# Web 应用程序框架教程

## Sun Java™ Studio Enterprise 7 2004Q4

Sun Microsystems, Inc.
www.sun.com

文件号码：819-1291-10
2004 年 12 月 修订版 A

请通过 http://www.sun.com/hwdocs/feedback 提交有关本文档的意见和建议

# 目录

# 前言

*《Web 应用程序框架教程》*向开发人员介绍使用 Web 应用程序框架工具构建 Web 应用程序的方法和技巧。

本文档的适用读者为：至少了解如何使用现有 Java 2 Platform Enterprise Edition (J2EE™ Platform) Web 技术构建 Web 应用程序，但对使用 Web 应用程序框架构建 Web 应用程序却比较陌生的开发人员。

## 阅读本书须知

本教程可以帮助开发人员更加熟悉如何使用 Sun Java Studio Enterprise 7 开发人员环境 （今后称为 IDE）中的 Web 应用程序框架工具。

阅读本书之前，您应该了解利用现有 J2EE Web 技术，如 servlet 和 JavaServer Pages™ （JSP™ pages）生成 Web 应用程序时用到的概念。

下列资源可提供更多信息：

- *《Java 2 Platform Enterprise Edition 规范》*
  `http://java.sun.com/j2ee/download.html#platformspec`
- *《J2EE 教程》*
  `http://java.sun.com/j2ee/tutorial`
- *《Java Servlet 规范 2.3 版》*
  `http://java.sun.com/products/servlet/download.html#specs`
- *《JavaServer Pages 规范 1.2 版》*
  `http://java.sun.com/products/jsp/download.html#specs`

**注 --** Sun 对本文档中提到的第三方 Web 站点的可用性不承担任何责任。对于此类站点或资源中的（或通过它们获得的）任何内容、广告、产品或其他材料， Sun 并不表示认可，也不承担任何责任。对于因使用或依靠此类站点或资源中的（或通过它们获得的）任何内容、产品或服务而造成的或连带产生的实际或名义损坏或损失， Sun 概不负责，也不承担任何责任。

# 本书的组织结构

您会在下面一章中了解开发企业 Web 应用程序的 Web 应用程序框架及工具箱 (IDE) 的主要功能概述。

- 第 1 章，第 15 页上的 “在开始本教程之前”。

在下面一章中，对使用 Web 应用程序框架工具生成 J2EE Web 应用程序的方法进行了概述。

- 第 2 章，第 17 页上的 “使用入门”。

下面章节指导您创建后续各章节所需的应用程序基础结构，并增加第一个 Web 应用程序框架页。

- 第 4 章，第 27 页上的 “教程 1.1 节 应用程序基础结构”。
- 第 5 章，第 35 页上的 “教程 1.2 节 创建登录页”。
- 第 6 章，第 47 页上的 “教程 1.3 节 测试运行登录页”。

下面章节指导您增加基于 SQL 的模型以及显示该模型数据的页，以此来扩展现有应用程序。然后将这两个应用程序页链接在一起，以使它们显示的数据协调一致。

- 第 7 章，第 51 页上的 “教程 2.1 节 准备应用程序来访问 SQL 数据库”。
- 第 8 章，第 57 页上的 “教程 2.2 节 创建 CustomerModel”。
- 第 9 章，第 65 页上的 “教程 2.3 节 创建客户页”。
- 第 10 章，第 83 页上的 “教程 2.4 节 测试运行客户页”。
- 第 11 章，第 85 页上的 “教程 2.5 节 将登录页链接到客户页”。
- 第 12 章，第 87 页上的 “教程 2.6 节 运行应用程序”。

下面章节指导您创建可以由同一应用程序内的多个按钮和 HREF 重用的 “命令”组件。对于父容器视图类内部的按钮或 HREF 的处理请求事件，这是实现请求处理代码的另一种技术。

- 第 13 章，第 89 页上的 “教程 3.1 节 创建命令组件”。
- 第 14 章，第 97 页上的 “教程 3.2 节 向客户页增加注销链接”。
- 第 15 章，第 103 页上的 “教程 3.3 节 测试运行登录 / 注销命令组件”。

下面章节指导您增加基于 Web 服务的模型以及显示该模型数据的页，以此来扩展现有应用程序。需要注册并下载 Google 开发人员的 SDK，才能为 Web 服务构建模型。

- 第 16 章，第 105 页上的 “教程 4.1 节 准备创建 Web 服务模型”。
- 第 17 章，第 111 页上的 “教程 4.2 节 创建 Google 搜索页”。
- 第 18 章，第 131 页上的 “教程 4.3 节 测试运行 Google 搜索页”。
- 第 19 章，第 133 页上的 “教程 4.4 节 将结果列表增加至 Google 搜索页”。
- 第 20 章，第 151 页上的 “教程 4.5 节 测试运行 Google 搜索页”。

# 排版惯例

| 字体 | 含义 | 示例 |
|---|---|---|
| AaBbCc123 | 命令、文件和目录的名称；计算机屏幕输出 | 编辑 .cvspass 文件。<br>使用 DIR 列出所有文件。<br>Search is complete. |
| **AaBbCc123** | 键入的内容，以便与计算机屏幕输出相区别 | > **login**<br>Password: |
| *AaBbCc123* | 书名、新词或术语以及要强调的词 | 请阅读 *《用户指南》* 中的第 6 章。<br>这些称为*类*选项。<br>*必须*保存您的更改。 |
| *AaBbCc123* | 命令行变量；用实际名称或值替换 | 要删除文件，请键入 **DEL** *filename*。 |

# 相关文档

Java Studio Enterprise 文档包括 Acrobat Reader (PDF) 格式的书籍和教程、发行说明、联机帮助以及 HTML 格式的教程。

## 访问联机文档

访问 docs.sun.com℠ Web 站点，或者通过 Sun Java Studio Enterprise 开发人员资源门户 (http://developers.sun.com/jsenterprise) 的文档链接，可以查看该部分所描述的文档。

您可以通过 Internet 上的 docs.sun.com Web 站点 (http://docs.sun.com)，阅读、打印和购买 Sun Microsystems 手册。

- *《Sun Java Studio Enterprise 7 发行说明》* - 文件号码 819-1303-10

  介绍最新的发行更改和技术说明。

- *《Sun Java Studio Enterprise 7 安装指南》*（PDF 格式）- 文件号码 819-1301-10

  介绍如何在每台支持的平台上安装 Sun Java Studio Enterprise 7 集成开发环境 (IDE)，包括其他相关信息，例如系统需求、升级指导、服务器信息、命令行开关、已安装的子目录、数据库集成以及有关如何使用 “更新中心”的信息。

- *《构建 J2EE 应用程序》* - 文件号码 819-1299-10

  描述如何将 EJB 模块和 Web 模块组装到 J2EE 应用程序中，以及如何部署和运行 J2EE 应用程序。

- Web 应用程序框架文档 （PDF 格式）
  - *《Web 应用程序框架组件编写人员指南》* - 文件号码 819-1285-10

    介绍 Web 应用程序框架组件体系结构以及设计、创建和分发新组件的过程。

  - *《Web 应用程序框架组件参考指南》* - 文件号码 819-1287-10

    介绍 Web 应用程序框架库中可用的组件。

  - *《Web 应用程序框架概述》* - 文件号码 819-1289-10

    介绍 Web 应用程序框架及其概念、工作原理以及与其他应用程序框架的区别。

  - *《Web 应用程序框架教程》* - 文件号码 819-1291-10

    介绍使用 Web 应用程序框架工具生成 Web 应用程序的机制和技术。

  - *《Web 应用程序框架开发人员指南》* - 文件号码 819-1293-10

    提供创建和使用应用程序组件的步骤 （这些应用程序组件经组装后通过 Web 应用程序框架可用来开发应用程序），并解释如何在大多数 J2EE 容器中部署该应用程序。

  - *《Web 应用程序框架 IDE 指南》* - 文件号码 819-1295-10

    介绍 Sun Java Studio Enterprise 7 2004Q4 IDE 的各个部分，着重介绍用于开发 Web 应用程序框架应用程序的可视工具的使用。

  - *《Web 应用程序框架标记库参考》* - 文件号码 819-1297-10

    概要介绍 Web 应用程序框架标记库，并对该库中可用的标记进行综合引用。

# 教程

Sun Java Studio Enterprise 7 教程可以帮助您了解 IDE 的功能。每个教程都提供技术和代码样例，您可以在开发大量的应用程序时使用或进行修改。所有的教程都采用 Sun Java System Application Server 来解释部署。

所有教程都可以通过开发人员资源门户中的 “教程和代码库”链接进行访问。在 IDE 中选择 “帮助” > “示例和教程”可以访问所有这些教程。

- **快速入门指南**概要介绍了 Sun Java Studio IDE。如果您初次使用 Sun Java Studio IDE 或者希望获取某个特定功能的快速介绍，请首先访问快速入门教程。这些教程描述了如何开发简单的 Web 应用程序和 J2EE 应用程序、生成 Web 服务；如何开始进行 UML 建模和重构。您只需花几分钟的时间便可以了解快速入门教程。
- **教程**着眼于 Sun Java Studio IDE 的单一功能。如果您希望了解某项特定功能的详细信息，请尝试使用教程。取决于示例的着眼点，有些教程从头开始生成应用程序，有些则是在提供的源文件上构建应用程序。您可以花一个小时或更少的时间完成该教程。
- **解说式教程利用视频来说明某项功能或技术。**请尝试使用解说式教程，获取关于 IDE 的可视化概述或某项特定功能的进一步说明。您只需花几分钟的时间便可以完成解说式教程。此外，您还可以随意启动和停止解说式教程。

## 联机帮助

您可以在 Sun Java Studio Enterprise 7 IDE 中获得联机帮助。通过按下帮助键（Microsoft Windows 环境中为 F1， Solaris 环境中为 Help 键），或者选择 “帮助” → “内容”可以打开帮助。执行以上任意操作都将显示一个帮助主题列表和一个搜索工具。

## 使用易读格式的文档

该文档以易读格式提供，以方便残障用户使用辅助技术进行阅读。您还可以按照下表所描述的信息找到文档的易读版本。

| 文档类型 | 易读版本的格式和位置 |
|---|---|
| 书籍和教程 | HTML 格式，位置：http://docs.sun.com |
| 教程 | HTML 格式，位置：开发人员资源门户 (http://developers.sun.com/jsenterprise) 的 “示例和代码库”链接 |
| 发行说明 | HTML 格式，位置：http://docs.sun.com |

# Sun 欢迎您提出意见和建议

Sun 致力于提高文档质量，并欢迎您提出宝贵的意见和建议。请通过电子邮件将您的意见发送至以下地址：

`docfeedback@sun.com`

请在电子邮件的主题栏中注明书名 （*《Web 应用程序框架教程》*）及其文件号码 (819-1291-10)。

第 1 章

# 在开始本教程之前

欢迎使用面向企业 Web 应用程序开发的 Web 应用程序框架、J2EE Web 应用程序框架和 Sun Java Studio Enterprise 7 工具集 (IDE)。

本章论述了 Web 应用程序框架的主要功能。

## Web 应用程序框架的主要功能

Web 应用程序框架的主要功能如下：

- 全套 J2EE™ 应用程序开发
- 高性能、可靠的 J2EE 框架运行环境
- 完全基于组件的开发
- 图形化应用程序生成器工具集如下：
  - 逻辑化应用程序树资源管理器视图
  - 应用程序组件和 JSP 间的更改自动同步
  - 高级向导
- 支持 Web 服务模型

Web 应用程序框架的使用者：

- 进行大中型或超大型企业 Web 应用程序开发的大型企业
- 金融、制造、政府、教育、保健和电信业

Web 应用程序框架作为一个极具价值的工具，具有以下功能：

- 为初级和资深的 Java™/J2EE 开发人员提供指导
  - 通过图形化开发工具实现了超乎寻常的易学易用性
  - 便于那些不具备深厚背景知识的人员利用复杂的 J2EE API
  - 为缺乏经验的开发人员提供了在构建高性能企业应用程序的过程中学习 J2EE 的功能
- 为高级 Java/J2EE 开发人员和设计师提供协助
  - 使高级开发人员从枯燥的低级 J2EE 开发中解脱出来，从而获得更高的工作效率
  - 为设计师提供了定义完善的应用程序体系结构扩展点
- 通过提供进入 J2EE API 世界的便利途径，加快 Web 应用程序开发和技术 / 组件的再利用

本指南说明如何利用 Web 应用程序框架功能来完成以下任务：

- 创建 Web 应用程序框架应用程序
- 创建页 （ViewBean 和 TiledView）及关联的 JSP
- 创建和使用 “模式”（JDBC™ SQL 和基于 WebService 的模型）
- 将页链接在一起

第 2 章

# 使用入门

本章概述使用 Web 应用程序框架工具构建 J2EE™ Web 应用程序的方法。

本章包含以下主题：

- 简介
- 编写 Web 应用程序框架应用程序
- 关于 Web 应用程序框架教程

## 简介

本文档向开发人员介绍使用 Web 应用程序框架工具构建 Web 应用程序的方法和技巧。

本文档所面向的读者是：对如何使用现有的 J2EE Web 技术 （servlet 和 JSP™ pages）来构建 Web 应用程序至少有所了解，但尚未了解如何使用 Web 应用程序框架来构建 Web 应用程序的开发人员。

本文档假定读者掌握了 Java 技术并熟悉要使用的特定 servlet 容器以及开发工具的开发和部署过程。

由于 Web 应用程序框架首先是一种设计模式和一组接口，因此本文档中的示例只演示了创建 Web 应用程序框架应用程序的最基本方法，即扩展现有的 Web 应用程序框架实现基类并手动构建某些应用程序对象。这只是创建 Web 应用程序框架应用程序的过程中一种可能的方法。

未在本文档中展示更高级技术的原因有两个。首先，从基础开始学习是初次接触 Web 应用程序框架的人员了解其工作方式的最直接方法。能够准确地理解框架与应用程序进行交互的过程对深入了解 Web 应用程序框架至关重要。

其次，使用这些基本技术来构建应用程序是透彻理解构建 Web 应用程序框架应用程序的多种可行方法的先决条件。通过扩展 Web 应用程序框架来增加其他性能的功能构建于本文档所展示的技术之上。理解了这些基本示例后，您便可以对这些功能扩展和补充 Web 应用程序框架核心的过程有更深刻的理解，并可以选择不使用它们而是构建自己的 Web 应用程序框架扩展 （或在必要时直接使用更基本的方法）。

本文档的最终目的是向开发人员介绍构建 Web 应用程序框架应用程序的最基本方法，以便他们可以逐渐熟悉 Web 应用程序框架与在其框架上构建的应用程序之间的交互，并能够更熟练地使用 Web 应用程序框架。

# 编写 Web 应用程序框架应用程序

编写 Web 应用程序框架应用程序由两个步骤组成：第一步是设计应用程序结构；第二步是逐渐向该结构添加 Web 应用程序框架对象。尽管此任务可以完全通过手动操作从头完成，但更简单的方法是为 Sun Java Studio Enterprise 7 创建 Web 应用程序框架工具模块，该模块可协助开发人员编写他们的 Web 应用程序框架应用程序。借助这些工具，创建 Web 应用程序框架应用程序变成了一个简单的过程：使用向导生成 Web 应用程序框架组件，然后将这些组件定制成应用程序。

在演示创建简单的 Web 应用程序框架应用程序前，先要了解 Web 应用程序框架应用程序结构的基础知识。

## J2EE/Web 应用程序框架术语

本文档中包含诸如应用程序、模块和组件等术语。这些术语有时会发生混淆，因为在较一般的 Web 体系结构和开发论述中也会用到它们。

下表包含本教程中出现的最重要术语的列表。

| 术语 | 描述 |
| --- | --- |
| *J2EE 组件 | 有时称作 J2EE 应用程序组件；在 J2EE 服务器上部署、管理和执行的具体软件组件，包括 EJB、Servlet 及 Java Server Pages™ (JSP™)；有些包含 HTML 和 Applet 的组件也是 J2EE 组件，但这些组件与 Web 应用程序框架 Web 应用程序的论述无关。 |
| *J2EE 模块 | 代表 J2EE 应用程序的基本组成单元。一个 J2EE 模块由一个或多个 J2EE 组件和一个组件级部署描述符组成。J2EE 模块可以作为独立单元部署，也可与 J2EE 应用程序部署描述符组装在一起，然后以 J2EE 应用程序形式部署。 Servlet 和 / 或 JSP 组件以 J2EE 模块形式打包，然后以 WAR 文件形式部署。 EJB 组件以 J2EE 模块形式打包，然后以 JAR 文件形式部署。可将任意数量的 WAR 文件和 JAR 文件组合在一起，组成 J2EE 应用程序，然后以 EAR 文件形式部署。可在 J2EE 服务器上独立部署 WAR 文件 （J2EE 模块，亦称 J2EE Web 应用程序）。 |
| *J2EE Web 应用程序 | 包含可在 J2EE servlet 容器 （Web 应用程序容器）内部署的 J2EE 组件的独立 J2EE 模块。“应用程序”或“J2EE 应用程序”一词可能指 J2EE Web 应用程序，具体应根据上下文判断。有些产品支持 J2EE Web 应用程序，如 Sun Java System Application Server Standard Edition 7 2004Q2 和 Apache Tomcat。因为它们可以管理包含 Servlet 和 JSP 的 J2EE 模块，但无法管理可能包含 EJB J2EE 模块的完整的 J2EE 应用程序。 |
| *J2EE 应用程序 | 由一个或多个 J2EE 模块和一个 J2EE 应用程序部署描述符组成，使用 Java 归档 (JAR) 文件格式打包成具有 `.ear` （企业归档）文件扩展名的文件。 |
| Web 应用程序框架模块 | 是指 Web 应用程序框架应用程序中的内容和组件的逻辑和物理分区两者 （勿与 J2EE 模块混淆）。 |
| Web 应用程序框架应用程序 | 简明地来讲， Web 应用程序框架应用程序是使用 Web 应用程序框架编写的 J2EE Web 应用程序。它至少由一个 J2EE 模块 （Web 应用程序）组成，但可能还包括其他标准 J2EE 组件或模块。最小的 Web 应用程序框架应用程序是由一个 WAR 文件组成的 J2EE Web 应用程序。严格地讲，Web 应用程序框架应用程序是在同一 servlet 上下文中运行的所有相关 Web 应用程序框架模块的集合。在这种意义上，Web 应用程序框架应用程序仅指此逻辑 Web 应用程序框架概要。 |

* 有关此术语的详细解释，请参阅 “Java 2 平台企业版规范 1.2 版” (J2EE) 的 J2EE8.1 部分。

# Web 应用程序框架应用程序的组织结构

Web 应用程序框架提供正式的应用程序和模块实体。 Web 应用程序框架应用程序是一个 Java 基包，包含一个或多个子包 （Web 应用程序框架模块）。一个应用程序只由一个模块组成是完全可行的，对于小型应用程序而言，这可能是常见情况。每个模块都继承其父应用程序级组件中的行为，也可以从其他模块单独定制此行为。

就 J2EE Web 应用程序容器而言， Web 应用程序框架应用程序与 servlet 上下文一一对应，因此会受到 servlet 上下文容器所施加的限制。

开始开发应用程序前，应先确定其组织结构：

- 确定哪些模块将一起组合到 Web 应用程序框架应用程序中。

  应避免仅仅因为 Web 应用程序框架提供了此功能而将应用程序过细地分成几个模块。许多情况下，使用一个模块即可。

- 确定应用程序包名。

  应用程序包名的复杂程度各异，并很可能反映您所在组织的打包策略。每个模块都会成为此应用程序包下的一个包。

- 分配部署时间或发布的 Web 应用程序名。

  在 Apache Tomcat 中， `/webapps` 的下一级目录会使用此名称。在 Sun Java System Application Server 中， `$instance_dir/applications/j2ee-modules` 的下一级目录会使用此名称。已部署的应用程序名与 WAR 文件名相同。

例如，如果 Web 应用程序框架应用程序由两个 “应用程序框架”模块 （名为 *module1* 和 *module2*）组成，可将此应用程序称为 *myapp*。完整的应用程序包名为 `com.mycompany.myapp`。

- 应用程序包应为 `com.mycompany.myapp`
- *module1* 包为 `com.mycompany.myapp.module1`
- *module2* 包为 `com.mycompany.myapp.module2`

通常，应用程序包不应与其任一模块同名。

例如，您的第一个直觉可能是将应用程序及其主模块都命名为 *foo*。这样做容易使试图了解您的应用程序及应用程序开发工具的人产生混淆。应考虑将应用程序包命名为类似 *fooapp* 这样的名称，或将主模块命名为类似 *main* 或 *module1* 这样的名称。这样做可以更容易地理解应用程序结构，对今后向其增加内容尤为有利。

# 关于 Web 应用程序框架教程

现在要做的是开发一个简单的应用程序来体验使用 Web 应用程序框架及其工具的感觉。此应用程序由两个页面组成：一个登录页面和一个客户帐户页面，可以展示以下内容：

- 检索用户提交的字段值。
- 将状态消息返回给用户。
- 使用 QueryModel 检索客户信息。
- 使用 QueryModel 更新客户信息。
- 利用 QueryModel SQL WHERE 条件协调用户输入。
- 从一页转到另一页。
- 使用 WebServiceModel 来执行 Google Internet 搜索。

- 显示 WebServiceModel 的多个搜索结果。

本教程分为多个小节，以任务的形式介绍了开发应用程序所需的步骤。每小节都针对一个重要主题，学完每一节后，您就构建了一个可以运行的应用程序。

其中每章中的每个任务都是一个相对独立的主题，由几个更详细的步骤组成。

第 3 章

# 教程章节 （链接）

本章概述了包含在此 *《Web 应用程序框架教程》* 中的各节。

本部分列出了指向各任务的链接：

- 第 1.1 - 1.3 节
- 第 2.1 - 2.6 节
- 第 3.1 - 3.3 节
- 第 4.1 - 4.5 节

## 第 1.1 - 1.3 节

学完 1.1 到 1.3 节部分后，可以创建以后各章所需的应用程序基础结构，并增加第一个 Web 应用程序框架页。

- 第 1.1 节

  任务 1：新建 Sun Java StudioWeb 应用程序

- 第 1.2 节

  任务 2：创建登录页

- 第 1.3 节

  任务 3：测试运行登录页

## 第 2.1 - 2.6 节

学完 2.1 到 2.6 节后，可以增加基于 SQL 的模型和显示该模型数据的页来扩展现有应用程序。然后将这两个应用程序页链接在一起，使它们显示的数据协调一致。

- 第 2.1 节

  任务 1：访问 SQL 数据库
- 第 2.2 节

  任务 2：创建 CustomerModel
- 第 2.3 节

  任务 3：创建客户页
- 第 2.4 节

  任务 4：测试运行客户页
- 第 2.5 节

  任务 5：将登录页链接到客户页
- 第 2.6 节

  任务 6：运行应用程序

# 第 3.1 - 3.3 节

学完 3.1 到 3.3 后，可以创建由同一应用程序内的许多按钮和 HREF 重用的“命令”组件。对于父容器视图类内部的按钮或 HREF 的处理请求事件，这是实现请求处理代码的另外一种技术。

- 第 3.1 节

  任务 1：创建命令组件
- 第 3.2 节

  任务 2：向客户页增加 HREF
- 第 3.3 节

  任务 3：测试运行登录 / 注销命令

# 第 4.1 - 4.5 节

学完 4.1 到 4.5 节后，可以增加基于 Web 服务的模型和显示该模型数据的页来扩展现有应用程序。需要注册并下载 Google 开发人员的 SDK，才能为 Web 服务构建模型。

- 第 4.1 节

  任务 1：Web 服务用户注册和下载

- 第 4.2 节

  任务 2：创建 Google 搜索页
- 第 4.3 节

  任务 3：测试运行 Google 搜索页
- 第 4.4 节

  任务 4：创建 TiledView 页面配件
- 第 4.5 节

  任务 5：测试运行 Google 搜索页并产生结果

第 4 章

# 教程 1.1 节
# 应用程序基础结构

本章介绍了如何创建所有后续任务所需的 Sun Java Studio Web 应用程序框架 （也称 Web 应用程序框架、 App 框架、 SJSAF 和 JATO）应用程序基础结构。

## 任务 1：新建 Sun Java StudioWeb 应用程序

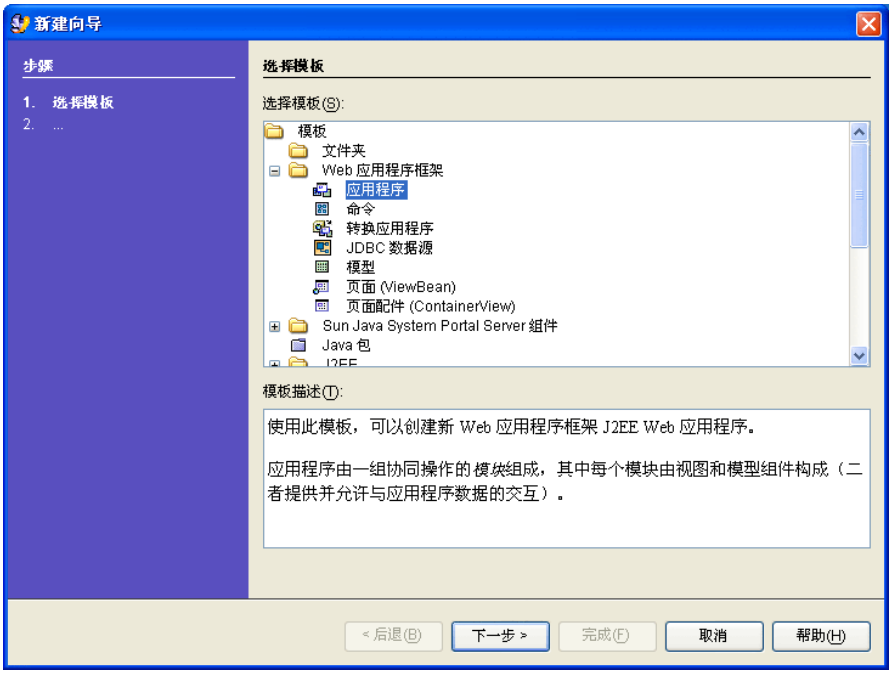
在开发任何页面之前，需要创建 Web 应用程序框架应用程序基础结构 （WAR 目录结构和支持文件）。这是每个 Web 应用程序框架应用程序的先决要求。

### 创建应用程序向导

创建应用程序之前，需要决定应用程序的放置位置。开发人员通常直接在 servlet 容器的 `webapps` 目录中开发应用程序，这样，应用程序不用部署到目标运行环境就可以进行测试。由于已使用 Sun Java Studio (Studio)，因此可以将应用程序放在任意位置，并且可以使用内置的 Sun Java System Application Server 模块就地对其进行测试。

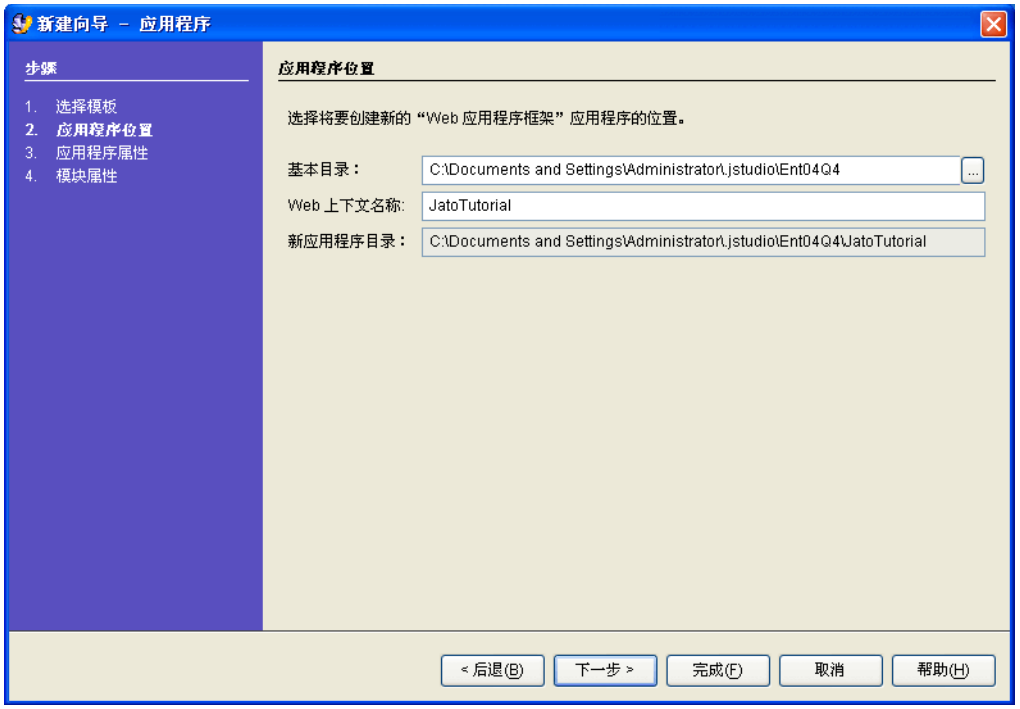
1. **选择 “Sun Java Studio Enterprise 7 菜单选项文件” -> “新建 Web 应用程序框架应用程序”。**

   显示 “选择模板” 面板。




2. **展开 “Web 应用程序框架” 文件夹。**
3. **选择 “应用程序”。**
4. **单击 “下一步”。**

   显示 “应用程序位置” 面板。




缺省的基本目录为 Sun Java Studio `user-dir`，该目录可能与本例中所示的目录不同。可以选择任何现有目录作为 Web 应用程序框架应用程序的基本目录。

