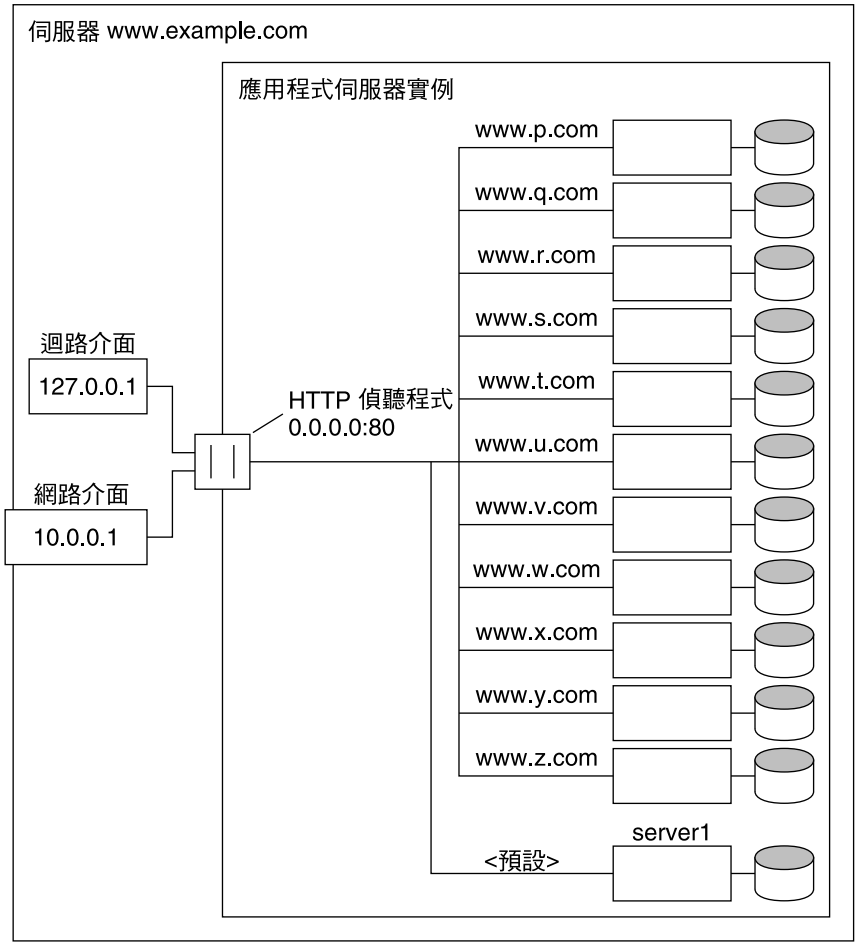
DNS

www.example.com	10.0.0.1
is.example.com	10.0.0.2
phone.example.com	10.0.0.3
maps.example.com	10.0.0.4

範例 4 ：大量主機作業

大量主機作業指的是您可以啓用許多低流量虛擬伺服器的配置。例如，管理許多低流量個人首頁的 ISP 將屬於此類。虛擬伺服器通常基於 URL 主機。

圖 15-5 大量主機作業



DNS

www.example.com	10.0.0.1
www.p.com	10.0.0.1
www.q.com	10.0.0.1
www.r.com	10.0.0.1
. . .	
www.z.com	10.0.0.1

請注意，預設虛擬伺服器 `server1` 仍然存在。

第 16 章

管理虛擬伺服器內容

本章描述配置與管理虛擬伺服器所提供檔案的方式。

本章包含以下主題：

- 變更文件根
- 設定附加文件目錄
- 啓動遠端檔案處理
- 使用 htaccess
- 限定符號式連結 (UNIX)
- 自訂使用者公用資訊目錄 (UNIX)
- 設定文件偏好設定
- 自訂錯誤回應
- 變更國際字元集
- 設定文件註腳
- 配置 URL 轉寄
- 設定伺服器剖析的 HTML
- 設定快取控制指令
- 使用更強密碼

變更文件根

文件根是您儲存所有檔案 (要讓遠端用戶端可以存取) 的中央目錄。

您加入虛擬伺服器時，可以指定具有絕對路徑的文件根。如需關於文件根及其使用方式的更多資訊，請參閱第 342 頁的「文件根」。

使用管理介面變更文件根，以使用不同路徑的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [General] 標籤。
5. 在 [Document Root] 欄位中輸入絕對目錄路徑。

   需要手動建立此目錄。
6. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

注意 通常，每個虛擬伺服器均具有其自己的文件根。

設定附加文件目錄

大部分時間，虛擬實例或伺服器實例的文件位於文件根中。有時，可能還是要提供文件根以外目錄中的文件。可以透過設定附加文件目錄來這樣做。提供文件根以外的文件目錄，可讓使用者在不需要存取主文件根的情況下，管理文件群組。

使用管理介面加入附加文件目錄的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Directories] 標籤。
5. 按一下 [Additional Doc Directories]。
6. 選擇要對映的 URL 字首。

   用戶端在需要文件時，將此 URL 發送至伺服器。

7. 請指定要將這些 URL 對映至的目錄。
8. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

您應該限定對附加文件目錄的存取，從而使用者無法寫入至這些目錄。

啟動遠端檔案處理

啓用遠端檔案操控後，用戶端可以在您的伺服器上上載檔案、刪除檔案、建立目錄、移除目錄、列示目錄內容以及重新命名檔案。虛擬伺服器配置檔案 `obj.conf` 包含您在啓動遠端檔案處理時啓動的指令。透過啓動這些指令，可讓遠端瀏覽器變更伺服器文件。您應該使用存取控制來限定對這些資源的寫入存取，以防止未授權竄改。

請注意，啓用遠端檔案操控，應該對使用內容管理系統 (例如 Microsoft Frontpage) 不產生影響。

UNIX：您必須具有正確的許可權存取檔案，否則此功能無效，即；文件根使用者必須與伺服器使用者相同。

使用管理介面以啓用遠端檔案操控的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Directories] 標籤。
5. 按一下 [Remote File Manipulation]。
6. 從資源挑選器中選擇 [Entire Server]，將變更套用至整個虛擬伺服器，或導覽至虛擬伺服器中的特定目錄。
7. 選擇啓動遠端檔案操控。
8. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

使用 htaccess

htaccess 檔案是儲存配置選項子集的動態配置檔案。您可以將 htaccess 檔案與 Sun ONE Application Server 標準存取控制結合使用 (在套用任何 htaccess 存取控制執行之前始終首先套用標準存取控制)。

如需關於使用 htaccess 的資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。

限定符號式連結 (UNIX)

您可以在伺服器中限制使用檔案系統連結。檔案系統連結是儲存在其他目錄或檔案系統中的檔案之參考。參考使得存取遠端檔案就象在目前目錄中存取此檔案一樣。有兩種類型的檔案系統連結：

- 強制連結 — 強制連結實際是指向同一資料區塊集的兩個檔案名稱；原始檔案與連結是完全相同的。出於這種原因，硬式連結不能位於不同的檔案系統中。
- 符號 (軟式) 連結 — 符號連結由兩個檔案組成，即包含資料的原始檔案，以及另一個指向原始檔案的檔案。符號式連結要比硬式連結更靈活。可以在不同的檔案系統中使用符號式連結，並且此連結可以連結至目錄。

如需關於硬式連結與符號式連結的更多資訊，請參閱 UNIX 系統說明文件。

檔案系統連結是在主文件目錄以外建立文件指標的一種簡易方法，任何人均可以建立這些連結。出於這種原因，您可能會擔心使用者建立敏感檔案 (例如，機密文件或系統密碼檔案) 的指標。

使用管理介面限定符號式連結的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Directories] 標籤。
5. 按一下 [Symbolic Links]。
6. 從資源挑選器中選擇 [Entire Server]，將變更套用至整個虛擬伺服器，或導覽至虛擬伺服器中的特定目錄。
7. 選擇是否啓用軟式連結和/或硬式連結，並選擇要開始的目錄。
8. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

自訂使用者公用資訊目錄 (UNIX)

有時，使用者要維護其自己的網頁。可以配置公用資訊目錄，這些目錄會讓伺服器上的所有使用者建立首頁與其他文件，而不會受到您的干預。

注意 雖然 Windows 系統的使用者文件目錄頁顯示在管理介面中，但是此功能不可用。

使用此系統，用戶端可以使用特定的 URL (伺服器將其識別爲公用資訊目錄) 存取您的伺服器。例如，假設您選擇字首 ~ 與目錄 `public_html`。如果請求使用 `http://www.sun.com/~jdoe/aboutjane.html`，則伺服器會識別 `~jdoe` 是指使用者公用資訊目錄。伺服器會在系統使用者資料庫中查找 `jdoe`，並尋找 Jane 的主目錄。然後，伺服器會查看 `~/jdoe/public_html/aboutjane.html`。

本章節包含下列主題：

- 配置公用資訊目錄
- 限定內容發佈
- 啓動時載入整個密碼檔案

配置公用資訊目錄

使用管理介面配置虛擬伺服器以使用公用目錄的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Handling] 標籤。
5. 按一下 [User Doc Directories]。
6. 選擇使用者 URL 字首。

   通常使用的前綴爲 ~，因爲波浪號字元是存取使用者主目錄的標準 UNIX 前綴。
7. 選擇使用者主目錄中的子目錄，伺服器從此子目錄中尋找 HTML 檔案。

   典型的目錄爲 `public_html`。

8. 指定密碼檔案。

   伺服器需要瞭解尋找檔案 (列示您系統中的使用者) 的位置。伺服器使用此檔案，來決定有效的使用者名稱，並尋找其主目錄。如果使用系統密碼檔案執行以上作業，則伺服器使用標準程式庫呼叫來查找使用者。或者，可以建立另一個使用者檔案來查找使用者。可以使用絕對路徑指定此使用者檔案。

   檔案中的每一行均應具有這種結構 (`/etc/passwd` 檔案中的非必要元素以 `*` 指示)：

   ```
   username:*:*:groupid:*:homedir:*
   ```

9. 選擇是否在啓動時載入密碼資料庫。

   如需更多資訊，請參閱第 367 頁的 「啓動時載入整個密碼檔案」。

10. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

向使用者指定單獨目錄的另一種方法是建立對映至 (所有使用者可以修改的) 中央目錄的 URL。

限定內容發佈

在某些情形下，系統管理員可能要限定能夠透過使用者文件目錄發佈內容的使用者帳號。若要限制使用者出版，請在 `/etc/passwd file` 的使用者主目錄路徑中加入尾隨斜線：

```
jdoe::1234:1234:John Doe:/home/jdoe:/bin/sh
```

成爲：

```
jdoe::1234:1234:John Doe:/home/jdoe/:/bin/sh
```

進行修改後，Sun ONE Application Server 將不提供此使用者目錄中的頁面。請求此 URI 的瀏覽器收到 「404 找不到檔案」錯誤，並且 404 錯誤將被記錄至存取日誌。

如果以後又決定允許使用者發佈內容，請從 `/etc/passwd` 項目中移除尾隨斜線，然後重新啓動 Application Server 實例。

啟動時載入整個密碼檔案

您還具有 [loading the entire password file on startup] 選項。如果選擇此選項，伺服器會在啓動時將密碼檔案載入至記憶體，便於使用者更快速查找。不過，如果密碼檔案很大，此選項會佔用太多的記憶體。

設定文件偏好設定

本章節包含下列主題：

- 輸入索引檔名
- 選取目錄索引
- 指定伺服器首頁
- 指定預設 MIME 類型

若要使用管理介面設定文件偏好設定，請執行以下步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Handling] 標籤。
5. 按一下 [Doc Preferences]。
6. 選擇適當的欄位值，如以下章節所述。
7. 按一下 [OK]。

在以下章節中將更詳細地論述您可以設定的偏好設定。如需附加資訊，請參閱線上輔助說明。

輸入索引檔名

如果在 URL 中未指定文件名稱，則伺服器會自動顯示索引檔案。預設索引檔案爲 `index.html` 與 `home.html`。如果指定了多個索引檔案，則伺服器會依顯示在此欄位中的檔案名稱順序查找，直至找到一個檔案。例如，如果索引檔案名稱爲 `index.html` 與 `home.html`，則伺服器會先查找 `index.html`，如果找不到，則會查找 `home.html`。

選取目錄索引

文件目錄有可能具有數個子目錄。例如，可能有一個目錄名爲 `products`，另一個目錄名爲 `people` 等等。讓用戶端存取這些目錄的簡介 (或索引) 通常很有幫助。

透過在目錄中搜尋名爲 `index.html` 或 `home.html` 的索引檔案 (是作爲目錄內容簡介而建立並維護的檔案)，伺服器將這些子目錄編入索引。如需更多資訊，請參閱第 367 頁的「輸入索引檔名」。您可以透過將檔案命名爲這些預設名稱中的一個名稱，將目錄中的任何檔案指定爲索引檔案，這意味著也可以使用 CGI 程式作爲索引。

如果找不到索引檔案，伺服器會產生列示文件根中所有檔案的索引檔案。

小心 如果伺服器在防火牆之外，請關閉目錄索引，以確保無法存取您的目錄結構與檔名。

指定伺服器首頁

終端使用者第一次存取伺服器時，其看到的第一個檔案通常稱爲首頁。通常，此檔案具有關於伺服器以及其他文件連結的一般資訊。

依預設，伺服器會尋找 [Document Preferences] 頁面之 [Index Filename] 欄位中指定的索引檔案，並使用此檔案作爲首頁。但是，也可以指定一個檔案作爲首頁。

指定預設 MIME 類型

文件傳送至用戶端時，伺服器會包含識別此文件類型的區段，從而用戶端可以以正確方式展示文件。但是，有時伺服器無法決定文件的正確類型，因爲沒有在伺服器中定義此文件的副檔名。在這些情況下，將會發送預設值。

預設值通常爲 text/plain，但是您應該將其設定爲儲存在伺服器中的最常用檔案類型。某些常用的 MIME 類型包括：

- `text/plain`
- `text/richtext`
- `image/jpeg`
- `application/x-tar`
- `application/x-gzip`
- `text/html`
- `image/tiff`
- `image/gif`
- `application/postscript`
- `audio/basic`

自訂錯誤回應

在用戶端遇到虛擬伺服器中的錯誤時，您可以指定將詳細訊息傳送至用戶端的自訂錯誤回應。您可以指定要發送的檔案或要執行的 CGI 程式。

例如，在用戶端取得有關特定目錄錯誤時，可以變更此虛擬伺服器的行爲方式。如果用戶端嘗試連線至受存取控制保護的伺服器部分，則可能傳回錯誤檔案，其中包括如何取得帳號的資訊。

在可以啓用自訂錯誤回應之前，必須建立要發送的 HTML 檔案或要執行的 CGI 程式以回應錯誤。執行完畢後，啓動管理介面中的回應。

使用管理介面啓用自訂錯誤回應的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Handling] 標籤。
5. 按一下 [Error Responses]。
6. 從資源挑選器中選擇 [Entire Server]，將變更套用至整個虛擬伺服器，或導覽至虛擬伺服器中的特定目錄。
7. 對於您要變更的每個錯誤碼，請指定檔案的絕對路徑或包含錯誤回應的 CGI。
8. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

變更國際字元集

文件的字元集在某種程度上由撰寫文件所使用的語言決定。您可以透過選取資源並輸入此資源的字元集，爲文件、文件集或目錄置換用戶端預設字元集設定。

瀏覽器可以使用 HTTP 中的 MIME 類型 `charset` 參數，變更其字元集。如果伺服器在其回應中包括此參數，則瀏覽器會相應變更其字元集。範例爲：

- `Content-Type:text/html;charset=iso-8859-1`
- `Content-Type:text/html;charset=iso-2022-jp`

在 RFC 1700 中指定了以下 `charset` 名稱 (以 `x-` 開頭的名稱除外)：

- us-ascii
- iso-2022-jp
- x-euc-jp
- iso-8859-1
- x-sjis
- x-mac-roman

此外，以下別名爲 us-ascii 可識別的別名：

- ansi_x3.4-1968
- ansi_x3.4-1986
- ascii
- us
- cp367
- iso-ir-6
- iso_646.irv:1991
- iso646-us
- ibm367

以下別名爲 iso_8859-1 可識別的別名：

- latin1
- iso_8859-1:1987
- ibm819
- iso_8859-1
- iso-ir-100
- cp819

使用管理介面變更字元集的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Handling] 標籤。
5. 按一下 [International Characters]。
6. 從資源挑選器中選擇 [Entire Server]，將變更套用至整個虛擬伺服器，或導覽至虛擬伺服器中的特定目錄。

7. 爲整個伺服器或部分伺服器設定字元集。

   如果保留此欄位爲空白，則字元集設定爲 NONE。

8. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

設定文件註腳

您可以在伺服器的特定區段中，爲所有文件指定文件註腳 (可以包括最後一次修改文件的時間)。除 CGI 程序檔的輸出或已剖析的 HTML (.shtml) 檔案之外，所有檔案均可使用此註腳。如果您需要文件註腳顯示在 CGI 程序檔輸出或已剖析的 HTML 檔案中，請將註腳文字輸入至單獨的檔案中，並加入一行程式碼或另一個伺服器端包含，將此檔案附加至此頁的輸出中。

若要使用管理介面設定文件註腳，請執行以下步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [Doc Handling] 標籤。
5. 按一下 [Doc Footer]。
6. 從資源挑選器中選擇 [Entire Server]，將變更套用至整個虛擬伺服器，或導覽至虛擬伺服器中的特定目錄。

   如果您選擇目錄，則文件註腳僅在伺服器收到對映此目錄的 URL 或此目錄中的任何檔案時套用。

7. 請指定您要包含此註腳的檔案之類型。
8. 指定日期格式。
9. 鍵入要在註腳中顯示的所有文字。

   文件註腳最多可達 765 個字元。如果您要包含最後一次修改文件的日期，請鍵入字串 :LASTMOD:。

如需更多資訊，請參閱線上輔助說明。

配置 URL 轉寄

URL 轉寄允許您將文件請求重新導向至另一個伺服器。轉寄 URL 或重新導向是伺服器通知使用者 URL 已變更 (例如，因爲已將檔案移至另一個目錄或伺服器) 的一種方法。如果使用者請求某個伺服器上的文件，您也可以使用重新導向，無縫地將其轉移爲請求另一個伺服器上的文件。

例如，如果您將 http://www.sun.com/info/movies 轉寄至字首 film.sun.com，則 URL http://www.sun.com/info/movies 會重新導向至 http://www.sun.com/info/movies。

有時，可能要將對某個子目錄中所有文件的請求重新導向至特定的 URL。例如，如果您必須移除目錄 (因爲造成太多的通訊或因爲任何原因不再提供這些文件)，可以將對其中任何一個文件的請求導向至解釋這些文件爲什麼不再可用的頁面中。例如，字首 /info/movies 可以重新導向至 http://www.sun.com/info/movies。

使用管理介面配置 URL 轉寄的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [HTTP/HTML] 標籤。
5. 按一下 [URL Forwarding]。
6. 鍵入您要重新導向的 URL 字首，並且是否要將其重新導向至另一個字首或靜態 URL。
7. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

設定伺服器剖析的 HTML

HTML 通常會準確地發送至用戶端，就像其存在於磁碟中一樣，不受任何伺服器干預。但是，伺服器在發送文件之前，可以在 HTML 檔案中搜尋特殊的指令 (即，伺服器可以剖析 HTML)。如果您要伺服器剖析這些檔案，並將針對請求的資訊或檔案插入至文件，必須首先啓用 HTML 剖析。

使用管理介面設定 HTML 剖析的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。

3. 按一下您要編輯的虛擬伺服器之名稱。
4. 按一下 [HTTP/HTML] 標籤。
5. 按一下 [Parse HTML]。
6. 從資源挑選器中選擇 [Entire Server]，將變更套用至整個虛擬伺服器，或導覽至虛擬伺服器中的特定目錄。

   如果您選擇目錄，則伺服器僅在伺服器收到此目錄對映的 URL 或此目錄中的任何檔案時才剖析 HTML。
7. 選擇是否啓用伺服器剖析的 HTML。

   您可以啓動 HTML 檔案，但不啓動 exec 標籤，或啓動 HTML 檔案與 exec 標籤，此標籤允許 HTML 檔案執行伺服器中的其他程式。
8. 選擇要剖析的檔案。

   可以選擇僅剖析具有 .shtml 副檔名的檔案，還是剖析所有 HTML 檔案 (這會降低效能)。如果使用 UNIX，也可以選擇剖析 UNIX 檔案 (執行許可權處於開啓狀態)，但這樣做不可靠。
9. 按一下 [OK]。

如需關於設定伺服器以接受已剖析的 HTML 之更多資訊，請參閱線上輔助說明。

如需關於使用伺服器剖析的 HTML 之更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to Web Applications*」。

設定快取控制指令

快取控制指令是 Sun ONE Application Server 使用的一種控制方法，用來控制代理伺服器快取的資訊。使用快取控制指令，您可以置換代理伺服器的預設快取法，以防止快取靈敏度高的資訊 (也許稍後會被擷取)。若要這些指令工作，代理伺服器必須遵守 HTTP 1.1。

如需關於 HTTP 1.1 的更多資訊，請參閱「超文件傳輸協定 --HTTP/1.1 規格 (RFC 2068)」，位於：

http://www.ietf.org/

使用管理介面設定快取控制指令的步驟：

1. 在左窗格中，針對應用程式伺服器實例，開啓 [HTTP Server]。
2. 開啓 [Virtual Servers]。
3. 按一下您要編輯的虛擬伺服器之名稱。

4. 按一下 [HTTP/HTML] 標籤。
5. 按一下 [Cache Control Directives]。
6. 填寫欄位。回應指令的有效值如下：
   - **Public**。此回應可透過任何快取記憶體快取。此爲預設值。
   - **Private**。此回應僅由專用的 (非共用的) 快取記憶體快取。
   - **No Cache**。在任何位置均不能快取此回應。
   - **No Store**。此快取記憶體不能將任何位置的請求或回應儲存在非揮發性儲存體中。
   - **Must Revalidate**。此快取項目必須由原始的伺服器重新驗證。
   - **Maximum Age (以秒表示)**。用戶端不接受比此時間長的回應。
7. 按一下 [OK]。

如需更多資訊，請參閱線上輔助說明。

使用更強密碼

如需關於設定更強密碼的資訊，請參閱「*Sun ONE Application Server Administrator's Guide to Security*」。

第 4 部分

附錄

附錄 A「使用指令行介面」

附錄 B「協力廠商版權備註」

使用指令行介面

本附錄提供在系統提示的單模式下 (即，每次從指令提示下運行一個指令)、多重模式下 (即，可以運行多個指令，而無需重新輸入環境層級資訊)、程序檔與程式中，使用指令行介面 (asadmin 公用程式) 的說明。您可以使用指令行介面代替管理介面螢幕。

