Introduction to the Programming Series

Forte™ for Java™, Internet Edition, 2.0

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

Part No. 806-7516-10
December 2000, Revision A

Contents

Overview of the Forte for Java Programming Series

Conventions 6

About the Forte for Java Programming Series 7

Application Types. 7

Structure of the Programming Series. 8

Application Design Issues 9

Forte for Java, Internet Edition Documentation Set 11

Documentation Set 11

Online Help 11

Javadoc 11

Overview of the Forte for Java Programming Series

Welcome to the *Introduction to the Programming Series*, the first book of the Forte™ for Java™ programming series. This book focuses on designing enterprise applications that will be developed with Forte for Java, Internet Edition. It explains the purpose of the Forte for Java programming series and how it is structured. It also discusses some preliminary application design considerations.

The two books of the Forte for Java programming series currently available are:

- *Building Web Components*
- *Programming Persistence*

Who should read this book? This book is written for programmers wishing to learn design principles that can be applied to the technologies available in Forte for Java in order to build enterprise applications. The book assumes a general knowledge of the Java programming language.

Before you read this book: The following list of resources can help you understand the concepts upon which this book is based:

- Java™ 2 Platform, Enterprise Edition Blueprints—`www.java.sun.com/j2ee/blueprints`
- *Java™ 2 Platform Enterprise Edition Specification*—`www.java.sun.com/products`
- *Java™ Servlet Specification, v2.2*—`www.java.sun.com/products/servlet/index.html`
- *JavaServer Pages™ Specification, v1.1*—`www.java.sun.com/products/jsp/index.html`

Conventions

This table provides information about the conventions used in this document.

Format	Description
italics	Italicized text represents a placeholder. Substitute an appropriate clause or value where you see italicized text. Italicized text is also used to designate a document title, for emphasis, or for a word or phrase being introduced.
`monospace`	Monospace text represents example code, commands that you enter on the command line, directory, file, or path names, error message text, class names, method names (including all elements in the signature), package names, reserved words, and URLs.
`monospace bold`	Monospace bold text represents user input contrasted with computer output.
ALL CAPS	Text in all capitals represents file system types (GIF, TXT, HTML and so forth), environment variables, or acronyms (FFJ, JSP).
Key+Key	Simultaneous keystrokes are joined with a plus sign. For example, Ctrl+A means press both keys simultaneously.
Key-Key	Consecutive keystrokes are joined with a hyphen. For example, Esc-S means press the Esc key, release it, then press the S key.

About the Forte for Java Programming Series

The purpose of the Forte for Java programming series is to help you design and implement the kind of applications that can be developed in the Forte for Java integrated development environment (IDE). These applications range from simple web-centric applications (web clients accessing a single application service running on a web server and accessing a database) to complex distributed enterprise applications.

Depending on the complexity of the applications, you might write JavaBeans™ components, applets, JavaServer Pages™ (JSP pages) or servlets. You might write your own database access code or use transparent persistence. You might be part of a small group of developers that requires minimal source code management or part of a large multi-team group that requires collaborative development features.

The Forte for Java programming series should help you understand and select from these various design options. Regardless of the application requirements, you will almost inevitably have to learn how to build some components and modules using the Forte for Java IDE. Hence, at a lower level, the programming series should help you build the constituent parts of your application, assemble them, test them, and so forth.

Application Types

The programming series is designed to help you develop applications of different types. Some technologies can be used in many types of applications, for example:

JSP Pages and Servlet Components: The most important programming decisions concern whether application logic is contained mainly in JSP pages, or whether the JSP pages provide only for presentation (while business logic is contained mainly in servlets), or whether servlets are used mainly to coordinate access to logic contained in classes or JavaBeans components. The complexity and scale of the application influences how you think about this problem, and hence, how to program your JSP pages and servlets. The documentation of JSP pages and servlets needs to take all these possibilities into account.

Persistence: The kind of persistence solution required depends upon the needs of an application. In some cases, SQL code is written by hand, in others it can be automated using a transparent persistence type of solution. Sometimes persistence is built into JSP pages, sometimes into servlets, sometimes into classes referenced by JSP pages or servlets. These are all important programming decisions based upon the needs of the application. Hence the documentation of persistence needs to take all these possibilities into account.

Collaborative development: The type of collaboration you need depends on the size of your development effort and the scope and complexity of your application. Forte for Java, Internet Edition offers solutions for versioning, and one of them, Forte™ TeamWare, supports parallel development in large teams. Other source-code management teams can also be used.

Structure of the Programming Series

The programming series must deal with an inherent complexity: On the one hand, Forte for Java contains a number of discrete tools for developing various component types. Each of these component types is somewhat independent, but depends in its programming logic on application context. On the other hand, there is a broad spectrum of applications, ranging from simple to complex, the programming of which involves choosing from among the different component types and their features.

Without first understanding the various types of components—their capabilities and features, it is hard to design an application. But without first designing an application it is hard to know what programming choices to make in using the various component types.

To address this complex matrix of choices, the programming series first presents an overview of application design issues, discussing a number of example applications, from simple to complex.

The other books of the programming series are more component-specific. These books present programming information about different Java™ 2, Platform, Enterprise Edition (J2EE™) technologies. Two books are currently available:

- *Building Web Components* for programming web (JSP pages and servlet) components
- *Programming Persistence* for programming with the JDBC and Transparent Persistence technologies.

Application Design Issues

This section looks briefly at J2EE applications, attempting to sort out the various internal functions they need to perform. The examples use a basic multi-tier approach, in which a client tier accesses services within an application services tier that, in turn, accesses data in a data sources tier. Most of the interesting logic is in the application services tier.