---

**注 --** 许多开发人员使用在其中部署应用程序的 servlet 容器的 `webapps` 目录。在本教程的后面，您将了解如何使用 Sun Java Studio 来运行 Web 应用程序框架应用程序，这样您便可以将 Web 应用程序放到您想要放置的任何地方。

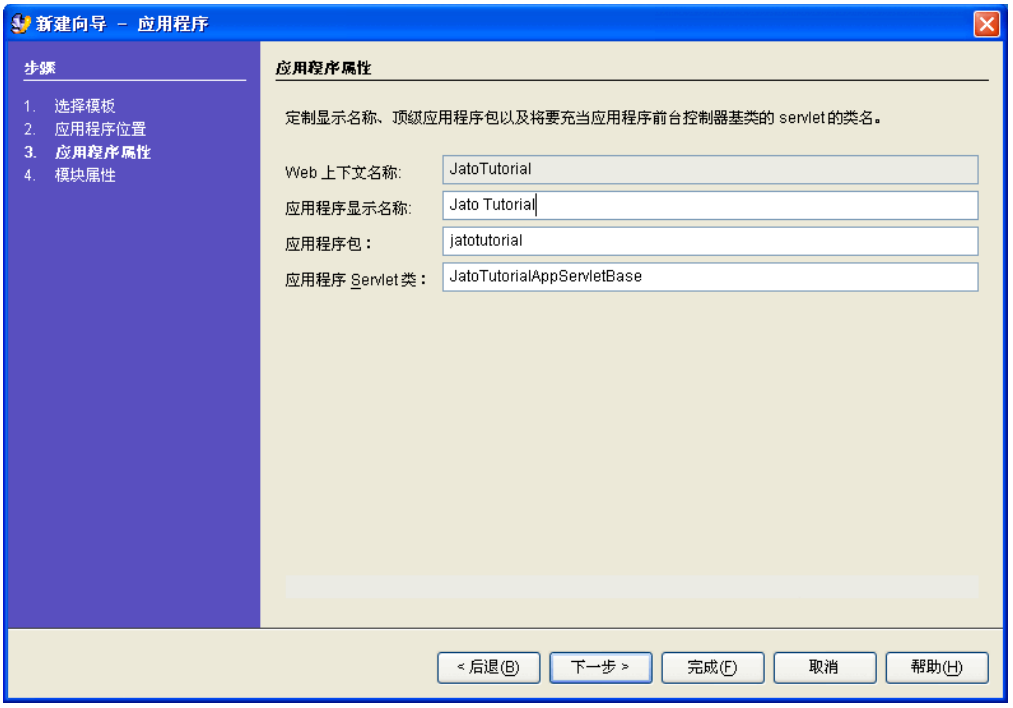
---

5. **在“Web 上下文名称”字段中输入 `JatoTutorial`。**

   在“基本目录”和“Web 上下文名称”字段中进行输入后，会填充“新应用程序目录”字段。

6. **单击“下一步”。**

   显示“应用程序属性”面板。

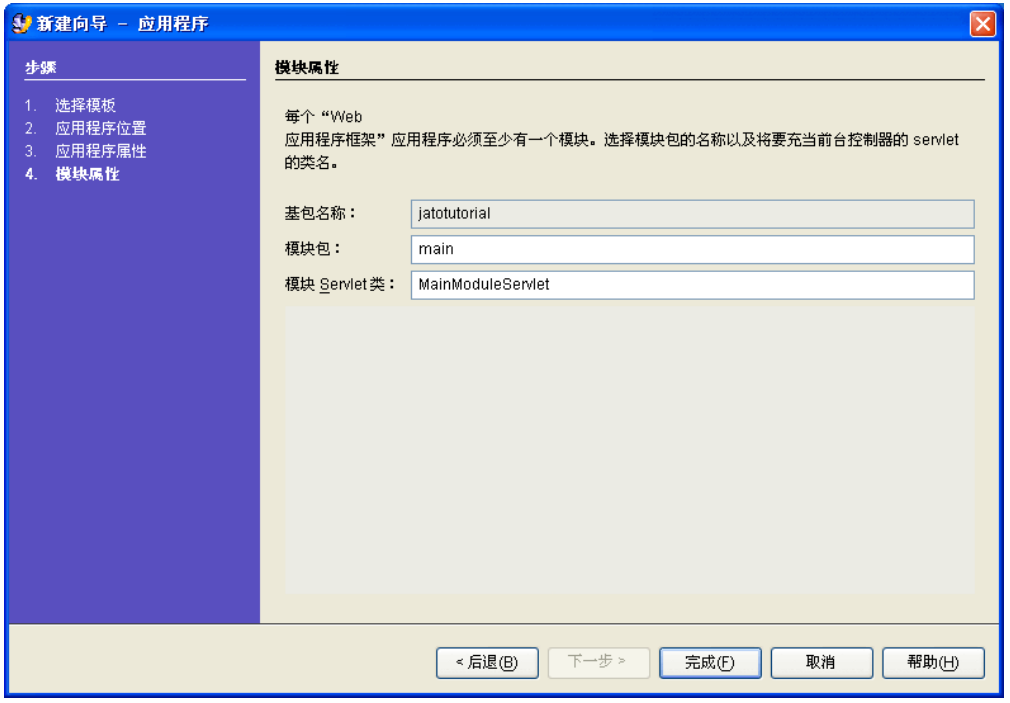



该面板中的字段使用前一面板中的 *“Web 上下文名称”* 字段值进行填充。

对于本教程，请接受缺省值。

7. **单击 “下一步”。**

显示 “模块属性” 面板。




对于本教程，请接受缺省值。

8. **单击 “完成”。**

9. **关闭弹出式对话框。**

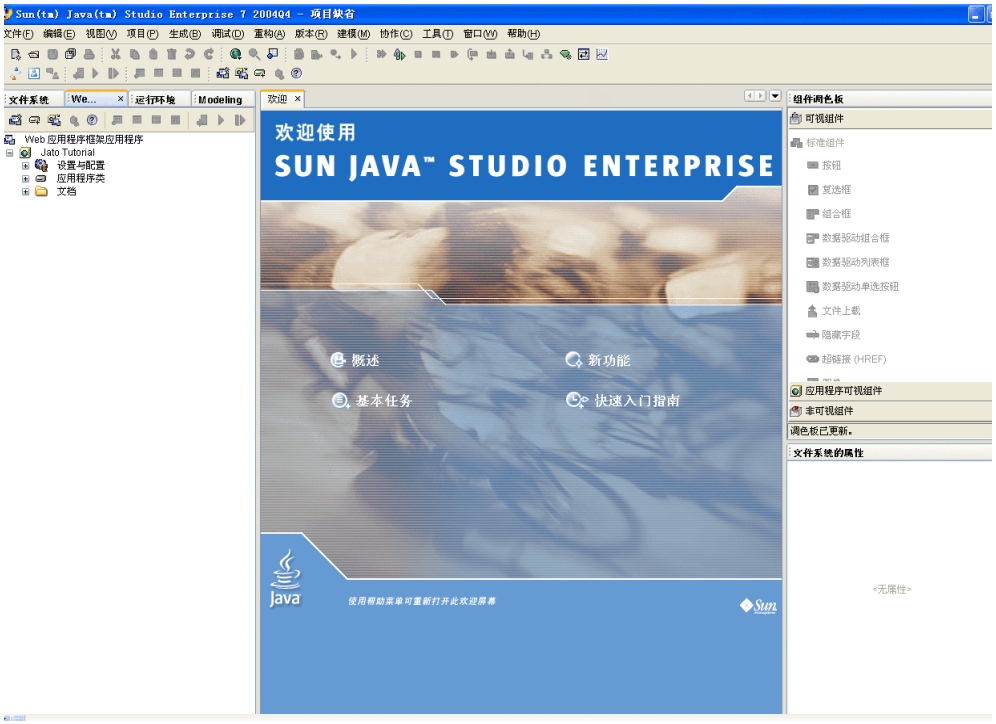
   将会创建应用程序。

---

**注 --** 处理时间取决于您使用的机器。

---

新应用程序显示在标为 “Web 应用程序框架应用程序” 的 IDE 资源管理器窗口的 “Web 应用程序框架应用程序” 树中。

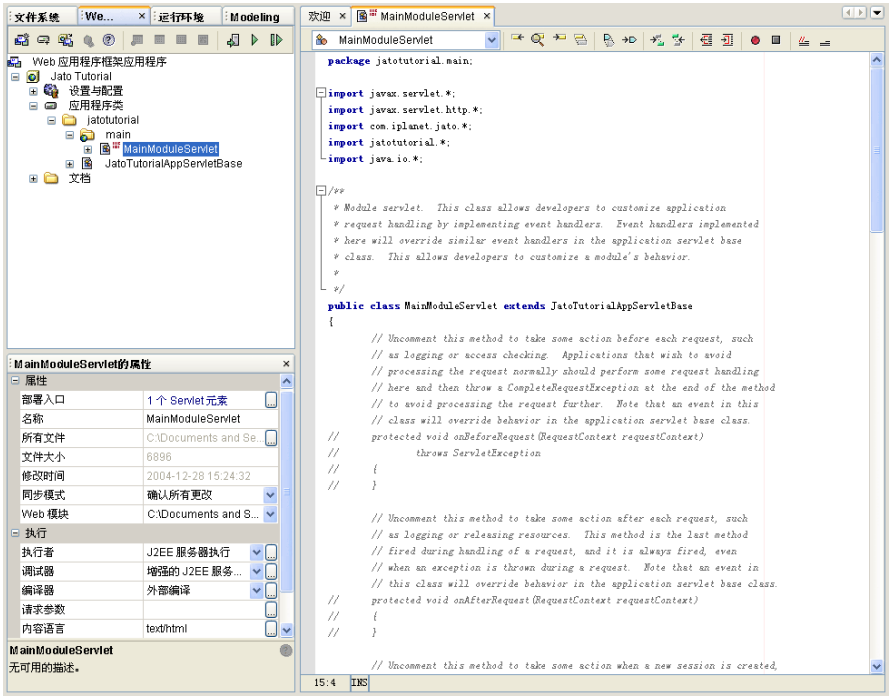



**注 --** 上图即为单击 “确定”后产生的结果。此时 “属性”窗口有可能不可见，特别是在安装新的 IDE 后会不可见。如果要切换显示各个窗口，请使用 “窗口”菜单，然后选择要显示或隐藏的窗口。

10. **展开 “Web 应用程序框架应用程序”资源管理器标签中的 “应用程序类”节点，来查看应用程序布局并观察所创建的两个 servlet 类中的代码。**
    - JatoTutorialAppServletBase
    - MainModuleServlet

11. **“属性”窗口的位置有可能位于您不希望的位置。为了在新的 IDE 中来回移动各个窗口，请选择窗口标题栏 （单击并按住鼠标左键），然后将其拖到您希望的位置。**

    在拖曳的过程中， IDE 将以红色显示窗口新位置的轮廓，以此表明所在的位置以及外观。各个窗口可以停靠在屏幕大小的四分之一处，或者将这些窗口作为标签面板与其他窗口停靠在一起。下图显示了放置在 “浏览器”视图下方左下角的 “属性”窗口。




# 应用程序 Servlet

应用程序 servlet `JatoTutorialAppServletBase` 除了表示它是应用程序中所有模块 servlet 的超类之外，对应用程序无任何特殊意义。

可以实现 Web 应用程序框架模块 servlet 所具有的事件来定制并控制会话和请求生命周期。

例如：

- onNewSession
- onSessionTimeout
- onBeforeRequest
- onAfterRequest

通常，同一应用程序中的所有模块 servlet 均要求所有事件具有相同的行为。因此，在所有模块 servlet 都可以扩展的类中为这些事件实现此种行为是一个不错的想法。

但是，从技术上讲，应用程序 servlet 不是必需的。可以定制模块 servlet 的分层结构，只需该分层结构从 Web 应用程序框架的 `com.iplanet.jato.ApplicationServletBase` 文件中派生而来即可。

此应用程序只有一个模块，定义为一个模块 servlet。因此，应用程序 servlet 并不具有它在多模块应用程序中的作用那样大。

## 模块 Servlet

模块 servlet `MainModuleServlet` 是为每个请求调用的实际 servlet。在将控制权交给适当的*请求处理程序*类（在本指南后面实现）之前，对此应用程序的所有访问都通过*前台控制器* servlet 进行。

此类中无需太多的代码。所有必需的请求处理代码位于 Web 应用程序框架的 `com.iplanet.jato.ApplicationServletBase` 文件中。通过查看 `com.iplanet.jato.ApplicationServletBase` 类中的源代码，高级开发人员可以对如何处理请求有一定的了解。

## 高级提示 - 模块

请注意，如果选择了 `main` 模块文件夹，其属性将在“Studio 资源管理器”窗口底端的属性表单中得以反映。注意，其“模块”属性为 *True*。将其更改为 *False*，该模块将变成一个普通的文件夹 / 包，并且会删除 `MainModuleServlet` 的 `Web.xml` 文件（标准 Web 应用程序配置文件）中的条目。

可以右键单击任何一个普通的文件夹并选择 *“转换为模块”* 操作，使该文件夹成为 Web 应用程序框架模块。然后，系统会提示您从该文件夹中选择一个 Java servlet 类作为模块 servlet，或者您也可以提供一个名称，创建一个新的模块 servlet。

第 5 章

# 教程 1.2 节
# 创建登录页

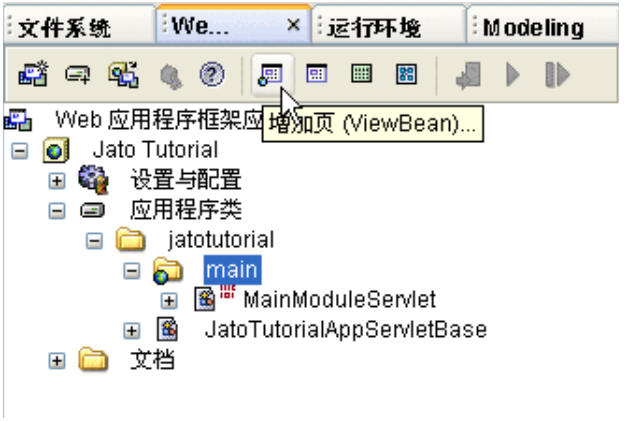
本章说明如何向您创建的应用程序基础结构增加第一个 Web 应用程序框架页。

## 任务 2：创建登录页

创建应用程序的第一页。

### 增加 ViewBean

1. **从 “Web 应用程序框架应用程序”资源管理器标签中选择 “main”模块文件夹。**




2. **单击 Web 应用程序框架工具栏上的 “增加页 (ViewBean)”按钮**

   或：

a. 选择 IDE 的菜单选项 “文件” -> “新建”。

b. 展开 “Web 应用程序框架” 节点。

c. 选择 “页 (ViewBean)”。

d. 单击 “下一步”。

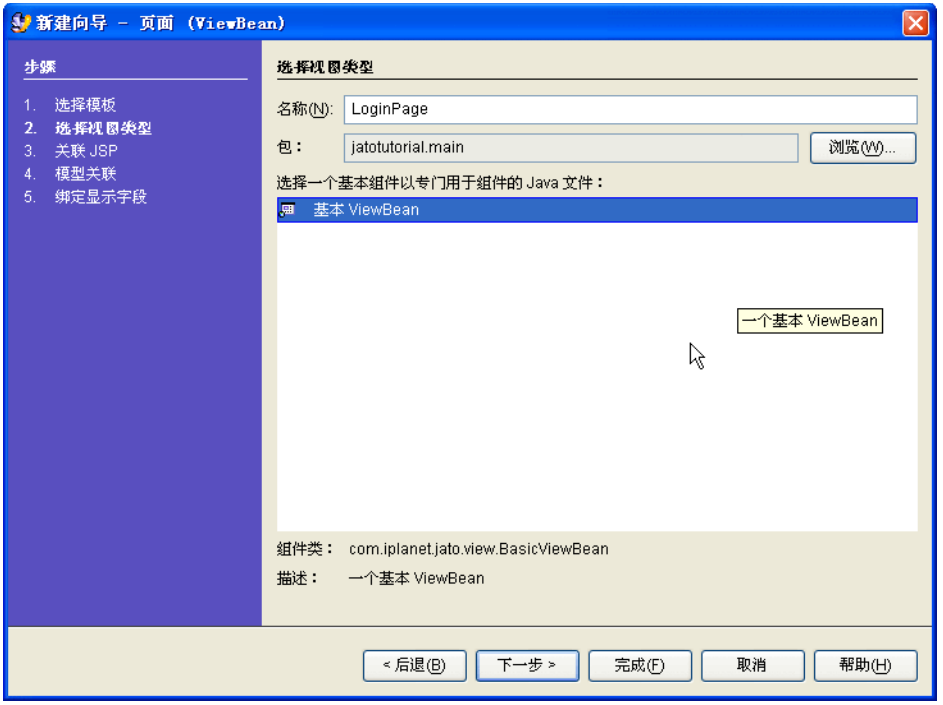
或：

a. 右键单击 “main” 模块文件夹。

b. 选择 “增加”。

c. 选择 “页 (ViewBean)”。

显示 “选择视图类型” 面板。

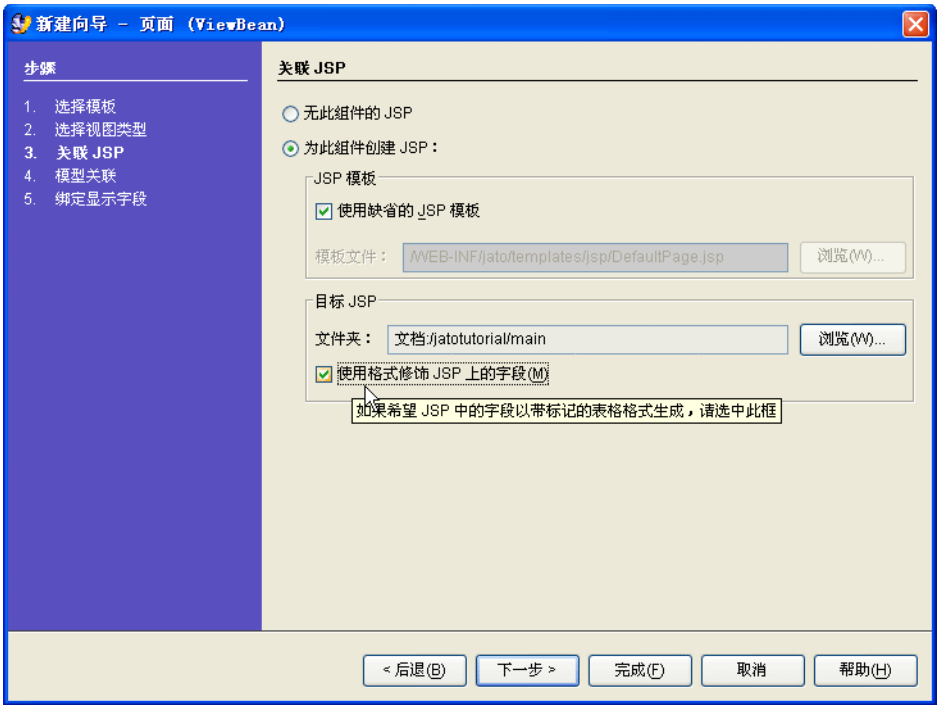



3. 在 “名称” 字段中输入 “LoginPage”（来替换 < *缺省值* >）。

4. 在基本组件列表中，选择 “*基本 ViewBean*”。

5. 单击 “下一步”。

   此时会显示 “关联 JSP” 面板。




6. **接受所有缺省选项，但是必须选中“*使用格式修饰 JSP 上的字段*”复选项。**

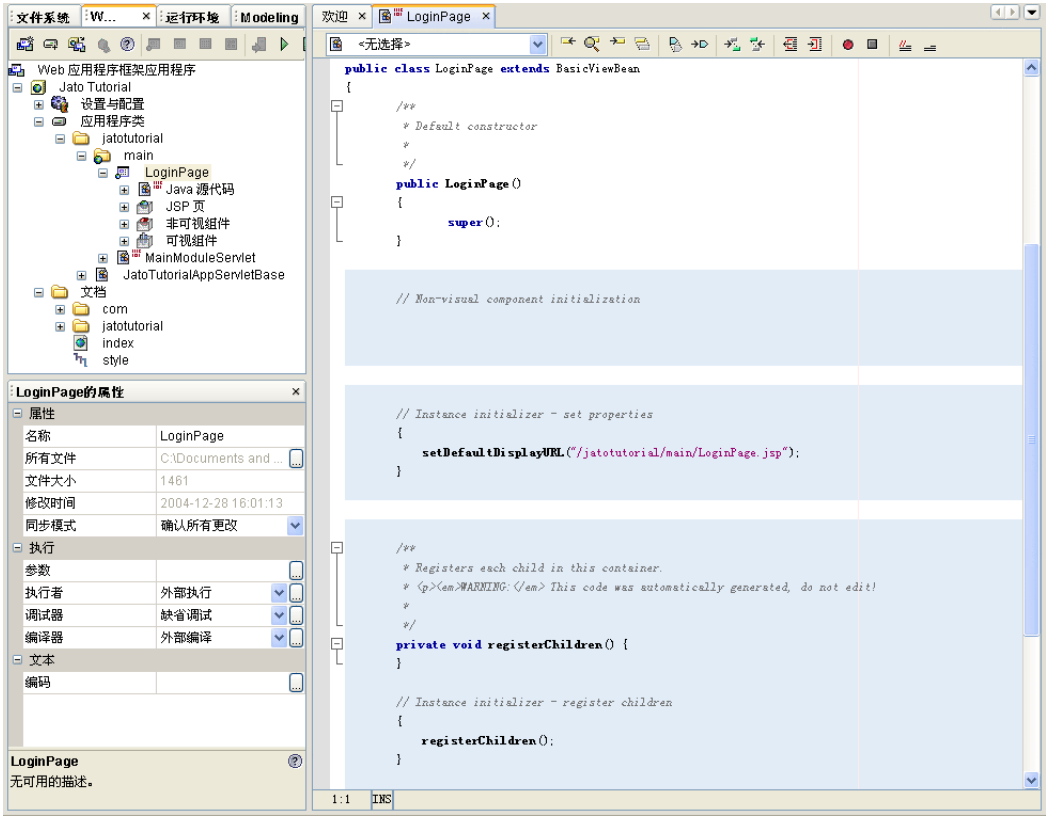
7. **单击“完成”。**

   ViewBean 创建完毕。

---

**注 --** “页”向导中还有其他一些步骤。但这些步骤涉及“LoginPage”并不需要的模型字段绑定。在以后的任务中，将会用到这些附加步骤。

---




8. **双击 “LoginPage”。**

   生成的源代码显示在 IDE 的源编辑器中。

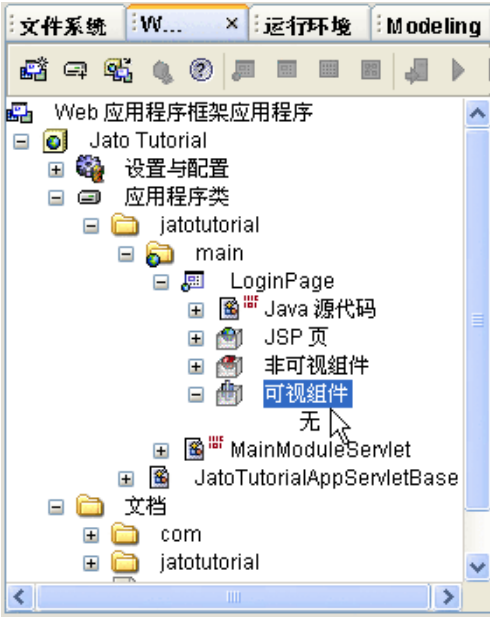
---

**注 --** 由于创建 LoginPage 时选择了创建 JSP，因此系统会将 JSP 增加到镜像 ViewBean 包结构的目录结构 (/jatotutorial/main) 的 “文档”文件夹中。为了方便起见，系统在 “JSP 页”节点中放置了一个使用 “LoginPage”的 JSP 的链接，它位于 “LoginPage”节点下。

---

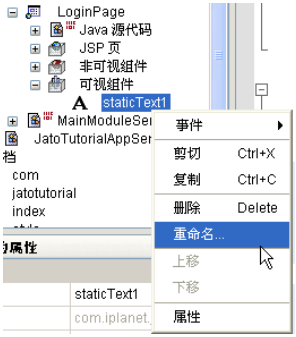
## 向登录页中增加显示字段

1. 展开 “LoginPage” 节点。
2. 选择 “LoginPage” 节点下的 “可视组件” 节点。




3. 在 Web 应用程序框架 “组件选择面板” 中，单击 “静态文本字段” 选项。


此时一个静态文本可视组件便被增加到 “可视组件”节点中。缺省名称为 “*staticText1*”。




4. **右键单击 “*staticText1*”字段名。**
5. **选择 “重命名”。**

6. **将字段重命名为 *message*。**

7. **再增加两个显示字段。**

   下表列出了两种可视组件类型，以及每种类型的名称和 *“按钮”*类型的初始值。

| 类型 | 名称 | 初始值 |
|---|---|---|
| 文本字段 | customerNum | |
| 按钮 | login | 对象类型：字符串<br>对象值：Login |

   这三个显示字段显示在 “LoginPage”的 “可视组件”节点下。


   向 “页”中增加显示字段，同时会将显示字段的相应 JSP 标记增加到使用此 “页”的 JSP 中。

8. **选择 *“login”*节点来设置按钮的 “初始值”属性。**

9. **单击 “初始值”属性值输入区域。**

10. **输入字符串 *Login*。**

    按钮的值便是显示在浏览器按钮上的字符串。


11. **打开 “LoginPage” 的 JSP 来查看三个显示字段的标记。**

    a. **展开 “LoginPage” 节点下的 “JSP 页” 节点。**

    b. **双击 “LoginPage JSP” 以在 IDE 的源编辑器中打开它。**


12. **自行设置 JSP 布局格式。**

---

**注 --** 由于在页向导中选中了修饰 JSP 页内容的选项，因此可能已在最初应用了某种基本的 HTML 格式。自动提供何种确切的附加格式取决于增加到父视图组件中的子视图组件种类和状态。因此，有可能需要修改更多的内容，因为在手动增加这一简单可视组件后没有使用其他的格式。

---

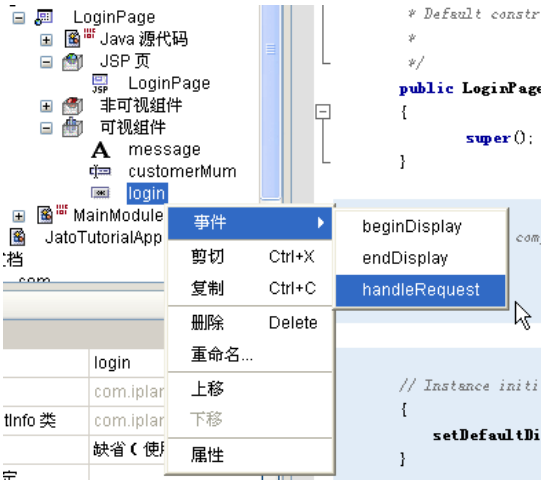
可以直接在 IDE 的源编辑器中编辑它，也可以使用自己喜欢的所见即所得 HTML 编辑器进行编辑。

在以下示例中，对 JSP 做了很少的更改 （此处只显示相关代码）。某些 HTML 源代码以**粗体**形式出现，以示强调。

```
<jato:form name="LoginPage" method="post">
<table border=0 cellspacing=2 cellpadding=2 width="100%">
<tr>
  <td align=right valign=middle width="20%"></td>
  <td align=left valign=middle><jato:text name="message"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"><b>Customer Num:</b></td>
  <td align=left valign=middle><jato:textField name="customerNum"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"></td>
  <td align=left valign=middle><jato:button name="login"/></td>
</tr>
</table>
</jato:form>
```

# 向 Login 按钮中增加代码

1. **右键单击 “login” 按钮。**
2. **选择 “事件” -> “handleRequest”。**




随即打开 `LoginPage.java` 文件，并插入 “`handleLoginRequest`” 方法桩模块。

3. **实现登录按钮处理请求代码。**

对于以下缺省代码：

```
getParentViewBean().forwardTo(getRequestContext());
```

请使用以**粗体**显示的代码替换：

```
public void handleLoginRequest(RequestInvocationEvent event) {
    // Retrieve the customer number
    String custNum = getDisplayFieldStringValue(CHILD_CUSTOMER_NUM);

    String theMessage = "";

    // Check the customer number
    if (custNum.equals("1") || custNum.equals("777") ||
            custNum.equals("410")) {
        theMessage = "Congratulations, " + custNum +
                ", you are now logged in!";
    } else {
        theMessage = "Sorry, " + custNum +
                ", your customer number was incorrect!";
    }

    // Set the output status message
    getDisplayField(CHILD_MESSAGE).setValue(theMessage);

    // Redisplay the current page
    forwardTo();
}
```