本附錄包含下列小節：

- 關於指令行介面
- 使用 asadmin
- 安全性考量
- 並行存取考量
- 指令參考

關於指令行介面

本章節包含下列主題：

- 關於 asadmin 公用程式
- 關於 Ant 工作
- 關於其他指令行公用程式

關於 asadmin 公用程式

asadmin 公用程式執行所有的配置與管理工作。您可以使用此公用程式代替使用管理介面。

關於 Ant 工作

許多開發人員使用 Ant 來加速 J2EE 應用程式的開發程序。Ant 程序檔會協助 `asadmin` 公用程式執行某些工作。開發人員可以使用 Ant 工作建立應用程式，部署與取消部署模組與應用程式，並控制 Sun ONE Application Server。

如需關於 Ant 工作的更多資訊，請參閱「*Sun ONE Application Server Developer's Guide*」。

如需關於 Ant 的更多資訊，請參閱 Jakarta Project 站點 (位於 http://jakarta.apache.org/ant/)。

關於其他指令行公用程式

Sun ONE Application Server 包含其他指令行公用程式。下表列示這些公用程式，並對每個公用程式進行簡要說明。

表格 A-1 其他指令行公用程式

公用程式	定義
appclient	會啓動應用程式用戶端容器，並且呼叫在應用程式 Jar 檔案中封裝的用戶端應用程式。
capture-schema	取得資料庫綱目與對映資訊。
flexanlg	產生關於伺服器的統計資料。
htpasswd	建立使用者認證檔案。
package-appclient	包裝應用程式用戶端容器程式庫與 jar 檔案。如需更多資訊，請參閱「*Sun ONE Application Server Developer's Guide to Clients*」。
verifier	使用 DTD 驗證 J2EE 部署描述元。如需更多資訊，請參閱「*Sun ONE Application Server Developer's Guide* 」。
wscompile	採用服務定義介面，並且產生用戶端存根或伺服器端骨架，或爲提供的介面產生 WSDL 集。
wsdeploy	產生可部署的 WAR 檔案。

如需關於這些公用程式的更多資訊，請參閱其線上輔助說明。

使用 asadmin

`asadmin` 公用程式具有執行管理工作的指令集。您可以將這些指令用於可透過管理介面執行的大多數工作。可以在 *install_dir*/bin 中找到 `asadmin` 公用程式，並從那裡執行它。在 Windows 中，按兩下 `asadmin.bat` 檔案，然後 `asadmin` 公用程式會在指令視窗中啓動，在多重模式下執行。

請注意，某些 HTTP 伺服器相關的特性與管理伺服器特性不能使用指令行設定，而必須使用管理介面設定。您可以設定儲存在 `server.xml` 配置檔案中的所有特性，但是不能設定儲存在 `init.conf` 或 `obj.conf` 中的特性。如需關於配置檔案的更多資訊，請參閱「*Sun ONE Application Server Administrator's Configuration File Reference*」。

如需關於個別指令的更多資訊，請參閱第 395 頁的「指令參考」，以及指令的說明。

本章節包含下列主題：

- 瞭解指令語法
- 使用單模式與多重模式
- 使用互動式與非互動式選項
- 使用環境指令
- 使用密碼檔案選項
- 本機或遠端運行 asadmin
- 使用指令行呼叫
- 使用逸出字元
- 使用 get 與 set 指令
- 使用輔助說明
- 檢視輸出與錯誤

瞭解指令語法

`asadmin` 公用程式具有以下語法：

asadmin command -short-option argument --long-option argument operand

指令

指令是執行的作業或工作。指令區分大小寫。

選項

選項修改公用程式執行指令的方式。選項區分大小寫。請注意，短選項在其前面具有單一破折號 (-)，長選項在其前面具有兩個破折號 (--)。對於許多選項，您可以使用長形式或短形式的選項，例如，使用者可以爲 `--user` 或 `-u`。某些選項是必需的，某些選項是選擇性的。選擇性的選項顯示在指令語法的括號中。在執行指令或收到錯誤訊息並且指令不執行的情況下，您必須包含所有必備選項。

如需關於可用長選項與短選項名稱的清單，請參閱第 395 頁的「指令參考」中的「短選項與長選項、預設值與環境變數」表格。

大多數選項需要引數值，例如 --port *port_number*。布林選項例外，其可以開啓與關閉功能，並且不需要引數值。

也可以在環境變數中儲存選項。如需更多資訊，請參閱第 383 頁的「使用環境指令」。如需對等於選項的環境變數之完整清單，請參閱第 428 頁的「長選項格式與短選項格式、預設值與對等的環境變數」。

布林選項

布林選項開啓或關閉 (例如 `--interactive` 使您處於互動式模式下，提示您輸入選項；`--no-interactive` 會關閉互動式模式)。在長選項的前面置入 --no-，會關閉此選項。指定短選項名稱時，總是會設定相反的預設值。

您可以群組短布林選項。例如，可以使用 `-Ie` 來指定互動式 (短選項 `-I`) 與回應 (短選項 `-e`)。

運算元

運算元由空格或標籤分隔。它們可以以任何順序排列在指令語法中。您可以在運算元之後使用 --no 選項，將選項與運算元分開。以下任何引數均可以被視爲運算元，即使其以破折號 (-) 開始。例如，於以下碼行中

```
asadmin> create-jvm-options --instance server1 -- -Xmx1500m
```

`-XMx1500m` 被視爲運算元 (雖然其以破折號開始)。

語法範例

```
asadmin create-instance [--user admin_user] [--password admin_password]
[-H host_name] [--port port_number] [--sysuser sys_user] [--domain
domain_name] [--local=true/false] [--passwordfile file_name] [--secure | -s]
--instanceport instance_port instance_name
```

在此語法範例中，`-H` 爲主機名稱的短選項；`--user` 爲長選項，*admin_user* 爲該選項的引數；*instance_name* 爲運算元。括號內爲選擇性選項。

以下範例顯示具有實數值的語法。某些選擇性選項未在此範例中使用。

```
asadmin create-instance --user admin --password password -H austen
--port 4848 --instanceport 1024 server2
```

使用單模式與多重模式

您可以在單模式或多重模式下執行 `asadmin`。在單模式下，每次從指令提示下僅執行一個指令。在多重模式下，您可以執行多個指令，而無需重新輸入環境層級資訊。

如果您正在使用檔案中的輸入，並且指令執行失敗，則在單模式下，此程式會結束。如果您正在使用多重模式，並且指令執行失敗，則會返回至 `asadmin` 提示。

單模式

如果您在指令行介面中從指令提示下呼叫單一指令，則您在單模式下執行。指令行介面會執行此指令，然後返回至指令提示。若要從指令提示下執行指令行介面，請移往 *install_dir*/appserv/bin 目錄，然後在指令提示下鍵入指令：

```
> asadmin command options arguments
```

例如：

```
> asadmin create-instance --user admin --password password -H austen
--port 4848 --instanceport 1024 server2
```

多重模式

多重模式讓您在開始時設定環境，以便無需重新輸入特定環境層級資訊 (例如伺服器名稱、連接埠與密碼) 便可以執行多個指令。使用多重模式的一個突出優點是可以更快速地輸入並執行指令，因爲 `asadmin` 儲存在記憶體中。如果在作業系統層級中設定這些環境變數，則多重模式會使用那些設定。`asadmin` 公用程式會一直使用這些設定，除非您變更它們。

在 Windows 中，當您執行 asadmin.bat 檔案時，會自動處於多重模式中。

在 UNIX 中，若要在多重模式下從指令行啓動 asadmin 公用程式，請鍵入：

```
> asadmin 多重模式
```

處於多重模式時，指令提示會變更爲 asadmin。然後您便可以在 asadmin 提示下鍵入指令。無需使用公用程式名稱。例如：

```
asadmin> create-instance --user admin --password password -H austen
--port 4848 --instanceport 1024 server2
```

鍵入 exit 或 quit 後，會結束多重模式。您返回至指令提示。

多重模式內的多重模式

也可以透過使用以下指令，從多重模式階段作業中呼叫多重模式：

```
asadmin> multimode
```

結束第二個多重模式環境後，您便會返回至原始的多重模式環境。

例如，如果您正在多重模式下管理 server1，並且要管理 server2，以比較這兩個指令，則可以從執行 server1 的多重模式內呼叫執行 server2 的多重模式。由於您無需結束目前的多重模式階段作業，因此可以保留環境設定。當您結束正在使用的 server2 多重模式階段作業時，會返回至 server1 多重模式環境。

使用互動式與非互動式選項

您使用指令行介面時，可以在互動式或非互動式模式下使用此介面。如果您選擇使用互動式模式，但未指定互動式模式，系統會提示您輸入密碼。依預設，會啓用互動式模式。

您可以透過使用 export 指令設定互動式環境變數，停用與啓用互動式模式。如需更多資訊，請參閱 「與 export 指令一起使用的環境變數」表格。

您可以在單模式的各種情況下，使用互動式選項。在每次從指令提示執行一個指令以及在多重模式下從檔案中執行指令時，可以使用多重模式中的互動式選項。但是，在多重模式下從輸入串流中傳輸的指令以及從另一個程式中呼叫的指令，不能在互動式模式下執行。

使用環境指令

asadmin 公用程式包含您可以使用環境指令設定的環境變數集。在多重模式下，一旦設定了這些變數，結束多重模式之前便無需重新設定您的環境。也可以在作業系統層級中設定這些環境變數，如果這樣做，在您輸入多重模式時會自動點選這些環境變數，並且在結束多重模式之後它們會繼續存在。

環境變數是可以在任何時間透過指定進行設定的名稱/值對。環境變數由大寫的選項名稱之字首 AS_ADMIN_ 構成。例如，若要設定管理伺服器使用者，可以鍵入：

```
export AS_ADMIN_USER=administrator
```

其中 *administrator* 是管理員的使用者名稱。

同時，AS_ADMIN_USER 的值也可用於 asadmin 指令，例如：

```
asadmin 多重模式
asadmin> export AS_ADMIN_HOST=austen
```

只要您在多重模式階段作業中，管理伺服器主機名稱就會設定爲 austen，除非您重新指定該名稱。

也可以在一個步驟中，設定並匯出多重環境變數的值，例如：

```
asadmin> export AS_ADMIN_PORT=4848 AS_ADMIN_USER=admin
```

若要查看目前的環境變數設定，請使用沒有引數的 export 指令：

```
asadmin> export
AS_ADMIN_HOST=austen
AS_ADMIN_PORT=4848
AS_ADMIN_USER=admin
```

使用 unset 指令，移除此環境中的變數及其值。例如：

```
asadmin> unset AS_ADMIN_HOST
```

您可以透過重新設定變數或設定作爲 asadmin 指令部分的另一個值，置換環境變數的設定值。例如：

```
asadmin> export AS_ADMIN_HOST=dickens
asadmin> show-instance-status --host austen instance-name
```

此範例顯示管理伺服器主機 austen 中的實例狀況，因爲該值置換了 dickens 的先前主機值。

如果您不使用已匯出的變數，則必須爲大多數指令提供以下選項或使用預設值 (如需預設值的清單，請參閱第 428 頁的 「長選項格式與短選項格式、預設值與對等的環境變數」)：

- --host
- --port
- --user
- --password 或 --passwordfile
- --secure=true (如果安全)
- --instance (如果需要)

下表「與 export 指令一起使用的環境變數」描述與 export 指令一起使用的一些環境變數。這些變數爲經常使用的變數，因爲其專用於設定環境。第一欄顯示環境變數名稱，第二欄顯示使用情況，並且如果未設定值，將使用預設值。如需環境變數的完整清單，請參閱第 428 頁的 「長選項格式與短選項格式、預設值與對等的環境變數」。

表格 A-2 與 export 指令一起使用的環境變數

環境變數	用法
AS_ADMIN_HOST	管理伺服器的主機名稱。如果未指定值，則使用 localhost。
AS_ADMIN_PORT	管理伺服器的連接埠號。如果未指定值，則使用 4848。
AS_ADMIN_USER	執行指令的使用者名稱。
AS_ADMIN_PASSWORD	執行指令的使用者名稱密碼。使用者名稱與密碼用於認證使用者，以確認允許使用者管理伺服器。這與透過管理介面存取管理伺服器時進行的認證相同。
AS_ADMIN_SECURE	如果安全的話，則爲 `=true`。
AS_ADMIN_INSTANCE	設定 Sun ONE Application Server 的實例。任何使用實例名稱作爲引數的後續指令 (但不是使用其作爲運算元的指令) 使用此指定的實例。

使用密碼檔案選項

如果您不想在指令行鍵入密碼或爲密碼設定環境變數，可以建立一個密碼檔案，並使用其作爲指令行中的一個選項。

每個具有 `password` 選項的指令也具有可以取代使用的 `passwordfile` 選項。密碼檔案包含以下行：

`AS_ADMIN_PASSWORD=`*value*

`AS_ADMIN_ADMINPASSWORD=`*value*

`AS_ADMIN_USERPASSWORD=`*value*

如果您使用 `passwordfile` 選項，則檔案中的密碼會匯出至多重模式環境中，並且沒有指定 `password` 選項的後續指令使用這些值。

如果您在指令行同時指定密碼與密碼檔案選項，則密碼檔案中的值會匯出至多重模式環境中，但是目前的指令仍使用密碼選項中指定的密碼，因爲密碼選項要優先於密碼檔案。

本機或遠端運行 asadmin

通常 `asadmin` 公用程式透過管理伺服器發送其指令。因此，不需要在安裝了 Sun ONE Application Server 的系統上執行 `asadmin`。但是，必須執行管理伺服器，以便讓大多數 `asadmin` 指令進行工作。

某些指令具有在本機執行的選項，例如，`create-instance`。如果您在 `create-instance` 中使用 `--local=true` 選項，則必須在安裝此伺服器的機器上執行它，但是無需執行管理伺服器來建立實例。

某些指令必須在本機執行。例如，啓動管理伺服器及其所有實例的 `start-appserv` 不能在遠端執行，因爲管理伺服器只有在該指令啓動它時才執行。

如需有關管理伺服器的更多資訊，請參閱第 2 章「設定管理伺服器偏好設定」。

以下指令可以在本機與遠端執行：

- create-instance
- delete-instance
- list-instances
- start-instance
- stop-instance
- display-license

- version
- stop-domain
- restart-instance
- list-domains

對於這些指令，無需您指定本機選項，便可選擇在本機執行指令。依預設，如果您在指令語法中指定使用者、密碼、主機或連接埠的值，則此指令會被視爲遠端指令 (雖然您仍然可以爲這些選項指定本機值)。如果您不指定這些選項的值，則依預設，此指令在本機執行。

在本機執行指令時，如果有一個領域選項，則其爲此指令所需的選項 (除非只有一個領域)；在遠端執行指令時，如果您指定了此指令，則會忽略此領域選項。

使用指令行呼叫

您可以以多種方式呼叫指令行，如以下主題所述：

- 從指令行使用 asadmin
- 以檔案 (程序檔) 中的輸入，使用 asadmin
- 以標準輸入 (管道) 使用 asadmin

從指令行使用 asadmin

使用指令的最簡便方式是在指令行上，一次僅執行一個指令。您鍵入公用程式、指令及其選項與引數。在多重模式下，無需重新鍵入公用程式名稱與環境選項 (如果您已設定環境變數)，便可鍵入多個指令。您可以互動式或非互動式執行單模式或多重模式指令 (提示進行其他需要的輸入，例如，密碼)。

如需關於單模式與多重模式的更多資訊，請參閱第 381 頁的「使用單模式與多重模式」。

如需關於互動式使用指令的更多資訊，請參閱第 382 頁的「使用互動式與非互動式選項」。

指令行中的範例

```
> asadmin create-instance --user admin --password password --host
austen --port 4848 --instanceport 1024 server2
```

指令執行結束後，會返回至作業系統提示。

以檔案 (程序檔) 中的輸入，使用 asadmin

您可以建立包含許多 `asadmin` 指令的程序檔。使用程序檔，您可以批次處理指令、設定在特定時間要執行的工作，並且還可簡化管理工作，使之自動化。

若要呼叫檔案中的程序檔，請使用此語法：

```
> asadmin multimode --file filename
```

以下是您可以以此方式呼叫的檔案中簡單程序檔範例：

```
# 建立新實例並啓動它。
export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=mypassword
AS_ADMIN_HOST=austen AS_ADMIN_PORT=4848
create-instance --instanceport 9000 austen3
start-instance austen3
```

此程序檔設定環境，建立名爲 `austen3` 的實例，然後啓動新的實例。以數字符號 (#) 開始的行被視爲注釋，並被忽略。

以標準輸入 (管道) 使用 asadmin

您可以使用以下語法，在 `asadmin` 公用程式中傳輸輸入：

```
cat filename | asadmin multimode
```

此語法可能在 Windows 中無法使用。

使用逸出字元

如果您在指令語法中使用某些字元，如冒號 (:)、星號 (*) 與反斜線 (\)，則會發生錯誤 (除非您使用逸出字元將其分開)。使用逸出字元的可能性因您使用的平台與您使用單模式還是多重模式而異。

注意 不需要在 `get` 與 `set` 指令中爲冒號使用逸出字元。

本章節包含下列主題：

- 在單模式下的 UNIX 中之逸出字元
- 在單模式下的 Windows 中之逸出字元
- 在單模式下的所有平台中之逸出字元
- 在多重模式下的所有平台中之逸出字元

在單模式下的 UNIX 中之逸出字元

在 Solaris 中，您可以使用兩個反斜線 (\\) 或雙引號 (“ ”)，以逸出受限的字元。

使用反斜線 (\\) 逸出

例如，在建立 JDBC 連線區時，其中其選項值包含冒號，則可以使用反斜線 (範例假設已爲某些特性設定了環境變數)：

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=jdbc\\:oracle\\:thin\\:@asperfsol8\\:1521\\:V8i:user=staging_lo
okup_app:password=staging_lookup_app OraclePoollookup
```

使用引號逸出

若要在以上同一個範例中使用引號，則需以雙引號 (") 包含此值，然後使用反斜線逸出此雙引號。

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=\"jdbc:oracle:thin:@asperfsol8:1521:V8i\":user=staging_lookup_a
pp:password=staging_lookup_app OraclePoollookup
```

也可以使用第 389 頁的「在單模式下的所有平台中之逸出字元」中所述的方法。

在單模式下的 Windows 中之逸出字元

在 windows 中，您可以使用反斜線字元逸出。例如，在建立 JDBC 連線區時，其中其選項值包含冒號，則可以使用反斜線 (範例假設已爲某些特性設定了環境變數)：

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=jdbc\:oracle\:thin\:@asperfsol8\:1521\:V8i:user=staging_lookup_
app:password=staging_lookup_app OraclePoollookup
```

也可以使用第 389 頁的「在單模式下的所有平台中之逸出字元」中所述的方法。

在單模式下的所有平台中之逸出字元

在任何平台中，您可以使用反斜線逸出字元，並且在雙引號中加上包含逸出字元的值。例如，在建立 JDBC 連線區時，其中其選項值包含冒號，則可以使用如下所示的逸出字元 (範例假設已爲某些特性設定了環境變數)：

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url="jdbc\:oracle\:thin\:@iasperfsol8\:1521\:V8i":user=staging_look
up_app:password=staging_lookup_app OraclePoollookup
```

在多重模式下的所有平台中之逸出字元

在多重模式下，您可以使用以下語法，其僅需要引號，而不需斜線或反斜線：

```
asadmin> create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url="jdbc:oracle:thin:@asperfsol8:1521:V8i":user=staging_lookup_app
:password=staging_lookup_app OraclePoollookup
```

使用 get 與 set 指令

使用 `get` 與 `set` 指令，可存取並變更 Sun ONE Application Server 中的配置設定。在大多數情況下，`asadmin` 指令僅設定**所需**的特性。使用 `set` 指令，變更選擇性特性的值。

表格 A-3 get 與 set 指令

指令	引數	用法
get	(scope)，其中 scope 表示屬性，爲有效名稱。	取得屬性值。
set	(scope=*value*)，其中 scope 表示屬性，爲有效名稱，而 *value* 是您要爲此屬性設定的值。	設定屬性值。
reconfig	instance-name	執行修改配置檔案的任何指令之後，需要執行 reconfig，以便使變更套用於伺服器。如需關於套用變更/重新配置伺服器的更多資訊，請參閱第 74 頁的「將變更套用至應用程式伺服器實例」。

您可以透過在屬性之間使用空格，在單一指令中取得或設定多個屬性值。例如：

```
set server1.appReloadPollInterval=20
server1.mime.mime1.file=mime.types
```

也可以使用 AS_ADMIN_PREFIX 環境變數，設定後續 get 與 set 指令使用的字首。小數點 (".") 隱式插入在 get 和 set 指令中的前綴字串與運算元之間。例如：

```
asadmin>export AS_ADMIN_PREFIX=server1
asadmin>get *
server1.locale = en_US
server1.appReloadPollInterval = 2
server1.name = server1
...
```

由於 get 與 set 指令需要小數點分隔符號，如果項目在其名稱中包含小數點，則必須在小數點之前使用逸出字元反斜線 (\)。以下範例顯示伺服器實例名稱 server2.sun.com，其小數點之前爲反斜線：

```
get server2\.sun\.com.*
```

如果您不包含反斜線，則會出現錯誤訊息。

get 與 set 指令範例

以下範例顯示如何使用 get 指令取得屬性值，以及如何使用 set 指令設定值。

MDB 容器服務範例

如果應用程式伺服器實例爲 server1，則可以使用多重模式中的以下指令，取得所有 mdb-container 屬性的值，其中環境設定爲：

```
asadmin> get server1.mdb-container.*
```

以下是此指令的輸出範例，顯示屬性的目前值：

```
server1.mdb-container.logLevel = null
server1.mdb-container.steadyPoolSize = 10
server1.mdb-container.idleInPoolTimeoutInSeconds = 600
server1.mdb-container.maxPoolSize = 60
server1.mdb-container.monitoringEnabled = false
server1.mdb-container.poolResizeQuantity = 2
```

若要僅取得 MDB 容器屬性 monitoringEnabled 的值，請使用以下內容：

```
asadmin> get server1.mdb-container.montioringEnabled
```

若要將 monitoringEnabled 屬性值設定爲 true，請使用以下內容：

```
asadmin> set server1.mdb-container.montioringEnabled=true
```

JMS 資源範例

如需配置任何資源，此屬性應該如下表示：

instancename.resource.primary_key_value.attribute_name

例如：

```
asadmin> get server1.jms-resource.myjms.*
```

取得名爲 `myjms` 的 JMS 目標資源之所有屬性。例如：

```
server1.jms-resource.myjms.resType = javax.jms.Topic
server1.jms-resource.myjms.enabled = true
server1.jms-resource.myjms.name = myjms
server1.jms-resource.myjms.description = null
```

取得單一屬性的值，例如 `resType`：

```
asadmin> get server1.jms-resource.myjms.resType
```

設定屬性，例如 `description`：

```
asadmin> set server1.jms-resource.myjms.description=mydescription
```

此範例將 description 設定爲 `mydescription`。

取得並設定多個值範例

您可以使用同一指令，取得並設定多個值。若要同時設定兩個屬性，請使用空格分隔屬性。例如：

```
set server1.appReloadPollInterval=20
server1.mime.mime1.file=mime.types
```