The application services tier typically performs a number of very different functions. At the highest level—for applications with web clients—these can be broken down as follows:

Control and Sequencing This involves receiving messages from web clients, determining if business logic needs to be executed (routing requests to objects that perform business logic), and gathering requested results. This function often involves deciding which screen is next provided to a user, depending on the history and current state of the user's session as well as the request the user is currently making.

Business logic This involves the basic work performed by the application and might include a number of different functions, depending on the application: enforcing business rules, performing complex calculations, coordinating multi-step operations, and possibly managing the interaction of multiple users.

Persistence This involves retrieving and updating persistent data.

Presentation This involves taking the results of business logic, formatting the data, and presenting it back to web clients.

These are not the only functions performed in the application services tier. The application must also manage its users, keeping track of user sessions and applying the appropriate security when one component attempts to access another.

The goal of this section is to explore control and sequencing, business logic, persistence, and presentation functions, looking at different J2EE technologies for implementing them. Depending on the scale and complexity of an application, different component types could have different advantages and disadvantages in performing these functions.

In general a very modular design is preferred, in which different functions are performed by different components. This approach has many advantages: encapsulation of function, reusability of code, conceptual clarity, and employing the power of various J2EE component types. Within this framework, the design shown in Figure 1 (for a web-centric application) is probably not sufficiently modular.

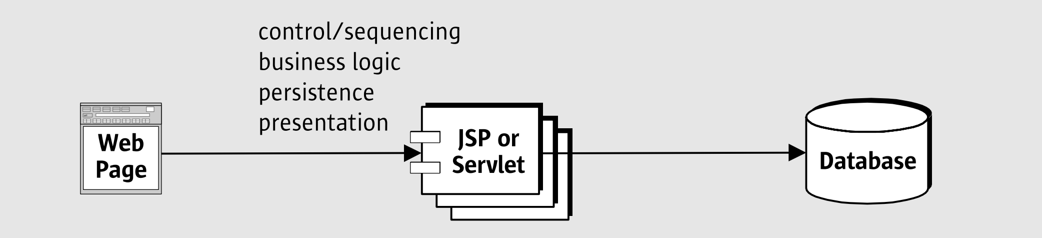

Figure 1 *Simple Design, With All Logic in JSP Pages or Servlet Components*

Instead, a design more similar to Figure 2, in which different functions are encapsulated in different objects, would be employed.

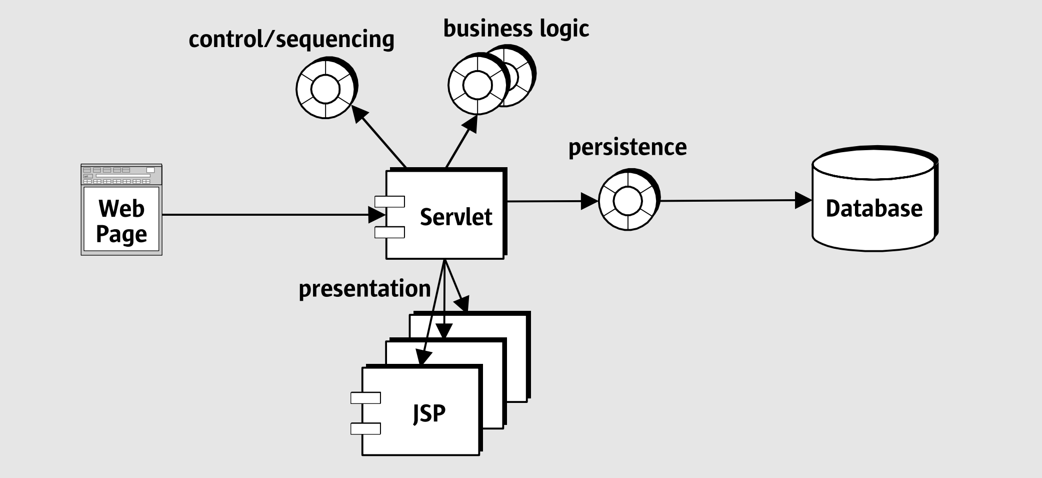

Figure 2 *Modular Architecture, With Different Types of Logic in Different Components*

Modularity is one consideration important in design. Others are performance, reliability, scalability, and security. These considerations often require applications to be distributed, where different components run on different computers in order to maximize performance, failover capability, load balancing, and secure access. Evaluating these design considerations should help you choose the appropriate technologies for your application.

Forte for Java, Internet Edition Documentation Set

Forte for Java offers a set of books delivered in Acrobat Reader (PDF) format and online help. This section provides descriptions of these documents.

Documentation Set

You can download the following documents from the Forte for Java web site:

- The Forte for Java programming series:
  - *Introduction to the Programming Series*

    Introduces the two books in the Forte for Java, Internet Edition programming series.
  - *Building Web Components*

    Describes how to build a web application as a J2EE web module using JSP pages, servlets, tag libraries, and supporting classes and files.
  - *Programming Persistence*

    Describes support for different persistence programming models provided by Forte for Java: JDBC and Transparent Persistence.
- *Forte for Java, Internet Edition Tutorial*

  Provides step-by-step instructions for building a simple web application using tools introduced in Forte for Java, Internet Edition, which facilitate creating a web module, as described in the *Java™ 2 Platform Enterprise Edition Specification*.

Online Help

Online help is available inside the Forte for Java development environment. You can access it by pressing the help key (Help on Solaris, F1 on Windows and Linux), or by choosing Help > Contents from the Help menu. This displays a list of help topics and a search facility.

Javadoc

Javadoc documentation is available within the IDE for many Forte for Java modules. Refer to the Release Notes for instructions for installing this documentation. When you start the IDE, you can access this Javadoc documentation within the Javadoc pane of the Explorer.