第 6 章

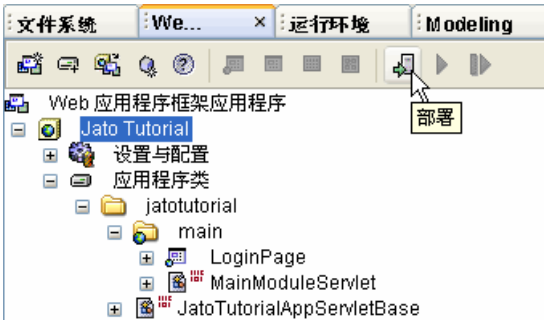
# 教程 1.3 节
# 测试运行登录页

本章说明如何运行 Web 应用程序框架应用程序。

## 任务 3：测试运行登录页

### 编译 Web 应用程序

1. **选择 “应用程序名称” 文件夹。**




2. **单击 “资源管理器” 窗口顶部的 Web 应用程序框架工具栏中的 “部署” 按钮。**

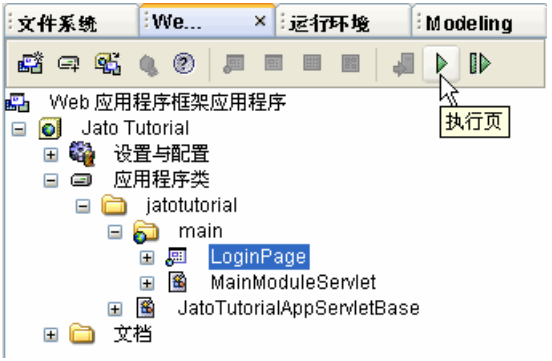
   这样会编译整个 Web 应用程序 （那些需要编译的类）并将其部署到 Sun Java System Application Server，所有操作只需一步完成。

   如果按照所有教程说明进行操作，便会无误地编译和部署 Web 应用程序。可以在 IDE 的输出窗口中查看是否有错误消息。

   在 Sun Java System Application Server（应用服务器）中运行 Web 应用程序时，对该应用程序中的任何资源所做的任何更改均必须执行这一部署步骤。

## 测试运行登录页

1. **选择 “LoginPage”。**




2. **单击 “资源管理器”窗口顶部的 Web 应用程序框架工具栏中的 “执行页”按钮。**

---

**注 --** “执行页 （重新部署）”按钮 （恰好位于 Web 应用程序框架工具栏上 “执行”按钮的右边）会强制 Sun Java System Application Server 重新装入所有资源 （例如，JSP 页、类等）。对于某些浏览器，可能需要关闭该浏览器的所有实例，才能在应用程序中重新运行任何页。

---

缺省浏览器会启动应用程序。

## 测试成功登录

1. **输入有效登录名 （例如， 1、 777 或 410 都是有效的 （硬编码）客户号码）。**
2. **单击 “Login”。**

---

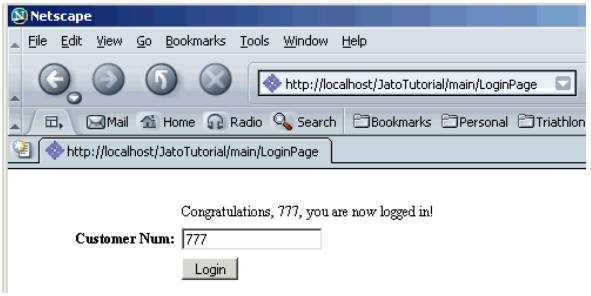
**警告 --** 有些 Web 浏览器，其中包括特定版本的 Microsoft Internet Explorer，如果在文本字段中按 Enter 键，则会为您提交此表单。但服务器并不知道执行此提交操作对应哪个按钮。*<jato:form>* 标记提供一个 *defaultCommandChild* 属性，用于告知服务器在缺省情况下应激活哪个按钮。

有关此功能的详细信息，请参阅标记库文档。

但目前只需直接单击该按钮即可。

---

此时登录页会刷新，并显示成功登录消息。

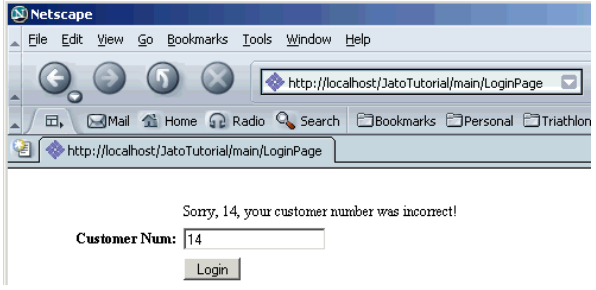



## 测试不成功登录

1. **输入一个无效的登录名 （例如， `foo`、 `8` 或 `14` - 除上述有效的硬编码客户号码外的任何字符）。**

2. **单击 “Login”。**

   登录页会刷新，显示失败消息。




## 其他运行环境

1. **如果想要在 IDE 外测试运行应用程序，请编译应用程序并将其打包到 WAR 文件中，然后将 WAR 文件放在 `webapps` 目录中 （目录名称会因容器而异，但大多数名为 `webapps`）。**

2. **需要将 PointBase 驱动程序文件增加到 servlet 容器的类路径中。驱动程序位于 IDE 的安装目录中，路径为：**

   在 Windows 中，位于 `<studio-install-dir>\AppServer7\pointbase\client_tools\lib\pbclient42RE.jar`。

   在 Solaris/Linux 中，位于 `<studio-install-dir>/AppServer7/pointbase/client_tools/lib/pbclient42RE.jar`。

   完成此操作的最简单方法是将驱动程序复制到应用程序的 `WEB-INF/lib` 目录中。

3. **打开浏览器，输入并运行与 servlet 容器对应的 URL。**

   唯一可能的变化是 URL 结尾的页名称 (LoginPage)。

   Apache Tomcat 或 Caucho Resin servlet 容器：
   `http://localhost:8080/JatoTutorial/main/LoginPage`

---

**注 --** 学习本教程期间，您可能认为有必要再次参阅此任务。

---

第 7 章

# 教程 2.1 节
# 准备应用程序来访问 SQL 数据库

本章说明如何扩展应用程序并准备 Web 应用程序框架应用程序访问 SQL 数据库。

通过增加基于 SQL 的模型和显示该模型数据的页来扩展现有应用程序。

将两个应用程序页链接在一起，以使它们显示的数据协调一致。

## 任务 1：访问 SQL 数据库

### 连接到样例数据库

教程中的以下内容假设有一个 RDBMS 数据库，它是向您介绍 Web 应用程序框架某些其他功能的先决条件。

并不要求 Web 应用程序框架应用程序一定要访问 RDBMS。因此，实际的应用程序可能不访问 RDBMS，而是访问要求另一种形式的准备、设置和连接的某个其他企业系统。

所执行的步骤取决于 IDE 的预配置 Pointbase 数据库，该数据库在 IDE 尝试使用预配置 JDBC 连接 （连接到样例数据库）时会自动启动。同时在自动安装的 Sun Java System Application Server 中，包含预配置数据库连接资源。每个新的 Web 应用程序框架应用程序将使用与先前存在的 Pointbase 样例数据库连接资源匹配的数据源进行预配置。这样，新的开发人员可以快速使用 Web 应用程序框架和 Pointbase 样例数据库，而不需要进行额外的配置。

### JDBC 数据源

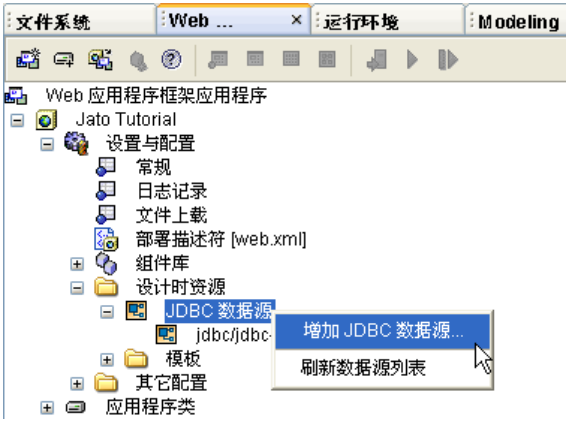
可以使用 Web 应用程序框架 JDBC 数据源向导来创建 JDBC 数据源。

但在缺省情况下，已创建了一个数据源，该数据源指向 IDE 随附的 PointBase 样例数据库。

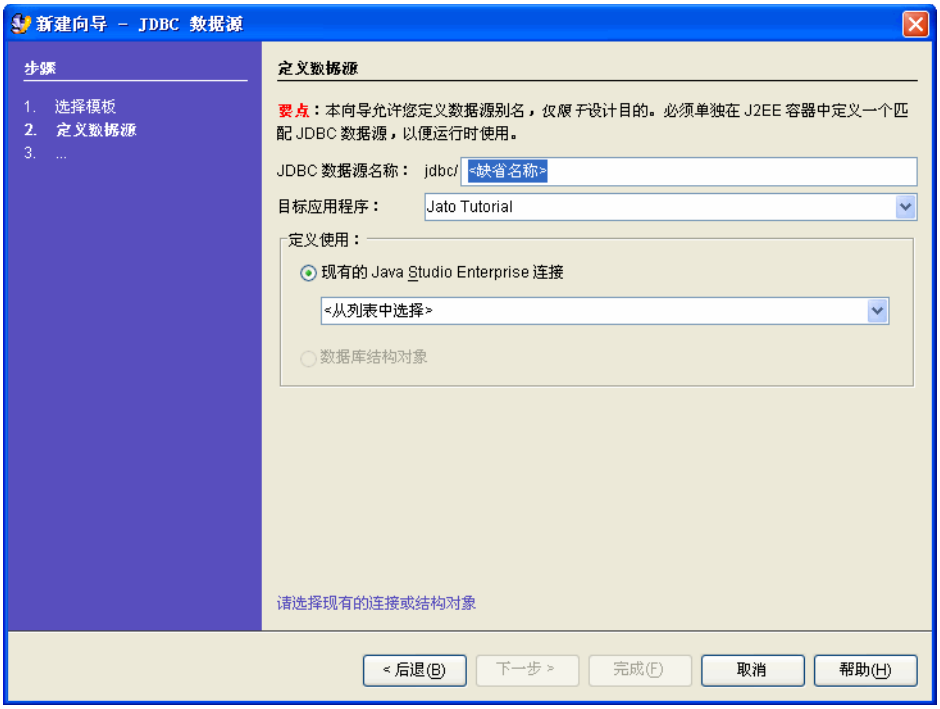
如果要为非本教程中使用的另一个数据库创建其他 JDBC 数据源，请执行以下步骤。

（否则，请通读本主题以了解其内容，或跳至 Tomcat （和其他非 JNDI 容器） SQL 连接准备部分。）

1. **在 Web 应用程序框架 Web 应用程序节点 (Jato Tutorial) 下，展开 “设置与配置”文件夹。**
2. **展开 “设计时资源”文件夹。**
3. **右键单击 “JDBC 数据源”节点。**
4. **选择 “增加 JDBC 数据源”。**



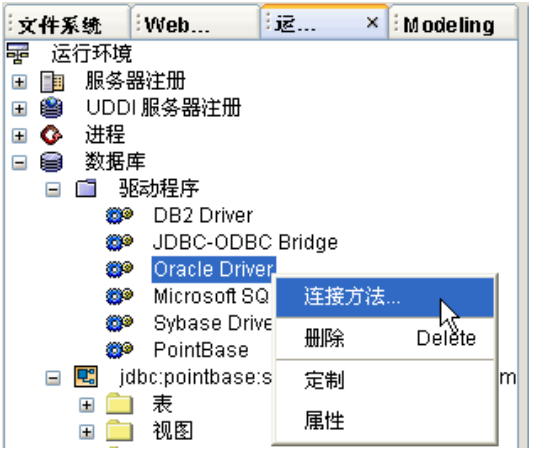

此时会显示 “定义 JDBC 数据源”面板。




5. **在新数据源名称文本框中输入希望使用的数据源名称。**

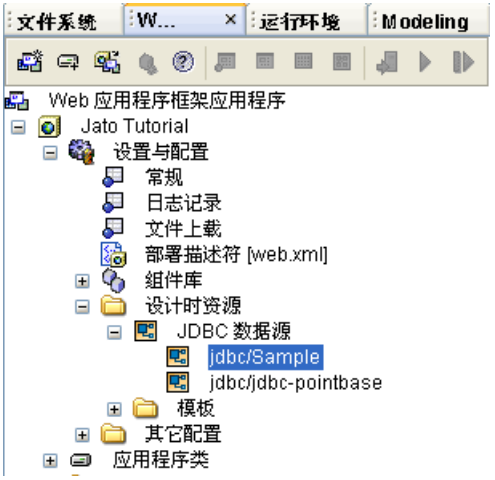
6. **在选择连接组合框中，选择合适的 JDBC 连接。**

   如果所需的连接不存在，则必须创建一个。由 Web 应用程序框架系列工具外的一个工具来执行创建操作。需要选择 “运行环境” 窗口，依次展开 “数据库” 节点和 “驱动程序” 节点，然后右键单击所需的驱动程序节点，并选择 “连接方法”。增加连接前可能需要为数据库增加驱动程序。有关详细信息，请参阅 IDE 联机帮助。




7. **单击 “完成”。**

此时便创建了一个新的 “JDBC 数据源”节点。




---

**注 --** 仅在创建 JDBC SQL 模型 （表和存储过程）的设计时才需要 JDBC 数据源。JDBC SQL 模型向导会提供一组已创建的数据源。

JDBC 数据源不包含在运行环境中。如果不使用直接连接到数据库的 JDBC URL，则必须用正确的 JNDI 设置配置运行环境容器。

---

# Tomcat （和其他非 JNDI 容器） SQL 连接准备

**注 --** 如果使用 Sun Java System Application Server 运行教程应用程序，可以跳过本步骤，因为系统支持 JNDI。

如果使用内置的 Tomcat 引擎，或在另一不支持 JNDI 的 servlet 容器中运行教程应用程序，则需要在应用程序中对应用程序 servlet 基类 (JatoTutorialAppServletBase) 做些细微的修改。

1. **展开 “应用程序类” 文件夹。**
2. **展开 “`jatotutorial`” 包文件夹。**
3. **双击 “`JatoTutorialAppServletBase`” 类，打开它。**

   其中有许多带有注释的事件代码，而注释描述了事件的用途。本教程应用程序并不需要它们，因此无需理会。

   需要增加静态初始化函数来执行以下操作：

   a. **指示 Web 应用程序框架不要使用 JNDI 查找**

   b. **装入 PointBase JDBC 驱动程序**

   c. **将 JDBC 数据源 (jdbc/jdbc-pointbase) 映射到 PointBase 样例数据库的 JDBC 连接 URL**

   下列代码样例显示需要增加到 JatoTutorialAppServletBase 类的代码。只需要增加**粗体**代码。此处省略了 JatoTutorialAppServletBase 类的许多代码 / 注释。

```
public class JatoTutorialAppServletBase extends ApplicationServletBase {
    static {
        // Turn off JNDI lookup (turn on DriverManager use)
        SQLConnectionManagerBase.setUsingJNDI(false);

        try {
            // load the PointBase JDBC driver
            Class.forName("com.pointbase.jdbc.jdbcUniversalDriver");
        } catch (ClassNotFoundException e) {
            // if the driver is unavailable, an exception will be thrown
            e.printStackTrace();
        }

        SQLConnectionManagerBase.addDataSourceMapping("jdbc/jdbc-pointbase",
                "jdbc:PointBase://localhost:9092/sample");
    } // static init
}
```

应用程序现在将使用 JDBC URL 直接连接到数据库，而不再通过 JNDI 使用连接池连接数据库。

**要点：**如果重视 Web 应用程序的性能，请使用 JNDI 来提高效率。

- 如果不在 Sun Java System Application Server 中测试教程，则需要将 PointBase 客户端库 JAR 文件 (`pbclient42RE.jar`) 复制到您的 `WEB-INF/lib` 目录中。

  可以从以下目录中获取 PointBase 客户端库：

  `<studio-install-dir>/appserver7/pointbase/client_tools/lib/`

- 如果使用的是另一数据库，可能需要将该数据库供应商的客户端库放入您的 `WEB-INF/lib` 目录或 servlet 容器的 `lib/ext` 目录 （在类路径中的某一位置）中。将它放入 Web 应用程序的 `WEB-INF/lib` 目录中。
- 对 Sun Java System Application Server 不必这样操作，因为 PointBase 库已包含在其类路径中。

第 8 章

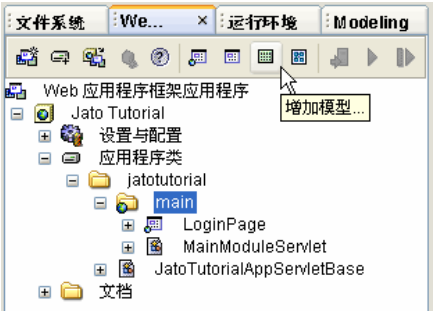
# 教程 2.2 节
# 创建 CustomerModel

本章说明如何在 Web 应用程序框架应用程序中创建模型来访问 RDBMS。

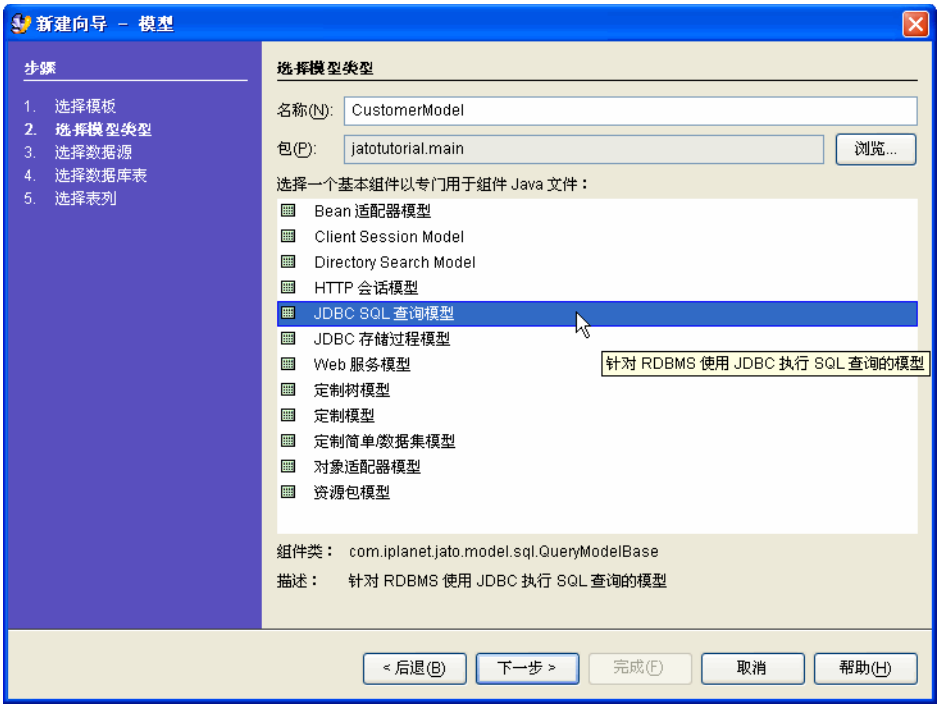
## 任务 2：创建 CustomerModel

### 创建 JDBC™ SQL 模型

1. **选择 “main” 模块文件夹。**
2. **单击 Web 应用程序框架工具栏上的 “增加模型” 按钮。**




此时会显示 “选择模型类型” 面板。



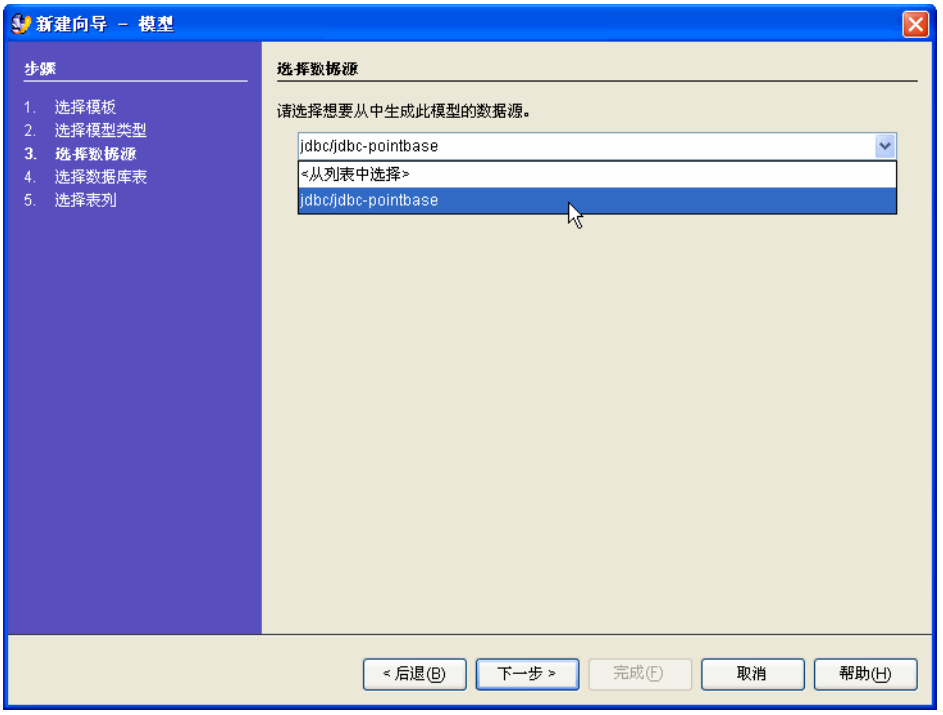

3. **在“名称”字段中输入 *CustomerModel*。**

4. **从模型组件列表中选择“JDBC SQL 查询模型”。**

   所见列表可能因 Web 应用程序框架版本及可能增加了定制或第三方组件库而有所不同。

5. **单击“下一步”。**

   此时会显示“选择数据源”页。




6. **从组合框中选择 “*jdbc/jdbc-pointbase*”。**

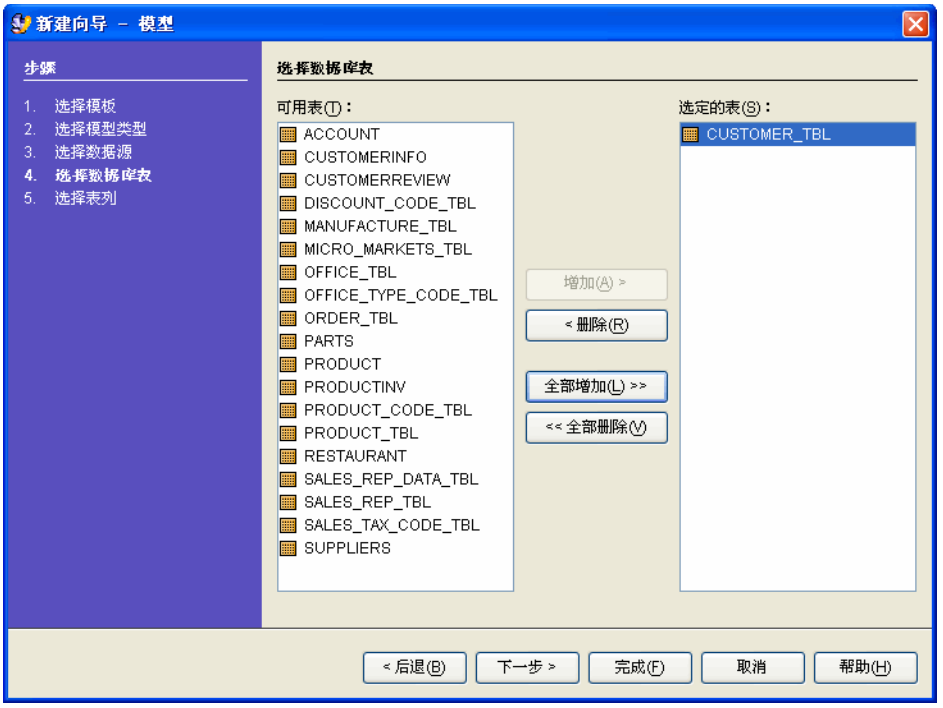
7. **单击 “下一步”。**

   此时会显示 “选择数据库表” 页。

---

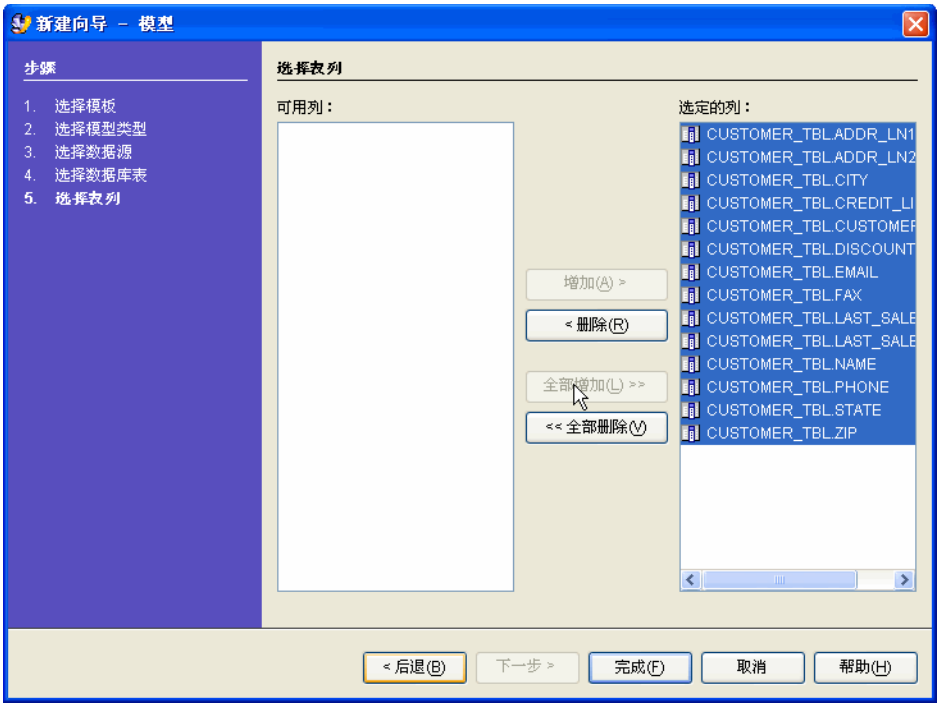
**注 --** 如果在单击 “下一步” 时出现 “连接” 对话框，可能需要输入连接凭证 （样例 Pointbase 数据库的登录用户名为 pbpublic，口令为 pbpublic）。

---




8. **选择 “CUSTOMER_TBL”。**
9. **单击 “增加”。**
10. **单击 “下一步”。**

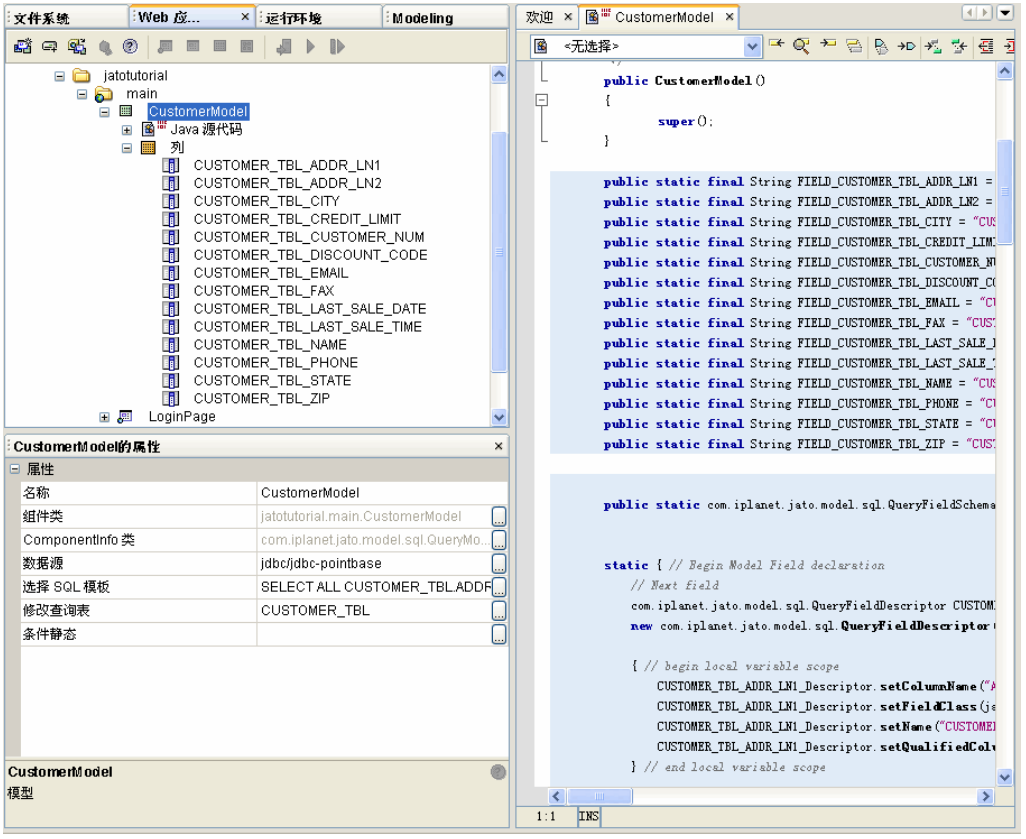
    此时会显示 “选择表列” 页。




11. **单击“全部增加”，在“模型”中包括全部列。**

12. **单击“完成”来创建“模型”。**

    此时 main 模块中便创建了“CustomerModel”对象。




13. **展开 “CustomerModel” 来查看全部列。**
14. **双击 “CustomerModel” 文件夹来查看 CustomerModel Java 类中的代码。**