同時，也可以使用環境變數 `AS_ADMIN_PREFIX`，設定許多 `get` 與 `set` 指令要使用的字首。

使用 get 與 set 指令監視

也可以使用 get 與 set 指令，監視執行中的伺服器。list 指令也用於監視。您可以將 `monitor` 選項設定爲 true 或 false。如果設定爲 true，則會監視已指定的屬性。如需關於使用指令行介面監視 Sun ONE Application Server 的更多資訊，請參閱第 119 頁的「使用 CLI 擷取監視資料」。

使用輔助說明

每個 asadmin 指令的輔助說明均可以透過鍵入 -h 或 --help，從指令提示下取得。例如，如需關於 asadmin 的輔助說明，請鍵入：

```
asadmin --help
```

您會看到所有 asadmin 指令的清單。

若要取得特定 asadmin 指令的輔助說明，請鍵入：

```
asadmin command -h
```

或

```
asadmin command --help
```

此輔助說明包含提要、指令說明、語法資訊、範例，以及相關指令的清單。

請注意，如果您在指令的任何位置使用 -h 或 --help，將取得該指令的輔助說明。此指令將不執行。

也可以在 UNIX 環境中，將指令行輔助說明頁存取爲線上援助頁。對於非隨附式安裝，請在 MANPATH 環境變數中加入 *install_dir*/man。完成之後，便可以存取 Sun ONE Application Server 公用程式的線上援助頁，例如，透過在指令提示下鍵入 man asadmin。

檢視輸出與錯誤

指令執行成功後，您將會看到通知您完成作業的訊息。如果指令執行失敗，則會看到錯誤訊息。

本章節包含下列主題：

- 檢視結束狀況
- 檢視用法

檢視結束狀況

除錯誤訊息外，asadmin 指令總是在結束狀況下結束。如果此指令執行成功，則結束狀況爲 0；如果指令執行失敗，則結束狀況爲 1。

在 *UNIX* 中的結束狀況

可以透過鍵入 `echo $?`，在指令提示下檢查結束狀況。

也可以在程序檔中使用結束碼，例如，以下 Korn shell 程序檔使用結束狀況，指出 `list-instances` 指令執行成功還是失敗：

```
#!/bin/ksh
asadmin list-instances
if  [[ $? = 0 ]]
   then
      echo "success"
   else
      echo "error"
fi
```

在 *Windows* 中的結束狀況

在 Windows 中，可以在 .bat 程序檔中檢查結束狀況。例如，以下兩個程序檔分別顯示成功的程序檔及其傳回的輸出，以及不成功的程序檔及其傳回的輸出：

Success Condition

```
myscript.bat
------------
echo off
echo Processing Command
call asadmin list-instances --domain domain1
if not %errorlevel% EQU 0  goto end
echo Command Succesful
goto program-end
:end
echo Command Failed
:program-end
```

Output:

```
Processing Command
admin-server <not running>
server1 <not running>
Command Succesful
```

Error Condition

```
myscript.bat
-------------
echo off
echo Processing Command
call asadmin list-instances
if not %errorlevel% EQU 0  goto end
echo Command Succesful
```

```
goto program-end
:end
echo Command Failed
:program-end
```

Output:

```
Processing Command
No default domain.Need to enter a domain.
Command Failed
```

檢視用法

如果鍵入沒有引數的指令，則會出現包含此指令語法的錯誤訊息。例如：

```
asadmin> create-instance

Invalid number of operands received

USAGE: create-instance [--user admin_user] [--password
admin_password] [--host localhost] [--port 4848] [--sysuser
sys_user] [--domain domain_name] [--local=false] [--passwordfile
file_name] [secure | -s] --instanceport instanceport instancename
```

安全性考量

您從指令行執行指令行介面時，必須提供使用所有指令的密碼。如果在多重模式下執行，則必須在設定環境時，首先提供密碼。如果結束多重模式，您再次啓動多重模式時，必須再次設定環境，包括密碼。使用環境指令設定密碼。如需更多資訊，請參閱第 383 頁的「使用環境指令」。

也可以設定密碼檔案，因此，無需在指令行鍵入密碼。如需更多資訊，請參閱第 385 頁的「使用密碼檔案選項」。

如果沒有有效使用者名稱與密碼的認證資訊，指令將不執行。

指令行介面具有您爲 Sun ONE Application Server 設定的安全性措施。如需關於 Sun ONE Application Server 中安全性的更多資訊，請參閱「*Sun ONE Application Server Administrator's Security Guide*」。

並行存取考量

有可能多個使用者使用指令行介面和/或管理介面，嘗試同時配置伺服器。如果這樣，第二個配置請求一直佇列到第一個配置請求完成爲止。如果此請求佇列的時間太長，則會逾時。

對於某些指令，只有您使用 reconfig 指令之後，這些變更才會生效。這意味著有多個使用者在變更套用於伺服器之前，可以編輯屬性。如需關於 reconfig 的更多資訊，請參閱第 74 頁的「將變更套用至應用程式伺服器實例」。

指令參考

本章節包含下列主題：

- 指令清單
- 點名稱與屬性清單
- 長選項格式與短選項格式、預設值與對等的環境變數

指令清單

下表顯示所有的 asadmin 指令及其用途。如需關於指令語法與用法的更多資訊，請參閱線上說明。

左欄顯示指令名稱，右欄顯示其用法。

表格 A-4 asadmin 指令

指令	用法
add-resources	加入類型爲 jdbc、jms 或 javamail 的一個或更多資源。
create-acl	建立 ACL (存取控制清單)。
create-authdb	建立認證資料庫。
create-auth-realm	建立認證範圍。
create-custom-resource	建立自訂資源。
create-domain	建立領域。
create-file-user	在 keyfile 中建立檔案範圍使用者。

表格 A-4 asadmin 指令

指令	用法
create-http-listener	建立 HTTP 偵聽程式。
create-http-qos	爲應用程式伺服器實例或虛擬伺服器建立服務設定的 HTTP 品質。
create-iiop-listener	建立 IIOP 偵聽程式。
create-instance	建立應用程式伺服器實例。
create-javamail-resource	建立 Java 郵件資源。
create-jdbc-connection-pool	建立 JDBC 連線區。
create-jdbc-resource	建立 JDBC 資源。
create-jmsdest	建立 JMS (Java 訊息服務) 目標。
create-jms-resource	建立 JMS 資源。
create-jndi-resource	建立 JNDI 資源。
create-jvm-options	在 java-config 中建立 JVM 選項或測量程式元素。
create-lifecycle-module	建立生命週期模組。
create-mime	建立 MIME 類型檔案。
create-persistence-resource	建立持續性管理程式 Factory 資源。
create-profiler	爲 JVM 建立測量程式。
create-ssl	建立 HTTP 偵聽程式、IIOP 偵聽程式或 IIOP 服務的 SSL 設定。
create-virtual-server	建立虛擬伺服器。
delete-acl	刪除 ACL。
delete-authdb	刪除認證資料庫。
delete-auth-realm	刪除認證範圍。
delete-custom-resource	刪除自訂資源。
delete-domain	刪除領域。此指令僅可以在本機執行。
delete-file-user	從 keyfile 中刪除檔案範圍使用者。
delete-http-listener	刪除 HTTP 偵聽程式。
delete-http-qos	刪除應用程式伺服器實例或虛擬伺服器的服務設定之 HTTP 品質。
delete-iiop-listener	刪除 IIOP 偵聽程式。

表格 A-4 asadmin 指令

指令	用法
delete-instance	刪除應用程式伺服器實例。
delete-javamail-resource	刪除 Java 郵件資源。
delete-jdbc-connection-pool	刪除 JDBC 連線區。
delete-jdbc-resource	刪除 JDBC 資源。
delete-jmsdest	刪除 JMS 目標。
delete-jms-resource	刪除 JMS 資源。
delete-jndi-resource	刪除 JNDI 資源。
delete-jvm-options	在 java-config 中刪除 JVM 選項或測量程式元素。
delete-lifecycle-module	刪除生命週期模組。
delete-mime	刪除 MIME 類型檔案。
delete-persistence-resource	刪除持續性管理程式 Factory 資源。
delete-profiler	刪除 JVM 測量程式。
delete-ssl	刪除 HTTP 偵聽程式、IIOP 偵聽程式或 IIOP 服務的 SSL 設定。
delete-virtual server	刪除虛擬伺服器。
deploy	部署 EJB、WEB、連接器、appclient 或應用程式伺服器實例的應用程式元件。
deploydir	部署 EJB、WEB、連接器、appclient 或在此目錄中的應用程式伺服器實例之應用程式元件。
disable	停用應用程式伺服器實例中的已部署元件。
display-license	顯示授權資訊。此指令僅可以在本機執行。
enable	啓用 (允許執行) 應用程式伺服器實例中的已部署元件。
export	匯出 asadmin 環境變數值，以便後續的 asadmin 指令可以使用該值。
get	取得屬性值。
help	顯示給定指令的輔助說明 (說明、用法、語法、範例)，或 asadmin 的一般輔助說明。
install-license	安裝授權檔案。此指令僅可以在本機執行。
jms-ping	ping JMS 提供者，以查看其是否在執行。

表格 A-4 asadmin 指令

指令	用法
list	列示可配置的元素。
list-acls	列示應用程式伺服器實例的 ACL。
list-authdbs	列示認證資料庫。
list-auth-realms	列示認證範圍。
list-components	列示伺服器實例的已部署元件。
list-custom-resources	列示伺服器實例中的自訂資源。
list-domains	列示領域。
list-file-users	列示伺服器實例中的所有檔案範圍使用者。
list-file-groups	列示指定的檔案範圍使用者之所有群組。如果您不指定使用者，則會列示伺服器實例的所有群組。
list-http-listeners	列示伺服器實例的 HTTP 偵聽程式。
list-instances	列示領域中的應用程式伺服器實例。
list-iiop-listeners	列示伺服器實例的 IIOP 偵聽程式。
list-javamail-resources	列示伺服器實例的 Java 郵件資源。
list-jdbc-connection-pools	列示伺服器實例的 JDBC 連線區。
list-jdbc-resources	列示伺服器實例的 JDBC 資源。
list-jmsdest	列示伺服器實例的 JMS 目標。
list-jms-resources	列示伺服器實例的 JMS 資源。
list-jndi-resources	列示伺服器實例的 JNDI 資源。
list-lifecycle-modules	列示伺服器實例的生命週期模組。
list-mimes	列示伺服器實例的 MIME 類型檔案。
list-persistence-resources	列示伺服器實例的持續性管理程式 Factory 資源。
list-profilers	列示伺服器實例的 JVM 測量程式。
list-sub-components	在已部署模組或已部署應用程式的模組中列示一個或多個 EJB 或 Servlet。
list-virtual-servers	列示伺服器實例的虛擬伺服器。
multimode	允許您執行多重指令，同時保留環境設定並保留在 asadmin 中。
reconfig	將變更套用至伺服器。大多數變更僅在套用至伺服器後，其才能生效。

表格 A-4 asadmin 指令

指令	用法
restart-instance	重新啓動伺服器實例。
set	設定屬性值。
show-component-status	顯示已部署元件的狀況。
show-instance-status	顯示伺服器實例的狀況 (即，該實例是否在執行)。
shutdown	關閉管理伺服器。
start-appserv	啓動管理伺服器及所有伺服器實例。此指令僅可以在本機執行。
start-domain	啓動領域中的所有實例。此指令僅可以在本機執行。
start-instance	啓動伺服器實例。
stop-appserv	停止管理伺服器及所有伺服器實例。此指令僅可以在本機執行。
stop-domain	停止領域中的所有實例。
stop-instance	停止伺服器實例。
undeploy	從伺服器實例中移出已部署的元件。
unset	取消設定 asadmin 的匯出環境變數。
update-file-user	更新現有的檔案範圍使用者。
version	顯示 Sun ONE Application Server 的版本資訊。

點名稱與屬性清單

您使用 get 或 set 指令取得並設定屬性時，需知道 asadmin 使用的服務與資源等的名稱，從而可以使用此名稱取得該特定物件的屬性。

由於使用這些名稱的語法在小數點之間分隔名稱，所以這些名稱稱爲點名稱。

在 asadmin 中使用的點名稱

下表列示使用 `asadmin` 配置項目所使用的名稱。這些名稱分爲以下幾類：

- 服務名稱
- 資源名稱
- 應用程式名稱
- 其他名稱

服務名稱

下表顯示取得並設定服務屬性要使用的服務名稱：

表格 A-5 指令行介面的服務名稱

服務	點名稱
JMS 服務配置	jms-service
作業事件服務配置	transaction-service
MDB 容器配置	mdb-container
EJB 容器配置	ejb-container
Web 容器配置	web-container
JVM 配置	java-config
ORB 配置	orb 或 iiop-service

ORB 偵聽程式配置	orblistener 或 iiop-listener
	請注意，orblistener 或 iiop-listener 不是有效名稱，兩者均需要偵聽程式的名稱跟隨其後。例如：
	`ORB 偵聽程式配置`
	`orblistener.< 偵聽程式名稱 > 或`
	`iiop-listener.< 偵聽程式名稱 >`

日誌配置	log-service
安全性配置	security-service
HTTP 配置	http-service

資源名稱

下表顯示取得並設定資源屬性要使用的資源名稱。請注意，這些名稱本身是無效的，其需要資源名稱跟隨其後。

表格 A-6 指令行介面的資源名稱

資源	點名稱
JDBC 資源配置	jdbc-resource
JNDI 資源配置	jndi-resource
JDBC 連線區資源配置	jdbc-connection-pool
自訂資源配置	custom-resource
JMS 資源配置	jms-resource
持續性管理程式 Factory 資源配置	persistence-manager-factory-resource
Java 郵件資源配置	mail-resource

應用程式名稱

下表顯示取得並設定應用程式相關的配置屬性要使用的點名稱。請注意，這些名稱本身是無效的，其需要應用程式名稱跟隨其後。

表格 A-7 指令行介面的應用程式名稱

應用程式元件	點名稱
應用程式配置	application
EJB 模組配置	ejb-module
Web 模組配置	web-module
連接器模組配置	connector-module

其他名稱

下表顯示使用 `get` 與 `set` 可以配置的其他項目之點名稱。請注意，這些名稱本身是無效的，其需要應用程式名稱跟隨其後。例如，`http-listener.`*listener_name*、`lifecycle-module.`*module-name* 等。

表格 A-8 指令行介面的其他項目名稱

項目	點名稱
HTTP 偵聽程式	http-listener 或 http-server.http-listener
MIME 類型檔案	mime
ACL	acl
虛擬伺服器	virtual-server
認證資料庫	auth-db
安全性範圍	authrealm
生命週期模組	lifecycle-module
測量程式配置	profiler
伺服器配置	server configuration (伺服器實例名稱)

屬性

以下章節顯示以上列示的每個具名項目的屬性，並且提供用法範例。請注意，某些屬性是唯讀的 (即，其僅可以用在 `get` 指令中，而不能用在 `set` 指令中)。

注意 本小節的範例假設已在環境變數中定義了使用者、密碼、主機與連接埠，並且在此語法中未列示這些選項。

jms-service

下表在左欄中顯示屬性的 server.xml 名稱，在右欄中顯示 asadmin 使用的名稱。

表格 A-9 JMS 服務屬性

server.xml 名稱	asadmin 名稱
port	port
admin-username	adminUserName
admin-password	adminPassword
log-level	logLevel
enabled	enabled
init-timeout-in-seconds	initTimeoutInSeconds
start-args	startArgs

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.jms-service.*
```

取得名爲 adminPassword 的屬性：

```
asadmin> get server1.jms-service.adminPassword
```

將名爲 adminPassword 的屬性設定爲值 admin：

```
asadmin> set server1.jms-service.adminPassword=admin
```

transaction-service

下表在左欄中顯示屬性的 server.xml 名稱，在右欄中顯示 asadmin 使用的名稱。

表格 A-10 作業事件服務屬性

server.xml 名稱	asadmin 名稱
automatic-recovery	automaticTransactionRecovery
timeout-in-seconds	transactionRecoveryTimeout
tx-log-dir	transactionLogFile
heuristic-decision	heuristicDecision

表格 A-10 作業事件服務屬性

server.xml 名稱	asadmin 名稱
keypoint-interval	keypointInterval
log-level	logLevel
monitoring-enabled	monitoringEnabled

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.transaction-service.*
```

取得名爲 `transactionRecoveryTimeout` 的屬性：

```
asadmin> get server1.transaction-service.transactionRecoveryTimeout
```

將名爲 `transactionRecoveryTimeout` 的屬性設定爲值 49：

```
asadmin> set
server1.transaction-service.transactionRecoveryTimeout=49
```

mdb-container

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-11 MDB 容器屬性

server.xml 名稱	asadmin 名稱
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
log-level	logLevel
monitoring-enabled	monitoringEnabled

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.mdb-container.*
```

取得名爲 `steadyPoolSize` 的屬性：

```
asadmin> get server1.mdb-container.steadyPoolSize
```

將名爲 `steadyPoolSize` 的屬性設定爲值 10：

```
asadmin> set server1.mdb-container.steadyPoolSize=10
```

ejb-container

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-12 EJB 容器屬性

server.xml 名稱	asadmin 名稱
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
cache-resize-quantity	cacheResizeQuantity
max-cache-size	maxCacheSize
pool-idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
cache-idle-timeout-in-seconds	idleInCacheTimeoutInSeconds
removal-timeout-in-seconds	removalTimeoutInSeconds
victim-selection-policy	victimSelectionPolicy
commit-option	commitOption
log-level	logLevel
monitoring-enabled	monitoringEnabled

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.ejb-container.*
```

取得名爲 maxPoolSize 的屬性：

```
asadmin> get server1.ejb-container.maxPoolSize
```

將名爲 maxPoolSize 的屬性設定爲值 12：

```
asadmin> set server1.ejb-container.maxPoolSize=12
```

web-container

下表在左欄中顯示屬性的 server.xml 名稱，右欄中顯示 asadmin 使用的名稱。

表格 A-13 Web 容器屬性

server.xml 名稱	asadmin 名稱
log-level	logLevel
monitoring-enabled	monitoring-enabled (不使用)

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.web-container.*
```

取得名爲 logLevel 的屬性：

```
asadmin> get server1.web-container.logLevel
```

將名爲 monitoringEnabled 屬性設定爲 WARNING：

```
asadmin> set server1.web-container.logLevel=WARNING
```

java-config

下表在左欄中顯示屬性的 `sserver.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-14 JVM 屬性

server.xml 名稱	asadmin 名稱
java-home	javahome
debug-enabled	debugEnabled
debug-options	debugOptions
javac-options	javacoptions
rmic-options	rmicoptions
classpath-prefix	classpathprefix
server-classpath	serverClasspath
classpath-suffix	classpathsuffix
native-library-path-prefix	libpathprefix
native-library-path-suffix	libpathsuffix
env-classpath-ignored	envpathignore

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.java-config.*
```

取得名爲 `classpathprefix` 的屬性：

```
asadmin> get server1.java-config.classpathprefix
```

將名爲 `classpathprefix` 的屬性設定爲值 `com.sun`：

```
asadmin> set server1.java-config.classpathprefix=com.sun
```

orb 或 iiop-service

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-15 ORB/IIOP 服務屬性

server.xml 名稱	asadmin 名稱
message-fragment-size	msgSize
steady-thread-pool-size	minThreads
max-thread-pool-size	maxThreads
max-connections	maxConnections
idle-thread-timeout-in-seconds	idleThreadTimeout
log-level	log
monitoring-enabled	monitor
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.orb.*
```

或

```
asadmin> get server1.iiop-service.*
```

取得名爲 `msgSize` 的屬性：

```
asadmin> get server1.orb.msgSize
```

或

```
asadmin> get server1.iiop-service.msgSize
```

將名爲 idleThreadTimeout 的屬性設定 300：

```
asadmin> set server1.orb.idleThreadTimeout=300
```

或

```
asadmin> set server1.iiop-service.idleThreadTimeout=300
```

orblistener 或 iiop-listener

下表在左欄中顯示屬性的 server.xml 名稱，在右欄中顯示 asadmin 使用的名稱。

表格 A-16 IIOP 偵聽程式屬性

server.xml 名稱	asadmin 名稱
id	id
address	address
port	port
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.orblistener.orb_listener_id.*
```

或

```
asadmin> get server1.iiop-listener.orb_listener_id.*
```

取得名爲 port 的屬性：

```
asadmin> get server1.orblistener.orb_listener_id.port
```

或

```
asadmin> get server1.iiop-listener.orb_listener_id.port
```

將名爲 address 的屬性設定爲 bluestar：

```
asadmin> set server1.orblistener.orb_listener_id.address=bluestar
```

或

```
asadmin> set server1.iiop-listener.orb_listener_id.address=bluestar
```

log-service

下表在左欄中顯示屬性的 server.xml 名稱，在右欄中顯示 asadmin 使用的名稱。

表格 A-17 日誌配置屬性

server.xml 名稱	asadmin 名稱
file	file
level	level
log-stdout	stdout
log-stderr	stderr
echo-log-messages-to-stderr	echoToStderr
create-console	createConsole
log-virtual-server-id	LogVirtualServerId
use-system-logging	useSystemLogging

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.log-service.*
```

取得名爲 level 的屬性：

```
asadmin> get server1.log-service.level
```

將名爲 echoToStderr 的屬性設定爲 true：

```
asadmin> set server1.log-service.echoToStderr=true
```

security-service

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-18 安全性範圍配置屬性

server.xml 名稱	asadmin 名稱
default-realm	defaultRealm
default-principal	defaultPrinicpal
default-principal-password	defaultPrinicpalPassword
anonymous-role	anonymousRole
audit-enabled	auditEnabled
log-level	logLevel

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.security-service.*
```

取得名爲 `anonymousRole` 的屬性：

```
asadmin> get server1.security-service.anonymousRole
```

將名爲 `encryptPasswords` 的屬性設定爲 true：

```
asadmin> set server1.security-service.auditEnabled=true
```

http-service

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-19 HTTP 服務屬性

server.xml 名稱	asadmin 名稱
qos-metrics-interval-in-seconds	qos-metrics-interval-in-seconds
qos-recompute-time-interval-in-millis	qos-recompute-time-interval-in-millis
qos-enabled	qos-enabled
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit

表格 A-19 HTTP 服務屬性

server.xml 名稱	asadmin 名稱
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.http-service.*
```

取得名爲 `bandwidthLimit` 的屬性：

```
asadmin> get server1.http-service.bandwidthLimit
```

將名爲 `qos-enabled` 的屬性設定爲 true：

```
asadmin> set server1.http-service.qos-enabled=true
```

jdbc-resource

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-20 JDBC 資源屬性

server.xml 名稱	asadmin 名稱
jndi-name	name
pool-name	pool
enabled	enabled
description	description

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.jdbc-resource.jdbc_resource_name.*
```

取得名爲 `pool` 的屬性：

```
asadmin> get server1.jdbc-resource.jdbc_resource_name.pool
```

將名爲 `enabled` 的屬性設定爲 true：

```
asadmin> set server1.jdbc-resource.jdbc_resource_name.enabled=true
```

jndi-resource

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-21 JNDI 資源屬性

server.xml 名稱	asadmin 名稱
jndi-name	name
jndi-lookup-name	LookupName
res-type	resType
factory-class	factory
enabled	enabled
description	description

