

# Rapid Application Development

## An introduction to OptimalJ

Development of enterprise applications using Java technologies is not for the faint hearted. Writing applications for the Java 2 Enterprise Edition (J2EE) platform is proving to be complex, difficult and tedious, slowing down advanced Java developers and creating a barrier to entry for many mainstream developers. Advanced Java developers are in short supply, and even among them experience with Enterprise Java Beans (EJB) development is rare. This is slowing time to market for business applications and challenging application reliability and performance.

To solve this problem Java development should be simplified. By providing them with a framework for delivering J2EE-compliant business applications, developers of all skill levels can build reliable, high-performance components. OptimalJ by Compuware offers a solution, a new breed of development environment enabling the rapid design, development and deployment of J2EE business applications.

OptimalJ is an advanced development environment which simplifies the rapid design, development and modification of J2EE applications. OptimalJ generates complete, working applications directly from a visual model, using active synchronization to keep both model and code up-to-date during rapid application changes. OptimalJ comes with a built-in web server and servlet engine, J2EE application server and DBMS for unit-testing purposes, allowing the developer to test the application without any deployment. For production purposes, OptimalJ generates archive packages and server deployment descriptors automatically, allowing administrators to deploy applications in the target environment rapidly.

The OptimalJ product includes the following critical features that enable organizations to build reliable J2EE business applications rapidly:

- Rapid Application Development
- Business Rules
- Pattern-Driven Generator
- Active Synchronization
- Integrated Deployment

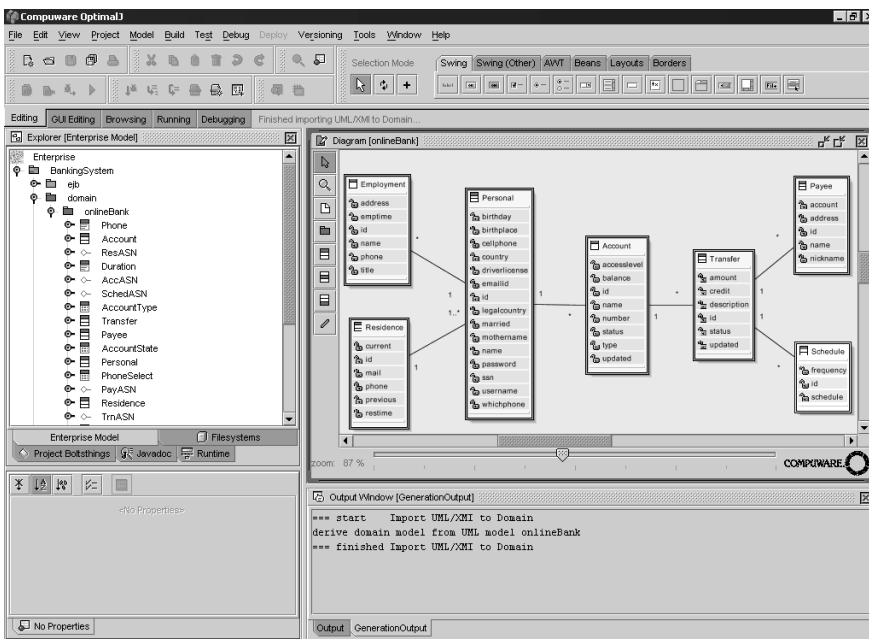
These five critical features are described in separate white papers. This white paper will focus on Rapid Application Development.

# Model-driven architecture

OptimalJ allows developers to start developing at a higher abstraction level, reducing the complexity of the J2EE platform from the start. From modeling through to deployment, application development in OptimalJ is model-driven. Through the visual model, OptimalJ ensures reuse of definitions automatically. The model-driven development paradigm allows developers to focus on what to build, not on how to build it.

## Unified Modeling Language (UML)

In a component-based development environment, comprehensive and accurate modeling is the first crucial step towards building a successful application. To manage large applications, model-based design and development is paramount. The methodology that OptimalJ promotes for developing component-based applications incorporates the industry-standard modeling techniques and notation used in the Unified Modeling Language. UML is a technology adopted by the Object Modeling Group (OMG), of which Compuware is a member.