# 标记模型的关键字字段

---

**注 --** 由于 PointBase 数据库结构元数据中存在一种特殊类型的关键字字段指示器，模型向导无法正确检测到关键字字段 `CUSTOMER_TBL_CUSTOMER_NUM`。因此，必须手动设置关键字字段。

在数据库结构对象中创建数据源，不存在任何问题；并且在非 PointBase 数据库 （如 Oracle）中创建数据源也不存在任何问题。

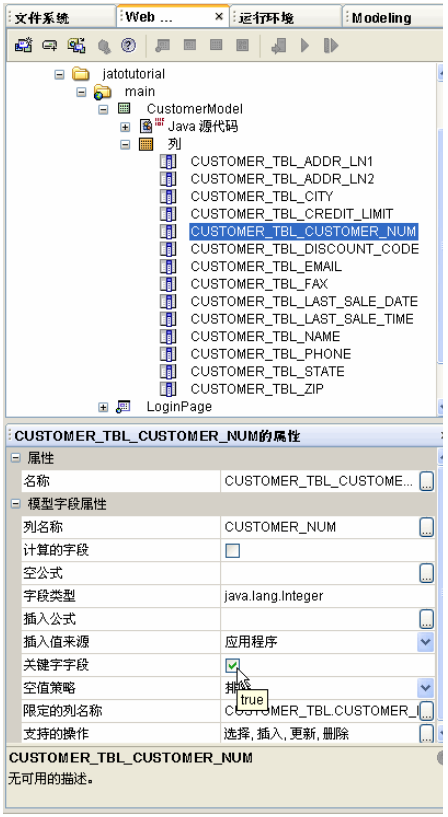
---

1. **在 “CustomerModel” 的 “列” 节点下，选择 “`CUSTOMER_TBL_CUSTOMER_NUM`” 模型字段。**

2. **在属性表单中，选择 “模型字段属性”标签。**

   如果 “属性”标签不可见，请单击 “窗口” -> “属性”菜单选项，或右键单击关键字字段列并选择 “属性”操作。

3. **将 “关键字字段”属性的值从 *false* 更改为 *true*。**




# 为不支持 JNDI 的容器增加连接代码

对于不支持 JNDI 数据源的 servlet 容器，则依赖对 JDBC 驱动程序的显式使用。

---

**注 --** 在本教程 2.1 节 （任务 1：访问 SQL 数据库）中指出，如果在不支持 JNDI 的 servlet 容器中测试此应用程序，则会禁用 JNDI 并在应用程序 servlet 类 (JatoTutorialAppServlet) 中声明对 PointBase JDBC 驱动程序的显式使用。

---

如果在不支持 JNDI 的 servlet 容器中测试，必须在模型中明确设置连接用户名和口令，这样在执行模型前便可创建正确的数据库连接。

---

**注 --** 对要求高效率的环境，应使用 JNDI 连接。

---

将下面的**粗体**代码增加到 “CustomerModel” 的构造函数中。

```
public CustomerModel() {
    super();
    setDefaultConnectionUser("pbpublic");
    setDefaultConnectionPassword("pbpublic");
}
```

对于实际使用的应用程序而言，提供如上所示的数据源用户名和口令并不是好的做法，但对本教程却很实用。应加倍小心，通过更安全、强健的实现过程来获取和提供用户名和口令。使用 JNDI 方法时，不必使用此代码，此登录信息由应用服务器中配置好的 JNDI 连接提供。

第 9 章

# 教程 2.3 节
# 创建客户页

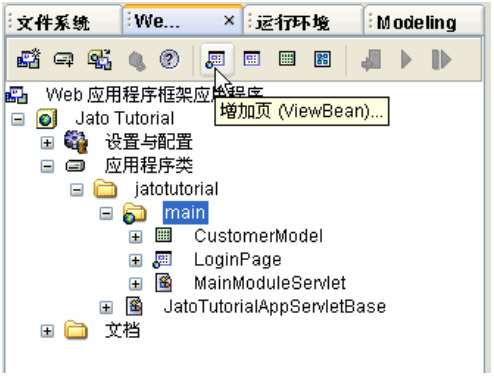
本章说明如何在 Web 应用程序框架中创建一个页面，以显示访问关系数据库的模型中的数据。

## 任务 3：创建客户页

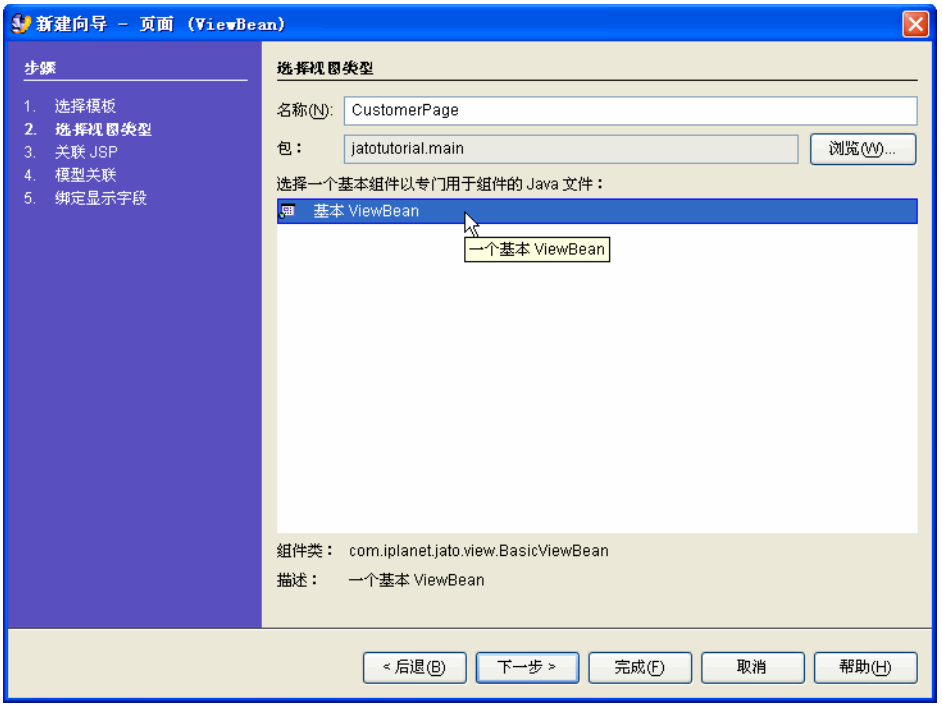
现在将创建应用程序的第二页。不过，此页将与模型绑定。绑定过程自动在显示数据的页上创建显示字段，该数据存储于模型字段中。

### 增加 ViewBean

1. **选择 “main” 模块。**
2. **单击 Web 应用程序框架工具栏上的 “增加页” 按钮。**

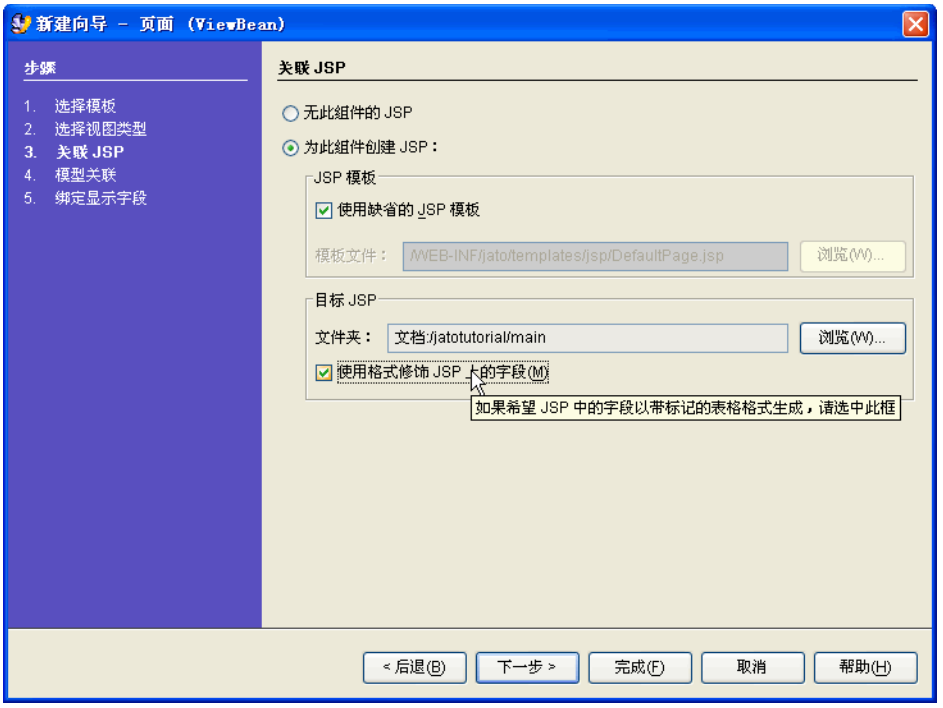



此时会显示 “选择视图类型” 面板。




3. **在“名称”字段中输入 CustomerPage （替换 < *缺省值* >）。**
4. **选择“基本 ViewBean”创建 ViewBean 类型的“页”组件。**
5. **单击“下一步”。**

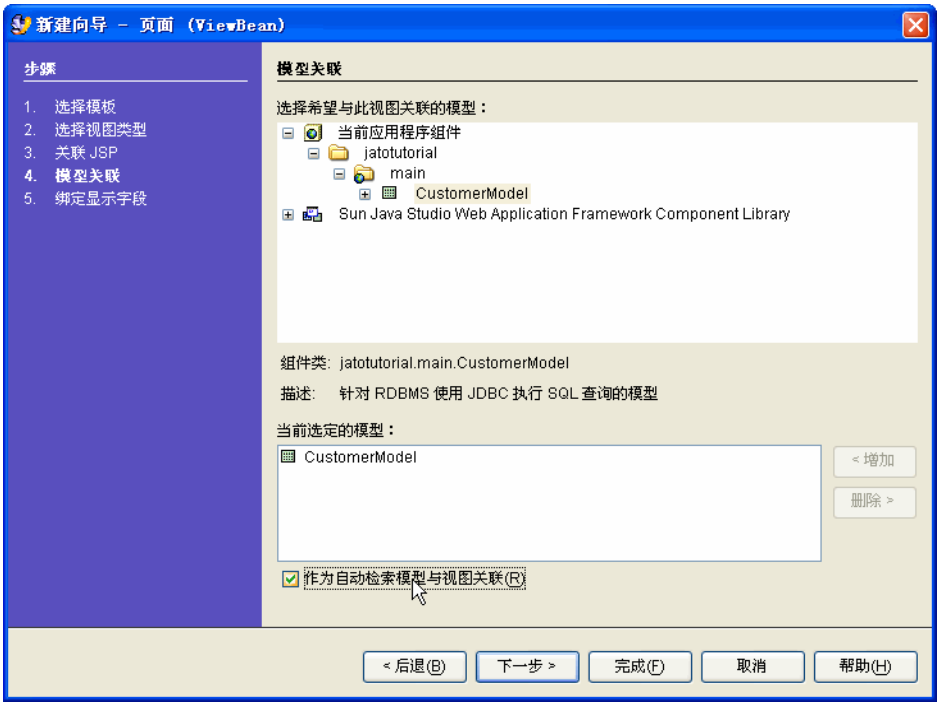
   此时会显示“关联 JSP”面板。



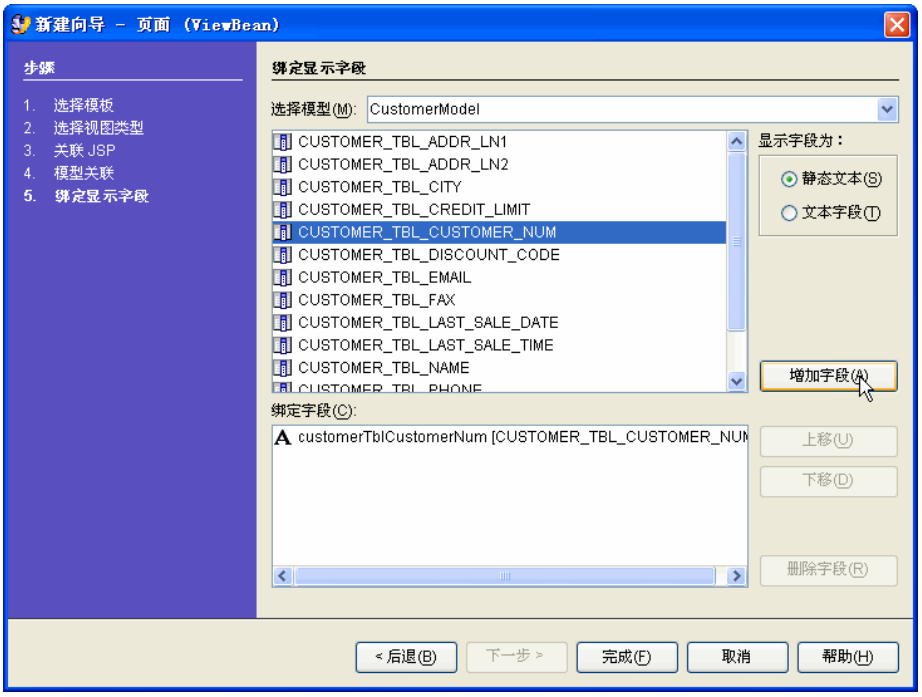

6. **单击“*使用格式修饰 JSP 上的字段*”复选框以应用某种基本格式设置。**

7. **单击“下一步”。**

   此时会显示“模型关联”面板。




8. **展开 “当前应用程序组件” 来显示 “jatotutorial” -> “main”。**
9. **选择 “CustomerModel”。**
10. **单击 “增加”。**
11. **选中 “*作为自动检索模型与视图关联*” 复选框。**
    这样，会自动将此模型配置为使用 RetrievingModel 功能。
12. **单击 “下一步”。**
    此时会显示 “绑定显示字段” 面板。




只需增加三个字段。

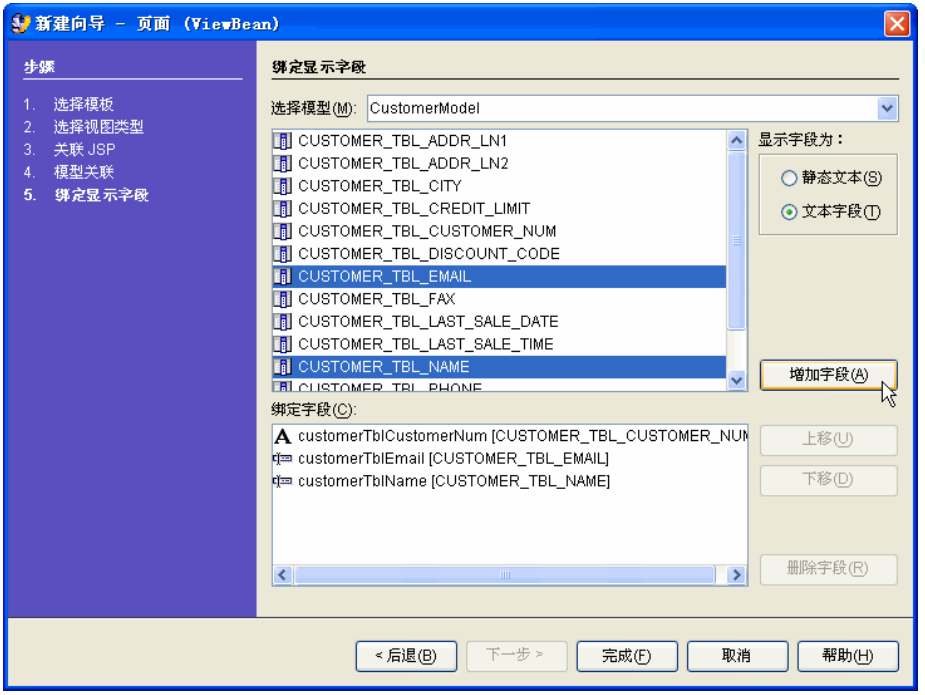
13. **增加第一个字段。**

    a. **选择 “CUSTOMER_TBL_CUSTOMER_NUM”字段。**

    接受缺省的 “静态文本”选项。

    b. **单击 “增加字段”。**

    “CUSTOMER_TBL_CUSTOMER_NUM”字段便会增加到 “绑定字段”列表框中。




14. **同时增加第二和第三个字段。**

    a. **选择 “CUSTOMER_TBL_EMAIL” 和 “CUSTOMER_TBL_NAME” 字段 （按住 Ctrl 键可选择多个非连续字段）。**

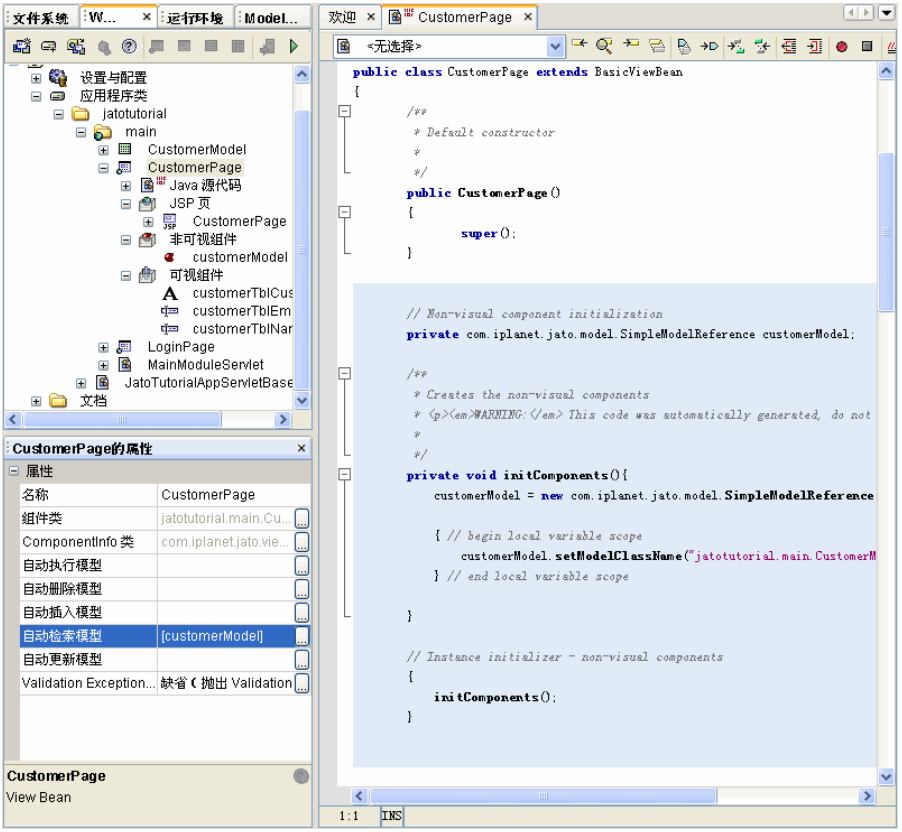
    b. **选择 “文本字段”。**

    c. **单击 “增加字段”。**

    “CUSTOMER_TBL_EMAIL” 和 “CUSTOMER_TBL_NAME” 字段便会增加到 “绑定字段” 列表框中。

15. **单击 “完成”。**

    ViewBean 创建完毕。


文件系统
W...
运行环境
Model...
设置与配置
应用程序类
jatotutorial
main
CustomerModel
CustomerPage
Java 源代码
JSP 页
CustomerPage
非可视组件
customerModel
可视组件
customerTblCus
customerTblEm
customerTblNar
LoginPage
MainModuleServlet
JatoTutorialAppServletBase
文档
CustomerPage的属性
属性
名称
CustomerPage
组件类
jatotutorial.main.Cu...
ComponentInfo 类
com.iplanet.jato.vie...
自动执行模型
自动删除模型
自动插入模型
自动检索模型
[customerModel]
自动更新模型
Validation Exception...
缺省 ( 抛出 Validation
CustomerPage
View Bean
欢迎
CustomerPage
<无选择>
public class CustomerPage extends BasicViewBean
{
/**
* Default constructor
*
*/
public CustomerPage()
{
super();
}
// Non-visual component initialization
private com.iplanet.jato.model.SimpleModelReference customerModel;
/**
* Creates the non-visual components
* <p>WARNING:  This code was automatically generated, do not
*
*/
private void initComponents(){
customerModel = new com.iplanet.jato.model.SimpleModelReference
{ // begin local variable scope
customerModel.setModelClassName("jatotutorial.main.CustomerM
} // end local variable scope
}
// Instance initializer - non-visual components
{
initComponents();
}
1:1
INS

16. **双击 “CustomerPage”。**

代码显示在右侧面板中。

展开 “CustomerPage”的全部子节点可查看由向导自动生成的 “JSP 页”、“可视组件”和 “非可视组件”。

---

**注 --** 与 LoginPage 一样， CustomerPage 的 JSP 也被增加到 “文档”文件夹 (`/jatotutorial/main`) 中，并且在此 ViewBean 的 JSP 文件夹下有指向该 JSP 的链接。

由于您指明要绑定到 “CustomerModel”的字段，因此可以看见已创建的三个可视组件。这样数据便可自动显示在 “客户”页上，对这些字段的更改会自动映射到模型中，因此您可以执行模型来对数据库进行更新。所有的 SQL 生成及连接创建均由 Web 应用程序框架来处理。

如果您确实想要 （需要）直接使用 JDBC API，该应用程序框架不会要求您使用它提供的所有功能，您可以自行随意使用所有的 JDBC 功能。换句话说，可以选择需要在 Web 应用程序框架中使用的功能，但大多数情况下 Web 应用程序框架的实现便可完全满足您的需要。

在 “非可视组件”节点下还会看到一个条目，它是对 “CustomerModel”类的一个*引用*。

---

## 增加按钮组件

1. **向 “CustomerPage”中增加按钮。**

   下表包含向 “CustomerPage”增加按钮的规范。使用 “组件选择面板”增加按钮与为 “登录页”增加字段的方法一样。

2. **如果 “组件选择面板”不可见，请选择菜单选项：**

   窗口 -> Web 应用程序框架 -> 组件选择面板。

   下表列出了向 “CustomerPage”增加按钮的说明。

| 类型 | 名称 | 初始值 |
|---|---|---|
| 按钮 | 更新 | 更新 |

3. **使按钮具有更新客户记录的功能。**

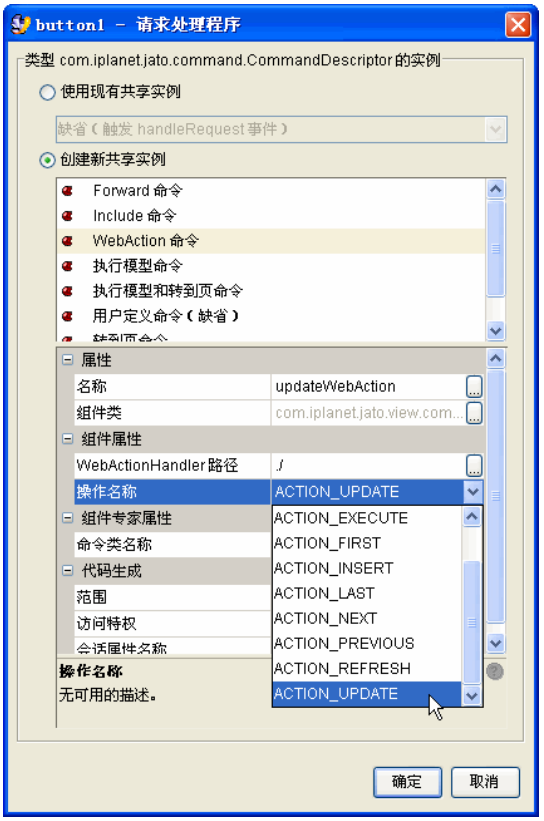
   a. **选择更新按钮字段。**

   b. **在按钮的属性表单中，单击 “*RequestHandler*”属性的值区域。**

      此时会显示省略号按钮。

   c. **单击省略号按钮。**

      此时会启动 “命令描述符”编辑器。




4. **选择 “创建新共享实例”。**
5. **从列表中选择 “*WebAction 命令*”。**
6. **在 “属性” 标签中，将名称更改为 “*updateWebAction*”。**
7. **选择编辑器底部的 “组件属性” 标签。**
8. **为 “操作名称” 属性选择 “ACTION_UPDATE”。**

   接受其他两个属性的缺省值。
9. **单击 “确定”。**

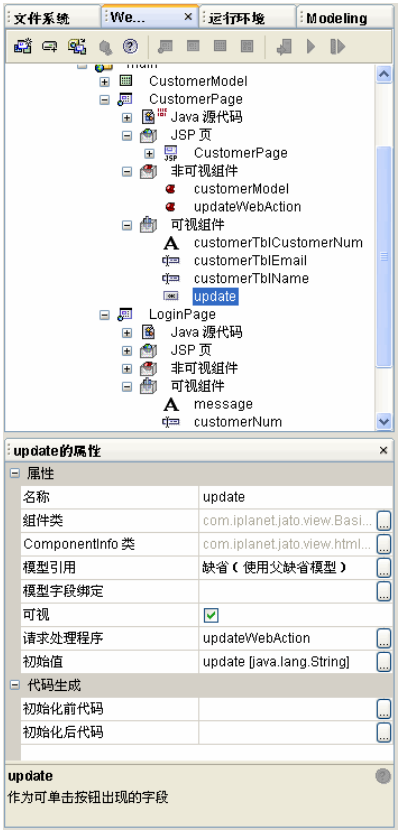
   此属性设置完毕。

---

**注 --** 此时会在 “非可视组件” 节点下方增加一个新条目，同时会设置 “命令描述符” 属性。

---

请参阅以下图像，该图像显示了 “模型”按钮的属性表单。“请求处理程序”属性现在用于指定前一步骤中配置的新命令描述符。




# 自动更新模型

现在需要在 “CustomerPage”上增加作为 “自动更新模型”的 customerModel 引用。

您可以使用适当的模型引用填充页面的自动更新模型属性来完成此操作，此处的引用就是您指定了模型关联或字段绑定后，向导为您创建的 customerModel 引用。

1. **选择 “CustomerPage”节点。**
2. **单击 “自动更新模型”属性的值区域。**

   此时会显示省略号按钮 ("...")。

3. **单击省略号按钮。**

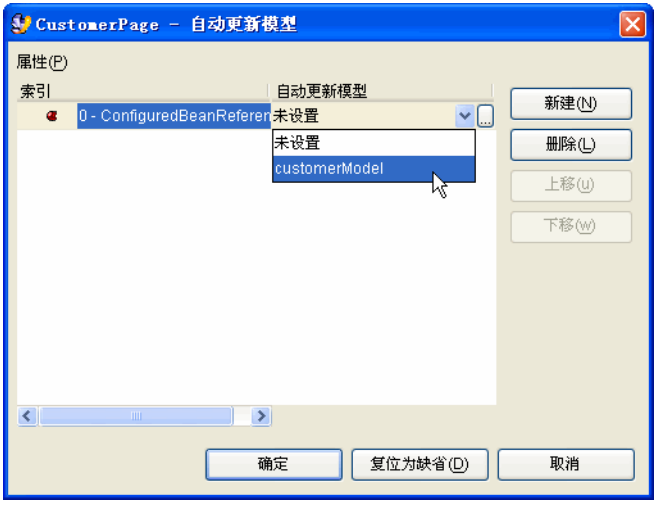
   会启动 “自动更新模型”定制编辑器。

   请注意第一次显示此编辑器时 “属性”区域为空。

4. **单击 “新建”。**

   这将添加一个条目。

5. **在 “自动更新模型”组合框中选择 “customerModel”。**




6. **单击 “确定”。**

   现在属性应具有 *[customerModel]* 条目了。

---

**注 --** 该按钮的更新 Web 操作命令描述符和自动检索或更新模型配置的组合使系统在显示 “客户”页或单击 “CustomerPage”的 “Update”按钮时执行 “CustomerModel”。

除了以声明的方式自动执行此模型外，还可以通过编写某种代码来实现这一相同目的。通常，此代码将在 “Update”按钮的 handle*Update*Request 事件中实现 （与实现 “登录”页上的 “Login”按钮的代码相似）。