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.jndi-resource.jndi_name.*
```

取得名爲 `factory` 的屬性：

```
asadmin> get server1.jndi-resource.jndi_name.factory
```

將名爲 `factory` 的屬性設定爲 `com.sun`：

```
asadmin> set server1.jndi-resource.jndi_name.factory=com.sun
```

jdbc-connection-pool

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-22 JDBC 連線區屬性

server.xml 名稱	asadmin 名稱
name	name
datasource-classname	dsClassName
res-type	resType
description	description
steady-pool-size	steadyPoolSize
max-pool-size	maxPoolSize

表格 A-22 JDBC 連線區屬性

server.xml 名稱	asadmin 名稱
max-wait-time-in-millis	maxWaitTime
pool-resize-quantity	resizeValue
idle-timeout-in-seconds	idleTimeout
transaction-isolation-level	transactionIsolationLevel
is-isolation-level-guaranteed	isIsolationLevelGuaranteed
connection-validation-method	validationMethod
is-connection-validation-required	isValidationRequired
fail-all-connections	failAll
validation-table-name	validationTable

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.jdbc-connection-pool.pool_name.*
```

取得名爲 `dsClassName` 的屬性：

```
asadmin> get server1.jdbc-connection-pool.pool_name.dsClassName
```

將名爲 `resizeValue` 的屬性設定爲 2：

```
asadmin> set server1.jdbc-connection-pool.pool_name.resizeValue=2
```

custom-resource

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-23 自訂資源屬性

server.xml 名稱	asadmin 名稱
jndi-name	name
res-type	resType
factory-class	factory
enabled	enabled
description	description

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.custom-resource.jndi_name.*
```

取得名爲 `factory` 的屬性：

```
asadmin> get server1.custom-resource.jndi_name.factory
```

設定名爲 `factory` 的屬性：

```
asadmin> set server1.custom-resource.jndi_name.factory=myclass
```

jms-resource

下表在左欄中顯示屬性的 `sserver.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-24 JMS 資源屬性

server.xml 名稱	asadmin 名稱
jndi-name	name
res-type	resType
enabled	enabled
description	description

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.jms-resource.jms_resource_name.*
```

取得名爲 `res-type` 的屬性：

```
asadmin> get server1.jms-resource.jms_resource_name.resType
```

將名爲 `enabled` 的屬性設定爲 true：

```
asadmin> set server1.jms-resource.jms_resource_name.enabled=true
```

persistence-manager-factory-resource

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-25 持續性管理程式 Factory 資源屬性

server.xml 名稱	asadmin 名稱
jndi-name	jndiName
jdbc-resource-jndi-name	JdbcResourceJndiName
factory-class	factoryClass
enabled	enabled
description	description

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.persistence-manager-factory-resource.jndi_name
```

取得名爲 `factoryClass` 的屬性：

```
asadmin> get
server1.persistence-manager-factory-resource.jndi_name.factoryClass
```

將名爲 `enabled` 的屬性設定爲 true：

```
asadmin> set
server1.persistence-manager-factory-resource.jndi_name.enabled=true
```

mail-resource

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-26 Java 郵件資源屬性

server.xml 名稱	asadmin 名稱
jndi-name	name
enabled	enabled
store-protocol	storeProtocol
store-protocol-class	storeProtocolClass
transport-protocol	transportProtocol

表格 A-26 Java 郵件資源屬性

server.xml 名稱	asadmin 名稱
transport-protocol-class	transportProtocolClass
host	host
user	user
from	from
debug	debug
description	description

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.mail-resource.jndi_name.*
```

取得名爲 `host` 的屬性：

```
asadmin> get server1.mail-resource.jndi_name.host
```

將名爲 `enabled` 的屬性設定爲 true：

```
asadmin> set server1.mail-resource.jndi_name.enabled=true
```

application

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-27 應用程式屬性

server.xml 名稱	asadmin 名稱
name	name
location	location
virtual-servers	virtualServers
description	description
enabled	enabled

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.application.application_name.*
```

在應用程式中取得名爲 location 的屬性：

```
asadmin> get server1.application.application_name.location
```

設定名爲 location 的屬性：

```
asadmin> set server1.application.application_name.location=
"/export/home/as7se/as1/repository/applications/ASConverter"
```

ejb-module

下表在左欄中顯示屬性的 server.xml 名稱，右欄中顯示 asadmin 使用的名稱。

表格 A-28 EJB 模組屬性

server.xml 名稱	asadmin 名稱
name	name
location	location
description	description
enabled	enabled

取得實例 (server1) 中獨立 EJB 模組的所有屬性：

```
asadmin> get server1.ejb-module.ejb_jar_name.*
```

爲實例 (server1) 取得應用程式中 EJB 模組的所有屬性：

```
asadmin> get
server1.j2ee-application.application_name.ejb-module.ejb_jar_name.*
```

或

```
asadmin>get server1.application.application_name.ejb-module.ejb_jar_name.*
```

從獨立的 EJB 模組中取得名爲 location 的屬性：

```
asadmin> get server1.ejb-module.ejb_jar_name.location
```

在應用程式的 EJB 模組中取得名爲 location 的屬性：

```
asadmin> get
server1.j2ee-application.application_name.ejb-module.ejb_jar_name.
location
```

或

```
asadmin> get
server1.application.application_name.ejb-module.ejb_jar_name.location
```

在獨立的 EJB 模組中設定名爲 location 的屬性：

```
asadmin> set
server1.ejb-module.ejb_jar_name.location="/export/home/as7se/as1/repos
itory/modules/ejb_jar_name"
```

在隨附於應用程式的 EJB 模組中設定名爲 location 的屬性：

```
asadmin> set
server1.j2ee-application.application_name.ejb-module.ejb_jar_name.
location="/export/home/as7se/as1/repository/modules/ejb_jar_name"
```

或

```
asadmin>set
server1.application.application_name.ejb-module.ejb_jar_name.location="/ex
port/home/as7se/as1/repository/modules/ejb_jar_name"
```

web-module

下表在左欄中顯示屬性的 server.xml 名稱，右欄中顯示 asadmin 使用的名稱。

表格 A-29 WEB 模組屬性

server.xml 名稱	asadmin 名稱
name	name
location	location
context-root	contextRoot
virtual-servers	virtualServers
description	description
enabled	enabled

取得實例 (server1) 中獨立 web 模組的所有屬性：

```
asadmin> get server1.web-module.web_war_name.*
```

爲實例 (server1) 取得應用程式中 web 模組的所有屬性：

```
asadmin> get server1.web-module.application_name.web_war_name.*
```

從獨立的 web 模組中取得名爲 `location` 的屬性：

```
asadmin> get server1.web-module.web_war_name.location
```

從應用程式的 web 模組中取得名爲 `location` 的屬性：

```
asadmin> get server1.web-module.application_name.web_war_name.location
```

在獨立的 web 模組中設定名爲 `location` 的屬性：

```
asadmin> set server1.web-module.war-ic.location=
"/export/home/as7se/as1/repository/modules/web_war_name"
```

在隨附於應用程式的 web 模組中設定名爲 `location` 的屬性：

```
asadmin> set server1.web-module.application_name.web_war_name.location=
"/export/home/as7se/as1/repository/modules/web_war_name"
```

connector-module

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-30 連接器模組屬性

server.xml 名稱	asadmin 名稱
name	name
location	location
description	description
enabled	enabled

取得實例 (server1) 中獨立連接器模組的所有屬性：

```
asadmin> get server1.connector-module.connector_rar_name.*
```

從獨立的連接器模組中取得名爲 `location` 的屬性：

```
asadmin> get server1.connector-module.connector_rar_name.location
```

在獨立的連接器模組中設定名爲 location 的屬性：

```
asadmin> set server1.connector-module.connector_rar_name.location=
"/export/home/as7se/as1/repository/modules/connector_rar_name"
```

http-listener 或 http-server.http-listener

下表在左欄中顯示屬性的 server.xml 名稱，在右欄中顯示 asadmin 使用的名稱。

表格 A-31 HTTP 偵聽程式屬性

server.xml 名稱	asadmin 名稱
id	id
address	address
port	port
family	family
acceptor-threads	acceptorThreads
blocking-enabled	blockingEnabled
security-enabled	securityEnabled
default-virtual-server	defaultVirtualServer
server-name	serverName
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.http-listener.http_listener_name.*
```

或

```
asadmin> get server1.http-server.http-listener.http_listener_name.*
```

取得名爲 `factory` 的屬性：

```
asadmin> get server1.httpl-listener.http_listener_name.address
```

或

```
asadmin> get server1.http-server.http-listener.http_listener_name.address
```

將名爲 `address` 的屬性設定爲 `0.0.0.0` IP 位址：

```
asadmin> set server1.http-listener.http_listener_name.address=0.0.0.0
```

或

```
asadmin> set
server1.http-server.http-listener.http_listener_name.address=0.0.0.0
```

mime

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-32 MIME 類型屬性

server.xml 名稱	asadmin 名稱
id	id
file	file

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.mime.mime_name.*
```

取得名爲 `file` 的屬性：

```
asadmin> get server1.mime.mime_name.file
```

將名爲 `file` 的屬性設定爲 `mime.types`：

```
asadmin> set server1.mime.mime_name.file=mime.types
```

acl

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-33 ACL 屬性

server.xml 名稱	asadmin 名稱
id	id
file	file

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.acl.acl_name.*
```

取得名爲 `file` 的屬性：

```
asadmin> get server1.acl.acl_name.file
```

設定名爲 `file` 的屬性：

```
asadmin> set server1.acl.acl_name.file=com/as1.acl
```

virtual-server

下表在左欄中顯示屬性的 `server.xml` 名稱，在右欄中顯示 `asadmin` 使用的名稱。

表格 A-34 虛擬伺服器屬性

server.xml 名稱	asadmin 名稱
id	id
http-listeners	httpListeners
config-file	configFile
default-object	defaultObject
accept-language	acceptLanguage
log-file	logFile
default-web-module	defaultWebModule
hosts	hosts

表格 A-34 虛擬伺服器屬性

server.xml 名稱	asadmin 名稱
mime	mime
state	state
acls	acls
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit
property name="dir" value=	property.dir
property name="nice" value=	property.nice
property name="user" value=	property.user
property name="group" value=	property.group
property name="chroot" value=	property.chroot
property name="docroot" value=	property.docroot
property name="accesslog" value=	property.accesslog

從實例 (server1) 中取得所有屬性：

```
asadmin> get instance_name.virtual-server.vserver_id.*
```

例如：

```
asadmin> get server1.virtual-server.server1.*
```

取得虛擬伺服器 server1 的名爲 `httpListeners` 之屬性：

```
asadmin> get server1.virtual-server.server1.httpListeners
```

將名爲 `acceptLanguage` 的屬性設定爲 false：

```
asadmin> set server1.virtual-acceptLanguage=false
```

auth-db

下表在左欄中顯示屬性的 server.xml 名稱，右欄中顯示 asadmin 使用的名稱。

表格 A-35 認證資料庫屬性

server.xml 名稱	asadmin 名稱
id	id
database	database
basedn	basedn
certmaps	certmaps

從實例中取得所有屬性：

```
asadmin> get instancename.virtual-server.vserver_id.auth-db.authdb_id.*
```

例如，對於實例 server1，虛擬伺服器 server1：

```
asadmin> get server1.virtual-server.server1.auth-db.authdb_id.*
```

取得名爲 database 的屬性：

```
asadmin> get server1.virtual-server.server1.auth-db.authdb_id.database
```

設定名爲 database 的屬性：

```
asadmin> set
server1.virtual-server.server1.auth-db.authdb_id.database=Oracle
```

authrealm

下表在左欄中顯示屬性的 server.xml 名稱，右欄中顯示 asadmin 使用的名稱。

表格 A-36 授權範圍屬性

server.xml 名稱	asadmin 名稱
name	name
classname	classname

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.authrealm.authrealm_id.*
```

取得名爲 classname 的屬性：

```
asadmin> get server1.authrealm.authrealm_id.classname
```

設定名爲 classname 的屬性：

```
asadmin> set
server1.authrealm.authrealm_id.classname=com.sun.as.security.auth.real
m.sharedpassword.SharedPasswordRealm
```

lifecycle-module

下表在左欄中顯示屬性的 server.xml 名稱，右欄中顯示 asadmin 使用的名稱。

表格 A-37 生命週期模組屬性

server.xml 名稱	asadmin 名稱
name	name
enabled	enabled
class-name	className
classpath	classPath
load-order	loadOrder
is-failure-fatal	isFailureFatal
description	description

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.lifecycle-module.lifecycle_module_id.*
```

爲生命週期模組取得名爲 className 的屬性：

```
asadmin> get server1.lifecycle-module.lifecycle_module_id.className
```

設定名爲 className 的屬性：

```
asadmin> set
server1.lifecycle-module.lifecycle_module_id.className=com.lifecycle_module_id.
lifecycle
```

profiler

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-38 JVM 效能評測器配置屬性

server.xml 名稱	asadmin 名稱
name	name
classpath	classPath
native-library-path	nativeLibraryPath
enabled	enabled

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.profiler.*
```

取得名爲 `enabled` 的屬性：

```
asadmin> get server1.profiler.enabled
```

將名爲 `enabled` 的屬性設定爲 false：

```
asadmin> set server1.profiler.enabled=false
```

server configuration (伺服器實例名稱)

下表在左欄中顯示屬性的 `server.xml` 名稱，右欄中顯示 `asadmin` 使用的名稱。

表格 A-39 伺服器配置屬性

server.xml 名稱	asadmin 名稱
instance-name	name
locale	locale
log-root	logRoot
session-store	sessionStore
application-root	applicationRoot
dynamic-reload-enabled	appDynamicReloadEnabled
dynamic-reload-poll-interval-in-seconds	appReloadPollInterval

從實例 (server1) 中取得所有屬性：

```
asadmin> get server1.*
```

取得名爲 `logRoot` 的屬性：

```
asadmin> get server1.logRoot
```

設定名爲 `logRoot` 的屬性：

```
asadmin> set server1.logRoot="/space/log"
```

長選項格式與短選項格式、預設值與對等的環境變數

下表列示指令行選項的長格式與短格式。如果沒有列示短格式，則此選項的短格式不可用。

表格 A-40 短選項與長選項、預設值與環境變數

選項名稱	長格式	短格式	預設值	環境變數
acceptlang	--acceptlang			AS_ADMIN_ACCEPT_
acceptorthreads	--acceptorthreads			AS_ADMIN_ACCEPTOR_THREADS
acls	--acls			AS_ADMIN_ACLS
address	--address			AS_ADMIN_ADDRESS
adminpassword	--adminpassword			AS_ADMIN_ADMINPASSWD
adminport	--adminport		4848	AS_ADMIN_ADMINPORT
adminuser	--adminuser			AS_ADMIN_ADMINUSER
basedn	--basedn			AS_ADMIN_BASEDN
blockingenabled	--blockingenabled			AS_ADMIN_BLOCKINGENABLED
bwlimit	--bwlimit			AS_ADMIN_BWLIMIT
certmaps	--certmaps			AS_ADMIN_CERTMAPS
certname	--certname			AS_ADMIN_CERTNAME
classname	--classname			AS_ADMIN_CLASSNAME
classpath	--classpath			AS_ADMIN_CLASSPATH
clientauthenabled	--clientauthenabled			AS_ADMIN_CLIENTAUTHENABLED

表格 A-40 短選項與長選項、預設值與環境變數

選項名稱	長格式	短格式	預設值	環境變數
configfile	--configfile			AS_ADMIN_CONFIGFILE
connectionpoolid	--connectionpoolid			AS_ADMIN_CONNECTIONPOOLID
connlimit	--connlimit			AS_ADMIN_CONNLIMIT
contextroot	--contextroot			AS_ADMIN_CONTEXTROOT
database	--database			AS_ADMIN_DATABASE
debug	--debug		false	AS_ADMIN_DEBUG
defaultobj	--defaultobj			AS_ADMIN_DEFAULTOBJ
defaultwebmodule	--defaultwebmodule			AS_ADMIN_DEFAULTWEBMODULE
description	--description			AS_ADMIN_DESCRIPTION
discardmanualchanges	--discardmanualchanges	-d	false	AS_ADMIN_DISCARDMANUALCHANGES
echo	--echo	-e	false	AS_ADMIN_ECHO
enabled	--enabled			AS_ADMIN_ENABLED
enforcebwlimit	--enforcebwlimit			AS_ADMIN_ENFORCEBWLIMIT
enforceconnlimit	--enforceconnlimit			AS_ADMIN_ENFORCECONNLIMIT
failconnection	--failconnection		false	AS_ADMIN_FAILCONNECTION
failurefatal	--failurefatal		false	AS_ADMIN_FAILUREFATAL
family	--family			AS_ADMIN_FAMILY
file	--file	-f		AS_ADMIN_FILE
force	--force	-F	true	AS_ADMIN_FORCE
help	--help	-h		AS_ADMIN_HELP
host	--host	-H		AS_ADMIN_HOST
hosts	--hosts			AS_ADMIN_HOSTS
httplistenerid	--httplistenerid			AS_ADMIN_HTTPLISTENERID
httplisteners	--httplisteners			AS_HTTP_LISTENERS
idletimeout	--idletimeout		300	AS_ADMIN_IDLETIMEOUT
instance	--instance	-i	server1	AS_ADMIN_INSTANCE
instanceport	--instanceport			AS_ADMIN_INSTANCEPORT

表格 A-40 短選項與長選項、預設值與環境變數

選項名稱	長格式	短格式	預設值	環境變數
interactive	--interactive	-I	true	AS_AMDIN_INTERACTIVE
isconnectvalidaterequired	--isconnectvalidaterequired		false	AS_ADMIN_ISCONNECTVALIDATEREQUIRED
jdbcjndiname	--jdbcjndiname	-a		AS_ADMIN_JDBCJNDINAME
jndilookupname	--jndilookupname	-l		AS_ADMIN_JNDILOOKUPNAME
keepmanualchanges	--keepmanualchanges	-k	false	AS_ADMIN_KEEPMANUALCHANGES
loadorder	--loadorder			AS_ADMIN_LOADORDER
local	--local	-l	false	
logfile	--logfile			AS_ADMIN_LOGFILE
maxpoolsize	--maxpoolsize		32	AS_ADMIN_MAXPOOLSIZE
maxwait	--maxwait		6000	AS_ADMIN_MAXWAIT
mime	--mime			AS_ADMIN_MIME
mimefile	--mimefile			AS_ADMIN_MIMEFILE
monitor	--monitor	-m	false	AS_ADMIN_MONITOR
name	--name	-n		AS_ADMIN_NAME
nativelibpath	--nativelibpath			AS_ADMIN_NATIVELIBPATH
objtype	--objtype	-o		AS_ADMIN_OBJTYPE
password	--password	-w		AS_ADMIN_PASSWORD
poolresize	--poolresize		2	AS_ADMIN_POOLRESIZE
port	--port	-p	8000	AS_ADMIN_PORT
prefix	--prefix	-x		AS_ADMIN_PREFIX
printprompt	--printprompt	-P	true	AS_ADMIN_PROMPT
property	--property			AS_ADMIN_PROPERTY
securityenabled	--securityenabled			AS_ADMIN_SECURITYENABLED
servername	--servername			AS_ADMIN_SERVERNAME
ssl2ciphers	--ssl2ciphers			AS_ADMIN_SSL2CIPHERS
ssl2enabled	--ssl2enabled			AS_ADMIN_SSL2ENABLED

表格 A-40 短選項與長選項、預設值與環境變數

選項名稱	長格式	短格式	預設值	環境變數
ssl3enabled	--ssl3enabled			AS_ADMIN_SSL3ENABLED
ssl3tlsciphers	--ssl3tlsciphers			AS_ADMIN_SSL3TLSCIPHERS
state	--state			AS_ADMIN_STATE
steadypoolsize	--steadypoolsize	8		AS_ADMIN_STEADYPOOLSIZE
storeprotocol	--storeprotocol			AS_ADMIN_STOREPROTOCOL
storeprotocolclass	--storeprotocolclass			AS_ADMIN_STOREPROTOCOLCLASS
tlsenabled	--tlsenabled			AS_ADMIN_TLSENABLED
tlsrollbackenabled	--tlsrollbackenabled			AS_ADMIN_TLSROLLBACKENABLED
transprotocol	--transprotocol		smtp	AS_ADMIN_TRANSPROTOCOL
type	--type			S_ADMIN_TRANSPROTOCOLCLASS
upload	--upload	-U	true	AS_ADMIN_TYPE
url	--url			AS_ADMIN_URL
user	--user	-u		AS_ADMIN_USER
validationmethod	--validationmethod		auto-commit	AS_ADMIN_VALIDATIONMETHOD
validationtable	--validationtable			AS_ADMIN_VALIDATIONTABLE
version	--version	-v		AS_AMDIN_VERSION
virtualserver	--virtualserver			AS_ADMIN_VIRTUALSERVER

協力廠商版權備註

本產品包含由 RSA Security, Inc. 授權的程式碼。

本產品的某些部分是採用 ANTLR 開發的。ANTLR 1989-2000 由 jGuru.com 開發，位於 http://www.ANTLR.org 與 http://www.jGuru.com。

本產品包含透過 Sun 公用授權下之 Netbeans Project (位於 http://www.netbeans.org) 開發的軟體。如果該軟體可用，可於網站 www.netbeans.org 中找到。

本產品包含 Perl。如果 Perl 的複本可用，可於網站 http://public.ActiveState.com/gsar/APC/ 中找到複本。

本產品包含透過 Exolab Project (http://www.exolab.org) 開發的軟體。

本產品包含透過 DOM4J Project (http://dom4j.org/) 開發的軟體。

本產品包含由 Apache Foundation 開發的軟體。版權所有 (c) 1999-2001 The Apache Software Foundation。保留所有權利。

本產品包含由 The Regents of University of California 開發的軟體。版權所有 (c) 1991, 1993 The Regents of University of California。保留所有權利。

本產品包含由 International Business Machines Corporation (IBM) 開發的軟體。版權所有 (c) 1995-2001 International Business Machines Corporation 及其他廠商。保留所有權利。於 ICU 授權下提供了 IBM 程式碼。請參閱下面的內容。

ICU 授權 - ICU 1.8.1 及更高版本。

版權與許可權備註

版權所有 (c) 1995-2001 International Business Machines Corporation 及其他廠商。保留所有權利。

詞彙表

本字彙表提供用於描述 Sun ONE Application Server 部署和開發環境的一般術語定義。如需標準 J2EE 術語的詞彙表，請參閱位於以下網站的 J2EE 詞彙表：