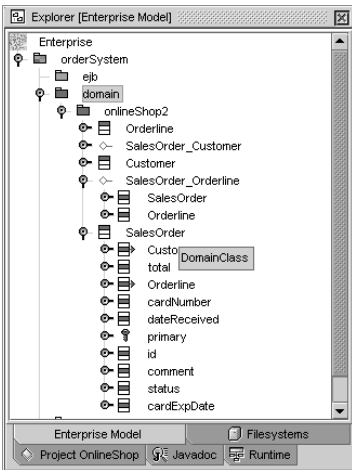
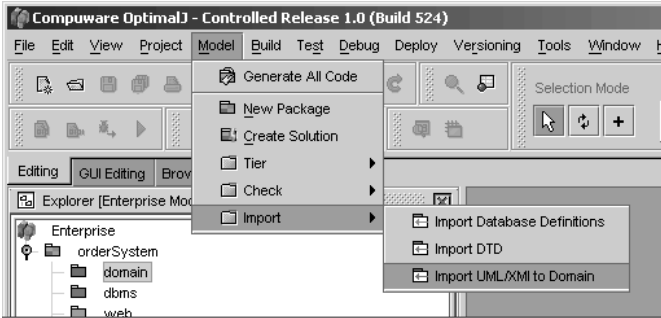


The OptimalJ workspace

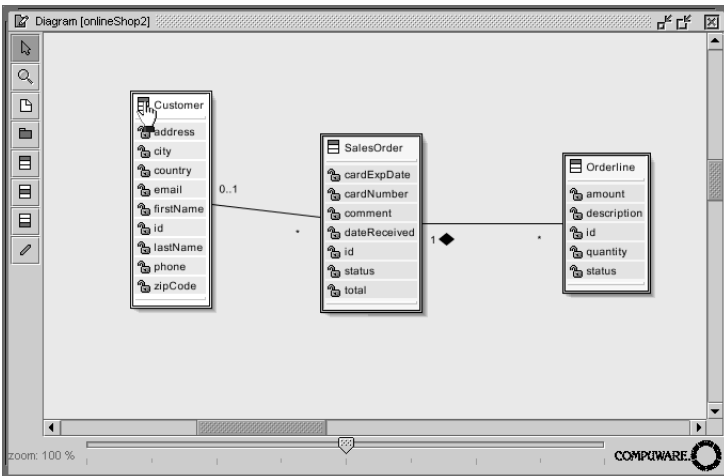
Developing in a model-based environment delivers several major advantages. The model:

- provides an overview of the application structure
- facilitates the reuse of objects and rules
- ensures consistency during development
- functions independently of implementation, so when changes occur (for example, to the technical infrastructure), the model remains valid.

The purpose of the modeling stage is to determine the appropriate scheme for defining components, including links and operations within a system. The components must fully address the functional requirements of the application, while remaining functionally independent. During the modeling phase, the components are not yet defined (no implementation exists at this stage).



Populating the Domain Model



Domain Model Editor

## The Domain Model

The OptimalJ modeling workspace is the starting point for a user's model repository. OptimalJ includes the Netbeans Integrated Development Environment and shares the Netbeans menus and windows to interface with the user. Additionally, the functionality of Netbeans can be used to work within the code generated by OptimalJ.

When developing an application with OptimalJ, the starting point is the Domain Model. The Domain Model is a high-level object model containing the information structure of the application and the relationships between the different data structures. The Domain Model focuses on business classes, like a customer, order, product, etc., their attributes, associations, aggregations, business methods and business rules. The Domain Model is geared toward integrity of the business information. The Domain Model does not contain implementation and coding details. This differentiates it from a Class Model, which also includes all technical classes necessary to implement applications that use business classes.