---

# 向客户页增加隐藏字段

1. **展开 “CustomerPage”节点。**
2. **展开 “可视组件”节点。**

3. **选择 “可视组件”节点。**

4. **选择使用组件选择面板来增加 “隐藏字段”组件。**

   此时一个隐藏字段便被增加到 “CustomerPage”的 “可视组件”节点中。

5. **将隐藏字段重命名为 “*hiddenKey*”。**

6. **将 “hiddenKey”字段绑定到 “customerTblCustomerNum”静态文本字段所绑定的同一模型字段中。**

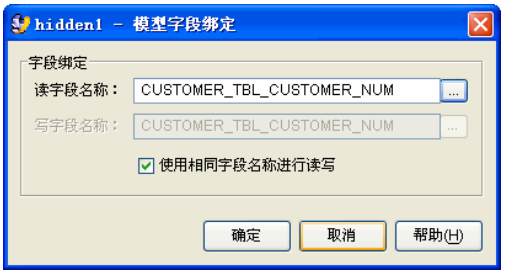
7. **选择 “hiddenKey”字段。**

8. **在属性表单中，通过从下拉框中进行选择，可以将 “模型引用”属性设置为 “customerModel”。**

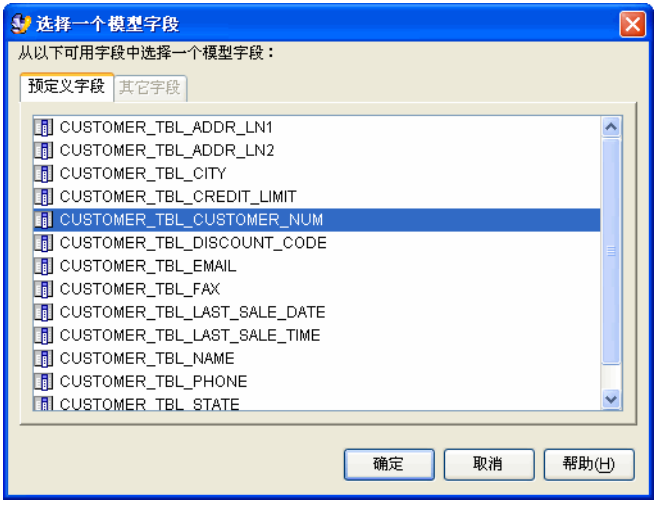

9. **设置 “模型字段绑定”属性。**

   a. **单击 “模型字段绑定”属性的值区域。**

   b. **单击省略号按钮来启动 “模型字段绑定”属性编辑器。**



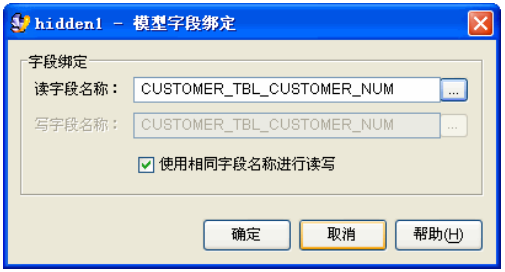

10. **单击 “*读字段名称*” 属性的省略号按钮，启动 “选择一个模型字段” 编辑器。**




11. **选择 “*CUSTOMER_TBL_CUSTOMER_NUM*”。**

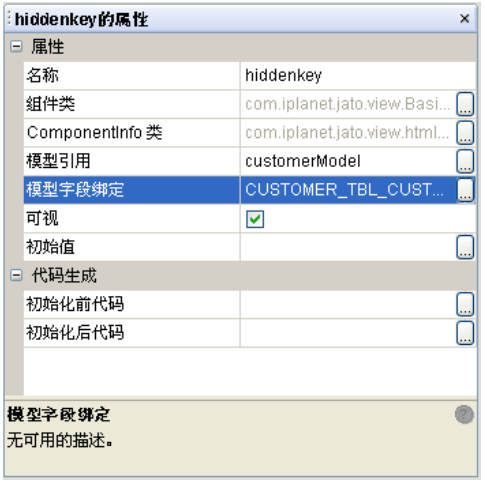
12. **单击 “确定”。**

    此时读取和写入字段都使用 “*CUSTOMER_TBL_CUSTOMER_NUM*” 模型字段填充。




13. **单击 “确定”。**

    这样便完成了对 “*hiddenKey*” 显示字段的 “模型字段绑定” 属性设置。




由于静态文本字段不是 HTML 输入字段，因此当单击更新按钮时，系统不会将其值提交回服务器。同时由于客户号码是数据库表中的关键字字段，更新逻辑需要此关键字值来限制对单个数据库行的更新。因此必须将此值连同其他输入字段值一起传回，这样便可对适当的客户记录进行更新，而不必更新表中的每条记录。为此，需在隐藏字段中保留客户号码字段值，该值将在表单提交时被传回，并映射回“*CUSTOMER_TBL_CUSTOMER_NUM*”模型字段。

**警告 --** 如果忽略此步骤，则关键字字段值不会随窗体一起提交。所产生的 JDBC 更新语句将缺少 WHERE 子句，因此会导致无意中对整个表进行修改。

这与实际实现方式并不完全相同。出于安全考虑，您可能不希望将关键字字段作为输入字段显示在 HTML 中，因为黑客可以修改该字段。

**注 --** 这并不是 Web 应用程序框架所独有的问题，它是任何 Web 应用程序（无论使用何种框架实现应用程序，或即便实现时未使用框架）都必须解决的问题。

Web 应用程序框架提供一个名为“页会话”的增值功能，该功能使用了一种以更安全的方式实现此解决方案的技术，但是这种技术不在本教程的学习范围内。有关详细信息，请参阅 JatoSample 应用程序和 *《Web 应用程序框架开发人员指南》*。

# 设置 JSP 格式

1. **展开“CustomerPage”节点下的“JSP”节点。**
2. **双击“CustomerPage JSP”在编辑器窗口中打开它。**
3. **为各个字段分别提供正确大小写的标签。**

   以下示例设置了最少的 JSP 格式（此处只显示相关代码）。某些 HTML 源代码以**粗体**显示，以示清晰。

```
<jato:form name="CustomerPage" method="post">

<table border=0 cellspacing=2 cellpadding=2 width="100%">
<tr>
  <td align=right valign=middle width="20%"><b>Customer #:</b></td>
  <td align=left valign=middle><jato:text name="customerTblCustomerNum"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"><b>Email:</b></td>
  <td align=left valign=middle><jato:textField name="customerTblEmail"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"><b>Name:</b></td>
  <td align=left valign=middle><jato:textField name="customerTblName"/></td>
</tr>
</table>

<jato:button name="update"/>
<jato:hidden name="hiddenKey"/>
</jato:form>
```

第 10 章

# 教程 2.4 节
# 测试运行客户页

本章说明如何运行 Web 应用程序框架应用程序。

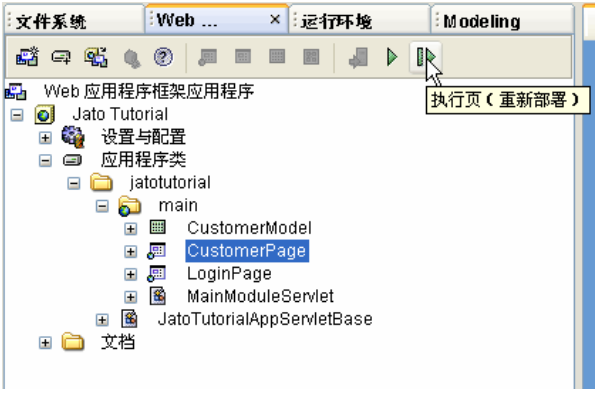
## 任务 4：测试运行客户页

**要点：**确保 PointBase 网络服务器在运行。如果没有运行，可以通过执行以下操作来启动它：

1. **选择菜单选项 “工具” -> “PointBase 网络服务器” -> “启动服务器”。**
2. **右键单击 “应用程序类”节点。**
3. **选择 “全部编译”操作。**

   如果在 Sun Java System Application Server 上运行应用程序，必须在对其进行更改后部署它。
4. **选择 “Web 应用程序框架应用程序”节点 (JatoTutorial)，然后单击 Web 应用程序框架工具栏上的 “部署”按钮。**
5. **选择 “CustomerPage”节点，然后单击 “执行页 （重新部署）”按钮。**

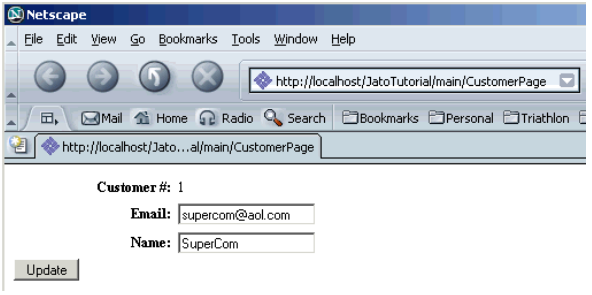
   使用此执行和重新部署选项会重新启动服务器，以确保服务器应用全部更改且不使用任何缓存资源。



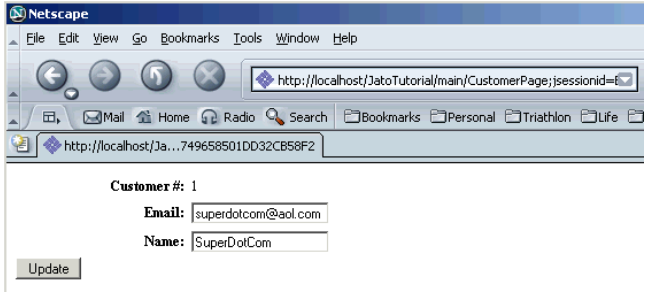

缺省浏览器将启动该应用程序。

# 测试客户更新

1. **更改其中一个或两个字段。**




2. **单击“Update”。**




在此图中，电子邮件名称和客户姓名已被更改。

第 11 章

# 教程 2.5 节
# 将登录页链接到客户页

本章说明如何在 Web 应用程序框架应用程序中将 LoginPage 链接到 CustomerPage 中，并根据客户的登录来过滤 “客户” 页显示的数据。

# 任务 5：将登录页链接到客户页

## 编辑 LoginPage 中的 handleLoginRequest 方法

编辑 `LoginPage.java` 文件。

按以下显示的代码示例来修改 `handleLoginRequest()` 方法中的逻辑，以便在成功登录后显示 “客户” 页，并在该页面上显示与 “用户名” 字段中输入的值对应的客户数据。

---

**注 --** 在以下代码示例中，“用户名” 字段的唯一合法值也是客户表中的 CustomerID 值。

因此，可以使用 “登录 ID” 值并将其应用于 CustomerModel 使用的 WHERE 子句。

这样可确保 CustomerModel 检索的数据与相应的 CustomerID 对应。

请慎重更改代码。

下方所示代码几乎替换了先前在此事件中出现的所有代码。

如果只在旧代码上进行一些修改，可能会导致错误。最好是删除当前代码，然后用以下代码替换。

---

以下是修改 handleLoginRequest() 方法中的逻辑而需输入的代码。

```
public void handleLoginRequest(RequestInvocationEvent event) {
    // Retrieve the customer number
    String custNum = getDisplayFieldStringValue(CHILD_CUSTOMER_NUM);

    String theMessage = "";

    // Check the customer number
    // Note, we don't check the password in this example
    if (custNum.equalsIgnoreCase("1") || custNum.equals("777") ||
            custNum.equals("410")) {
        // Instead of returning the login page, display the Customer
        // page for the customer that matches the customer number
        // Get a reference to the CustomerModel
        CustomerModel model = (CustomerModel)getModel(CustomerModel.class);

        // Modify the where criteria to reflect
        // the customer number used to login
        model.clearUserWhereCriteria();
        model.addUserWhereCriterion(
                "CUSTOMER_TBL_CUSTOMER_NUM", new Integer(custNum));

        // Display the Customer page
        getViewBean(CustomerPage.class).forwardTo(event.getRequestContext());
    } else {
        theMessage = "Sorry, "+ custNum +
                ",your customer number was incorrect!";

        // Set the output status message
        getDisplayField(CHILD_MESSAGE).setValue(theMessage);

        forwardTo();
    }
}
```

第 12 章

# 教程 2.6 节
# 运行应用程序

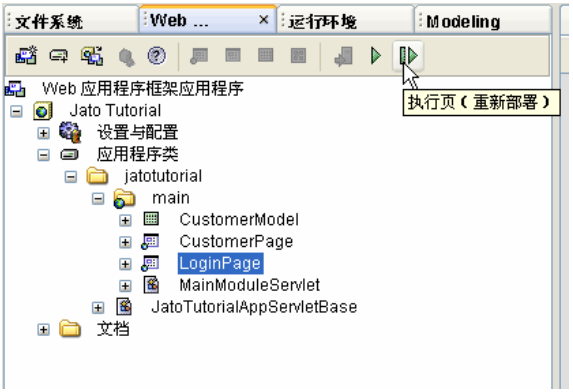
您已经学习了如何为应用程序增加一个附加页并将其链接到第一页。接下来，本章将介绍如何运行 Web 应用程序框架应用程序。

## 任务 6：运行应用程序

**要点：**确保 PointBase 网络服务器在运行。

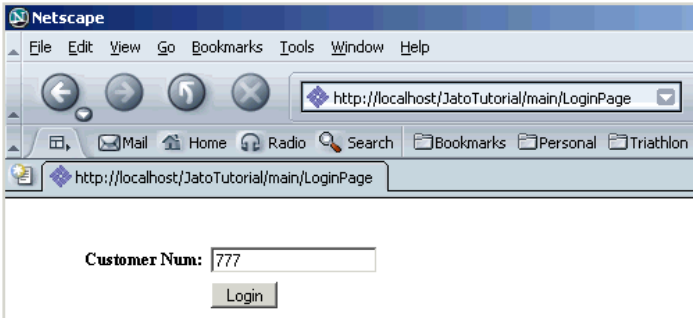
如果没有运行，可以在 IDE 中通过以下操作来启动：

1. **选择菜单选项 “工具” -> “PointBase 网络服务器” -> “启动服务器”。**
   由于已对若干个类进行了修改，所以请务必编译该应用程序。
2. **右键单击 “应用程序类”节点，然后选择 “全部编译”操作。**
   如果在 Sun Java System Application Server 上运行应用程序，必须在对其进行更改后部署它。
3. **选择 “Web 应用程序框架应用程序”节点 (JatoTutorial)，然后单击 Web 应用程序框架工具栏上的 “部署”按钮。**
4. **选择 “LoginPage”节点，然后单击 “执行页 （重新部署）”按钮**
   使用此执行和重新部署选项会重新启动服务器，以确保服务器应用全部更改且不使用任何缓存的资源。



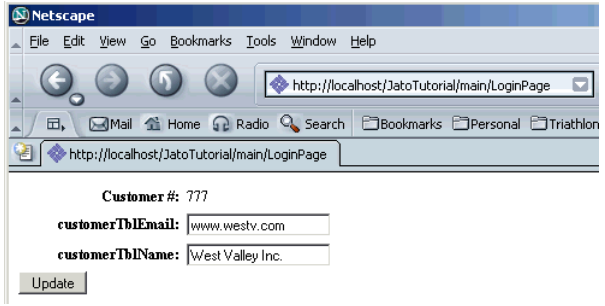

缺省浏览器将启动该应用程序。

5. **输入有效客户号码（1、777 或 410）。**




6. **单击“Login”。**

会看到具有客户记录的客户页，该记录与您进行登录的客户号码对应。



第 13 章

# 教程 3.1 节
# 创建命令组件

本章说明如何创建可以由同一应用程序内部的许多命令字段 （按钮和 HREF 组件）重用的 “命令”组件。这是在命令字段的处理请求事件 （在其父页 / 页面配件类中，例如， LoginPage 中的 handleLoginRequest）中实现请求处理代码的另一种可行方法。

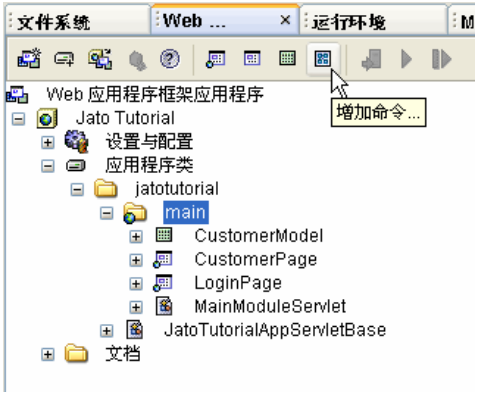
当实现代码重用时，命令可提供强大的功能和灵活性。任何 Java 类都可以简单地通过实现 com.iplanet.jato.command.Command 接口而成为 “命令”组件。本教程中，您将使用 Web 应用程序框架命令向导创建一个新的 “命令”类，用来创建登录 / 注销命令，以替换 “Login”按钮的请求处理程序事件。如果需要，此 “命令”组件随后可以被其他页面和页面配件上的命令字段重用。

## 任务 1：创建命令组件

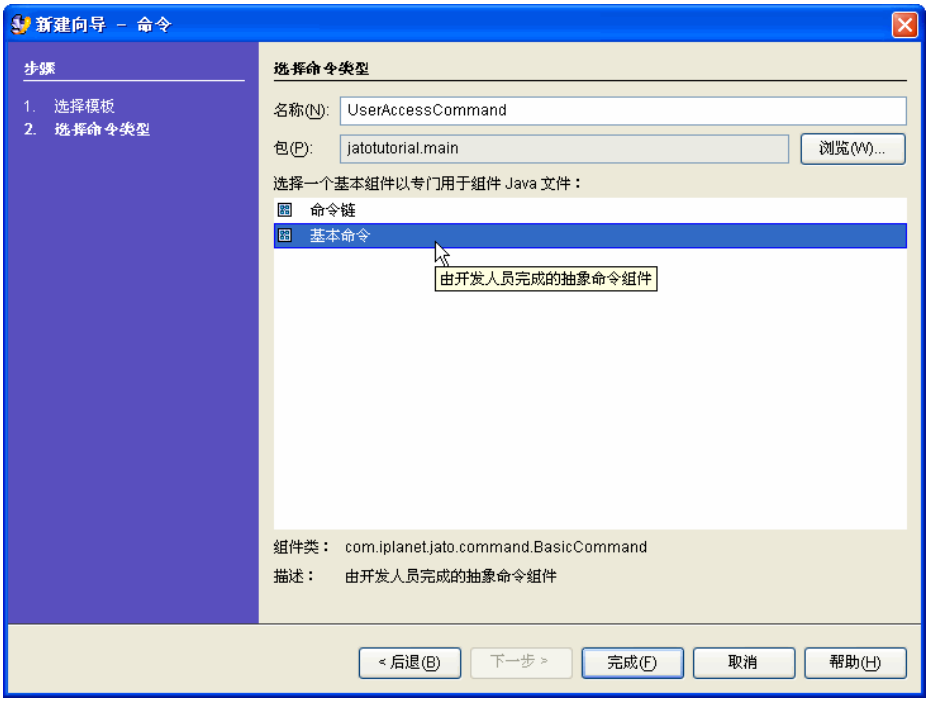
### 创建 UserAccessCommand 组件

使用 Web 应用程序框架命令向导创建一个 “命令”组件。

1. **选择 “main”模块文件夹，然后在 Web 应用程序框架工具栏中单击 “增加命令”按钮。**

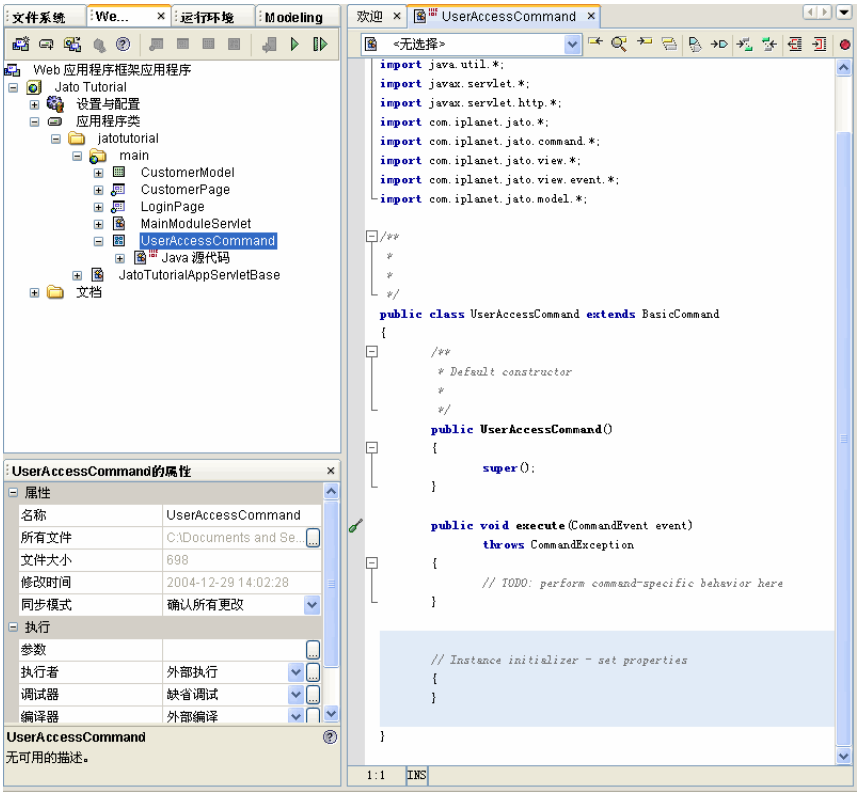



将会显示“选择命令类型”面板。




2. **在“名称”文本框中输入 *UserAccessCommand*。**
3. **选择“基本命令”。**
4. **单击“完成”。**

“UserAccessCommand”组件即被增加到应用程序中。




双击 “UserAccessCommand”，在编辑器窗口中打开此组件的 Java 源代码。

它是一个扩展 BasicCommand 的非常简单的类。 BasicCommand 实现仅声明一个方法的 “命令”接口：

```
public void execute(CommandEvent) throws CommmandException
```

此命令当前无法执行任何操作。需要向执行方法增加一些代码，以便执行您需要其执行的操作，也就是，基于通过 CommandEvent 参数传递进来的*操作名称*执行用户登录或注销。操作名称完全取决于开发人员进行命名。以下步骤和任务向您说明了如何传递和计算定制的操作名称。

# 向执行方法中增加代码

此步骤只需您编写一个小代码。看起来似乎有很多代码，但大部分代码都是 LoginPage 中 handleLoginRequest 事件的重新实现。因此无需您使用按钮事件。

将下列代码增加到 UserAccessCommand 类的执行方法中。

```
public void execute(CommandEvent event) throws CommandException {
 // get the RequestContext
 RequestContext requestContext = event.getRequestContext();

 // get the J2EE HttpSession
 javax.servlet.http.HttpSession session =
 requestContext.getRequest().getSession();

 // get the operation name that was passed in
 // by the commandfield object (button/href)
 String opName = event.getOperationName();

 // get the LoginPage
 LoginPage loginVB = (LoginPage)requestContext
 .getViewBeanManager().getViewBean(LoginPage.class);

 // perform user login
 if (opName.equals("login")) {
 // get the customer number that was entered
 int custNum = loginVB.getDisplayFieldIntValue(
 LoginPage.CHILD_CUSTOMER_NUM);

 // get the Customer model
 CustomerModel customerModel = (CustomerModel)requestContext
 .getModelManager().getModel(CustomerModel.class);

 // execute the CustomerModel with the customer number as criteria
 // to see if the user exists in the database
 customerModel.clearUserWhereCriteria();
 customerModel.addUserWhereCriterion(
 "CUSTOMER_TBL_CUSTOMER_NUM", new Integer(custNum));
 try {
 customerModel.executeSelect(null);
 } catch (ModelControlException e) {
 Log.log("Exception caught in UserAccessCommand.execute(): "
 + e.toString());
 } catch (java.sql.SQLException e) {
 Log.log("Exception caught in UserAccessCommand.execute(): "
 + e.toString());
 }

 // valid customer number entered
 if (customerModel.getNumRows() == 1) {
 // Display the Customer page
 requestContext.getViewBeanManager().getViewBean(
 CustomerPage.class).forwardTo(requestContext);
```

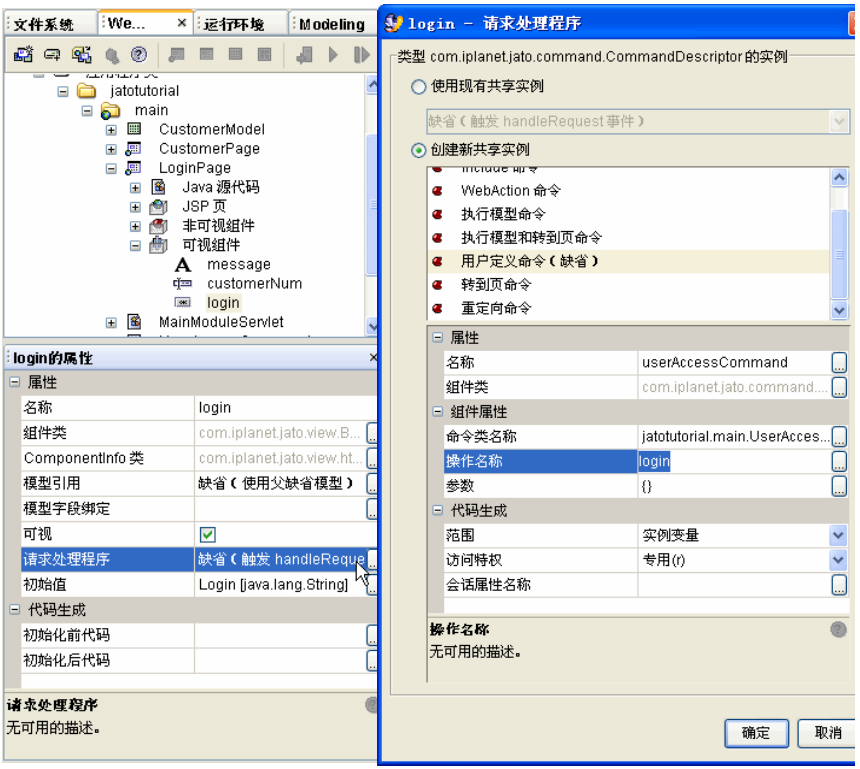
```
  // put the customer number into an HttpSession attribute
  // for potential use in a later request
  session.setAttribute("hsaCustNum", new Integer(custNum));
  }
  // invalid customer number entered
  else {
  String msg = "Sorry, " + custNum +
  " is not a valid customer number.";

  // Set the output status message
  loginVB.getDisplayField(
  LoginPage.CHILD_MESSAGE).setValue(msg);

  // Display the Login page
  loginVB.forwardTo(requestContext);
  }
  } // if opName = login

  // perform user logout
  else if (opName.equals("logout")) {
  // get the customer number from session to use in the logout message
  String hsaCustNum = session.getAttribute("hsaCustNum").toString();

  // Set the status message value
  String msg = "Customer " + hsaCustNum +
  ", you have logged out successfully.";

  // invalidate the user's HttpSession
  session.invalidate();

  // Set the logout message and display the Login page
  loginVB.getDisplayField(LoginPage.CHILD_MESSAGE).setValue(msg);

  // Display the Login page
  loginVB.forwardTo(requestContext);
  } // else if opName = logout

  // unknown operation name
  else {
  throw new CommandException(
  "Unknown UserAccessCommand operation name: " + opName);
  }
 }
```

在测试运行此代码之前，需要配置一个命令字段（按钮或 HREF）来使用它。

# 配置按钮的命令描述符

通过按钮的 “命令描述符”属性，配置登录按钮以使用 “UserAccessCommand”组件。此方法同样也适用于配置 HREF。

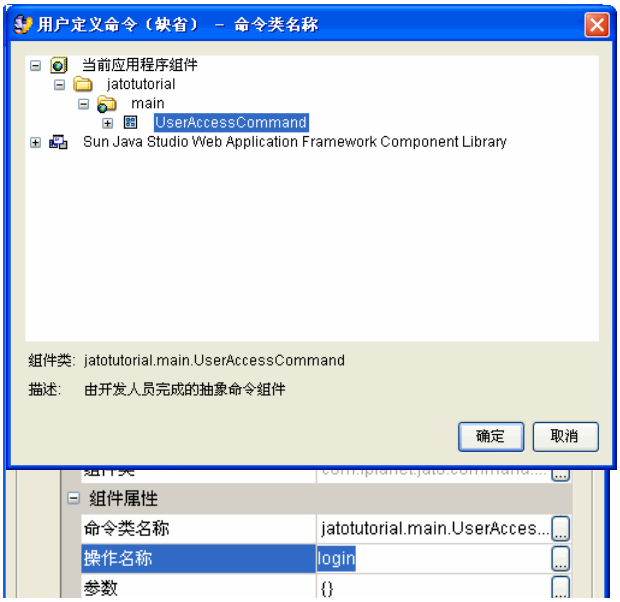
1. **展开 “LoginPage”节点，并展开 “可视组件”节点。**
2. **选择 “可视组件”下的登录按钮。**




3. **单击 “*请求处理程序*”属性的省略号按钮。**
   此时会显示 “命令描述符”编辑器。
4. **从 “*创建新共享实例*”单选按钮选项下的列表中选择 “*用户定义命令（缺省）*”。**
5. **将 “名称”属性更改为 “userAccessCommand”。**
6. **选择编辑器底部的 “组件属性”标签。**
7. **单击 “*命令类名称*”属性的省略号按钮。**
   此时会显示 “命令类名称”选择器对话框。
8. **依次展开 “当前应用程序组件”节点、“jatotutorial”节点和 “main”节点。**

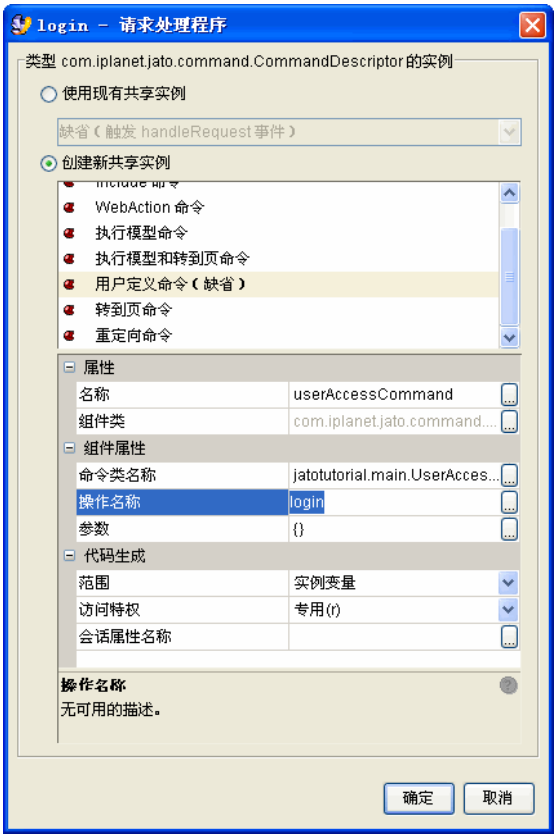
9. **选择“UserAccessCommand”命令组件。**

10. **单击“确定”。**




11. **将“操作名称”从 *DEFAULT* 改为 *login*。**

    重新调入为 UserAccessCommand 类中的执行方法实现的代码。您有一个需要以 *login* 或 *logout* 作为操作名称的 if/else 块。由于操作名称区分大小写，因此需要确保设置正确的名称，否则在测试运行此命令时，会收到 CommandException （未知的操作名称）。