```
http://java.sun.com/j2ee/glossary.html
```

ACL　存取控制清單。ACL 是指包含清單的文字檔，這些清單能夠識別出哪些使用者可以存取儲存在 Sun ONE Application Server 上的資源。**另請參閱**一般 ACL。

API　應用程式介面。電腦程式可以用來與解譯該 API 的軟體或硬體進行通訊的指令集。

applet　用 Java 語言編寫，並在 Web 瀏覽器中執行的小型應用程式。通常，applet 透過網頁呼叫，或嵌入網頁來提供特殊功能。相比之下，*Servlet* 則是在伺服器上執行的小型應用程式。

Bean 管理持續性　實體 Bean 的變數與資料儲存區之間的資料傳輸。資料存取邏輯通常由開發人員使用 Java 資料庫連結性 (JDBC) 或其他資料存取技術來提供。**另請參閱**容器管理持續性。

Bean 管理作業事件　在此處，企業 Bean 的作業事件分隔由開發人員有計劃地控制。**另請參閱**容器管理作業事件。

BLOB　二進制大型物件。用於儲存和擷取複雜物件欄位的資料類型。BLOB 是轉換爲大型位元組陣列的二進位物件或可串列轉換的物件 (例如圖片)，隨後，這些物件串列轉換爲容器管理持續性欄位。

BMP　**請參閱** Bean 管理持續性。

BMT　**請參閱** Bean 管理作業事件。

CA　請參閱證書授權中心或連接器架構。

CKL　洩露的密鑰清單。一個由證書授權中心發佈的清單，指出用戶端使用者或伺服器使用者不應該再信任的證書。在這種情況下，密鑰已經被洩露。**另請參閱** CRL。

CLI　指令行介面。一個可讓您在使用者提示下鍵入可執行指令的介面。**另請參閱**管理介面。

CMP　**請參閱**容器管理持續性。

CMR　**請參閱**容器管理關係。

CMT　**請參閱**容器管理作業事件。

cookie　一個小型的資訊集合，可以被傳輸到 Web 呼叫瀏覽器，然後從該瀏覽器的每次呼叫上進行擷取，這樣，伺服器便能識別來自於同一個用戶端的呼叫。Cookies 是領域特定的資訊集合，可以利用 Web 伺服器安全性功能，這與應用程式和伺服器之間進行其他資料交換利用的功能相同。

CORBA　共用物件請求代理程式架構。一個用於物件導向分散式運算的標準架構定義。

COSNaming 服務　一項基於 IIOP 的命名服務。

CosNaming 提供者　爲了支援全域 JNDI 名稱空間 (IIOP 應用程式用戶端可以存取)，Sun ONE Application Server 包含基於 J2EE 的 CosNaming 提供者，該程式支援連結 CORBA 參考 (遠端 EJB 參考)。

CRL　證書廢止清單。一個由證書授權中心發佈的清單，指出用戶端使用者或伺服器使用者不應該再信任的證書。在這種情況下，證書已經被廢止。**另請參閱** CKL。

DataSource 物件　DataSource 物件有一個特性集，用於識別並描述其所表示的真實資料來源。

DN　區別名稱。一種字串表示法，用於目錄伺服器中的項目名稱。

DN 屬性　區別名稱屬性。一個包含關聯使用者、群組或物件之識別資訊的文字字串。

DTD　文件類型定義。對一類 XML 檔案的結構與特性的描述。

EAR 檔案　企業歸檔檔案。一個含有 J2EE 應用程式的歸檔檔案。EAR 檔案的副檔名爲 `.ear`。**另請參閱** JAR 檔案。

EIS　企業資訊系統。可以被解譯爲封裝的企業應用程式、作業事件系統或使用者應用程式。經常被稱爲 EIS。EIS 的範例包括：R/3、PeopleSoft、Tuxedo 以及 CICS。

EJB QL　EJB 查詢語言。一種查詢語言，用於在實體 Bean 的網路 (由容器管理關係定義) 上進行導航。

EJB 技術　企業 Bean 是伺服器端的元件，用於封裝應用程式的企業邏輯。企業邏輯是用於完成應用程式目標的程式碼。例如，在庫存控制應用程式中，企業 Bean 可能使用 `checkInventoryLevel` 方法或 `orderProduct` 方法來執行企業邏輯。透過呼叫這些方法，遠端用戶端可以存取該應用程式提供的庫存服務。**另請參閱**容器、實體 Bean、訊息導引 bean、階段作業 Bean。

EJB 容器　**請參閱**容器。

ejbc 公用程式　企業 Bean 的編譯程式。它會檢查所有的 EJB 類別、介面是否符合 EJB 規格，並產生存根介面與骨架介面。

ERP　企業資源計劃。一個支援企業資源計劃的多模組軟體系統。通常，ERP 系統包括關聯式資料庫和應用程式，以管理採購、庫存、人事、客戶服務、運輸、財政計劃以及企業的其他重要方面。

Factory 類別　一種建立持續性管理程式的類別。**另請參閱**連線 Factory。

finder 方法　用戶端在全域可用目錄中查找 Bean 或 Bean 集合的方法。

FQDN　完整的領域名稱。系統的全名，包含其主機名稱與領域名稱。

HTML　超文件標示語言。一種編碼標示語言，用於建立可由 Web 瀏覽器顯示的文件。每個文字區塊均由指示該文字特性的程式碼包圍。

HTML 頁面　一個用 HTML 進行編碼的頁面，旨在使用 Web 瀏覽器顯示。

HTTP　超文件傳送協定。可以從遠端主機擷取超文件物件的網際網路協定。它以 TCP/IP 爲基礎。

HTTP Servlet　一個延伸 `javax.servlet.HttpServlet` 的 Servlet。這些 Servlet 具有對 HTTP 協定的內建支援。請對照一般 Servlet。

HTTPS　安全超文件傳輸協定。安全作業事件的 HTTP。

IDE　整合式開發環境。可讓您經由易於使用的單一介面建立、組合、佈署並除錯程式碼的軟體。

IDL　介面定義語言。一種用於定義遠端 CORBA 物件介面的語言。此介面獨立於作業系統與程式設計語言。描述遠端程序呼叫 (RPC) 的功能介面，這樣，編譯器可以產生代理與存根代碼，以對機器之間的參數進行編組。

IIOP　網際網路 ORB 交換協定。由 IIOP 上遠端方法調用 (RMI) 和共用物件請求代理程式架構 (CORBA) 共同使用的傳輸層級協定。

IIOP 偵聽程式　IIOP 偵聽程式是一個偵聽套接字，可以偵聽指定的連接埠，並經由基於用戶端應用程式的 CORBA 接受進來的連線。

IMAP　網際網路訊息存取協定。

IP 位址　識別 TCP/IP 網路上的電腦或其他裝置的結構化數字識別碼。IP 位址的格式爲 32 位元數字位址，該位址由四組透過小數點進分隔的數字組成。每組編號可以爲 0 到 255 之間的任意數字，例如，123.231.32.2 可以作爲 IP 位址。

J2EE　Java 2 企業版。用於開發、佈署基於 Web 的多層式企業應用程式的環境。J2EE 平台由一組服務、應用程式設計介面 (API)，以及提供應用程式開發功能的協定組成。

JAF　JavaBeans 啓動框架 (JAF) 將對 MIME 資料類型的支援整合至 Java 平台。請參閱 Mime 類型。

JAR 檔案　Java 歸檔檔案。用於將眾多檔案合併爲一個檔案的檔案。JAR 檔案的副檔名爲 `.jar`。

JAR 檔案合約　Java 歸檔檔案合約指定企業 Bean 套裝軟體中必須包含的資訊。

JAR 檔案格式　Java 歸檔檔案格式。一種與平台無關的檔案格式，可以將眾多檔案合併爲一個檔案。可以在 JAR 檔案中附帶多個 applet 及其必備元件 (類別檔案、影像、聲音以及其他資源檔案)，然後將其下載到單一 HTTP 作業事件中的瀏覽器。JAR 檔案格式還支援檔案壓縮與數位簽名。

Java IDL　Java 介面定義語言。使用 Java 程式設計語言編寫的 API，提供標準的相容性以及與共用物件請求代理程式架構 (CORBA) 的連接性。

JavaBean　一個與平台無關，並可重複使用的可攜式元件模型。

JavaMail 階段作業　應用程式用來與郵件儲存區進行互動的物件。應用程式碼使用 JNDI 服務來尋找使用 JNDI 名稱的 JavaMail 階段作業資源。

JAX-RPC　用於遠端程序呼叫 (基於 XML) 的 Java API。可讓開發人員依據基於 XML 的 RPC 協定，建立具有協作性的 Web 應用程式以及 Web 服務。

JAXM　用於 XML 訊息傳送的 Java API。可讓應用程式使用 SOAP 標準發送與接收文件導向 XML 訊息。這些訊息可以帶有附件、也可以不帶有附件。

JAXP　用於 XML 處理的 Java API。支援使用 DOM、SAX 與 XSLT 處理 XML 文件的 Java API。可讓應用程式剖析與轉變不受特定 XML 處理執行支配的 XML 文件。

JAXR　用於 XML 註冊的 Java API。提供統一的標準 Java API 來存取各種 XML 註冊。可讓使用者建立、佈署以及找到 Web 服務。

JDBC　Java 資料庫連結性。一個基於標準的類別與介面集合，其中類別與介面可讓開發人員建立可察覺資料的元件。JDBC 以與平台和供應商無關的方式執行各種方法來連線至資料來源，以及與資料來源進行互動。

JDBC 連線區　一個將 JDBC 資料來源特性 (用於指定到資料庫的連線) 與連線區特性進行結合的儲存區。

JDBC 資源　該資源用於將應用程式伺服器中正在執行的應用程式連線到使用現有 JDBC 連線區的資料庫。由一個 JNDI 名稱 (由應用程式使用) 和現有 JDBC 連線區的名稱組成。

JDK　Java 開發工具。該軟體包括開發人員爲 Java 2 Platform 之前的 Java 平台版本建立應用程式時所需的 API 和工具。**另請參閱** JDK。

JMS　Java 訊息服務。一個標準的介面與語義集，可定義 JMS 用戶端存取 JMS 訊息服務工具的方式。這些介面爲 Java 程式提供建立、發送、接收、讀取訊息的標準方式。

JMS 用戶端　通過使用 JMS 訊息服務交換訊息，與其他 JMS 用戶端進行互動的應用程式 (或軟體元件)。

JMS 目標　JMS 訊息服務中的實體目標，產生的訊息會首先發送給該目標進行路由，然後發送給使用者。實體目標由 JMS 管理物件識別與封裝，JMS 用戶端會使用該物件指定目標，從而爲該目標產生訊息和/或經由該目標使用訊息。

JMS 服務　爲 JMS 訊息傳送系統提供發送服務的軟體，包括至 JMS 用戶端的連線、訊息路由與發送、持續性、安全性以及記錄。訊息服務維護 JMS 用戶端向其發送訊息的實體目標，訊息會從該目標發送至最終用戶端。

JMS 訊息　由 JMS 用戶端使用的非同步請求、報告或事件。每個訊息具有一個標頭 (可在其中加入附加欄位) 一個內文。訊息標頭指定標準欄位與選擇性特性。訊息內文包含被傳輸的資料。

JMS 連線 Factory　JMS 管理物件，被 JMS 用戶端用來建立與 JMS 訊息服務的連線。

JMS 提供者　一個產品，用於執行訊息傳送系統 JMS 介面，並加入完整產品必備的管理與控制功能。

JMS 管理物件　由管理員建立的預先配置的 JMS 物件 (連線 Factory 或目標)，可由一個或多個 JMS 用戶端使用。使用管理物件可讓 JMS 用戶端獨立於提供者，即，將用戶端從提供者的專用方面隔離出來。這些物件由管理員置入 JNDI 名稱空間，並由 JMS 用戶端使用 JNDI 查找進行存取。

JNDI Java 命名與目錄介面。這是一個 Java 平台的標準延伸，爲使用 Java 技術的應用程式提供統一的介面，以用於多重命名與企業中的目錄服務。作爲 Java Enterprise API 集合的一部分，JNDI 可讓您無縫連接至不同的企業命名與目錄服務。

JNDI 名稱 一個用於存取在 JNDI 命名服務中已註冊資源的名稱。

JRE Java 執行環境。Java 開發工具 (JDK) 的子集，由 Java 虛擬機器、Java 核心類別以及支援檔案組成，爲使用 Java 程式設計語言編寫的應用程式提供執行期間的支援。**另請參閱** JDK。

JSP JavaServer 頁面。一個結合使用 HTML 或 XML 標記、JSP 標記以及 Java 程式碼編寫的文字頁面。JSP 結合了標準瀏覽器頁面的佈局功能與程式設計語言的功能。

jspc 公用程式 JSP 的編譯程式。它會檢查所有的 JSP 是否符合 JSP 規格。

JTA Java 作業事件 API。可讓應用程式與 J2EE 伺服器存取作業事件的 API。

JTS Java 作業事件服務。處理作業事件的 Java 服務。

LDAP 輕型目錄存取協定。LDAP 是一個在 TCP/IP 上執行的開放式目錄存取協定。它可以調整到全域大小，包含上百萬個項目。使用 Sun ONE Directory Server (提供的一個 LDAP 伺服器)，可以將所有企業資訊儲存在單一的中央目錄資訊儲存庫，任何應用程式伺服器均可以透過網路存取該庫。

LDIF LDAP 資料交換格式。用於以文字形式表示 Sun ONE Directory Server 項目的格式。

MDB **請參閱**訊息導引 bean。

MIME 資料類型 MIME (多用途網際網路郵件延伸標準) 類型可控制您系統所支援的多媒體檔案類型。

NTV 名稱、類型、值。

O/R 對映工具 物件至關聯式 [資料庫] 工具。Sun ONE Application Server 管理介面中可以爲實體 Bean 建立 XML 部署描述元的對映工具。

POP3 郵局協定

QOS QOS (服務品質) 是指您爲伺服器實例或虛擬伺服器設定的效能限定。例如，如果您是 ISP，可能需要依據所提供的頻寬，針對虛擬伺服器收取不同數額的費用。您可以限定兩個方面：頻寬量與連線數。

RAR 檔案 資源歸檔檔案。一個含有資源介面的 JAR 歸檔檔案。

RDB　關聯式資料庫。

RDBMS　關聯式資料庫管理系統。

ResultSet　一個執行 `java.sql.ResultSet` 介面的物件。`ResultSets` 用於封裝經由資料庫或其他表格資料來源所擷取的列集。

RMI　遠端方法調用。標準的 Java API 集，可讓開發人員撰寫遠端介面來將物件傳輸至遠端程序。

RMIC　遠端方法調用編譯程式。

RowSet　一個物件，用於封裝經由資料庫或其他表格資料來源所擷取的列集。`RowSet` 延伸了 `java.sql.ResultSet` 介面，可讓 `ResultSet` 作爲 JavaBeans 元件發揮作用。

RPC　遠端程序呼叫。一種存取遠端物件或服務的機制。

SAF　伺服器應用程式功能。一項參與請求處理和其他伺服器活動的功能。

Servlet　一個 `Servlet` 類別實例。Servlet 是在伺服器上執行，並可重複使用的應用程式。在 Sun ONE Application Server 中，Servlet 透過執行表示邏輯、呼叫企業邏輯以及呼叫或執行表示佈局，從而作爲應用程式中每次互動的中央派送程式發揮作用。

Servlet 引擎　一個處理所有 Servlet 媒介功能的內部物件。爲 Servlet 共同提供服務的程序集，包括創設與執行。

Servlet 執行程式　Servlet 引擎的一部分，可以透過請求物件與回應物件呼叫 Servlet。**請參閱** Servlet 引擎。

SMTP　簡單郵遞傳輸協定

SNMP　SNMP (簡單網路管理協定) 是一項用於交換網路活動相關資料的協定。透過 SNMP，資料在管理裝置與網路管理站 (NMS) 之間傳輸。管理裝置是指所有執行 SNMP 的裝置：主機、路由器、Web 伺服器以及網路上的其他伺服器。NMS 是用於遠端管理網路的機器。

SOAP　簡單物件存取協定 (SOAP) 結合使用基於 XML 的資料結構與超文件傳輸協定 (HTTP) 來定義一種標準化方法，用於呼叫物件中的方法 (分散在網路上的各種作業環境中)。

SQL　結構化查詢語言。一種在關聯式資料庫應用程式中普遍使用的語言。SQL2 與 SQL3 表明語言的版本。

SSL　安全套接字層。用於提供網際網路上安全通訊的協定。

Sun ONE Directory Server　輕型目錄存取協定 (LDAP) 的 Sun ONE 版本。Sun ONE Application Server 的每個實例都使用 Sun ONE Directory Server 儲存共用伺服器資訊，包括有關使用者與群組的資訊。**另請參閱** LDAP。

Sun ONE Message Queue　Sun ONE 企業訊息傳送系統，執行 Java 訊息服務 (JMS) 開放式標準：這是 JMS 提供者。

TLS　傳輸層安全性。在傳輸層提供加密與驗證的協定，這樣，資料便可通過安全的通道流動，而無需對用戶端與伺服器應用程式做重大變更。

UDDI　通用說明、開發與整合。針對開發與整合，提供 Web 服務的全球性註冊。

URI　通用資源識別碼。描述領域中的特定資源。在本機作爲基本目錄的子集進行描述，因此 `/ham/burger` 爲基本目錄，URI 指定了 `toppings/cheese.html`。相應的 URL 爲 `http://domain:port/toppings/cheese.html`。

URL　統一資源位址。一個唯一識別 HTML 頁面或其他資源的位址。Web 瀏覽器使用 URL 指定要顯示的頁面。URL 描述傳輸協定 (例如，HTTP、FTP)、領域 (例如，`www.my-domain.com`)，並隨意描述 URI。

WAR 檔案　Web 歸檔檔案。一個含有 Web 模組的 Java 歸檔檔案。WAR 檔案的副檔名爲：`.war`。

Web 伺服器　儲存與管理 HTML 頁面和 Web 應用程式的主機，但不是管理全部的 J2EE 應用程式。Web 伺服器經由 Web 瀏覽器回應使用者的請求。

Web 伺服器外掛程式　Web 伺服器外掛程式是 HTTP 反向代理外掛程式，可讓您指導 Sun One Web Server 或 Sun ONE Application Server，將特定的 HTTP 請求轉寄到另一個伺服器。

Web 快取　一項 Sun ONE Application Server 功能，可讓 Servlet 或 JSP 在特定的時間內快取其結果，以便提昇效能。爲持續時間內對 Servlet 或 JSP 的後續呼叫提供快取結果，從而該 Servlet 或 JSP 不必再次執行。

Web 服務　通過 Web 提供的服務。一個自我包含、自我描述的模組化應用程式，可在網際網路上或企業網路上經由一個系統接受請求，處理請求，然後傳回一個回應。

Web 容器　**請參閱**容器。

web 連接器外掛程式　Web 伺服器的延伸功能，可讓該伺服器與 Sun ONE Application Server 進行通訊。

Web 模組　一個單獨部署的 Web 應用程式。**請參閱** Web 應用程式。

Web 應用程式　一個 Servlet、JavaServer 頁面、HTML 文件以及其他 Web 資源的集合，可以包括影像檔、壓縮的歸檔檔案以及其他資料。Web 應用程式可以封裝到歸檔檔案中 (WAR 檔案)，或存在於開放式目錄結構中。Sun ONE Application Server 也支援一些非 Java Web 應用程式技術，例如 SHTML 與 CGI。

WSDL　Web 服務描述語言。一種基於 XML 的語言，用於以標準的方法定義 Web 服務。其從實質上描述 Web 服務的三個基本特性：Web 服務的定義、Web 服務的存取方式以及 Web 服務的位置。

XA 協定　分散式作業事件的資料庫工業標準協定。

XML　可延伸式標示語言。一種使用 HTML 樣式標記的語言，用於識別文件中使用的資訊類型，並格式化文件。

一般 ACL　Sun ONE Directory Server 中的具名清單，可將使用者或群組與一項或多項許可權關聯起來。可以定義與隨意存取該清單以記錄任何許可權集。

一般 Servlet　一個延伸 `javax.servlet.GenericServlet` 的 Servlet。一般 Servlet 與協定無關，表示其本身並不支援 HTTP 或任何其他傳輸協定。請對照 HTTP Servlet。

已儲存的程序　一組用 SQL 撰寫的敘述，儲存在資料庫中。您可以使用已儲存的程序執行任意類型的資料庫作業，例如修改、插入或刪除記錄。使用已儲存的程序會透過降低在網路上發送的資訊數量來提昇效能。

元素　大型集合中的成員，例如，陣列中的資料單元，或邏輯元素。在 XML 檔案中，其爲基本的結構單元。XML 元素包含子元素或資料，也可能包含屬性。

公用密碼學　一種每個使用者可具有一個公用鍵值和一個專用鍵值的密碼學。訊息使用接收者的公用鍵值加密後發送，然後接收者再使用其專用鍵值進行解密。使用這種方法，專用鍵值便永遠不會洩露給使用者之外的任何人。

分散式作業事件　單一的作業事件，可套用於駐留在各個伺服器內的多個異質資料庫。

文件根　文件根 (有時稱爲主文件目錄) 是一個中央目錄，包含要用於遠端用戶端的所有虛擬伺服器檔案。

主機 IP 認證　一種安全性機制，使用特定的電腦規定僅有用戶端才能使用網站上的管理伺服器，或者檔案與目錄，從而限制對它們的存取。

主鍵　可讓用戶端尋找特定實體 Bean 的唯一識別碼。

主鍵類別名稱　一個指定 Bean 主鍵完全合格類別名稱的變數。用於 JNDI 查找。

主題　一個由管理員建立，用於執行發佈/訂購發送模型的物件。主題可以被視爲內容階層中的節點，負責收集與散發發送至其的訊息。透過將主題作爲媒介使用，訊息的發佈者與訂購者保持分離狀態。

代理程式　管理 JMS 的訊息路由、發送、持續性、安全性以及記錄的 Sun ONE Message Queue 實體，其所提供的介面允許管理員監視並調整效能與資源的使用。

加密　轉變資訊，使得僅有預定接受者才能理解它的程序。

可分配階段作業　可在叢集內所有伺服器中進行分配的使用者階段作業。

可串列轉換物件　一個可以被解構或重新建構的物件，這樣其便可在多個伺服器之間儲存或分配。

可呼叫敘述　一種封裝各個資料庫中資料庫程序或函式呼叫的類別，這些資料庫支援從已儲存的程序中傳回結果集。

可信任的資料庫　包含公用鍵值與專用鍵值的安全性檔案，也稱爲鍵對檔案。

可重複使用的元件　爲了在多個容量中使用而建立的元件，例如，透過多個資源或應用程式。

可插接式認證　允許 J2EE 應用程式經由 J2SE 平台使用 Java 認證與授權服務 (JAAS) 功能。開發人員可插入其自己的認證機制。

外部 JDNI 資源　允許 JNDI 服務充當連接至遠端 JNDI 伺服器的橋接器。

本地介面　一種方法定義機制，所定義的方法可讓用戶端建立並移除企業 Bean。

本機介面　一個爲用戶端機制 (該用戶端位於同一個 Java 虛擬機器 [JVM] 內) 提供階段作業或實體 Bean 來存取該 Bean 的介面。

本機作業事件　一個資料庫中特有的，並在單一程序中受到限制的作業事件。本機作業事件總是逆單一後端發揮作用。通常，使用 JDBC API 區分本機作業事件。**另請參閱**全域作業事件。

本機階段作業　僅有一個伺服器可視的使用者階段作業。

本機資料庫連線　本機連線中的作業事件環境對於目前程序和目前資料來源而言爲本機環境，並非分散式跨程序或跨資料來源。

永久性狀態　表示物件的狀態保留在永久性儲存體中，通常是指資料庫。

生命週期事件　伺服器生命週期的某個階段，例如啓動或關機。

生命週期模組　一個用於偵聽並執行其工作的模組，回應伺服器生命週期中的事件。

用戶端合約　確定用戶端與 EJB 容器之間通訊規則的合約，爲使用企業 Bean 的應用程式建立了統一開發模型，並通過標準化與用戶端之間的關係，保證能夠更多次地重複使用 Bean。

用戶端認證　認證用戶端證書的程序，其方法爲使用密碼檢驗證書簽名與通向信賴 CA 清單上 CA 的證書鏈。另請參閱認證、證書授權中心。

目標資源　表示主題或佇列目標的物件。由應用程式使用，讀取/寫入佇列或發佈/訂購主題。應用程式碼使用 JNDI 服務來尋找使用 JNDI 名稱的 JMS 資源。

目錄伺服器　請參閱 Sun ONE Directory Server。

交談式狀態　在此處，物件的狀態隨著與同一個用戶端重複地互動而不斷變更。**另請參閱**永久性狀態。

企業邏輯　用於執行應用程式之基本企業規則的程式碼，而不是資料整合或表示邏輯。

全域作業事件　一個由作業事件管理者管理與協調的作業事件，可以跨越多個資料庫與程序。作業事件管理者通常使用 XA 協定與資料庫後端進行互動。**請參閱**本機作業事件。

全域資料庫連線　一個可用於多重元件的資料庫連線。需要一個資源管理者。

列　表格內，每行包含值的單一資料記錄。

回應物件　一個參考呼叫用戶端的物件，提供爲用戶端產生輸出的方法。

存取控制　透過控制可存取 Sun ONE Application Server 的使用者或其他事物來對其進行保護的方法。

安全性　確保只有取得授權的用戶端才能存取應用程式資源的審查機制。

安全套接字層　**請參閱** SSL。

有狀態階段作業 Bean　一個透過特定用戶端表示階段作業的階段作業 Bean，其會自動維護跨用戶端呼叫方法的狀態。

佇列　一個由管理員建立，用於執行點對點發送模型的物件。即使當使用訊息的用戶端不在作用中，佇列也總可以接收訊息。佇列作爲產生器與使用者之間的中間接收位置發揮作用。

伺服器實例　一個 Sun ONE Application Server 在同一台機器、同一個安裝版本中可以含有多個實例。每個實例都有其自己的目錄結構、配置以及部署的應用程式。每個實例也可含有多個虛擬伺服器。**另請參閱**虛擬伺服器。

佈署　將應用程式所需的檔案分配給應用程式伺服器，以便使應用程式可以在應用程式伺服器上執行的程序。**另請參閱**組合。

佈署描述元　提供有每個模組和應用程式的 XML 檔案，用於描述模組和應用程式的佈署方式。佈署描述元使用佈署工具，透過特定的容器選項佈署模組或應用程式，並描述佈署程式必須解決的特定配置需求。

快取列集　`CachedRowSet` 物件允許您從資料來源中擷取資料，然後在檢查、修改資料的時候將其從資料來源中分離出來。快取列集會記錄已擷取的原始資料以及應用程式對資料所作的任何變更。如果應用程式嘗試更新原始資料來源，列集會重新連線至資料來源，僅有那些已經變更的列才被合併回資料庫。

快取控制指令　快取控制指令是 Sun ONE Application Server 使用的一種控制方法，用來控制代理伺服器快取的資訊。使用快取控制指令，您可以置換代理伺服器的預設快取法，以防止快取靈敏度高的資訊 (也許稍後會被擷取)。若要這些指令工作，代理伺服器必須遵守 HTTP 1.1。

改版　**請參閱**動態重新載入。

系統管理員　管理 Sun ONE Application Server 軟體並部署 Sun ONE Application Server 應用程式的人員。

角色　應用程式中按照主題功能進行的分組，由已部署的環境中的一個或多個群組表示。**另請參閱**使用者、群組。

防火牆　允許網路管理員限制網路上的資訊流以增強安全性的電子邊界。