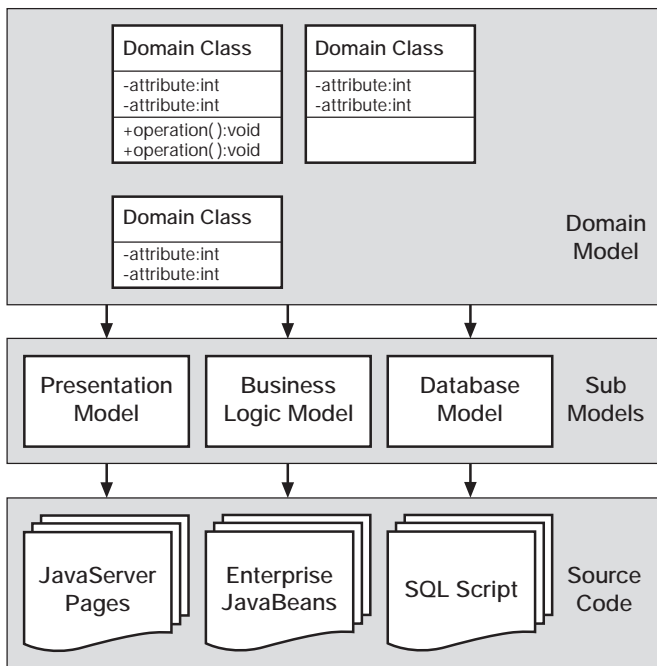
## UML/XMI Import Utility

When a separate UML modeling tool is used, developers want to economize on effort by reusing as much as possible from the modeling stage in the development stage. There are several phases in the UML-based modeling, each with different modeling activities. These range from the high-level analysis in the inception phase to the final detailed modeling of the individual components in the component modeling phase. When the modeling is complete, a UML-based model becomes available. This can be exported to an XMI (XML Metadata Interface) file. Using the UML/XMI import utility, this XMI file can help populate OptimalJ's Domain Model.

## Integrated graphical Domain Model Editor

OptimalJ includes an integrated graphical Domain Model Editor. After importing a model with the UML/XMI Import Utility, the model becomes visible in the Domain Model Editor automatically.

Instead of importing the model, developers can also use the Domain Model Editor to draw the Domain Model quickly and accurately.



Graphical overview of the generation process

Load definitions from a database

OptimalJ allows developers to upload the Domain Model from an existing relational database using JDBC. By using this utility, OptimalJ converts the database model into a Domain Model automatically, by using relational to OO mapping patterns.

Generation of the three sub-models: Web, EJB, DBMS  
From the Domain Model, OptimalJ derives three types of sub-models:

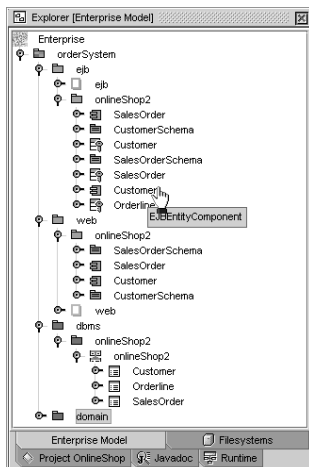
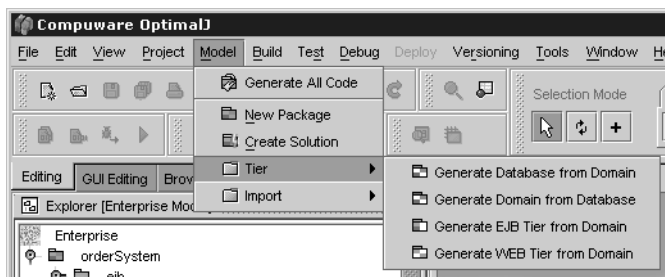
- Business Logic (EJB) Model
- Database (DBMS) Model
- Presentation (Web) Model.

Each model contains all the definitions of the components required to implement the functionality. The actual implementations will be generated out of these sub-models, consisting of JavaServer pages, servlets, Enterprise JavaBeans and SQL scripts.

The three sub-models can be generated separately or all at once, automatically. OptimalJ ensures that the models are derived from and kept in sync with the Domain Model. These models automatically inherit all relevant definitions from the Domain Model when created and/or updated.

When all models are in place, the actual code can be generated to implement the components of the different models. The automatic generation process ensures consistency with the Domain Model, saving a great deal of time and potential programming errors. Naturally, the Presentation Model depends on the Business Logic Model, and the Business Logic Model depends on the Database Model. This dependency is automatically maintained by OptimalJ.