**12. 单击 “确定”完成登录按钮的 “命令描述符”属性设置。**

现在，如果运行 “登录”页并单击 “Login”按钮，“UserAccessCommand”组件将处理请求，而不是 LoginPage 中的 handleLoginRequest 事件代码。

可以保留 handleLoginRequest 事件中的代码不变，因为永远都不会调用它，除非您重新配置登录按钮以使用请求处理程序事件而不是命令组件。

这是因为 Web 应用程序框架首先为命令字段查找命令描述符。如果未实现命令描述符，那么会尝试调用 handle<CommandField>Request 方法。如果未实现该方法，会收到*找不到请求处理程序*异常消息。

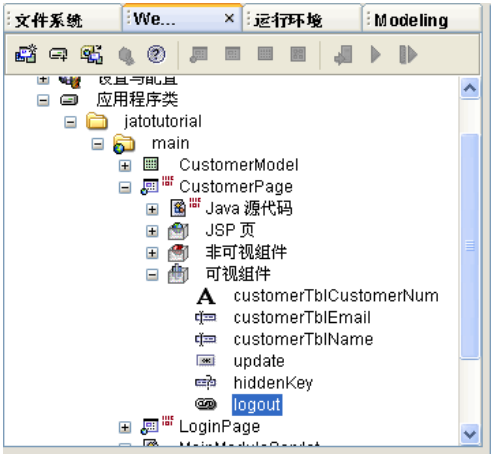
第 14 章

# 教程 3.2 节
# 向客户页增加注销链接

本章说明如何向使用 “命令”组件的页增加 HREF。

## 任务 2：向客户页增加 HREF

1. **选择 “CustomerPage”节点。**
2. **使用 “组件选择面板”增加一个 “超链接 (HREF)”组件。**
   一个 HREF 命令字段随即被增加到 “CustomerPage”的 “可视组件”节点中。
3. **将 HREF 重命名为 *logout*。**

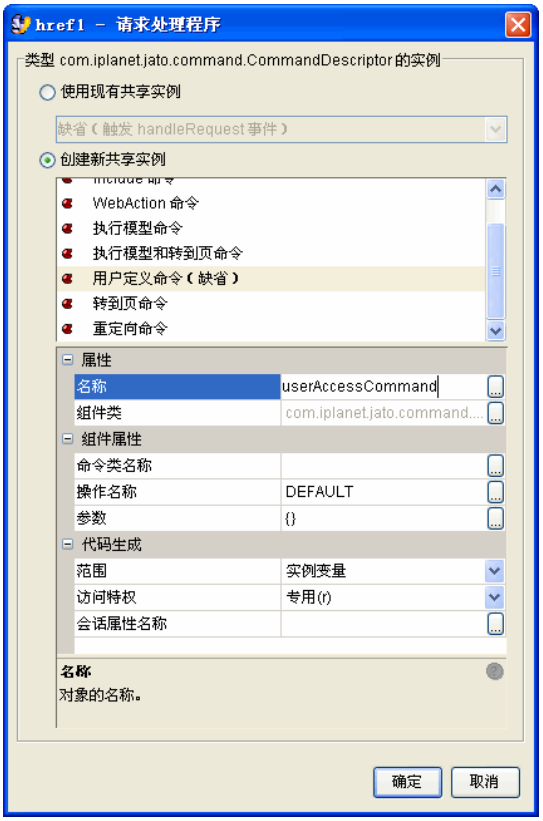


# 配置 HREF 的命令描述符

通过按钮的 “命令描述符”属性，配置注销 HREF 以使用 “UserAccessCommand” 组件。本操作与前一任务的按钮 “命令描述符”配置相同，只不过*操作名称*为 *logout* 而不是 *login*。

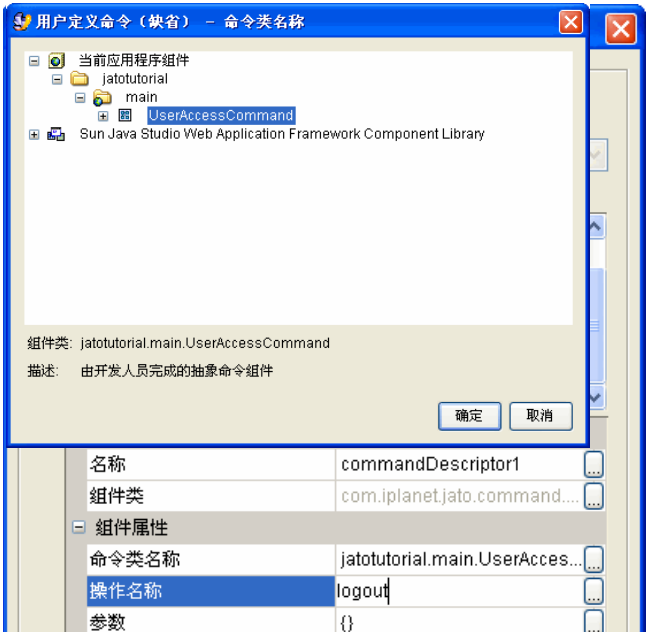
1. **选择 “CustomerPage”的 “可视组件”节点下的注销 HREF。**
2. **单击 “请求处理程序”属性的省略号按钮。**

   此时会显示 “请求处理程序”编辑器。




3. **从 “*创建新共享实例*”单选按钮选项下的列表中选择 “*用户定义命令（缺省）*”。**
4. **将名称改为 userAccessCommand。**
5. **选择编辑器底部的 “组件属性”标签。**
6. **单击 “*命令类名称*”属性的省略号按钮。**

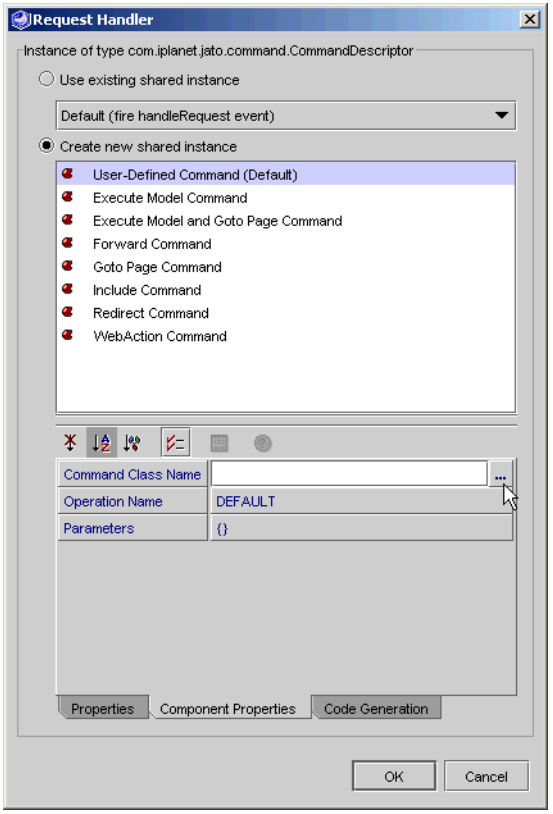
此时会显示“命令类名称”选择器对话框。




7. **会依次展开“当前应用程序组件”节点、“jatotutorial”节点和“main”节点。**
8. **选择“UserAccessCommand”命令组件。**
9. **单击“确定”。**
10. **将“操作名称”从 *DEFAULT* 改为 *logout*。**

    重新调入为 UserAccessCommand 类中的执行方法实现的代码。您有一个需要以 *login* 或 *logout* 作为操作名称的 if/else 块。

    由于操作名称区分大小写，因此请确保设置正确的名称，否则在测试运行此命令时，会收到 CommandException（未知的操作名称）。




11. **单击 “确定”以完成注销 HREF 的 “请求处理程序”属性设置。**

    当使用有效客户号码登录时，会进入 “客户”页。显示有注销链接。单击链接时，注销操作名称被传送至 UserAccessCommand，从而使用户会话无效，然后在 “登录”页上显示注销消息。

## 设置客户 JSP 中的 HREF 标记格式

向 CustomerPage 增加*注销* HREF 字段后，HREF 标记被增加至 `CustomerPage.jsp` 文件中。但是，链接以缺省 HREF 名称 *href1* 显示，而不是所需的文本。

1. **展开 “CustomerPage”下的 JSP 节点。**
2. **双击 “CustomerPage JSP”节点，在编辑器窗口中打开 JSP。**

3. **查找注销 HREF 标记并修改主体内容部分，以显示 *Logout* 而不是 *href1*。**

```
<jato:href  name="logout">Logout</jato:href>
```

可以将 HREF 标记放在您喜欢的任何位置，只要它嵌套在 *useViewBean* 标记之间，并且是 HTML 主体区段的一部分 （位于*主体*标记之间）。

与按钮不同， HREF 并不是表单所需的部分，因此它可以放在*表单*标记 (<*jato:form*>) 外。

第 15 章

# 教程 3.3 节
# 测试运行登录 / 注销命令组件

本章说明如何运行 Web 应用程序框架应用程序。

## 任务 3：测试运行登录 / 注销命令

**要点：**确保 PointBase 网络服务器正在运行。如果没有运行，可以在 IDE 中通过以下操作来启动：

1. **选择菜单选项 “工具” -> “PointBase 网络服务器” -> “启动服务器”。**

   由于已经创建了一个新类并对其他两个类进行了修改，所以请务必编译 / 部署应用程序。

2. **右键单击 “应用程序类”节点，然后选择 “全部编译”操作。**

3. **如果在 Sun Java System Application Server 上运行应用程序，则必须先对其进行更改，然后部署它。**

   选择 “Web 应用程序框架应用程序”节点 (JatoTutorial)，然后单击 Web 应用程序框架工具栏上的 “部署”按钮。

4. **选择 “LoginPage”节点，然后单击 “执行页 （重新部署）”按钮**

   使用此执行和重新部署选项会重新启动服务器，以确保服务器应用全部更改且不使用任何缓存的资源。

   此时缺省浏览器会启动应用程序。

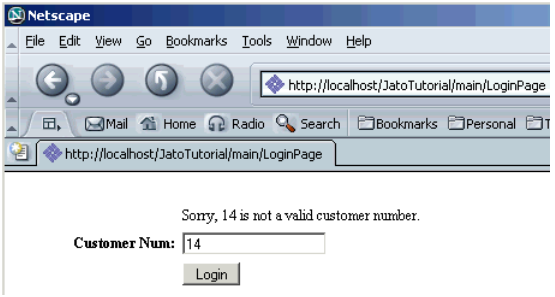
---

**注 --** 在 3.1 和 3.2 节中，已将三个客户号码硬编码到登录验证中。新的 UserAccessCommand 将根据数据库来验证输入的客户号码。

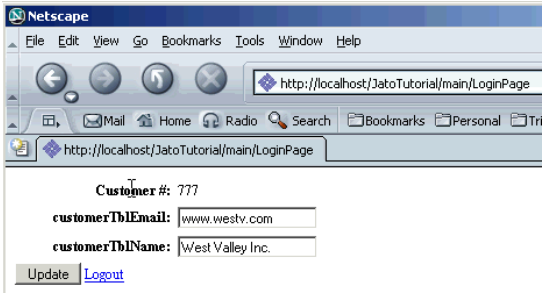
为了方便起见，样例 PointBase 数据库中的有效客户号码列表为：1、 2、 3、 25、 36、 106、 149、 409、 410、 722、 753、 777、 863。

---

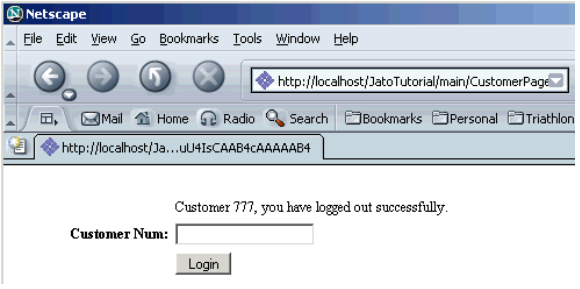
5. **首先输入无效的客户号码。**




6. 输入一个有效的客户号码。




7. 尝试注销链接。



第 16 章

# 教程 4.1 节<br>准备创建 Web 服务模型

本章说明如何通过 Web 服务扩展应用程序来访问数据。必须运行一个包含 Web 服务模型向导的 IDE 版本。同时必须连接到 Internet 上，并且不使用代理 / 防火墙以防干扰与 Web 服务的通信。

可通过增加基于 Web 服务的模型和显示该模型数据的页来扩展现有应用程序。首先，需要下载一些资源以构建 Web 服务模型，并且需要注册为此 Web 服务的用户。

## 任务 1：Web 服务用户注册和下载

### 下载 Web 服务 SDK

下载包含 WSDL 文件的 Google Web 服务软件 SDK，该文件是 Web 应用程序框架创建 Web 服务模型所需的文件。

1. **要下载 Google Web 服务 SDK，请转到**
   `http://www.google.com/apis/download.html`
2. **接受协议。**
3. **单击“下载”按钮。**
4. **将文件保存在您的硬盘驱动器上。**

   打开 zip 文件，并将 `googleapi/GoogleSearch.wsdl` 文件解压缩到应用程序的 lib 目录 (`.../JatoTutorial/WEB-INF/lib`) 中。此文件在 zip 文件中有三个版本。请务必找到*不*在 *dotnet* 目录下的那一个版本。这就是您构建 Web 服务模型所需的那个版本。

**注 --** 将新文件复制到应用程序文件结构中时，有时 IDE 会花费一段时间来刷新其状态。

如果发现 IDE 花很长时间识别新文件，则可以转到 “文件系统”或 “项目”标签，右键单击 lib 目录，然后选择 “刷新文件夹”操作。

# 注册使用 Web 服务

要使用 Google Web 服务，必须注册为用户，以接收每个查询传递给 Web 服务的关键字。

1. **要注册 Google 服务，请转到**

   https://www.google.com/accounts/NewAccount?continue=http://api.google.com/createkey&followup=http://api.google.com/createkey

2. **要注册新帐户，请输入电子邮件地址和口令。**

   您将收到一个用来验证帐户的电子邮件。验证帐户后，会收到另外一个带有密钥（一个由字母和数字组成的长字符串）的电子邮件。将此电子邮件保存在方便的位置，因为创建 Web 服务模型时会用到它。

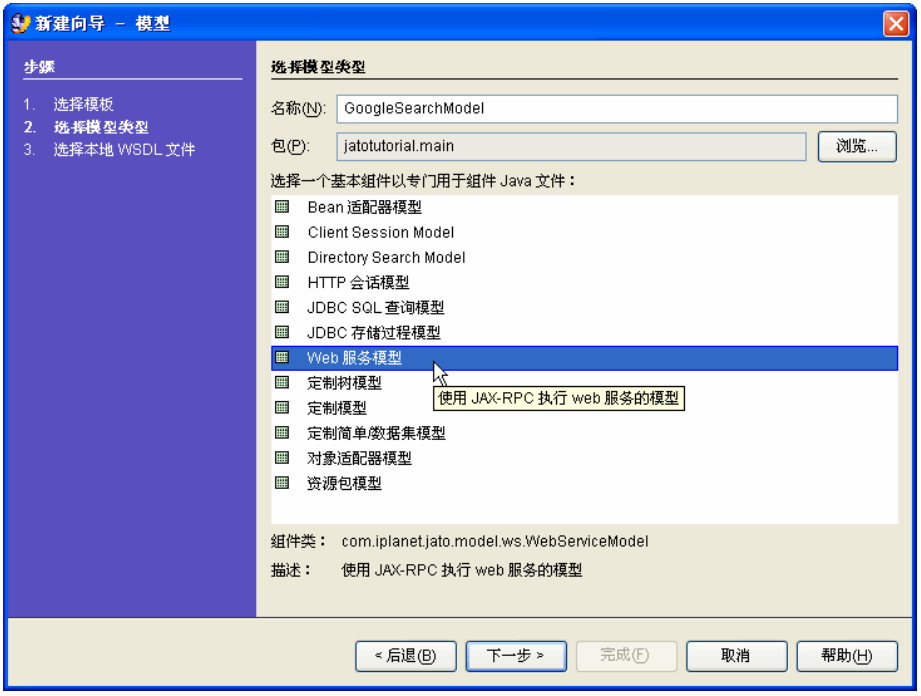
# 创建 Web 服务模型

使用 WSDL 文件可创建 Web 服务模型，该模型使用 Google Internet 搜索引擎，通过 Google Web 服务来执行 Internet 搜索。

1. **选择 “main”模块文件夹。**

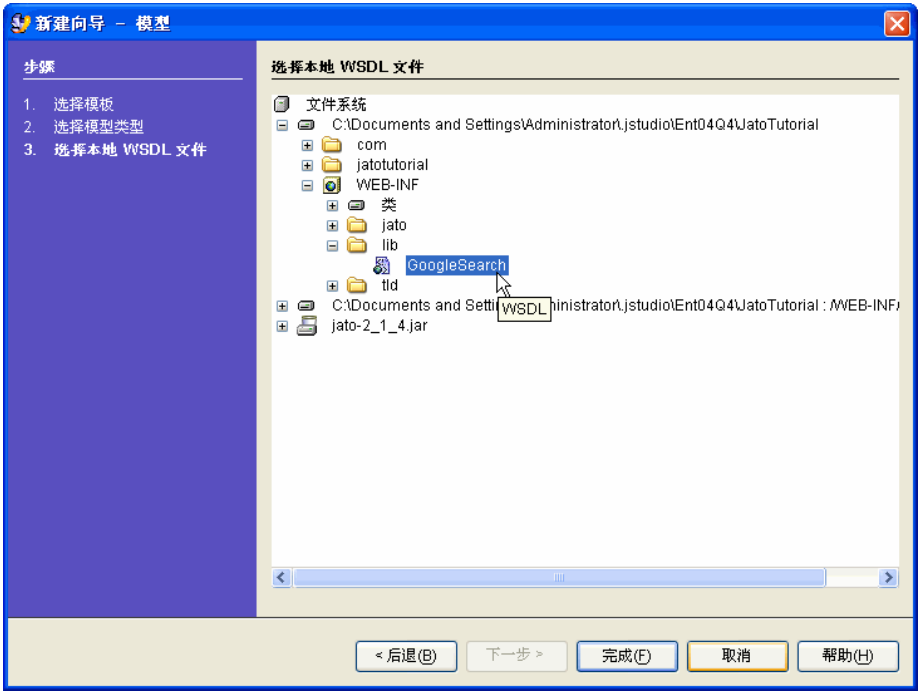
2. **单击 Web 应用程序框架工具栏上的 “增加模型”按钮。**

   此时会显示 “选择模型类型”面板。




3. **在模型名称文本框中输入 *GoogleSearchModel*。**
4. **从模型组件列表中选择 “*Web 服务模型*”。**
5. **单击 “下一步”。**

   此时显示 “选择本地 WSDL 文件” 面板。

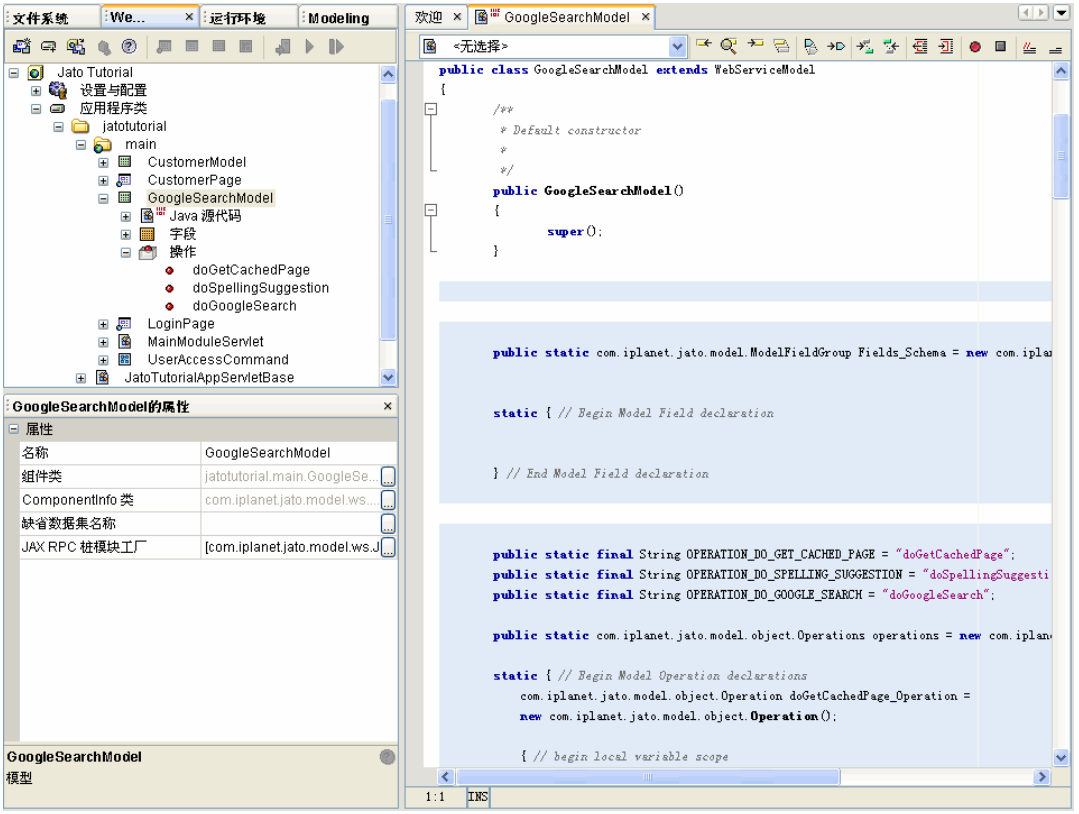



在 JatoTutorial 应用程序目录结构中向下导航至 `lib` 目录 (`JatoTutorial/WEB-INF/lib`)，然后选择 “GoogleSearch”（WSDL 文件）。

文件可能位于 `lib` 目录的子目录中。从 zip 文件中对其解压缩时，它在名为 `googleapi` 的基本目录中。

6. **单击 “完成”来创建 Web 服务模型。**

   即在 main 模块中创建了 GoogleSearchModel 对象。




7. **双击“GoogleSearchModel”节点可查看 GoogleSearchModel 类中的代码。**

   此 Web 服务有几个可以使用的操作。下列任务的重点在 *doGoogleSearch* 操作上。

---

**注 --** 查看应用程序文件结构时，可以看到一个名为 stubs 的新文件夹。此文件夹由 Web 服务模型向导将其创建为包，用以存储支持使用 Web 服务所需的所有桩模块类。

它是 Web 服务模型向导所提供的众多好处之一。要使用 Web 服务，需要创建许多类。浏览此包文件夹可查看实际完成了多少工作。

您大可不必查看这些文件。所有的繁琐工作都已经为您完成。您只需处理 Web 服务模型类，而它只需少量的手动编码，而多数情况下不需要进行任何编码。

---

第 17 章

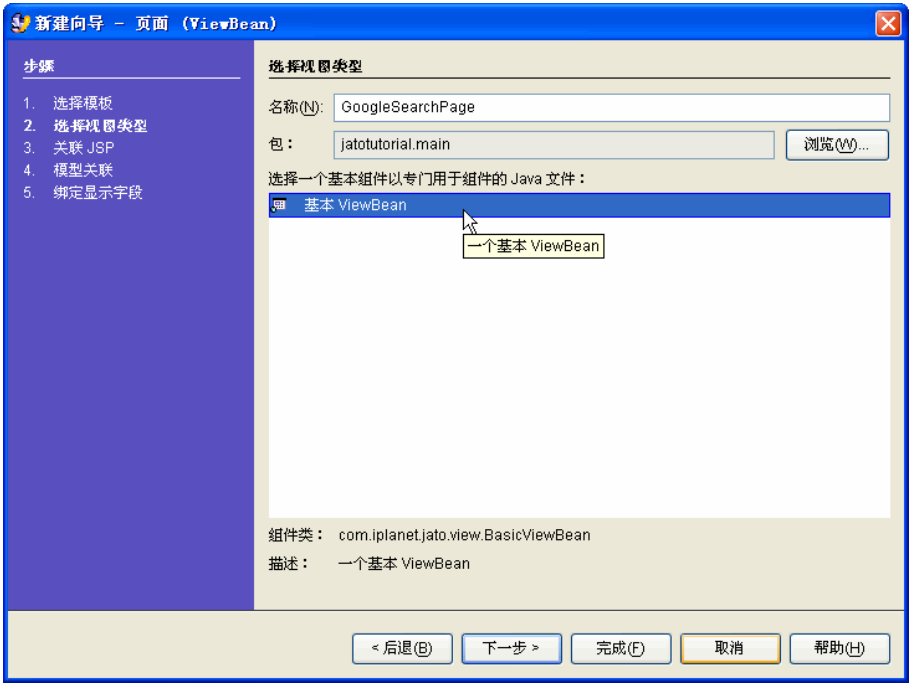
# 教程 4.2 节
# 创建 Google 搜索页

本章说明如何在 Web 应用程序框架中创建一个页面，以显示来自访问 Web 服务数据的模型中的数据。

## 任务 2：创建 Google 搜索页

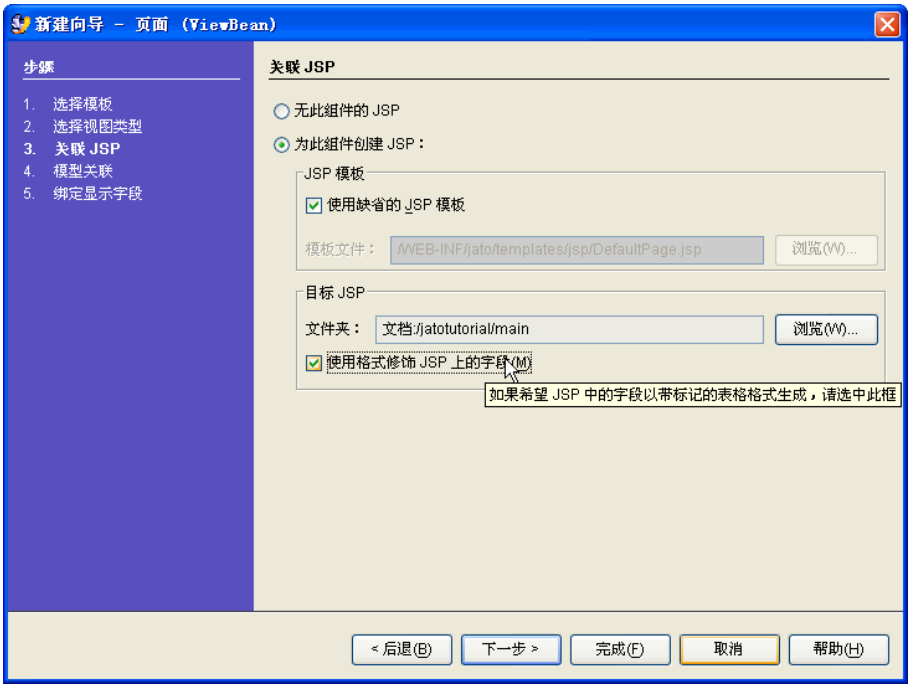
### 增加页组件

1. **选择 “main” 模块文件夹。**
2. **然后单击 Web 应用程序框架工具栏上的 “增加页” 按钮。**
   此时会显示 “选择视图类型” 面板。




3. **在“名称”字段中输入 GoogleSearchPage（替换 < *缺省* >）。**
4. **选择“基本 ViewBean”。**
5. **单击“下一步”。**

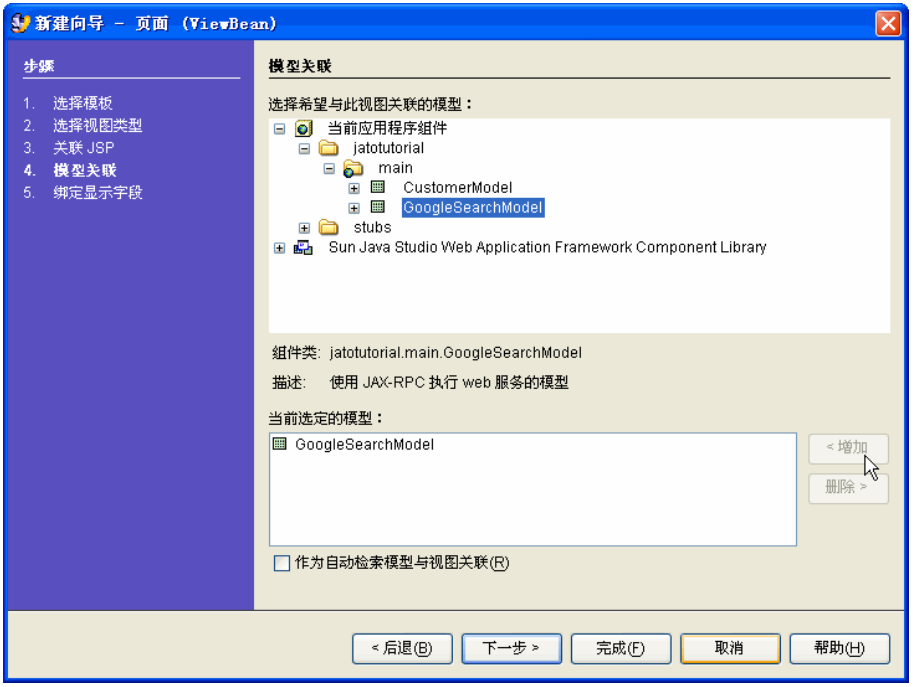
   此时显示“关联 JSP”面板。




6. **选中“*使用格式修饰 JSP 上的字段*”选项。**

7. **单击“下一步”。**

   此时显示“模型关联”面板。




8. **展开“当前应用程序组件”来显示“jatotutorial”->“main”。**
9. **选择“GoogleSearchModel”。**
10. **单击“增加”。**
11. **单击“下一步”。**