事件　一個具名動作，可以觸發模組或應用程式的回應。

使用者　使用應用程式的人員。依照一般規則，使用者由使用者名稱、密碼、可讓應用程式辨識用戶端的屬性集組成。**另請參閱**群組、角色。

使用者階段作業　由伺服器追蹤的一系列使用者應用程式互動。階段作業維護使用者狀態、永久性物件以及身份認證。

委託人　認證後，指定給實體的身份。

物件持續性　**請參閱**持續性。

狀態　**1.** 任意給定時間內實體的環境或狀況。**2.** 一種分散式資料儲存機制，透過該機制，您可以使用 Sun ONE Application Server 功能介面 `IState2` 儲存應用程式的狀態。**另請參閱**交談式狀態、永久性狀態。

表示佈局　網頁內容格式。

表示邏輯　使用應用程式建立頁面的活動，包括處理請求、產生回應內容、格式化用戶端頁面。通常由 Web 應用程式處理。

表面　在此處，特定的應用程式有狀態階段作業 Bean 用於管理各種企業 JavaBeans (EJBs)。

表格　資料庫內用列和行表示的相關資料的具名群組。

表格動作處理程式　一個在 Servlet 或應用程式邏輯中特別定義的方法，可基於表格上的具名按鈕執行動作。

非同步通訊　一種通訊模式，在該模式下，訊息發送者無需等到發送方法傳回，便可繼續進行其他的工作。

宣告性安全性　宣告元件配置檔案中的安全性特性，並允許元件的容器 (例如，Bean 的容器或 Servlet 引擎) 隱式地管理安全性。此類安全性不需要有計劃地控制。其與程式化安全性正好相反。**請參閱**容器管理持續性。

宣告性作業事件　請參閱容器管理作業事件。

封裝　用於本土化模組中的內容。由於物件封裝了資料與執行，物件的使用者便可以將物件作爲提供服務的黑箱進行檢視。可以加入、刪除或變更實例變數和方法，但是，如果物件所提供的服務仍保持不變，則使用物件的程式碼可以繼續使用它，而不必重新寫入該物件。

建立方法　一種在建立企業 Bean 時自訂該 Bean 的方法。

持續性　對於企業 Bean，其指在實體 Bean 的實例變數與基礎資料庫之間傳輸實體 Bean 狀態的協定。其與暫態性正好相反。對於階段作業，其指階段作業儲存機制。

持續性管理程式　負責安裝在容器中實體 Bean 之持續性的實體。

故障恢復　Bean 可以從伺服器當機中明顯恢復的程序。

流線　一種管理資料如何透過 HTTP 進行通訊的技術。當結果成爲流線型時，便可以立刻使用資料的第一部分。當結果未形成流線型時，必須在使用結果的任意部分之前，首先接收整個結果。流線技術可讓大量的資料以更有效的方式傳回，提昇應用程式的感知效能。

套裝軟體　儲存於共用目錄中的相關類別的集合。它們通常整齊地共同封裝在一個 Java 歸檔檔案 (JAR) 中。**另請參閱**組合、佈署。

容器　一個爲特定類型的 J2EE 元件提供生命週期管理、安全性、佈署以及執行期間服務的實體。Sun ONE Application Server 提供 Web 與 EJB 容器，並支援應用程式用戶端容器。**另請參閱**元件。

容器管理持續性　在此處，EJB 容器負責實體 Bean 的持續性。實體 Bean 的變數與資料儲存區之間的資料傳輸，其中，資料存取邏輯由 Sun ONE Application Server 提供。**另請參閱** Bean 管理持續性。

容器管理作業事件　在此處，企業 Bean 的作業事件分隔以宣告的方式指定，並由 EJB 容器自動控制。**另請參閱** Bean 管理作業事件。

容器管理關係　一對類別中欄位之間的關係，其中，關係一側的作業會影響另一側。

特性　定義應用程式元件行爲的單一屬性。在 `server.xml` 檔案中，特性便是含有名稱/數值對的元素。

訊息傳送　企業應用程式使用的非同步請求、報告或事件系統，可讓松藕合的應用程式可靠、安全地傳輸資訊。

訊息導引 bean　作爲非同步訊息使用者的企業 Bean。訊息導引 Bean 對於特定的用戶端而言，不具有狀態，但是其實例變數可能含有跨用戶端訊息處理的狀態，包括一個開放式資料庫連線以及一個 EJB 物件的物件參考。用戶端透過將訊息發送至訊息導引 Bean 偵聽的目標，可以存取訊息導引 Bean。

配置　調整伺服器或爲元件提供複合資料的程序。通常，特定元件的配置保留在元件的部署描述元檔案中。**另請參閱**管理伺服器、佈署描述元。

偵聽程式　一個用郵寄物件註冊的類別，可在事件發生的時候指出處理的方法。

動態重新佈署　在不重新啓動伺服器的情況下重新佈署元件的程序。

動態重新載入　在不重新啓動伺服器的情況下更新與重新載入元件的程序。依預設，Servlet、JavaServer 頁面 (JSP) 以及企業 Bean 元件可以被動態地重新載入。也稱爲改版。

區別名稱　**請參閱** DN、DN 屬性。

參數　一個從用戶端發送的名稱/數值對，包括表格欄位資料，HTTP 標頭資訊等等，封裝在請求物件中。請與屬性對照。更廣義地講，是一個 Java 方法或資料庫預備指令的引數。

唯讀 Bean　一個永遠不被 EJB 用戶端修改的實體 Bean。**另請參閱**實體 Bean。

執行系統　執行程式的軟體環境。執行系統包括載入用 Java 程式設計語言撰寫的程式，動態地連結本端方法，管理記憶體，以及處理異常時所需的全部程式碼。包括 Java 虛擬機器的執行，可以爲 Java 解譯程式。

執行緒　程序中的執行序列。一個程序可同時執行多個執行緒，這種情況稱爲多執行緒。如果程序依次執行每個執行緒，稱爲單一執行緒。

密碼　一種加密演算法 (一種數學函式)，用於加密或解密。

專用鍵值　**請參閱**公用密碼學。

控制描述元　企業 Bean 配置項目集，可讓您指定選擇性的個別特性置換，以用於 Bean 方法、企業 Bean 作業事件以及安全性特性。

控點　一個識別企業 Bean 的物件。用戶端可以串列化控點，稍後對其進行串列反轉換以取得 Bean 的參考。

授權　決定方法或資源存取的程序。J2EE 平台中的授權取決於通過認證與請求關聯的使用者是否以給定的安全性角色執行作業。例如，人力資源應用程式可以授權管理者檢視所有員工的個人資訊，但是每個員工僅能檢視自己的個人資訊。

授權　一種物件導向技術，將物件組合作爲執行策略。一個對作業結果負責的物件將授權執行另一個物件，即其受權物件。例如，類別載入器經常將某些類別的載入授權給其父元件。

啟動　將企業 Bean 的狀態從輔助儲存體傳輸到記憶體的程序。

啟發式決策　特定作業事件所使用的作業事件模式。作業事件必須進行「確定」或「回轉」。

作業事件　一個資料庫指令集，作爲一個群組，可以成功，也可以失敗。只有所包含的全部指令成功，整個作業事件才會成功。

作業事件恢復　分散式作業事件的自動或手動恢復。

作業事件隔離層　決定資料庫上並存作業事件相互可見的程度。

作業事件管理者　一個控制全域作業事件的物件，通常使用 XA 協定。**請參閱**全域作業事件。

作業事件環境　作業事件的範圍，本機或全域。**請參閱**本機作業事件、全域作業事件。

作業事件屬性　作業事件屬性控制作業事件的範圍。

元件　一個 Web 應用程式、企業 Bean、訊息導引 Bean、應用程式用戶端或連接器。**另請參閱**應用程式、模組。

元件合約　用於建立企業 Bean 與其容器之間關係的合約。

組合　將分散的元件結合爲一個可佈署單元的程序。**另請參閱**佈署。

組配　將一個元件與其相關的元件放置於同一個記憶體空間，以避免遠端程序呼叫，並提昇效能。

許可權　授與或不授與使用者或群組的權限集。**另請參閱** ACL。

連接器　一種標準的容器延伸機制，可用於連接至 EIS。連接器專用於 EIS，由資源介面和用於 EIS 連接性的應用程式開發工具組成。資源介面可以插入至容器，因爲其支援連接器架構中定義的系統層級合約。

連接器架構　J2EE 應用程式與 EIS 的整合架構。此架構中包含兩個部分：EIS 供應商提供的資源介面與可插入此資源介面的 J2EE 伺服器。該架構定義了合約集，要求資源介面必須支援插入 J2EE 伺服器，例如，作業事件、安全性與資源管理。

連線 Factory　一個能產生連線物件的物件，所產生的物件可讓 J2EE 元件存取資源。用於建立 JMS 連線 (TopicConnection 或 QueueConnection)，這些連線可讓應用程式碼利用所提供的 JMS 執行。應用程式碼使用 JNDI 服務來尋找使用 JNDI 名稱的連線 Factory 物件。

連線區　可讓您透過快取或重複使用實體連線來高效地存取資料庫，從而避免連線耗用時間，並允許在大量的執行緒之間共用少量的連線。**另請參閱** JDBC 連線區。

粘滯 cookie　一個傳回至用戶端以使自己總連線至同一個伺服器程序的 cookie。**另請參閱**階段作業 cookie。

粘滯載入平衡　一種載入平衡方法，其中，初始用戶端請求被平衡載入，但隨後的請求被導向至與初始請求相同的程序。**請參閱**載入平衡。

備份儲存區　一個資料儲存庫，通常是指檔案系統或資料庫。備份儲存區可以由背景執行緒 (清掃器執行緒) 進行監視，以便移除多餘的項目。

單一登入　使用者的認證狀態可在單一虛擬伺服器實例中的多個 J2EE 應用程式進行共用的情形。

無狀態階段作業 Bean　表示無狀態服務的階段作業 Bean。無狀態階段作業 Bean 完全是暫時的，其會封裝特定用戶端在特定時間段內所需的暫存企業邏輯片段。

發佈/訂購發送模型　發佈者與訂購者通常使用匿名，可以動態地發佈或訂購主題。系統將來自於該主題的多個出版社的訊息發行給該主題的多個訂閱者。

程式化安全性　明確地以程式碼控制安全性的程序，而不允許元件的容器 (例如，Bean 的容器或 Servlet 引擎) 來處理它。其與宣告性安全性正好相反。

程式設計者區分作業事件　**請參閱** Bean 管理作業事件。

程序　作用中程式的執行順序。該程序由一個或多個執行緒組成。

虛擬伺服器　一個虛擬 Web 伺服器，用於伺服針對特定 URL 的內容。多個虛擬伺服器可以使用相同或不同的主機名稱、連接埠號或 IP 位址伺服內容。HTTP 服務可根據 URL，將進來的 Web 請求導向其他虛擬伺服器。也稱爲虛擬主機。可將 Web 應用程式指定給特定的虛擬伺服器。一個伺服器實例可以具有多個虛擬伺服器。**另請參閱**伺服器實例。

鈍化　在不銷毀 Bean 的情況下，從記憶體釋放 Bean 資源的方法。使用這種方法，Bean 可以保持永久性，並且可在不耗用實例化時間的情況下重新被呼叫。

階段作業　Servlet 追蹤使用者與 Web 應用程式 (跨多個 HTTP 請求) 之間的互動時所使用的物件。

階段作業 Bean　一個由用戶端建立的企業 Bean，通常只在單一主從式階段作業發生時存在。階段作業 Bean 針對用戶端執行作業，例如計算或存取其他 EJB。雖然階段作業 Bean 是可作業事件的，但是如果系統當機，其便不能恢復。階段作業 Bean 物件可以是無狀態的 (與特定用戶端無關聯)，也可以是有狀態的 (與特定用戶端相關聯)，即，其可以維護跨方法與作業事件的交談式狀態。**另請參閱**有狀態階段作業 Bean、無狀態階段作業 Bean。

階段作業 cookie　傳回至用戶端 (包含使用者階段作業識別碼) 的 cookie。**另請參閱**粘滯 cookie。

階段作業逾時　一段指定的持續時間，超過該時間，Sun ONE Application Server 便可令使用者階段作業無效。**請參閱**階段作業。

匯集　提供一些預先配置的資源以提昇效能的程序。如果匯集了資源，元件便可以經由儲存區使用現有的實例，而不必創設新的實例。在 Sun ONE Application Server 中，資料庫連線、Servlet 實例以及企業 Bean 實例均可被匯集。

群組　一組以某種方式關聯的使用者。群組成員關係通常由本機系統管理員維護。**請參閱**使用者、角色。

解密　變換加密資訊，以便其重新變得可以理解的程序。

資料存取邏輯　與資料來源互動的企業邏輯。

資料來源　一個資料來源控點，例如資料庫。資料來源透過 iPlanet Application Server 註冊，然後有計劃地擷取，以便建立連線，並與資料來源互動。資料來源的定義指定了連線至資料來源的方式。

資料庫　一個用於關聯式資料庫管理系統 (RDBMS) 的一般術語。它是一個套裝軟體，可用於建立並操控大量有組織的相關資料。

資料庫連線　資料庫連線是與資料庫或其他資料來源之間的通訊連結。元件可以同時操控數個資料庫連線以存取資料。

資源參考　部署描述元中的一個元素，可以爲資源識別元件的編碼名稱。

資源管理者　一個作爲資源 (例如資料庫或訊息代理程式) 與資源用戶端 (例如 Sun ONE Application Server 程序) 之間便利工具的物件。控制可在全域中獲得的資料來源。

載入平衡　在一個叢集內的多個伺服器之間平均分佈使用者載入的技術。**請參閱**粘滯載入平衡。

隔離層　**請參閱**作業事件隔離層。

電子商業　電子商業。一個描述企業在網際網路上進行經營的術語。

預備指令　一個預先編譯以使重複執行作業更加有效的資料庫指令 (使用 SQL 編寫)。預備指令可以含有參數。一個預備敘述可以含有一個或多個預備指令。

預備敘述　用於封裝 `QUERY`、`UPDATE` 或 `INSERT` 敘述的類別，該敘述可以重複使用以擷取資料。一個預備敘述可以含有一個或多個預備指令。

實體 Bean　一個與實體資料相關的企業 Bean，如資料庫中的一列。實體 Bean 可以長期存在，因爲其與永久性資料連接在一處。實體 Bean 總是可作業事件的，並且可察覺多使用者。**請參閱**訊息導引 bean、唯讀 Bean、階段作業 Bean。

對映　將一個物件導向模型與相關資料模型進行連結的功能，通常爲關聯式資料庫的綱目。將綱目轉換爲其他結構的程序。也指將使用者對映於安全性角色。

摘要認證　一種允許使用者依據使用者名稱與密碼進行認證的認證方式，無需將使用者名稱與密碼作爲清除文字發送。

管理介面　用於配置和管理 Sun ONE Application Server 的基於瀏覽器的表格集。**另請參閱** CLI。

管理伺服器　一個專門提供 Sun ONE Application Server 管理功能的應用程式伺服器實例，包含部署、基於瀏覽器的管理、經由指令行介面 (CLI) 與整合式開發環境 (IDE) 存取。

管理資訊庫 (MIB)　一種樹狀結構，用於定義 SNMP 主代理程式可存取的變數。使用 MIB，可存取 HTTP 伺服器的網路配置、狀態以及統計資料。使用 SNMP，您可以經由網路管理工作站 (NMS) 檢視該資訊。**另請參閱**網路管理站 (NMS) 與 SNMP。

管理領域　多管理領域是 Sun ONE Application Server 的一個特徵，其允許不同的管理使用者建立並管理他們各自的領域。一個領域就是一個實例集，其是使用單一系統內已安裝的二進位制共用集合建立的。

網路管理站 (NMS)　用於遠端管理特定網路的機器。通常，NMS 軟體會提供圖形來顯示收集到的資料，或使用此資料確定伺服器在特定的容許度下作業。**另請參閱** SNMP。

綱目　基礎資料庫的結構，包括表格名稱、欄名稱與欄類型、索引資訊以及關係 (主鍵值與外來鍵值) 資訊。

認證　一個實體 (例如使用者) 向另一個實體 (例如應用程式) 證明其正在以特定身份 (使用者的安全性身份) 進行作業的程序。Sun ONE Application Server 支援基於表格的基本 SSL 相互認證。**另請參閱**用戶端認證、摘要認證、主機 IP 認證、可插接式認證。

遠端介面　用於企業 JavaBean 的一個或兩個介面。遠端介面定義了可由用戶端呼叫的企業方法。

領域註冊　網域註冊是包含網域特定資訊的單一資料結構，用於在安裝 Sun ONE Application Server 時建立並配置的所有網域，例如網域名稱、網域位置、網域連接埠、網域主機。

數位簽名　一種對訊息和簽名者進行認證的電子安全性機制。

暫態性　一個在資源不使用時釋放資源的協定。其與持續性正好相反。

模組　一個在應用程式之外已個別部署的 Web 應用程式、企業 Bean、訊息導引 Bean、應用程式用戶端或連接器。**另請參閱**應用程式、元件、生命週期模組。

確定　透過將所需的指令發送至資料庫來完成作業事件。**請參閱**轉返、作業事件。

稽核　通常是指在錯誤或安全性被破壞的情形下，用於記錄重要事件以進行後續檢查的方法。

範圍　在該範圍中，可以定義一般安全性策略，並且安全性服務管理員可以增強該策略安全性。在 J2EE 規格中也稱爲**安全性策略領域**或**安全性領域**。

複合資料　有關元件的資訊，例如該元件的名稱，及其行爲規格。

請求物件　一個含有用戶端所產生的頁面與階段作業資料的物件，作爲輸入參數傳送至 Servlet 或 JavaServer 頁面 (JSP)。

應用程式　一組封裝在 `.ear` 檔案中，且具有 J2EE 應用程式佈署描述元的元件。**另請參閱**元件、模組。

應用程式用戶端容器　**請參閱**容器。

應用程式伺服器　一個執行企業應用程式的可靠又安全的可調式軟體平台。應用程式伺服器通常爲各種應用程式提供高階服務，例如，元件的生命週期、位置、分配以及作業事件資源的存取。

應用程式層　J2EE 應用程式的一個概念部分。**用戶端層**：使用者介面 (UI)。終端使用者與用戶端軟體 (例如 Web 瀏覽器) 通過互動使用應用程式。**伺服器層**：組成應用程式的企業邏輯與表示邏輯，在應用程式的元件中定義。**資料層**：資料存取邏輯，可讓您的應用程式與資料來源進行互動。

檔案快取記憶體　檔案快取記憶體包含有關檔案與靜態檔案內容的資訊。依預設，檔案快取記憶體總處於開啓狀態。

鍵對檔案　**請參閱**可信任的資料庫。

顆粒性層級　將應用程式分爲多個片段的方法。**高階顆粒性**表示應用程式被分爲許多更加細小、更加嚴格定義的企業 JavaBeans (EJBs)。**低階顆粒性**表示應用程式被分爲較少的片段，產生更大的程式。

點對點發送模型　產生器將訊息發送至特定的佇列，使用者然後從接收訊息的佇列中擷取訊息。一則訊息只能發送至一個訊息使用者。

叢集　經由高速網路連線的協同作業的一組電腦，就如同它們是一台具有多個 CPU 的機器。如果叢集中的某個伺服器無法使用，可以將其服務故障恢復至正常作業的伺服器。**另請參閱**故障恢復。

轉返　取消作業事件。

證書　指定個人、公司或其他實體名稱，並驗證該實體證書中公用鍵值的數位資料。用戶端與伺服器均可具有證書。

證書授權中心　通過網際網路出售憑證的公司，或是針對企業網路或企業間網路負責發行憑證的部門。

類別路徑　一個路徑，用於識別儲存 Java 類別的目錄與 JAR 檔案。**另請參閱**類別載入器。

類別載入器　一個 Java 元件，用於依據特定的規則載入 Java 類別。**另請參閱**類別路徑。

屬性　請求物件中可以由 Servlet 設定的名稱數值對。也是一個可修改 XML 檔案元素的名稱數值對。請對照**參數**。更廣義地講，一個屬性便是一個複合資料單位。

欄　資料庫表格中的欄位。

索引

符號

[TLS Rollback] 選項
    密碼 304

字母

access.log 82
acl，點名稱 423
ACL，屬性 423
AddLog 168
add-resources 指令 296，395
admin-service 92
afterBegin 213
afterCompletion 213
ansi_x3.4-1968 370
ansi_x3.4-1986 370
Ant 工作 378
appclient 公用程式 378
Application Server
    產品系列簡介 24
    線上說明文件，Web 網站位置 25
application.xml 部署描述元 309
application-client.xml 部署描述元 309
applications
    監視物件類型 125
appserv.mib 146
    管理物件與描述 147
appservd 72
appservd-wdog.exe 72
appserv-wdog 72
AS_ADMIN_HOST 384
AS_ADMIN_INSTANCE 384
AS_ADMIN_PASSWORD 384
AS_ADMIN_PORT 384
AS_ADMIN_PREFIX 390
AS_ADMIN_SECURE 384
AS_ADMIN_USER 384
asadmin 公用程式
    export 383
    get 389
    JVM 設定 79
    reconfig 389
    set 389
    unset 383
    互動式 382
    本機 385
    用法 394
    多重模式 381
    安全性 394
    作業事件管理 219
    並行存取 395
    長選項 428
    非互動式 382
    指令 380
    指令行 386
    指令語法 379
    重新啓動實例 71
    密碼檔案選項 385
    授權指令 43
    啓動與停止實例 66
    單模式 381

短選項 428
程序檔 387
結束狀況 392
逸出字元 387
資料庫，管理與監視作業事件 218
運算元 380
預設值 428
管道 387
輔助說明 392
遠端 385
選項 380
環境指令 383
環境變數 428
點名稱 399
擷取監視資料 119
關於 377
屬性 399
ascii 370
auth-db 425
auth-passthrough 169，170
authrealm 425
AuthTrans 168
AuthTrans qos-handler 143
AuthTrans-class 170
auto-commit 連線驗證 258
avax.transaction.UserTransaction 214
Bean 管理式的作業事件
不允許用於實體 Bean 214
bean-cache
監視物件類型 126
監視屬性名稱 129
beanIdleTimeoutInSeconds 198
bean-method
監視物件類型 126
監視屬性名稱 130
bean-pool，監視物件類型 126
Bean，訊息導引
特徵 196
beforeCompletion 213
BM1168210 131
CacheBucket
監視 HTTP 伺服器元素 131
監視屬性 136
cacheFaultsPercentage 199
cache-hits 129
cache-misses 129
cache-resize-quantity 129
capture-schema 公用程式 378
CGI 369
使用虛擬伺服器 344
虛擬伺服器的設定 350
check-passthrough 172
chroot 設定 350
classpathprefix 79
ConnectionQueue 136
監視 ConnectionQueueBucket 屬性 134
監視 HTTP 伺服器元素 131
ConnectionQueue 屬性
監視 132
ConnectionQueueBucket
監視 HTTP 伺服器元素 131
ConnectionQueueBucket 監視屬性 134
connector-module 420
context-root 188
CORBA，關於 297
COSNaming 服務 229
Count200 至 Count503 137
Count2xx 至 Count5xx 137
CountAsyncAddrLookups 135
CountAsyncLookupsInProgress 135
CountAsyncNameLookups 135
CountBytesReceived 137
CountBytesTransmitted 137
CountCacheEntries 135
CountCacheHits 135
CountCacheMisses 135
Countcalls 138
CountConfigurations
監視 Process 屬性 133
CountConnections 135
CountContentHits 136
CountContentMisses 136
CountEntries 136
CountFlushes 135
CountHits 135，136
CountInfoHits 136

CountInfoMisses 136
CountMisses 136
CountOpenConnections 137
CountOpenEntries 136
CountOther 137
CountOverflow
    監視 ConnectionQueueBucket 屬性 134
CountQueued 134
    監視 ConnectionQueueBucket 屬性 134
CountRefusals 135
CountRequests 137，138
CountThreads 134
CountThreadsIdle 134
CountTimeouts 135
CountTotalConnection
    監視 ConnectionQueueBucket 屬性 134
CountTotalQueued
    監視 ConnectionQueueBucket 屬性 134
cp367 370
cp819 370
create-acl 指令 395
create-authdb 指令 395
create-auth-realm 指令 395
create-custom-resource 指令 395
create-domain 指令 57，395
create-file-user 指令 395
create-http-listener 指令 345，396
create-http-qos 指令 332，350，396
create-iiop-listener 指令 396
create-instance 指令 73，396
create-javamail-resource 指令 396
create-jdbc-connection-pool 指令 255，396
create-jdbc-resource 指令 246，257，396
create-jmsdest 指令 296，396
create-jms-resource 指令 296，396
create-jndi-resource 指令 396
create-jvm-options 指令 79，396
create-lifecycle-module 指令 323，396
create-mime 指令 335，396
create-persistence-resource 指令 396
create-profiler 指令 396
create-ssl 指令 396
create-virtual-server 指令 348，396
cron 95
    排程 logadm 的執行 100