During the generation phase, generated code is derived from the Business Logic (EJB) Model, Database (DBMS) Model and Presentation (Web) Model. The generated code exists of .java files and the .jsp files. OptimalJ does not generate code-skeletons, but delivers fully functional J2EE-compliant components and SQL scripts. These scripts create the database tables.



Generation of the source code: Web, EJB and DBMS

```

Output Window [GenerationOutput]
generate deployment descriptors for EJBModule ejb
generate JOnAS deployment descriptors for EJBModule ejb
generate J2EE deployment descriptors for EJBModule ejb

(Re)Generating web deployment descriptor
Generating action mappings for web
Generating resource properties for web
Generating mainMenu.jsp for web
Generating index.html
generate java classes for EJBEntityComponent Customer
generate java classes for EJBKeyClass Customer
generate java classes for EJBKeyClass Orderline
generate java classes for EJBEntityComponent SalesOrder
generate java classes for EJBKeyClass SalesOrder
(Re)Generating WebComponent Customer
Generating CustomerQuery.jsp
Generating CustomerMaintChange.jsp
Generating CustomerUpdateAction.java
Generating CustomerSelectAction.java
Generating CustomerCreateAction.java
Generating CustomerDeleteAction.java
Generating CustomerBrowseAction.java
Generating CustomerSetProfileAction.java
Generating CustomerNewAction.java

```

Generation of the source code

## The OptimalJ models explored

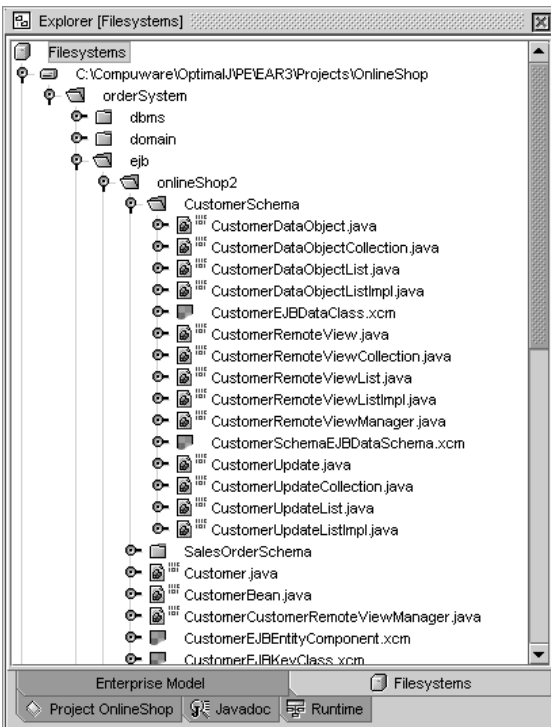
### The Business Logic (EJB) Model

OptimalJ simplifies the development of J2EE applications, enabling developers of varying experience levels to produce reliable applications rapidly. J2EE is a concept that is based on a combination of JavaServer pages, servlets and Enterprise JavaBeans (EJBs).

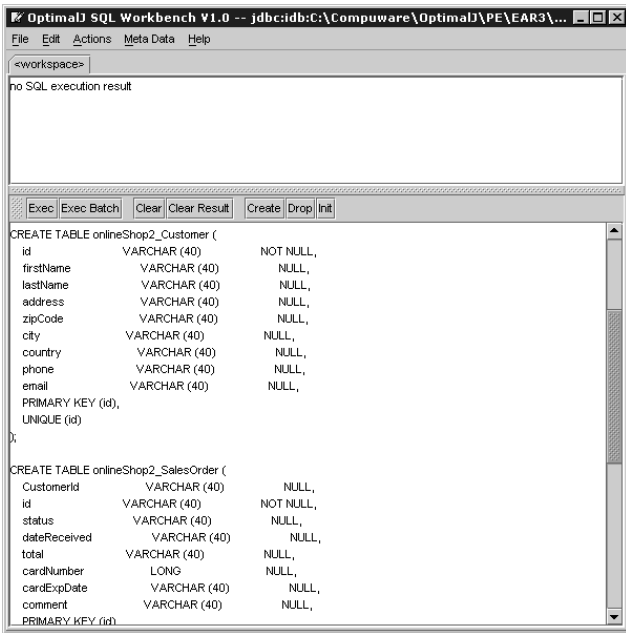
OptimalJ includes a set of major innovations that tackles the complexity of the J2EE platform. To increase productivity, it is necessary to address not only the coding aspects of the development process of EJBs, but also other ways of increasing productivity, such as integrated testing of EJBs in a runtime environment (see white paper on Deployment).