    此时显示“绑定显示字段”面板。


12. **增加第一个字段 （如上图所示）：**

    a. **展开 “doGoogleSearch” 操作节点。**

    b. **展开 “参数” 节点。**

    c. **选择 *q* 字段 （q 表示查询字符串：q -> *String*）。**

    d. **选择 “文本字段” 选项。**

    e. **单击 “*增加字段*”。**

    将 q 字段增加到 “*绑定字段*” 列表框中。

---

**注 --** 您尚未完成此向导面板设置。

---

13. **现在还要将下列字段增加为 “文本字段”（与 JDBC 模型字段不同， WebService 模型字段不是多选字段）：**

    a. **start**

    b. **maxResults**

    c. **restrict**

14. **现在将这些字段增加为 “静态文本”字段 （而不是 “文本字段”）：**
（展开 “*返回: ...*”节点 [ 在 “参数”节点上面 ]）

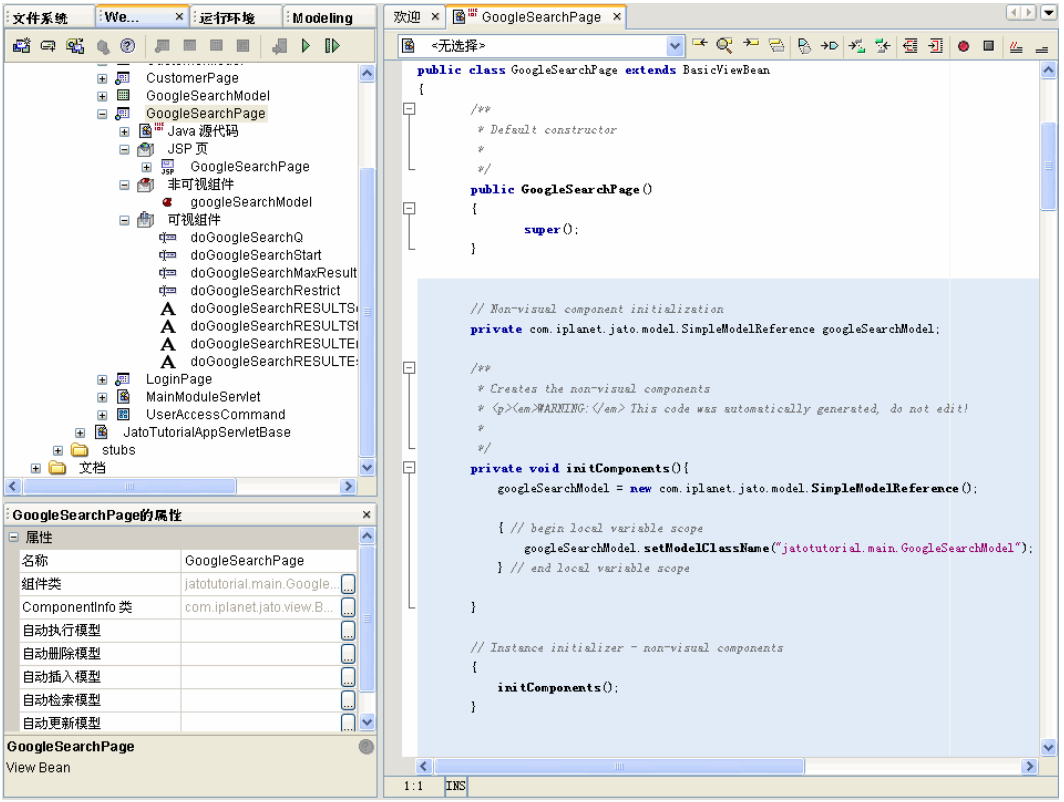
    **a. searchTime**

    **b. startIndex**

    **c. endIndex**

    **d. estimatedTotalResultsCount**

15. **单击 “完成”。**
GoogleSearchViewBean 创建完毕。




16. **重命名字段以使用更简短的名称 （选择字段，单击 F2 进行重命名）。**
下表左列显示较长字段名称，右列显示较短名称。

| | |
|---|---|
| doGoogleSearch**Q** | queryString |
| doGoogleSearch**Start** | start |
| doGoogleSearch**MaxResults** | max |
| doGoogleSearch**Restrict** | restrict |
| doGoogleSearchRESULTS**SearchTime** | searchTime |
| doGoogleSearchRESULTS**StartIndex** | startIndex |
| doGoogleSearchRESULTS**EndIndex** | endIndex |
| doGoogleSearchRESULTS**EstimatedTotalResultsCount** | estTotal |

按照下表为 **start** 和 **max** 文本字段设置属性。

请注意无需设置 **restrict** 字段的属性。

**要点：**记住要通过单击 “...” 按钮来启动 “初始值” 编辑器，以便能选择 “整型” 类型。如果在属性表单中键入值，则会将其视作 “字符串”。

| 名称 | 初始值 |
|---|---|
| start | 类型：整型<br>值：0 |
| maxResults | 类型：整型<br>值：5 |

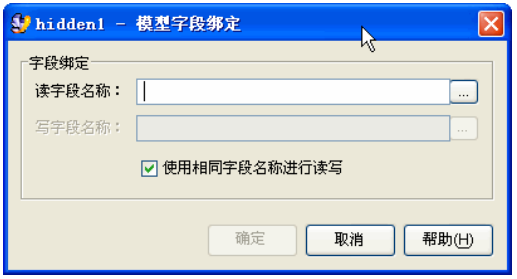
您已经为此页组件创建了四个*搜索*字段和四个*结果*字段，但还需要几个搜索字段（Google Web 服务必需的字段）。这些字段将被逐个增加并绑定到 GoogleSearchModel 中。您需要将这些字段作为文本或静态文本字段以外的内容增加，这就是在页向导之外增加这些字段的原因。

## 将更多可视组件增加到页中

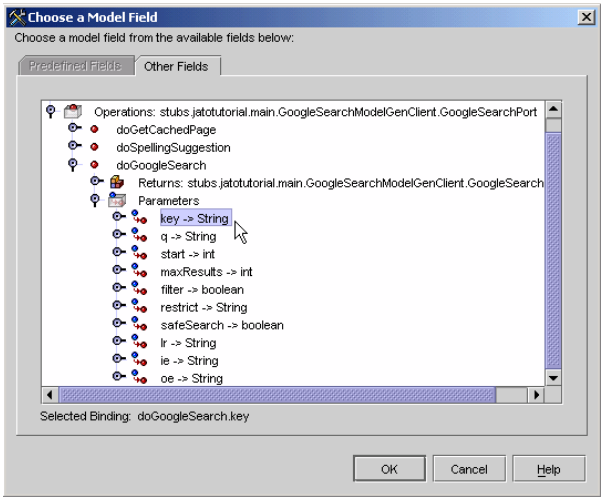
1. **选择 “`GoogleSearchPage`”。**
2. **使用 “组件选择面板” 来增加 “基本隐藏字段”。**
   缺省名称是 *hidden1*。
3. **将缺省名称重命名为 *key*。**
4. **为 *key* 字段设置 “模型引用” 属性。**


5. 从下拉列表中选择 “googleSearchModel”。
6. 设置 “模型字段绑定”属性。


7. 单击省略号按钮来启动 “模型字段绑定”编辑器。




8. 在此编辑器中，单击 “读字段名称”属性的省略号按钮。




9. **展开 “doGoogleSearch” 操作节点，然后展开 “Parameters” 节点。**
10. **选择 “key -> String”。**
11. **单击 “确定”。**

    读取和写入字段都使用 “doGoogleSearch.key” 模型字段填充。


12. **单击 “确定” 以完成关键字隐藏显示字段的 “模型字段绑定” 属性设置。**
13. **使用 Google 通过电子邮件发送给您的关键字，为关键字字段设置初始值属性 （在 “模型字段绑定” 属性上方）。**

    “初始值” 属性的缺省类型是 “字符串”。您无需启动编辑器。只需直接在属性单元格中输入字符串值。

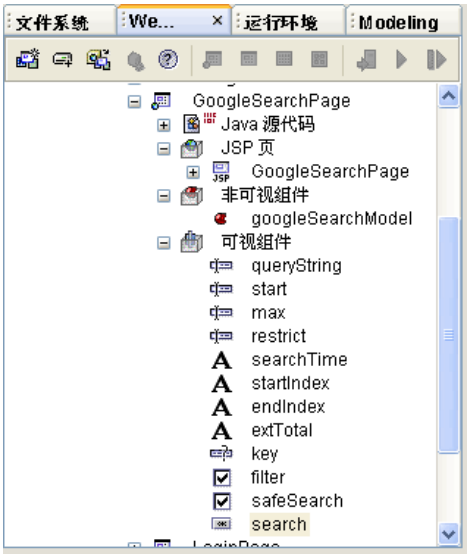

关键字字段的属性内容与上图相似，只是关键字的初始值不同。

14. **使用 “组件选择面板”来增加其他三个显示字段。**

下表包含三个显示字段以及所需属性设置的列表。

| 类型 | 名称 | 初始值 | 模型引用 | 模型字段绑定 |
|---|---|---|---|---|
| 复选框 | filter | | googleSearchModel | doGoogleSearch/Parameters/filter |
| 复选框 | safeSearch | | googleSearchModel | doGoogleSearch/Parameters/safeSearch |
| 按钮 | search | 类型：字符串<br>值：搜索 | | |

“GoogleSearch 页”节点结构应如下图所示：




# 启用搜索按钮

目前搜索按钮在被单击时无法实现任何操作。在单击搜索按钮的同时，还需要执行 Web 服务模型，然后重新装入页来查看结果。现在看到的内容均为统计信息：

- 开始 / 结束索引
- 统计的结果数
- 查询时间

接下来，将增加可视组件以显示实际搜索结果列表。

要启用按钮，可以选择两种方法来执行 Web 服务模型和重新装入页。一种方法是编写几行代码。另一种方法是全部进行点击操作。请选择其中的一种方法来实现。

## 手动编码方法

1. **右键单击搜索按钮。**
2. **选择“事件”。**
3. **选择“handleRequest”。**

   这样会将“handleSearchRequest”事件桩模块插入“GoogleSearchPage”类中。
4. **执行搜索按钮处理请求代码。**

   对于以下缺省代码：

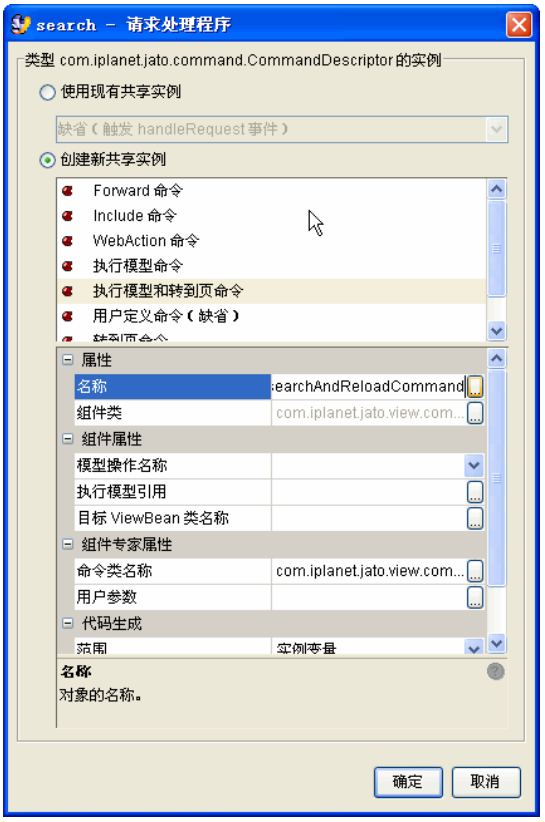
   ```
   getParentViewBean().forwardTo(getRequestContext());
   ```

   请使用以**粗体**显示的代码替换：

```
public void handleSearchRequest(RequestInvocationEvent event)
        throws Exception {
    // get a reference to the Google web service model
    GoogleSearchModel model = (GoogleSearchModel)getModel(
            GoogleSearchModel.class);

    // execute the model using the doGoogleSearch operation
    // (the model execution context)
    model.execute(new ModelExecutionContextBase("doGoogleSearch"));

    // redisplay the page which will now show the query statistical results
    forwardTo();
}
```

## 点击方法（非代码方法）

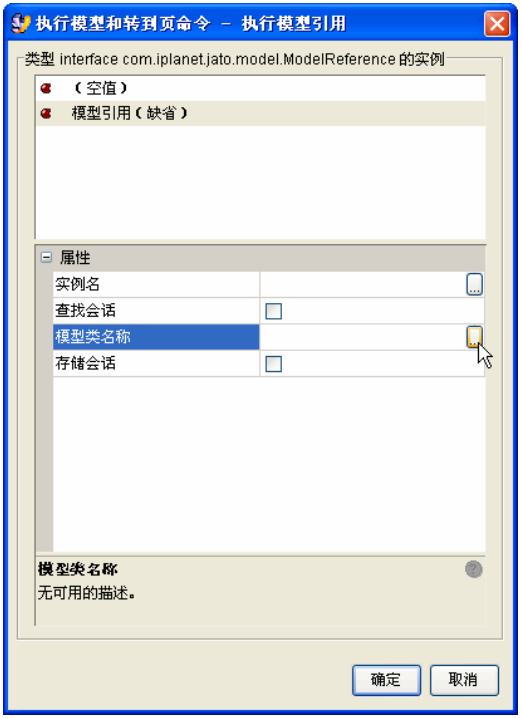
1. **选择搜索按钮。**
2. **通过单击省略号按钮来启动搜索按钮“请求处理程序”属性的编辑器。**




3. **选择 “*执行模型和转到页命令*” 选项。**

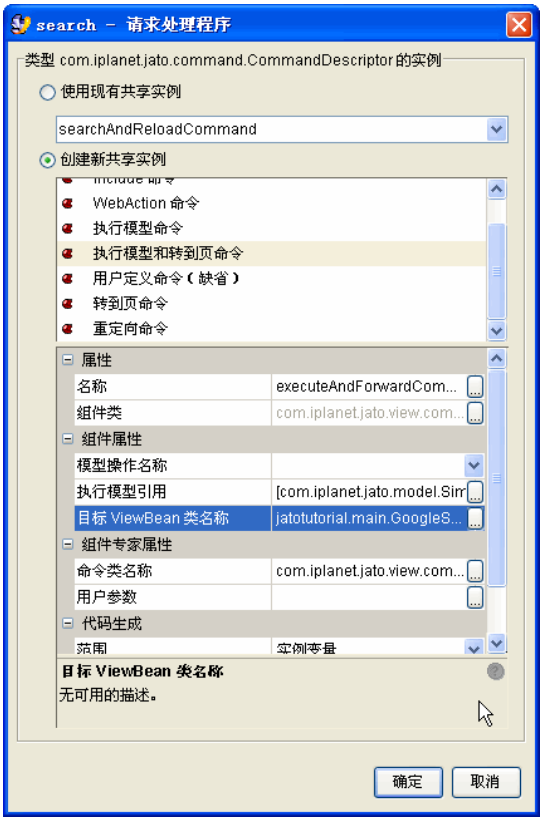
4. **将 “名称” 设置为 “*searchAndReloadCommand*”。**

   在 “组件属性” 标签中，需要设置所有三个属性。

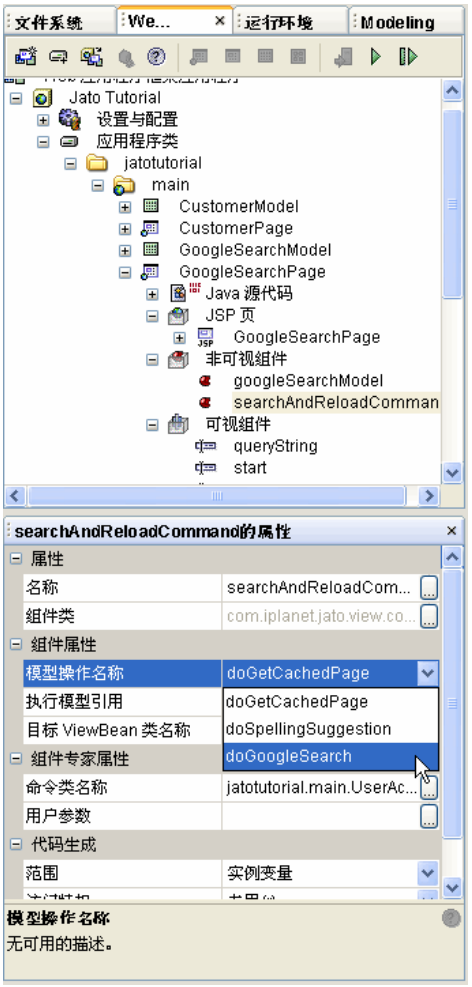



5. 对于 “执行模型引用” 属性，单击省略号按钮来启动其编辑器。

6. 选择 “模型引用 （缺省）” 选项。




7. **启动“模型类名称”属性编辑器。**
8. **浏览并选择“GoogleSearchModel”。**
9. **返回到上述编辑器，正确设置“模型类名称”属性后，请单击“确定”。**
   “模型操作名称”是一个下拉式控件。




10. 从其选项列表中选择 “*doGoogleSearch*”。
11. 启动 “*目标 ViewBean 类名称*” 编辑器。
12. 浏览并选择 “*GoogleSearchPage*”。
13. 单击 “确定” 以完成设置按钮的 “请求处理程序” 属性。

    在 GoogleSearchPage 的 “非可视组件” 节点上增加的内容显示为：*“searchAndReloadCommand”*。

# 设置 JSP 内容格式

测试运行此页之前，请务必根据需要设置 JSP 格式。

1. **在“GoogleSearchPage”下，展开 JSP 节点并双击“GoogleSearchPage JSP”，以便在 IDE 源编辑器中打开它。**
2. **为这些字段提供合适的大小写名称。**
3. **向复选框字段增加标签属性，并删除自动创建的标签。**
4. **为该页提供一个标题，并用水平线规则将其分为两部分：输入字段在顶部，仅供显示的静态文本字段在底部。**

最吸引人的 JSP/HTML 代码段以**粗体**显示在下方：

```
<jato:form name="GoogleSearchPage" method="post">
<h2>Google Search</h2>
<table border=0 cellspacing=2 cellpadding=2 width="100%">
<tr>
  <td align=right valign=middle width="20%"><b>Search for:</b></td>
  <td align=left valign=middle><jato:textField name="queryString"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"><b>Start:</b></td>
  <td align=left valign=middle><jato:textField name="start"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"><b>Max Results:</b></td>
  <td align=left valign=middle><jato:textField name="max"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"><b>Restrict Search:</b></td>
<td align=left valign=middle><jato:textField name="restrict"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"><b>Filter:</b></td>
  <td align=left valign=middle><jato:checkbox name="filter" label=
"Filter?"/></td>
</tr>
<tr>
  <td align=right valign=middle width="20%"></td>
  <td align=left valign=middle><jato:checkbox name="safeSearch" label="Safe
Search?"/></td>
</tr>
</table>
<jato:button name="search"/>
<hr>
Search Time: <jato:text name="searchTime"/>
Results <jato:text name="startIndex"/>
 to <jato:text name="endIndex"/>
 of <jato:text name="estTotal"/>
<jato:hidden name="key"/>
</jato:form>
```

第 18 章

# 教程 4.3 节
# 测试运行 Google 搜索页

本章说明如何运行 Web 应用程序框架应用程序。

## 任务 3：测试运行 Google 搜索页

由于已对若干个类进行了修改，所以请务必编译该应用程序。

1. **右键单击 “应用程序类”节点，然后选择 “全部编译”操作。**

   如果在 Sun Java System Application Server 上运行应用程序，必须在对其进行更改后部署它。

2. **选择 “Web 应用程序框架应用程序”节点 (JatoTutorial)，然后单击 Web 应用程序框架工具栏上的 “部署”按钮。**

3. **选择 “GoogleSearchPage”节点，然后单击 “执行页 （重新部署）”按钮。**

   使用此执行和重新部署选项会重新启动服务器，以确保服务器应用全部更改且不使用任何缓存的资源。

   缺省浏览器会启动应用程序。

   页的结果部分最初为零值。

   搜索将返回这些字段的值。

**警告 --** 如果您收到以下异常，您可能忘记执行 4.2.1 部分的第 15 步 （从 GoogleSearchPage 的 “自动检索模型”属性中删除 “googleSearchModel”）：

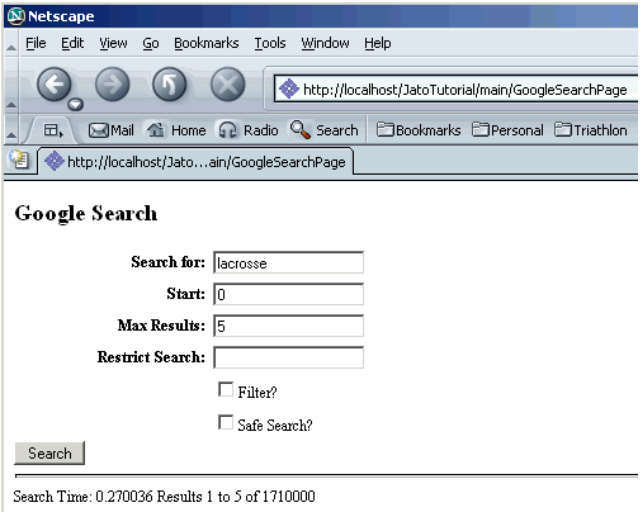
```
com.iplanet.jato.NavigationException: Exception encountered
during forward
Root cause = [com.iplanet.jato.model.ModelControlException: no
current dataset assigned yet]
```

## 尝试搜索

1. **输入搜索查询字符串。**

   在下图中，*搜索*内容为 *lacrosse*。

2. **单击 “Search” 按钮。**




3. **尝试其他搜索以查看获得的结果。**

第 19 章

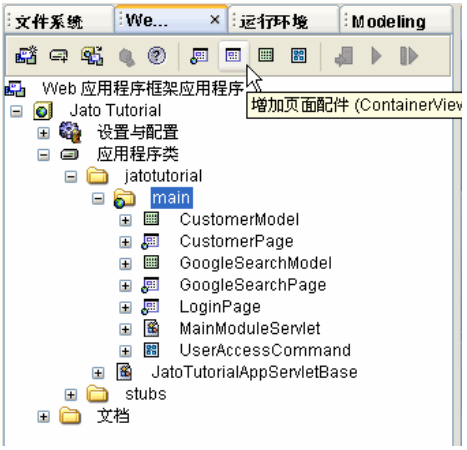
# 教程 4.4 节
# 将结果列表增加至 Google 搜索页

本章说明如何创建 TiledView 页面配件组件，以显示来自 Web 服务模型的结果列表。TiledView 将被增加到现有页组件中。

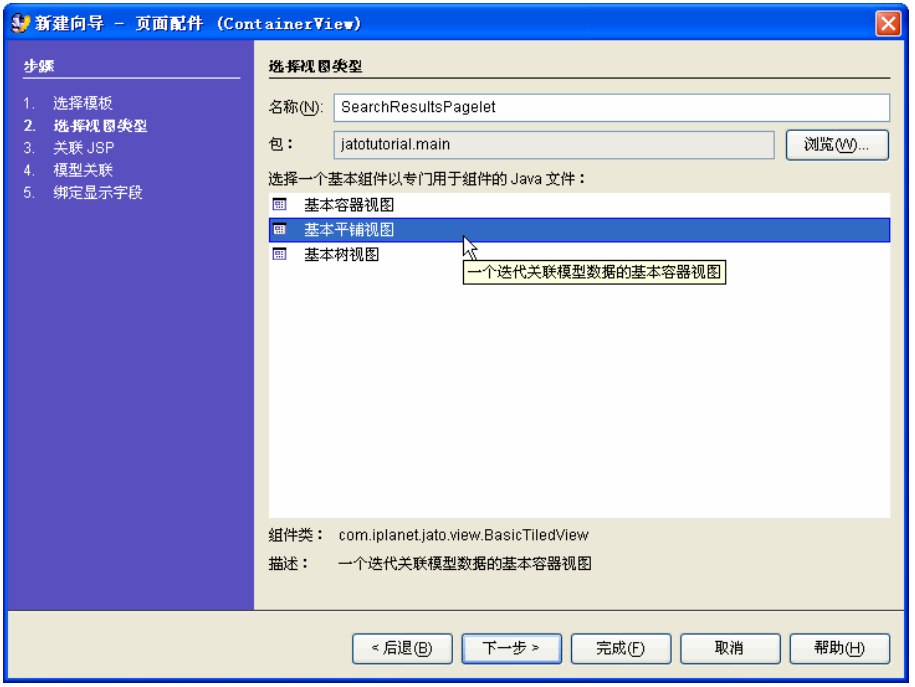
## 任务 4：创建 TiledView 页面配件

### 增加 TiledView

1. **选择 “main” 模块，并在 Web 应用程序框架工具栏上单击 “增加页面配件” 按钮。**

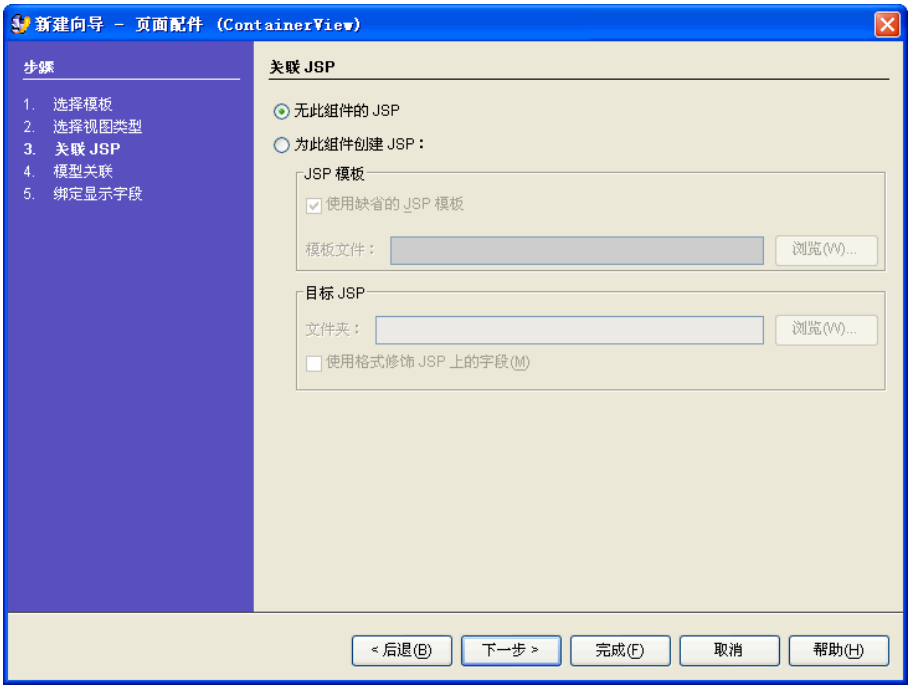



此时会显示 “选择视图类型” 面板。




2. 在“*名称*”字段中输入 *SearchResultsPagelet*（来替换 < *缺省值* >）。
3. 从页面配件组件类型列表中选择“*基本平铺视图*”。
4. 单击“下一步”。

   此时显示“关联 JSP”面板。




没有为此页面配件组件创建匹配的 JSP。此页面配件组件的 JSP 标记及内容将直接自动增加到父页组件的 JSP 页中。

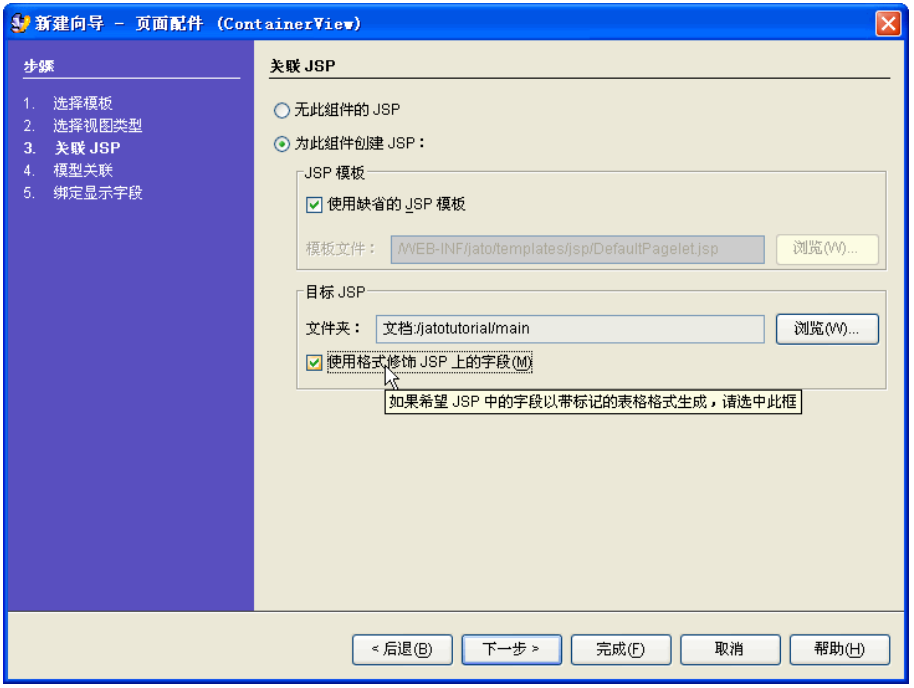
问题在于是否要为页面配件组件创建 JSP，两者均可。决定因素是页面配件组件在 JSP 端重用的方式。如果无论页面配件由哪个页面 （或另一个页面配件）生成，它都以相同的方式呈现，则应为该页面配件创建 JSP。这个 JSP 页面配件文件将包含在需要该文件的各个父页及页面配件 JSP 中 （JSP 文件包含指令）。因此，对 JSP 页面配件文件的任何更改都将反映在包含该文件的相应位置处。