crontab 99
crontab，項目格式 100
custom-resource 414
DataSource 230
DataSource 物件 244
delete-acl 指令 396
delete-authdb 指令 396
delete-auth-realm 指令 396
delete-custom-resource 指令 396
delete-domain 指令 58，396
delete-file-user 指令 396
delete-http-listener 指令 347，396
delete-http-qos 指令 332，350，396
delete-iiop-listener 指令 396
delete-instance 指令 73，397
delete-javamail-resource 指令 397
delete-jdbc-connection-pool 指令 397
delete-jdbc-resource 指令 397
delete-jmsdest 指令 296，397
delete-jms-resource 指令 296，397
delete-jndi-resource 指令 397
delete-jvm-options 指令 79
delete-lifecycle-module 指令 323，397
delete-mime 指令 335，397
delete-persistence-resource 指令 397
delete-profiler 指令 397
delete-ssl 指令 397
delete-virtual server 指令 397
delete-virtual-server 指令 352
deploy 指令 319，397
deploydir 指令 320，397
Developer 掇 Guide to Web Applications
    說明文件，說明 26
disable 指令 397
discardmanualchanges 74
display-license 指令 43，397
DnsBucket
    監視 HTTP 伺服器元素 131
    監視屬性 135

DnsBucket 屬性 135
domains.bin 58
domains.lck 58
DTD 檔案
  應用程式 XML 309
EJB
  MDB 儲存區設定，配置 201
  快取設定，配置 200
  參考 226
  啓動 192
  設定，配置 199
  鈍化 192
  模組屬性 418
  儲存區設定，配置 199
  類型 193
EJB JAR 模組
  部署 322
EJB JAR 檔案 308
EJB 容器
  可以監視的屬性 198
  負責 193
  配置日誌層級 196
  關於 191
  屬性 405
ejb-container 92，405
EJBContext 212
ejb-jar.xml 227
ejb-jar.xml 部署描述元 309
ejb-link 元素 227
ejbLoad 213
ejb-module 418
  監視物件類型 125
ejb-name 元素 227
  對映 237
EJBObject 193
ejb-ref-name 元素 227
enable 指令 397
enabled 屬性 189
entity-bean
  監視物件類型 126
Error qos-error 143
Error 指令 168
ErrorLogDateFormat 106
execution-time-millis 130
export 指令 383，397
Factory 物件 227
fail-all-connections 特性 259
FlagAsyncEnabled 135
FlagCacheEnabled 135
FlagEnabled 136
FlagProfilingEnabled
  監視 HTTP 伺服器屬性 132
FlagVirtualServerOverflow
  監視 HTTP 伺服器屬性 132
flexanlg 378
  用法和語法 111
FractionSystemMemoryUsage
  監視 Process 屬性 133
get 指令 397
  asadmin 389
  監視資料 120
getUserTransaction 214
help 指令 397
home.html 367
Hosts 137
htaccess 檔案 364
HTML，伺服器剖析，設定 372
htpasswd 公用程式 378
HTTP 166
  監視 117
HTTP 伺服器
  可監視屬性 130
HTTP 伺服器元素
  監視 131
HTTP 伺服器的可監視屬性 130
HTTP 伺服器屬性
  監視 132
HTTP 服務
  屬性 411
HTTP 偵聽程式 338
  http-listener-1 338，344
  SSL/TLS 安全性設定，啓動 53
  建立 345
  接受者執行緒，指定數目 53
  設定 53
  管理伺服器 53
  屬性 421
HTTP/1.1 協定 166

http-listener 346，421
http-server
    監視物件類型 125
    監視屬性名稱 126
http-server.http-listener 421
http-service 75，411
ibm367 370
ibm819 370
Id 137
    監視 ConnectionQueue 屬性 132
    監視 HTTP 伺服器屬性 132
idle-timeout-in-seconds 129，202
IIOP 服務
    屬性 408
IIOP 偵聽程式
    SSL/TLS 設定 304
    建立 302
    連接埠 305
    屬性 409
iiop-listener 409
iiop-service 92，408
    監視物件類型 125
IIOP，關於 298
IIS
    Web 伺服器外掛程式，配置以使用 177
    Web 伺服器外掛程式，配置用於 176
index.html 367
inflight-tx 128
INFO
    預設日誌層級 88
INIT 161
init.conf 75，169
    啓動時的全域變數設定 333
    終止逾時 68
initialBeansInPool 198
init-passthrough 170
inittab 47，67，69
    自動重新啓動伺服器 70
    啓動伺服器以 69
    編輯 70
install-license 指令 42，397
Interfaces 137
IP 位址，在 HTTP 偵聽程式中 338
isFrozen 128
iso_646.irv
    1991 370
iso_8859-1 370
    1987 370
iso-2022-jp 370
iso646-us 370
iso-8859-1 370
iso-ir-100 370
iso-ir-6 370
iwsCpuID 147
iwsCpuIdleTime 147
iwsCpuKernelTime 147
iwsCpuTable 147
iwsCpuUserTime 147
iwsInstanceContact 147
iwsInstanceCount200 (至 404) 147
iwsInstanceCount2xx - 5xx 147
iwsInstanceCount3xx 147
iwsInstanceCount4xx (& 5xx) 147
iwsInstanceCount503 149
iwsInstanceCountOther 147
iwsInstanceDeathCount 147
iwsInstanceDescription 147
iwsInstanceId 147
iwsInstanceInOctets 147
iwsInstanceLoad15MinuteAverage 148
iwsInstanceLoad1MinuteAverage 148
iwsInstanceLoad5MinuteAverage 148
iwsInstanceLocation 147
iwsInstanceNetworkInOctets 148
iwsInstanceNetworkOutOctets 148
iwsInstanceOrganization 147
iwsInstanceOutOctets 147
iwsInstanceRequests 147
iwsInstanceStatus 147
iwsInstanceTable 147
iwsInstanceUptime 147
iwsInstanceVersion 147
iwsListenAddress 149
iwsListenId 149
iwsListenPort 149
iwsListenSecurity 149

iwsListenTable 149
iwsProcessConnectionQueueCount 149
iwsProcessConnectionQueueMax 149
iwsProcessConnectionQueueOverflows 149
iwsProcessConnectionQueuePeak 149
iwsProcessConnectionQueueTotal 149
iwsProcessId 149
iwsProcessKeepaliveCount 149
iwsProcessKeepaliveMax 149
iwsProcessTable 149
iwsProcessThreadCount 149
iwsProcessThreadIdle 149
iwsThreadPoolTable 149
iwsVsCount200 (至 404) 148
iwsVsCount2xx - 5xx 148
iwsVsCount503 149
iwsVsCountOther 148
iwsVsId 148
iwsVsInOctets 148
iwsVsOutOctets 148
iwsVsRequests 148
iwsVsTable 148
J2EE
    Web 容器，關於 185
    作業事件 204
    作業事件應用程式 207
J2EE 連接器
    資源管理員 206
J2EE 模組
    命名 311
    定義 308
    動態重新載入 318
    執行環境 313
J2EE 應用程式
    EJB 規格 276
    JMS，與 276
    服務 221
    訊息導引 Bean，**請參閱** MDB
    資源 221
Java Message Service，**請參閱** JMS
Java Virtual Machine，**請參閱** JVM
Java 資料庫連結性 (JDBC) API
    使用實體 Bean 進行資料存取 194
java.sql.Connection 214
java.util.Properties 193
java-config 407
JavaMail
    Folder 物件 264
    JAF 264
    Message 子類別 264
    Message 類別 263
    Session 類別 264
    Store 類別 264
JavaMail API
    訊息處理 262
    關於 261
JavaMail 階段作業
    建立 268
    配置 269
    部署描述元 267
    資源 Factory 227
JavaMail 資源
    配置參數 265
    關於 261
    屬性 416
javax.ejb.EJBContext 214
javax.ejb.EntityBean 192
javax.ejb.EntityContext 192
javax.ejb.MDBContext 193
javax.ejb.SessionBean 192
javax.ejb.SessionContext 192
javax.ejb.SessionSynchronization 192
javax.sql.DataSource 206
javax.sql.XADataSource 206
JDBC
    API 194，222，243
    DataSource 物件 222
    URL 249
    作業事件 260
    連線 248
    連線 Factory 227
    資料來源 222，244
JDBC 連線區
    fail-all-connections 特性 259
    作業事件隔離 253
    建立 250
    特性 252
    連線驗證 253，258

監視 259
儲存區設定 252
屬性 413
JDBC 資源
建立 246
註冊 246
屬性 412
jdbc-connection-pool 256，413
監視物件類型 125
監視屬性名稱 128
jdbc-resource 412
JMS
API，規格清單 196
目標，**請參閱** JMS 目標
系統架構 273
服務，內建 278
服務，**請參閱** JMS 服務
訊息使用者 275
訊息偵聽程式 277
訊息提供者 275
訊息發送模型 273
訊息結構 274
訊息傳送系統概念 272
訊息導引 Bean 196
規格 272，274
提供者，內建 278
提供者，**請參閱** JMS 提供者
程式設計模型 274
資源，**請參閱** JMS 管理物件
實體目標，**請參閱** JMS 目標
管理物件**請參閱** JMS 管理物件
關於 272
JMS 目標 222
主題 280
佇列 280
管理 287
關於 280
JMS 服務
MQ 用戶端執行期間，與 283
MQ 訊息伺服器，與 283
MQ 管理物件，與 283
內建 283
外部 283
架構 282
配置 285
停用 283
管理 284
管理工具 283
屬性 403
JMS 提供者
原生 271，283
資源管理員 206
關於 271，278
JMS 管理物件
目標 281
連線 Factory 281
管理 290
關於 276
屬性 415
jms-max-messages-load 129
jms-ping 指令 296，397
jms-resource 296，415
jms-service 92，296，403
JNDI
JMS 管理物件，與 281
MDB 與 277
外部資源，建立 233
外部儲存庫 235
名稱 224
自訂資源，建立 232
架構 223
查找 276
查找及關聯的參考 225
查找方法 223
查找名稱 311
連線 Factory 230
資源屬性 413
jndi-resource 413
JVM
除錯選項 68
設定
配置 76，79
選項 78
屬性 407
JVM 測量程式
透過管理介面配置 78
屬性 427
KeepaliveBucket
監視 HTTP 伺服器元素 131
keepmanualchanges 74
latin1 370

lifecycle-module 426
list 指令 398
  監視 120
list-acls 指令 398
list-authdbs 指令 398
list-auth-realms 指令 398
list-components 指令 398
list-custom-resources 指令 398
list-domains 指令 59，398
list-file-groups 指令 398
list-file-users 指令 398
list-http-listeners 指令 345，398
list-iiop-listeners 指令 398
list-instances 指令 398
list-javamail-resources 指令 398
list-jdbc-connection-pools 指令 256，398
list-jdbc-resources 指令 257，398
list-jmsdest 指令 296，398
list-jms-resources 指令 296，398
list-jndi-resources 指令 398
list-lifecycle-modules 指令 323，398
list-mimes 指令 335，398
list-persistence-resources 指令 398
list-profilers 指令 398
list-sub-components 指令 398
list-virtual-servers 指令 398
Load15MinuteAverage
  監視 HTTP 伺服器屬性 132
Load5MinuteAverage
  監視 HTTP 伺服器屬性 132
LoadMinuteAverage
  監視 HTTP 伺服器屬性 132
location 188
log service 元素 91
LOG_ALERT 90
LOG_CRIT 90
LOG_DEBUG 90
LOG_ERR 90
LOG_INFO 90
LOG_WARNING 90
logadm 97
logadm.conf
  位置和範例 97
logchecker 99
LogFlushInterval 106
log-service 92，94，410
log-virtual-server-id 91
mail-resource 416
maxBeansInCache 198
max-beans-in-cache 129
maxBeansInPool 198
MaxByteTransmissionRate 137
MaxCacheEntries 135
MaxConnections 135
MaxEntries 136
MaxHeapCacheSize 136
MaxMmapCacheSize 136
MaxOpenConnections 137
MaxOpenEntries 136
max-pool-size 128
MaxProcs
  監視 HTTP 伺服器屬性 132
MaxQueued 134
  監視 ConnectionQueueBucket 屬性 134
MaxThreads 134
  監視 HTTP 伺服器屬性 132
MaxVirtualServers
  監視 HTTP 伺服器屬性 132
MDB
  JNDI 與 277
  作業事件 212，214
  部署描述元 277
  關於 195，277
MDB 容器
  關於 277
  屬性 404
MDB 儲存區設定
  爲 EJB 進行配置 201
mdb-container 92，404
message-driven-bean
  監視物件類型 126
MessageListener 196
meta-data 連線驗證 258

Microsoft 網際網路資訊服務
配置以使用 Web 伺服器外掛程式 175
MIME 類型
charset 參數 369
使用虛擬伺服器 348
定義 334
定義與存取頁面 440
建立新檔案 334
指定預設 368
虛擬伺服器設定，配置 348
編輯定義 334
屬性 422
mime，點名稱 422
minBeansInCache 199
minBeansInPool 198
Mode 136，137
監視 Process 屬性 133
MQ
代理程式 279
用戶端執行期間 280
訊息伺服器 279
訊息傳送系統，部分 278
資源管理員 206
管理工具 281
管理物件 281
與 Sun ONE Application Server 整合 282
說明文件，Web 網站位置 27
關於
multimode 398
NameTrans 168
nice 350
nsfc.conf
檔案快取記憶體設定 330
numBeansCreated 198
numBeansDestroyed 198
numBeansInPool 198
num-beans-in-pool 129
num-expired-sessions-removed 130
num-passivation-errors 129
num-passivations 129
num-passivation-success 130
numThreadsWaitaing 198
num-threads-waiting 129
obj.conf 檔案 75
設定 SAF 以使用服務品質 140
虛擬伺服器 341
範本 341
ObjectType 168
ObjectType-class 172
onMessage 126，212
ORB
IIOP 偵聽程式配置 302
介紹 298
服務，監視 118
附帶的 ORB 之功能性 299
配置 300
偵聽程式屬性 409
執行緒儲存區 301
屬性 408
orb
監視物件類型 125
點名稱 408
orb-connection
監視物件類型 125
監視屬性名稱 127
orblistener 409
orb-thread-pool
監視物件類型 125
監視屬性名稱 127
package-appclient 公用程式 378
password.conf 65
PathCheck 168
PeakQueued 134
監視 ConnectionQueueBucket 屬性 134
persistence-manager-factory-resource 416
Pid
監視 Process 屬性 133
PidLog 106
pkgadd 42
PooledConnection 物件 245
pool-resize-quantity 128
PR_Recv()/net_read 143
PR_Send()/net_write 143
PR_TransmitFile 143
process
監視 HTTP 伺服器元素 131
監視物件類型 125

監視屬性 133
監視屬性名稱 127
屬性 130
Process 元素 130
Profile 138
監視 HTTP 伺服器元素 131
監視屬性 133
Profile 元素 130
ProfileBucket
監視 HTTP 伺服器元素 131
ProfileBucket 元素 130
qos-error，Error 143
qos-handler，AuthTrans 143
ra.xml 部署描述元 310
RAR 檔案 308
RateBytesReceived
監視 HTTP 伺服器屬性 132
RateBytesTransmitted 137
監視 HTTP 伺服器屬性 132
rc.2.d，啓動伺服器以 69
reconfig 指令 60，74，255，389，398
RemoteException 211
removal-timeout-in-seconds 201
RequestBucket
監視 HTTP 伺服器元素 131
resources
監視物件類型 125
res-sharing-scope 208
restart-instance 指令 71，399
restartserv 71
RMI
介紹 298
RMI/IIOP 用戶端
部署 324
root
監視物件類型 125
SAF
auth-passthrough 170
check-passthrough 172
init-passthrough 170
service-passthrough 171
sagt 153
sagt，啓動 SNMP 代理程式的指令 153
schedulerd 97
SecondsMaxAge 136
SecondsRunning
監視 HTTP 伺服器屬性 132
SecondsTimeouts 135
security-service 92，411
server.log 82
預設日誌層級 88
預設記錄 83
範例 83
server.xml 75，87，91，94，216，317，329，338
不需要重新啓動的設定 75
預設 Web 應用程式 189
server.xml 檔案
屬性 94
server1 64
Service 168
service-passthrough 169，170，171
SessionSynchronization 213
set 指令 389，399
setAutoCommit 214
setRollbackOnly 212
SEVERE 89
show-component-status 指令 399
show-instance-status 指令 75，399
shutdown 指令 50，399
SizeHeapCache 136
SizeMmapCache 136
SizeResident
監視 Process 屬性 133
SizeVirtual
監視 Process 屬性 133
SMUX 152
SNMP
GET 與 SET 訊息 150
子代理程式 145
子代理程式，啓用 163
主代理程式 145
手動配置 158
在其他連接埠上啓動 157
安裝 152，154，157
啓用與啓動 157
啓動 161

代理程式
安裝 153
啓動 153
關於 152
本端常駐程式，重新啓動 154
在伺服器上設定 151
常駐程式，重新啓動 154
陷阱 150
團體字串 151
團體字串，配置 151
監視 116
簡單網路管理協定，介紹 144
snmpd，重新啓動本端 SNMP 常駐程式的指令 154
Solaris 8 與 9 封裝式非評估、非隨附安裝
預設安裝目錄的文件慣例 23
Solaris 9 隨附安裝
配置 33
預設安裝目錄的文件慣例 23
SSL/TLS
HTTP 偵聽程式設定 53
IIOP 偵聽程式設定 304
配合虛擬伺服器使用 343
standalone-ejb-module
監視物件類型 125
start-appserv 指令 399
start-domain 指令 48，59，399
start-instance 指令 66，68，399
startserv 67
啓動管理伺服器 47
stateful-session-bean
監視物件類型 126
stateless-session-bean
監視物件類型 125
stderr 83，94
stdout 83，94
steady-pool-size 128
stop-appserv 指令 50，399
stop-domain 指令 50，60，399
stop-instance 指令 66，399
stopserv 67
關閉管理伺服器 49
summary
可監視屬性 126
Sun ONE Message Queue，請參閱 MQ
Sun ONE Studio
部署使用 321
關於 42
Sun 客戶支援 27
sun-acc.xml 94
sun-application.xml 部署描述元 310
sun-application-client.xml 部署描述元 310
sun-cmp-mapping.xml 部署描述元 310
sun-ejb-jar.xml 部署描述元 310
sun-passthrough.properties 178
檔案範例 178
sun-web.xml 188
sun-web.xml 部署描述元 310
sysContact 158
sysLocation 158
syslog
用來識別應用程式伺服器訊息的資訊 87
用於配置的日誌層級 90
記錄 84
訊息
範例 87
syslog.conf 84
配置以儲存嚴重性較低的訊息 85
配置檔案的範例 85
syslogd 84
System.getCurrentTimeInMillis 217
table 連線驗證 258
Thread
監視 HTTP 伺服器元素 131
監視屬性 136
ThreadPool
監視 HTTP 伺服器元素 131
監視屬性 133
Thread-pool 134
ThreadPoolBucket
監視 HTTP 伺服器元素 131
監視屬性 134
thread-pool-size 127
TicksDispatch 138
TicksFunction 138
TicksPerSecond
監視 HTTP 伺服器屬性 132
TicksTotalQueued
監視 ConnectionQueueBucket 屬性 134

timeStamp 217
TimeStarted 136
    監視 HTTP 伺服器屬性 132
    監視 Process 屬性 133
total-beans-created 129
total-beans-destroyed 129
total-beans-in-cache 129
total-connections-timed-out 128
total-inbound-connections 127
total-num-calls 130
total-num-errors 130
total-num-success 130
total-outbound-connections 127，128
total-threads-waiting 128
total-tx-completed 128
total-tx-inflight 128
total-tx-rolled-back 128
TransactionRequiredException 210
transactionsCompleted 217
transaction-service 92，403
    監視物件類型 125
    監視屬性名稱 128
transactionsInFlight 217
transactionsRecovered 217
transactionsRolledBack 217
ulimit 65
undeploy 指令 320，399
unset 指令 383，399
update-file-user 指令 399
URL 連線 Factory 資源 236
URL 轉寄，配置 372
URL，JDBC 249
us 370
us-ascii 370
UserTransaction 物件 229
use-system-logging 85
verifier 公用程式 378
version 指令 399
VersionServer
    監視 HTTP 伺服器屬性 132
VirtualServer
    監視 HTTP 伺服器元素 131
    監視屬性 137
virtual-server 351，423
    監視物件類型 125
    監視屬性名稱 127
VirtualServer 元素 130
virtual-server 屬性 130
waiting-thread-count 127
WAR 模組，部署 322
WAR 檔案 308，349
Web 伺服器外掛程式
    IIS，配置以使用 177
    init.conf 169
    加入 173
    配置 169
    配置 Microsoft 網際網路資訊服務 175
    關於 165
Web 容器
    Web 應用程式部署 189
    介紹 185
    在虛擬伺服器中部署 Web 應用程式 187
    預設記錄行爲 190
web 容器
    屬性 406
Web 模組
    屬性 188
web 模組屬性 419
Web 應用程式 308
    元素 187
    使用虛擬伺服器 349
web.xml 部署描述元 309
web-container 75，92，406
WEB-INF 目錄 188
web-module 419
wscompile 公用程式 378
wsdeploy 公用程式 378
XATransaction 模式 206
x-euc-jp 370
x-mac-roman 370
x-sjis 370

三畫

三層資料庫存取 244
子代理程式
SNMP 145
SNMP，啓用 163
子系統
記錄控制，位於 92
記錄預設處理程式 92
工具
可實現管理功能 32

四畫

不可分性 204
互動式 asadmin 382
元件，MDB，**請參閱** MDB
內部常駐程式日誌旋轉 96
公用目錄
配置 365
公制間隔時間
流量計算使用的時間 139
分析程式，日誌
執行 (在使用前歸檔伺服器日誌) 111
分散式作業事件 245
分散式作業事件與本機作業事件 206
支援，客戶
聯絡資訊 27
文件
存取的日誌項目清單 110
文件目錄
主 342，362，443
附加 362
限定內容發佈 366
文件根 342，443
設定 362
文件偏好設定
目錄索引 368
伺服器首頁 368
剖析接受語言標頭 349
索引檔名 367
虛擬伺服器，設定 367
預設 MIME 類型，指定 368
文件註腳，設定 371
日誌分析程式
flexanlg，用法和語法 111
在使用前歸檔伺服器日誌 111
執行 111
從指令行執行 111
日誌服務屬性 410
file 104
log-stderr 104
log-stdout 104
use-system-logs 104
層級 104
日誌旋轉
內部常駐程式 96
基於 cron 99
執行 (四種方法) 95
排程程式 96
日誌層級
ALERT 89
用於 syslog 配置 90
表格 89
配置，EJB 容器 196
關於 88
嚴重性次序 89
日誌檔
存取 106
配置 110
虛擬伺服器 344
錯誤 106
歸檔 95
日誌歸檔檔案格式 95

五畫

主文件目錄，設定 342，362，443
主代理程式
SNMP 145
SNMP，手動配置 158
SNMP，在其他連接埠上啓動 157
SNMP，安裝 152，154，157
SNMP，啓用與啓動 157
SNMP，啓動 161
在非標準的連接埠上啓動 161
配置 檔案，編輯 158

主機屬性
針對主題型樣核取 342
主題 **請參閱** JMS 目標
代理程式，SNMP 152
安裝 153
啓動 153
外掛程式，Web 伺服器
請參閱 Web 伺服器外掛程式
外部資源
建立 233
關於 231
外部儲存庫，存取 235
布林選項 380
平台版
Application Server 7 24
本指南中所使用的慣例 22
本端 SNMP 常駐程式
重新啓動 154
本機作業事件最佳化 206
本機作業事件與分散式作業事件 206
本機選項 385
生命週期模組
部署 322
屬性 426
用戶端
存取清單 110
請求 166
用戶端名稱對映範例 122
用法，asadmin 394
目標，對於 JMS 訊息，**請參閱** JMS 目標
目錄，附加文件 362
目錄結構，部署 312

六畫

交談式狀態 194
企業 Java Bean
訊息導引 Bean 195
階段作業 Bean，關於 194
實體 Bean，關於 195
類型 193
企業 Java Bean 容器
關於 191
企業方法，作業事件 212，213
企業版
Application Server 7 25
共用程式庫，使用 325
再計算間隔時間 139
回轉，**請參閱**作業事件，回轉
多重伺服器儲存區
配置 178
多重模式 381
字元集
iso_8859-1 370
us-ascii 370
變更 369
存取 110
存取日誌檔 95，106
配置 110
旋轉 95
檢視 106
存取控制，使用虛擬伺服器 344
安全性，asadmin 394
安全性服務
屬性 411
自訂資源
建立 232
關於 231
屬性 414

七畫

佇列，**請參閱** JMS 目標
伺服器
手動重新啓動 (Unix) 47，67
手動停止 49
手動停止 (Unix) 67
重新啓動 (Unix) 69
配置屬性 427
停止 50
啓動 69
監視 HTTP 伺服器元素 131
請求的處理 166
伺服器元素 130

伺服器日誌 111
伺服器剖析的 HTML 372
伺服器實例
  加入 72
  刪除 73
作業事件
  afterBegin 方法範例 213
  afterCompletion 方法範例 213
  Bean 管理式的 198
  J2EE 204
  Mandatory 屬性 210
  Never 屬性 211
  NotSupported 屬性 210
  required 屬性 210
  RequiresNew 屬性 210
  Supports 屬性 210
  一致性 204
  介紹 204
  分散式 245
  本機最佳化 206
  本機與分散式 206
  回轉 198，212，219
  使用者應用程式 205
  恢復 208
  相同層級的，J2EE 207
  容器管理式的 208
  訊息導引 Bean 212，214
  執行中 219
  透過管理介面進行管理 216
  階段作業 Bean 213
  資料庫，使用 asadmin 進行管理與監視 218
  實體 Bean 212
  監視 220
  確定 198
  屬性 209，450
作業事件使用者應用程式 205
作業事件服務
  凍結與取消凍結的範例 219
  經由 asadmin 管理 138
  監視 118
  屬性 403
作業事件資源管理員 206
作業事件管理員 204
快取控制指令，設定 373
快取設定，配置 EJB 200
更強密碼 374
更精細 89
系統 RC 程序檔
  自動重新啓動伺服器 70

八畫

並行存取，asadmin 395
並行連線
  虛擬伺服器，服務品質 144
事件，檢視 (Windows 2000 Pro) 113
事件日誌檔
  檢視 108
事件檢視器
  監視事件 (Windows 2000 Pro) 113
事件變數
  陷阱 146
使用者目錄
  自訂 365
  配置 365
使用者作業事件參考 229
使用者應用程式，作業事件 205
兩層資料庫存取 244
協定資料單元 (PDU) 150
命名
  COSNaming 229
  J2EE 模組 311
  JNDI 查找 311
  JNDI 與資源參考 224
  服務 249
  初始環境 229
  標準 311
服務品質 139
  在 obj.conf 中設定 SAF 以使用 140
  並行連線，虛擬伺服器 144
  使用 119
  配置 140
  配置 HTTP 伺服器 331
  虛擬伺服器，配置設定 350
  僅測量應用程式層級的 HTTP 頻寬 143
  監視 119
  範例 139
服務控制台
  啓動管理伺服器 48

物件類型，監視 124
狀況，應用程式伺服器實例 75
狀態，虛擬伺服器 349
初始命名環境 229
長選項 428
附加文件目錄 362
非互動式 asadmin 382

九畫

客戶支援，聯絡資訊 27
恢復，作業事件 208
持續性
    Bean 管理式的 195
    容器管理式的 195
    資料儲存與 238
    實體 Bean 239
    關於 238
持續性管理程式
    Factory 資源屬性 416
    角色 239
    建立 241
指令
    asadmin 380
    授權 42
指令，配置記錄 106
指令行，asadmin 386
指令行介面
    名稱對映，監視 121
    啓動管理伺服器 48
    關閉管理伺服器 50
指定
    日誌層級 105
    日誌檔 105
    作業事件日誌位置 106
流量
    設定，計算統計資料 139
相同層級的作業事件，J2EE 207
重新部署應用程式 317
重新載入，動態 318
限定符號式連結 364
首頁 368

十畫

容器
    EJB，負責 193
    MDB 277
    Web，關於 185
容器管理式的作業事件 208
效能
    使用服務品質 (QOS) 119
    動態重新載入 318
根目錄
    安裝，慣例 22
索引檔名 367
記錄
    UNIX 83
    Web 容器，預設行爲 190
    Windows 83
    元件和子系統，配置 105
    元件和子系統，清單 105
    用戶端 94
    存取檔案，檢視 106
    事件檔案，檢視 108
    使用 syslog 84
    指令，配置 106
    指令行選項置換 logadm.conf 選項 99
    重新導向應用程式與伺服器日誌輸出 94
    特徵和功能 81
    訊息
        提供的資訊 82
    偏好設定 110
    透過指令行介面配置 101
    透過管理介面配置 102
    透過管理介面配置屬性 103
    虛擬伺服器實例 90
    應用程式用戶端容器 (ACC) 94
    關於 82
訊息，記錄
    提供的資訊 82
訊息代理程式，**請參閱** MQ 代理程式
訊息偵聽程式 275，277
訊息傳送
    JMS **請參閱** JMS
    非同步 271
訊息導引 Bean，**請參閱** MDB
配接卡，資源 205

配置 89，153，157，158
主代理程式，編輯 158
配置檔案，關於 42
除錯 68
除錯模式
啓動應用程式伺服器實例 68

十一畫

停止此伺服器 50
停用已部署的應用程式或模組 317
偵聽套接字，**請參閱** HTTP 偵聽程式
偵聽程式，HTTP
編輯 53
偏好設定，日誌
設定 110
動態重新部署
在不重新啓動伺服器的情況下，重新部署現有應用程式 189
動態重新載入 318
動態部署 317
基於 cron 的日誌旋轉 99
基於 IP 位址的虛擬伺服器 340
基於 URL 主機的虛擬伺服器 340
執行中的作業事件 219
執行緒儲存區
ORB 301
指定要加入的資訊 333
執行環境 313
密碼，[TLS Rollback] 選項 304
密碼檔案，啓動時載入 367
密碼檔案選項 385
常駐程式
本端 SNMP，重新啓動 154
控制設定，檢視管理伺服器 52
接受者執行緒
虛擬伺服器 339
藉由 HTTP 偵聽程式指定數目 53
接受語言標頭，剖析 349
授權指令 42
授權範圍屬性 425
排程程式日誌旋轉
排程程式連結 96
歸檔日誌檔 97
啓動，定義 192
啓動此伺服器 69
產品系列
簡介，Application Server 7 24
異常
回轉作業事件 212
符號式 (軟式) 連結，定義 364
符號式連結，限定 364
統計資料
動態地重新配置伺服器之後會遺失服務品質頻寬 144
測量流量的設定 139
監視 116
終止逾時
init.conf 68
設定 68
設定
Java Virtual Machine (JVM)，配置 76
管理伺服器，存取 51
軟式 (符號式) 連結 364
連接埠
HTTP 偵聽程式 339
IIOP 偵聽程式 305
連接器模組
部署目錄結構 313
屬性 420
連線 Factory
JNDI 230
URL 236
定義 230
連線，可共用或不可共用 208
連線共用 259
連線物件 230
連線匯集
datasource 物件 245
關於 258
連線驗證 258
部署
COSNaming 服務 230
EJB JAR 模組 322
RMI/IIOP 用戶端 324
WAR 模組 322

生命週期模組 322
目錄結構 312
使用 asadmin 319
使用 Sun ONE Studio 321
使用管理介面 320
重新部署 317
個別模組 321
停用 317
動態 317
執行環境 313
部署，熱
在不重新啓動的情況下，於伺服器執行期間部署應用程式 189
部署描述元
J2EE 標準 309
Sun ONE Application Server 310
作業事件屬性 211
項目 267
陷阱
SNMP 150
包含事件變數的訊息 146

十二畫

最佳化，本機作業事件 206
最精細 89
單一登入，關於 190
單模式 381
測量程式 79
點名稱 427
屬性 427
短選項 428
硬式連結，定義 364
程式庫，共用，使用 325
程式設計，JMS 程式設計模型 274
程序檔，asadmin 387
結束狀況，asadmin 392
虛擬伺服器
HTTP 偵聽程式 338
HTTP 偵聽程式，建立 345
大量主機作業範例 358
介紹 337
公用目錄，配置使用 365
文件偏好設定，設定 367
日誌檔 344
企業網路主機作業範例 356
安全伺服器範例 355
刪除 352
並行連線，服務品質 144
使用 SSL 343
使用存取控制 344
使用服務品質 119
服務品質，配置設定 350
狀態 349
建立 347
記錄實例 90
配置 MIME 設定 348
將 Web 容器配置爲部署 Web 應用程式 187
接受者執行緒 339
設定附加文件目錄 362
部署 353
單一登入 190
預設 341
預設 Web 應用程式 189
預設配置範例 353
編輯一般設定 351
請求處理 342
類型 340
屬性 187，423
註冊，領域
重新建立 60
逸出字元，asadmin 387
鈍化 192
階段作業
JMS 訊息傳送 275
動態重新載入 317，318
階段作業 Bean
同步化實例變數 213
有狀態 194
作業事件 212，213
無狀態 194
實例變數，同步化 213
關於 194

十三畫

資料庫
  JDBC API 243
  JNDI 名稱 224
  三層存取模型 244
  兩層存取模型 244
  命名服務 249
  連線驗證 253
  透過 CLI 管理與監視 218
  資源參考 224
  資源管理員 205
資料儲存 238
資訊 89
資源
  JMS，**請參閱** JMS 管理物件
  外部 231
  自訂 231
資源 RAR 檔案 308
資源介面 205
資源參考 224，235
資源管理員
  J2EE 連接器 206
  JMS 提供者 206
  作業事件 206
  定義 205
  資料庫 205
資源環境參考 228，236
運算元，asadmin 380
逾時，終止
  設定 68
隔離 204
預設 HTTP 偵聽程式
  HTTP 伺服器 344
  管理伺服器 53
預設 Web 模組 350
預設處理程式
  子系統記錄 92
預設選項值 428

十四畫

團體字串，與 SNMP 代理程式 151
實例
  應用程式伺服器
    存取 41
    關於 64
實體 Bean
  作業事件 212
  透過 JDBC 處理資料存取 194
  禁用的 Bean 管理式的作業事件 214
  關於 195
監視 134，135，136，137
  bean-cache 屬性 129
  bean-method 屬性 130
  CacheBucket 131
  CLI 名稱對映 121
  ConnectionQueue 131
  ConnectionQueue server 屬性 132
  ConnectionQueueBucket 131
  ConnectionQueueBucket ConnectionQueue 屬性 134
  ConnectionQueueBucket CountOverFlow 屬性 134
  ConnectionQueueBucket CountQueued 屬性 134
  ConnectionQueueBucket CountTotalConnection 屬性 134
  ConnectionQueueBucket CountTotalQueued 屬性 134
  ConnectionQueueBucket MaxQueued 屬性 134
  ConnectionQueueBucket PeakQueued 屬性 134
  ConnectionQueueBucket TicksTotalQueued 屬性 134
  ConnectionQueueBucket 屬性 134
  DnsBucket 131
  FlagProfilingEnabled 132
  FlagVirtualServerOverflow 132
  HTTP 117
  HTTP 伺服器元素 131
  HTTP 伺服器屬性 130，132
  http-server 屬性 126
  Id 132
  JDBC 連線區 259
  jdbc-connection-pool 屬性 128
  KeepaliveBucket 131
  Load15MinuteAverage 132
  Load5minuteAverage 132
  LoadMinuteAverage 132
  MaxProcs 132

MaxThreads 132
MaxVirtualServers 132
ORB 服務 118
orb-connection 屬性 127
orb-thread 屬性 127
Pid Process 屬性 133
process 131
Process CountConfigurations 屬性 133
Process FractionSystemMemoryUsage 屬性 133
Process SizeResident 屬性 133
Process SizeVirtual 屬性 133
process 屬性 127，133
Profile 131
Profile 屬性 133
ProfileBucket 131
RateBytesReceived 132
RateBytesTransmitted 132
RequestBucket 131
SecondsRunning 132
SNMP 116
Thread 131
ThreadPool 131
ThreadPool 屬性 133
ThreadPoolBucket 131
TicksPerSecond 132
TimeStarted 132
TimeStarted Process 屬性 133
transaction-service 屬性 128
VersionServer 132
VirtualServer 131
virtual-server 屬性 127
用戶端名稱對映，範例 122
伺服器 131
作業事件服務 118
使用 asadmin 擷取資料 119
使用 get 指令 120
使用 list 指令 120
服務品質 (QOS) 119
物件類型 124
附加的子系統與元件 117
容器子系統 118
統計資料 116
節點 Process 屬性 133
關於 115
監視程式 72
管理 216
管理，工具與關聯功能 32
管理介面
JVM 測量程式，配置 78
JVM 選項，配置 78
一般設定，配置 76
存取 35，46
自動恢復作業事件 208
使用 35
配置日誌服務屬性 103
路徑設定，配置 77
管理作業事件 216
標準按鈕 39
標籤，使用 37
線上輔助說明，存取 39
關閉管理伺服器 49
管理伺服器
套用變更 52
控制設定，檢視 52
啓動 SNMP 主代理程式 162
啓動，方法用於 47
設定，存取 51
關於 46
關閉，方法用於 49
管理物件 146，150
管理物件**請參閱** JMS 管理物件
管理資訊庫 (MIB)
定義管理物件 146
管理領域
建立 33
關於 55
管道，使用 asadmin 387
精細 89
網路管理站 (NMS) 144
關於 145
語法，asadmin 379
認證 299
認證資料庫屬性 425
說明文件
手冊概論 25
輔助說明
asadmin 公用程式 392
管理介面 39
遠端檔案操控
啓用 363
領域
建立 57

配置 56
管理，非 root 使用者執行的建立與刪除作業 58
管理，關於 55
領域目錄 56
領域註冊
重新建立 60

十五畫

標準版
Application Server 7 24
熱部署
在不重新啓動的情況下，於伺服器執行期間部署應用程式 189
確定，**請參閱**作業事件，確定
線上說明文件
Web 網站位置 25
線上輔助說明
asadmin 公用程式 392
管理介面，存取 39
請求
方法 166
伺服器處理的方式 166
處理步驟 168
請求處理，針對虛擬伺服器 342

十六畫

選項 380
布林 380
預設值 428
錯誤日誌檔 106
錯誤回應，自訂 369

十七畫

儲存區，多重伺服器
配置 178
儲存區設定
EJB，配置 199
應用程式
J2EE，介紹 309
JMS 與 276
JNDI 查找名稱 311
Web 元素 187
目錄結構 312
命名標準 311
停用 317
動態重新載入 318
執行環境 313
連線共用 259
資源參考 235
資源環境參考 236
環境項目 226
屬性 417
應用程式，點名稱 417
應用程式用戶端 JAR 檔案 308
應用程式用戶端容器 94
應用程式用戶端容器 (ACC)
用戶端記錄 94
應用程式伺服器
記錄特徵和功能 81
概論與主要功能 32
應用程式伺服器實例
手動啓動 70
存取 41
狀況，檢視 75
套用變更 74
啓動與停止 65
進階設定 80
關於 64
應用程式與伺服器日誌輸出，重新導向 94
檔案快取記憶體 330
檔案操控，遠端
啓用 363
檢視事件 113
檢驗器公用程式 309
環境指令，asadmin 383
環境項目 226
環境類別路徑
忽略 77
環境變數
AS_ADMIN_PREFIX 390
asadmin 428

ASADMIN_HOST 384
ASADMIN_INSTANCE 384
ASADMIN_PASSWORD 384
ASADMIN_PORT 384
ASADMIN_SECURE 384
ASADMIN_USER 384
點名稱，asadmin 399

十八畫

歸檔，日誌檔 95
簡單網路管理協定 (SNMP)
介紹 144

二十畫以上

嚴重 89
警示 89
警告 89
屬性
EJB 容器 (可以監視) 198
Web 模組 188
作業事件 209
作業事件，部署描述元 211
虛擬伺服器 187
屬性，asadmin 399
變數
全域
init.conf 中的設定 333
事件
陷阱 146