但是，对于包含页面配件的各个父 JSP 页和页面配件来说，如果需要以不同方式灵活呈现页面配件，则必须在每个父 JSP 页或页面配件文件中内联并定制 JSP 页面配件内容。

---

**注 --** 在下图中，显示了为新页面配件创建 JSP 文件的其他页面配件技术。在这里选择“为组件创建 JSP”单选按钮，同时选中 “使用格式修饰 JSP 上的字段”框。此外，本技术与以上显示的技术之间的区别在于，新的页面配件具有关联的 JSP 片段。将此页面配件增加为另一个页面或页面配件的子视图组件时，如果有一个关联的 JSP，则父组件将使用包含指令来呈现 JSP 片段。本教程中，两种方法均可使用。

---

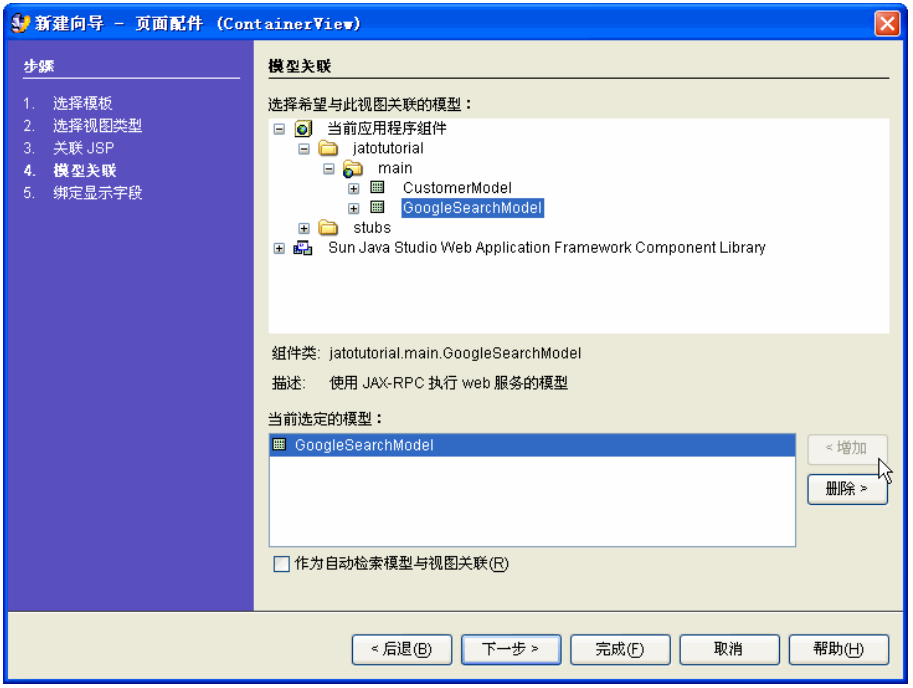



您可能无法很快了解这些技术的细微差别，但请不要担心。当您更加熟练使用 JSP 页和 Web 应用程序框架后，就会开始完全了解 Web 应用程序框架页及页面配件组件的灵活性和重用性的强大功能。

有关详细信息，请阅读本向导面板包含的说明，并请参阅 *《Web 应用程序框架开发人员指南》*和 *《Web 应用程序框架组件编写人员指南》*。

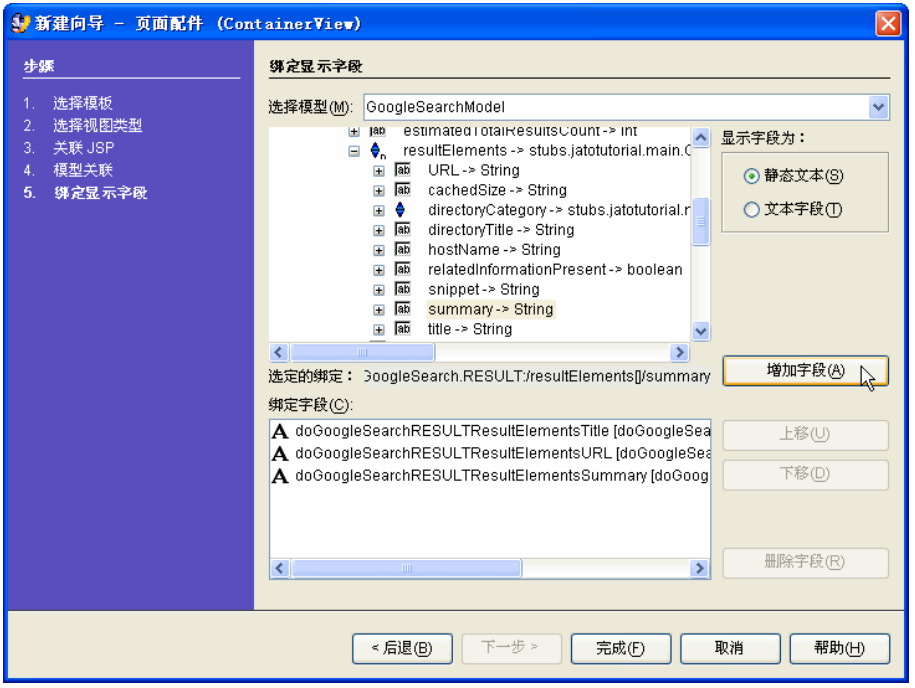
5. **单击 “下一步”。**

   此时显示 “模型关联”面板。



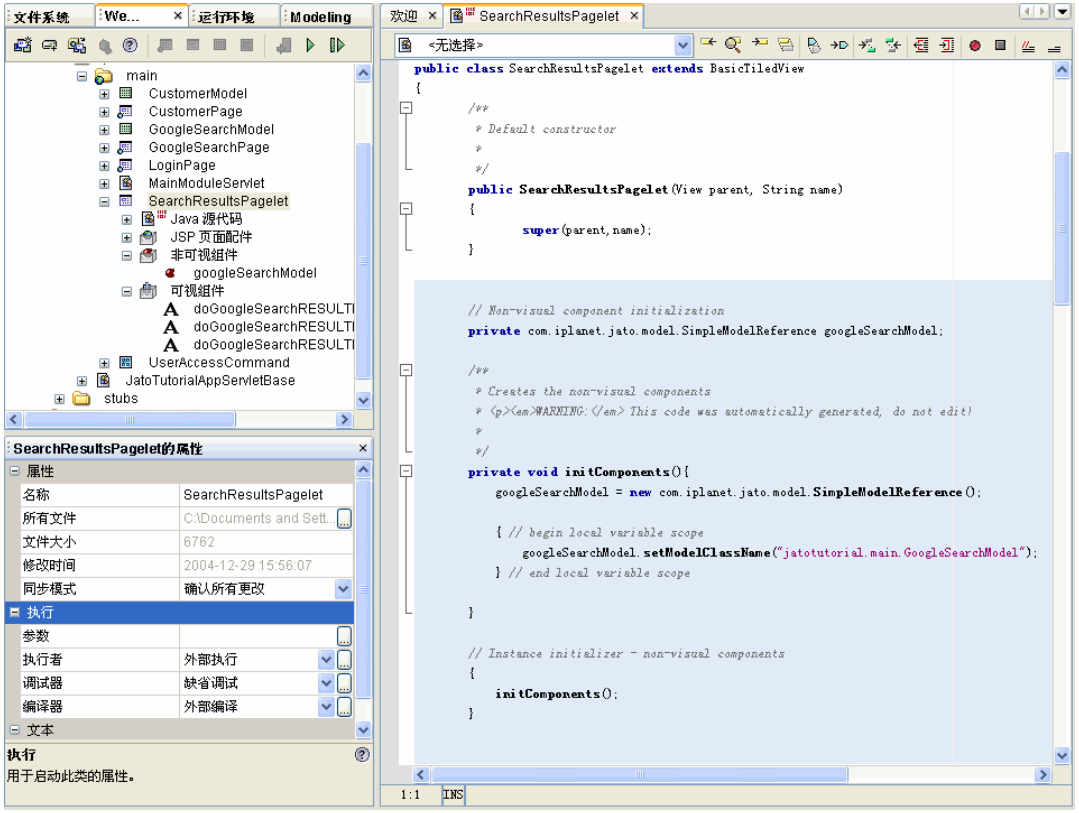

6. **展开“当前应用程序组件”以显示** `jatotutorial/main`。
7. **选择“*GoogleSearchModel*”。**
8. **单击“增加”。**
9. **单击“下一步”。**

   此时显示“绑定显示字段”面板。




10. **展开 “*doGoogleSearch*” 节点。**
11. **展开 “*返回*” 节点。**
12. **展开 “*resultElements*” 节点。**
13. **增加以下三个返回参数作为 “静态文本” 字段：**
    - title
    - URL
    - summary
14. **单击 “完成”。**

    您已经创建了 SearchResultsPagelet TiledView，其中三个字段绑定到 GoogleSearchModel 中的某些返回参数中。




**15. 以更简短的名称来重命名这些字段。**

下表左列显示较长字段名称，右列显示较短名称。

| | |
|---|---|
| doGoogleSearchRESULTS**Title** | `title` |
| doGoogleSearchRESULTS**URL** | `url` |
| doGoogleSearchRESULTS**Summary** | `summary` |

## 配置 TiledView 页面配件组件

需要为 TiledView 页面配件组件设置三个属性。

1. **选择 “SearchResultsPagelet TiledView”。**
2. **在属性表单中，通过从下拉列表中选择 “*googleSearchModel*” 来设置 “主模型引用”。**

   主模型是在显示 TiledView 时控制其迭代的模型。
3. **将 “*最大平铺显示*” 设置为 5。**

   这将显示结果数限制为 5 项。

   值 -1 （缺省值）表明检索 / 显示所有可能结果。
4. **将 “*主模型数据集名称*” 属性设置为 *doGoogleSearch.RESULT:/resultElements*。**

## 获取正确的主模型数据集名称

TiledView 需要 *DatasetModel* 类型的主模型，以使视图具有平铺域。如果主模型是 *MultiDatasetModel*，可以选择指定 “*主模型数据集名称*”，以便在显示和提交周期内，TiledView 会自动在 MultiDatasetModel 上设置 *CurrentDatasetName*。

然而，无需急于完全理解这一概念。简单而言， Web 服务模型可以有多个结果集。而 “主模型数据集名称” 属性则为特定 TiledView 指定缺省情况下使用的结果集。

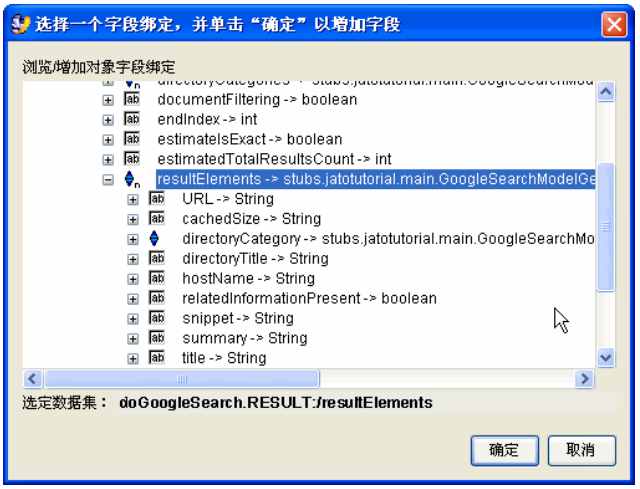
本教程已为您提供了主模型数据集名称值，但是如果由您自己指定该值，您如何知道应选择哪一个值呢？如果您对 Web 服务非常熟悉，很可能知道主模型数据集名称值。目前，最令人心烦的是输入错误，会导致一些讨厌的运行环境异常。

目前，应用程序框架工具还没有一种通过浏览 Web 服务并选择*主路径*的直接方式来设置此值。但是可以使用一次性浏览 Web 服务技术获得此值，从而复制它，然后将其粘贴到属性中。

首先在应用程序中选择 “GoogleSearchModel”。

1. **右键单击模型，选择 “浏览 / 增加对象字段绑定”。**


启动 “Web 服务字段绑定”编辑器。

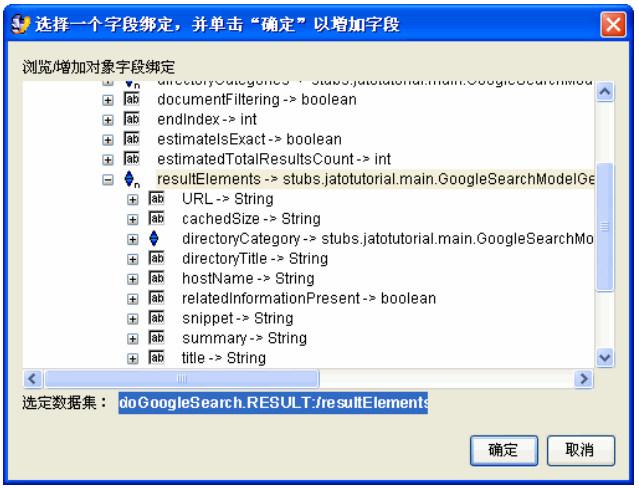



2. **现在需要向下导航到 TiledView 中的字段被绑定的操作路径。**

3. **选择字段的父节点：*resultElements***

   请注意此编辑器底部以粗体显示的 “选定数据集”值：*doGoogleSearch.RESULT:/resultElements*。

   它尽管看起来不可选，但实际上可以使用鼠标单击 / 拖动选择它。




4. **突出显示它之后，按 Ctrl-C，将该值复制到缓存。**

5. **现在，可以选择 “取消”退出此编辑器。**

   不要单击 “确定”，因为这样做会在 Web 服务模型中增加一个字段。尽管这样没有任何坏处，但也完全不需要该字段。

6. **选择 “SearchResultsPagelet TiledView”节点，并将该值粘贴至其 “主模型数据集名称”属性中。**

# 向页增加页面配件

必须借助*根视图*才能显示页面配件。页 (ViewBean) 是根视图。根视图是容器视图，它可以包含其他视图，但不能被其他容器视图所包含。所有视图分层结构必须有根视图。根视图下面有多少级子视图完全是任意的，取决于开发人员的设置。

这与包含驱动器及目录的 PC 文件系统很相似。驱动器 （与页相似）总是位于各个绝对路径 （根）的顶级，所有驱动器都位于路径的顶级。

目录 （与页面配件相似）必须包含在驱动器或其他目录下。这些目录可以任意深层地嵌套在驱动器下。

文件 （与显示字段相似）必须被驱动器或目录包含。文件不能包含其他文件、目录或驱动器。

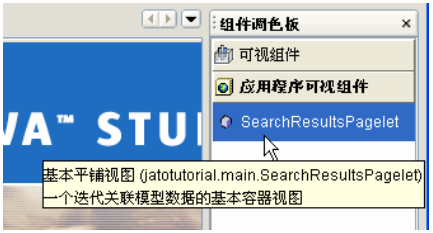
1. **展开 “GoogleSearchPage” 节点。**
2. **右键单击 “GoogleSearchPage” 的 “可视组件” 节点，然后选择 “增加可视组件”。**
   此时将启动 “组件浏览器”。


3. **展开 “当前应用程序组件” 节点。**
4. **展开 “jatotutorial”。**
5. **展开 “main” 节点。**
6. **选择 “SearchResultsPagelet” TiledView 组件。**
7. **单击 “确定”。**

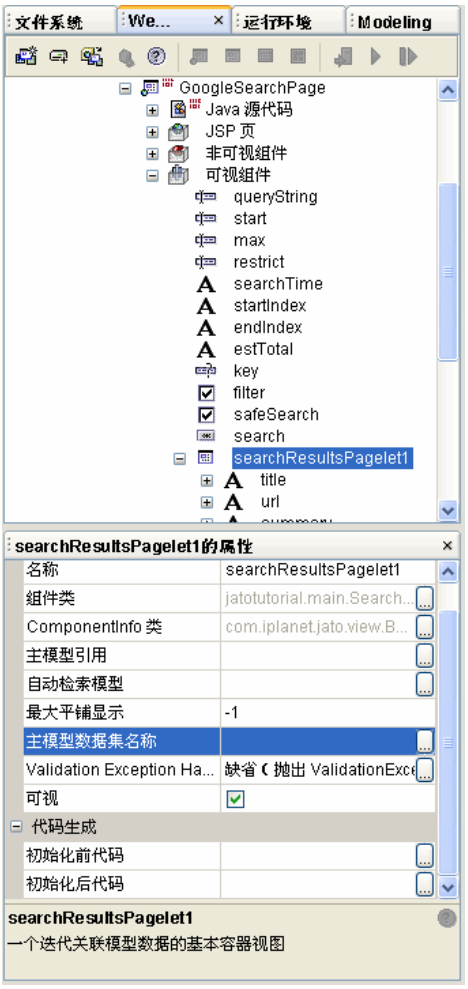
---

**注 --** 下图显示了使用组件选择面板增加页面配件组件的另一种可行方法。

---




如果在 “组件选择面板” 窗口中选择 “应用程序可视组件” 部分，将看到所有目前已在应用程序中创建的页面配件组件。例如，可以找到 SearchResultPagelet。无需执行步骤 2-7，只需在 “组件选择面板” 中单击 SearchResultPagelet 即可。




SearchResultsPagelet TiledView 被作为可视组件增加到 GoogleSearchPage 下，与其他可视组件位于相同位置下。请注意，该页面配件本身并没有 JSP。该页面配件及其它所包含的可视组件的标记会增加到父页组件的 JSP 页中。页面配件组件可被多个页组件重用，但此主题不在本教程讨论之列。

---

**注 --** 在第 133 页上的“增加 TiledView”中，显示了另一种创建页面配件的方法。在该方法中，创建了一个新的 JSP 片段成为新页面配件的关联 JSP。如果使用这一替代方法，那么不会在父页 GoogleSearchPage 中增加新增页面配件的标记，而只增加页面配件 JSP 片段的 JSP 包含指令。上述两种方法在逻辑上增加页面配件的步骤均相同，但在更改 GoogleSearchPage JSP 上有所不同，原因是存在页面配件的关联 JSP 片段。

---

# 设置 JSP 格式

1. **双击 “GoogleSearchPage JSP”页节点，打开 JSP。**

   在 JSP 底部，已插入 TiledView 页面配件及包含的显示字段标记，但未设置任何格式。

   可以增加需要的所有 HTML 标记，但属于 SearchResultsPagelet TiledView 的显示字段标记必须嵌套在 `jato:tiledView` 标记内。

2. **在 HTML 格式设置方面，可以尽情发挥您的创造性，也可以利用以下内容帮您快速入门。**

---

**注 --** 在结束 JATO 表单标记 (`</jato:form>`) 之前插入所有内容，并将当前内容保持不变。正文标记之间的所有内容都显示在这里。然而，只是需要增加**粗体**代码。

---

```
<body>
<jato:form name="GoogleSearch" method="post">

<jato:hidden name="key"/>
<h1>Google Search</h1>

<h2>Search Criteria</h2>
Search for: <jato:textField name="queryString"/><br>
Start: <jato:textField name="start"/><br>
Max results: <jato:textField name="max"/><br>
<jato:checkbox name="filter" label="Filter?"/><br>
<jato:checkbox name="safeSearch" label="Safe Search?"/><br>
Restrict: <jato:textField name="restrict"/><br>
<jato:button name="search"/>
<hr size="3">

<h2>Results</h2>
Search Time: <jato:text name="searchTime"/><br>
<jato:text name="startIndex"/> to <jato:text name="endIndex"/>
of <jato:text name="estTotal"/>

<jato:tiledView name="searchResultsPagelet1">
<hr size="1">
Title: <jato:text name="title" escape="false"/><br>
<a href="<jato:text name='url'/>" target="_blank">
  <jato:text name="url"/>
</a><br>
Summary: <jato:text name="summary" escape="false"/><br>
<br>
</jato:tiledView>

</jato:form>
</body>
```

需要对以上几点进行说明：

首先，查看 *title* 及 *summary* 字段标记的 *escape* 属性。缺省值为 *true* 时，表明对所有特殊字符进行转义。这意味着对最终用户而言，所有在字段值中转义的 HTML 标记均可见。此 Web 服务将粗体标记 (<b>) 置于所有与最终用户输入的查询字符串匹配的词的两端。因此，当指定 `escape` 为 `false` 时，可以告诉此标记，要使该标记呈现为 HTML 标记，而不显示给最终用户。可以利用此属性，对其设置 *false* 标记，然后再设置 *true* 标记来查看有何区别。

其次，请注意表示 *url* 字段的标记出现过两次。简单地复制 / 粘贴标记可以多次显示相同的动态数据。在这种情况下，标记的第一个实例将填充锚点标记 (`<a href="..."`) 的 *href* 属性。

另请注意，此 *url* 字段标记实例的名称属性使用单引号。这是因为此标记包含在 href 属性值的双引号内部。如果在双引号内部嵌套双引号，会在运行环境中呈现最佳效果，但许多 HTML 编辑器将在这种情况下报错。在双引号内部使用单引号将会补救这些情形。

*url* 字段标记的第二个实例是将其作为链接文本显示给用户。

位于两个 `jato:tiledView` 标记之间的所有内容将在返回的每行数据中出现一次（本例中共出现五次）。

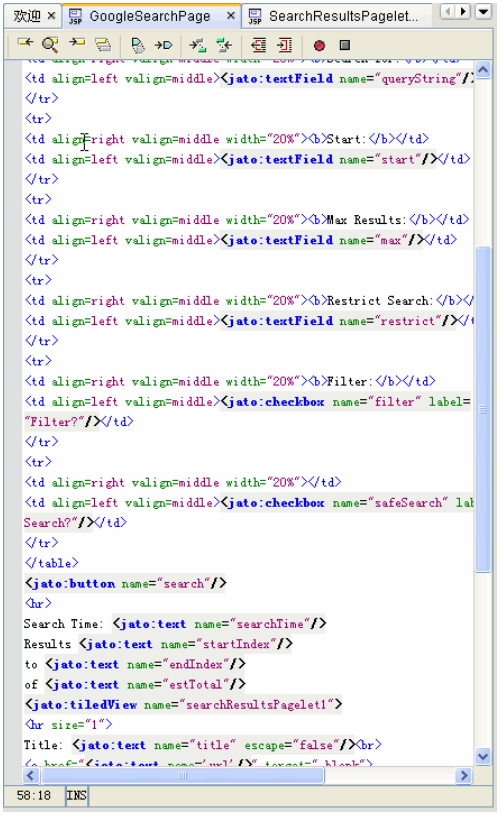
---

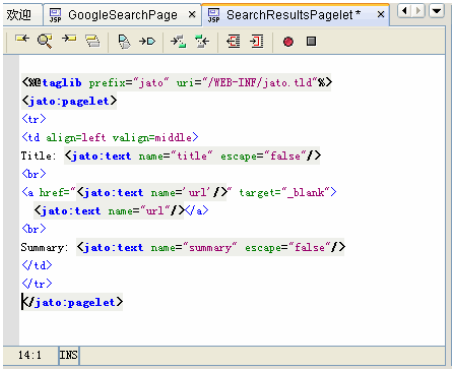
**注 --** 后面两部分，代码示例及其解释，与创建 SearchResultPagelet 但没有关联的 JSP 片段有关。因为 GoogleSearchPage.jsp 为页面配件子视图自动生成标记，因此产生了许多需要的标记格式。如果已按照第 133 页上的 “增加 TiledView” 中的说明，使用另一种方法创建了关联的 JSP 片段，则请参阅第 147 页上的“设置 JSP 和页面配件 JSP 片段格式 （另一种方法）”。

---

# 设置 JSP 和页面配件 JSP 片段格式 （另一种方法）

1. **双击 “GoogleSearchPage JSP” 页节点，打开该页的 JSP。双击 “SearchResultPagelet JSP” 片段节点，打开页面配件 JSP 片段。JSP 源代码图将显示在下方。**


欢迎
GoogleSearchPage
SearchResultsPagelet...
58:18
INS


欢迎
GoogleSearchPage
SearchResultsPagelet*
14:1
INS

如果回忆一下第 133 页上的 “增加 TiledView”中的内容，就会知道我们创建 SearchResultPagelet 时，不仅创建了 JSP 片段与新的页面配件关联，而且自动选择了设置格式复选框。请参阅上图。请注意，GoogleSearchPage 随同 HTML 表格标记和表头标记一同被自动更新，并且自动增加了 searchResultPagelet1 tileview，包含页面配件 JSP 片段的指令。您可以接受目前的自动格式更改，或者更改上图显示的两个 JSP 文件，使搜索结果更具可视性。

第 20 章

# 教程 4.5 节
# 测试运行 Google 搜索页

本章说明如何运行 Web 应用程序框架应用程序。

## 任务 5：测试运行 Google 搜索页并产生结果

由于已经创建了一个新类并对其他两个类进行了修改，所以请务必编译 / 部署应用程序。

1. **右键单击 “应用程序类” 节点，然后选择 “全部编译” 操作。**

   如果在 Sun Java System Application Server 上运行应用程序，必须在对其进行更改后部署它。

2. **选择 “Web 应用程序框架应用程序” 节点 (JatoTutorial)，然后单击 Web 应用程序框架工具栏上的 “部署” 按钮。**

3. **选择 “GoogleSearchPage” 节点，然后单击 “执行页 （重新部署）” 按钮**

   使用此执行和重新部署选项会重新启动服务器，以确保服务器应用全部更改且不使用任何缓存的资源。

   缺省浏览器会启动应用程序。

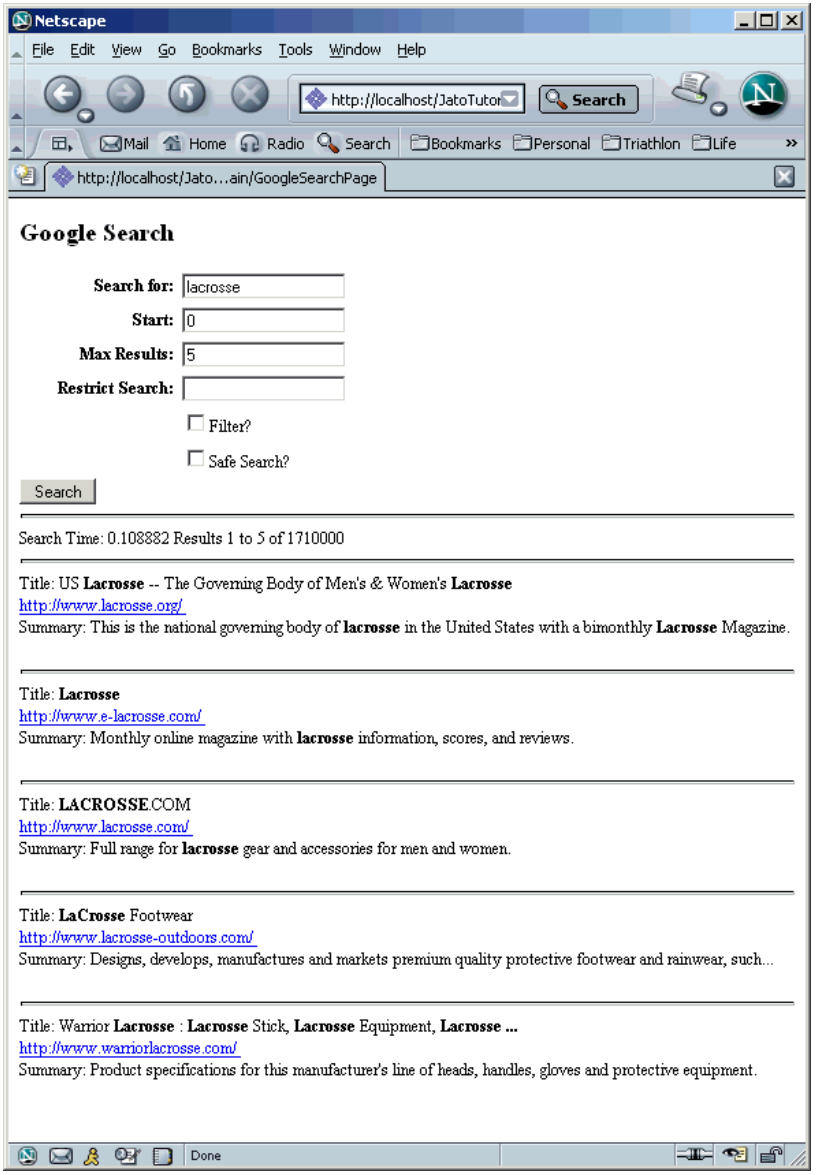
   页的结果部分最初为零值。搜索操作将返回这些字段的值，并列出五个与查询字符串匹配的链接。

### 尝试搜索

1. **输入搜索查询字符串。**

在下图中，*搜索*内容为 *lacrosse*。

2. **单击 “Search” 按钮。**




3. **尝试进行其他搜索以查看获得的结果。**

# 索引

## H



## J



## K



## L



## M



## P