OptimalJ does not just generate EJB skeletons with interfaces and attributes, but also fully implemented EJBs. The EJBs generated by OptimalJ are consistent regarding the definitions in the Domain Model. They also contain full functionality to interact with the front-end components (JSPs) and contain all the functionality and methods that interact with the database and EJB server.

As shown (left), many Java source code files are generated in the EJB tier and the web tier. The generation of this code shows the increased productivity of OptimalJ. No manual coding is required. Creating Enterprise JavaBeans in OptimalJ reduces complexity and helps developers maximize productivity, flexibility and quality. OptimalJ ensures that effort is targeted more effectively at writing applications that truly differentiate a company's business and reduce time-to-market.



EJBs generated by OptimalJ



OptimalJ's integrated SQL Workbench

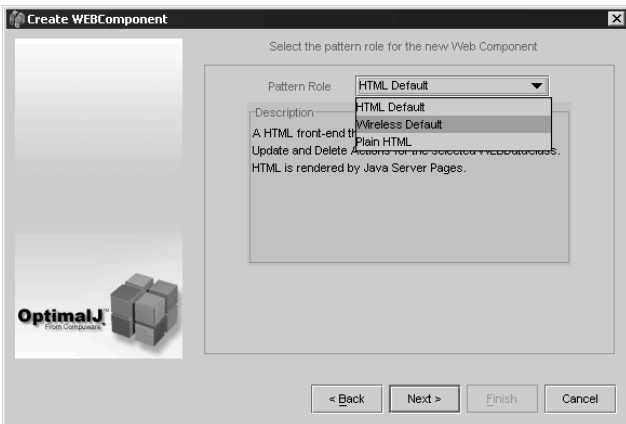
## The Database Model

Developing a new application requires database scripts to create database tables. When the Domain Model is complete, the menu option “Generate Database from Domain” will derive the Data Model from the Domain Model. The Data Model contains the relational database definitions for the application. OptimalJ generates the SQL command scripts needed to build the corresponding database tables in the underlying DBMS. The developer needs to define for which database the SQL code must be generated. To execute the SQL scripts, OptimalJ offers an integrated SQL workbench.

For database access, OptimalJ makes use of JDBC (Java Database Connectivity).

## The Presentation Model

The presentation tier defines a set of default presentation components. These components are based on JavaServer pages and servlets and can be implemented for HTML or WML clients. This presentation tier model is derived from the Domain Model, which ensures consistency and quality of the front-tier components.



Presentation patterns

The split between presentation layer and business (EJB) tier enables developers to create a new web front-end on top of an existing EJB tier. Therefore, OptimalJ includes a set of Presentation Patterns, used to create additional components. For example, by selecting different Presentation Patterns, the same data can be shown in a HTML-based web component as well as in a WML-based WAP component using the same EJB tier (see white paper on Patterns).

After generating the web components, the developer has a first-draft presentation layer and menu structure in place. Since the web components are linked to the generated EJBs, database actions like read, create, delete and update are available immediately.

OptimalJ offers a plug-in to Macromedia's Dreamweaver to make changes in the layout. This allows a web designer to change the look and feel of the web components.

## Remote View Manager

The Remote View Manager (RVM) is a key functionality of OptimalJ, creating a clear split between the presentation layer and business-tier layer. The RVM is used by the components in the web-tier layer and manages the communication with the EJB tier. In Java programs, data is sent attribute-by-attribute from the EJB to the presentation component. This is a slow and inefficient way of data transport. The RVM transports the data more efficiently. The data is first collected and packaged before it is sent to the presentation component. This is preferable from the standpoint of performance and network load. The RVM handles data transport in a transparent way, so a developer does not have to focus on optimizing the data exchange between the EJB and the Presentation Component. No manual coding is required.

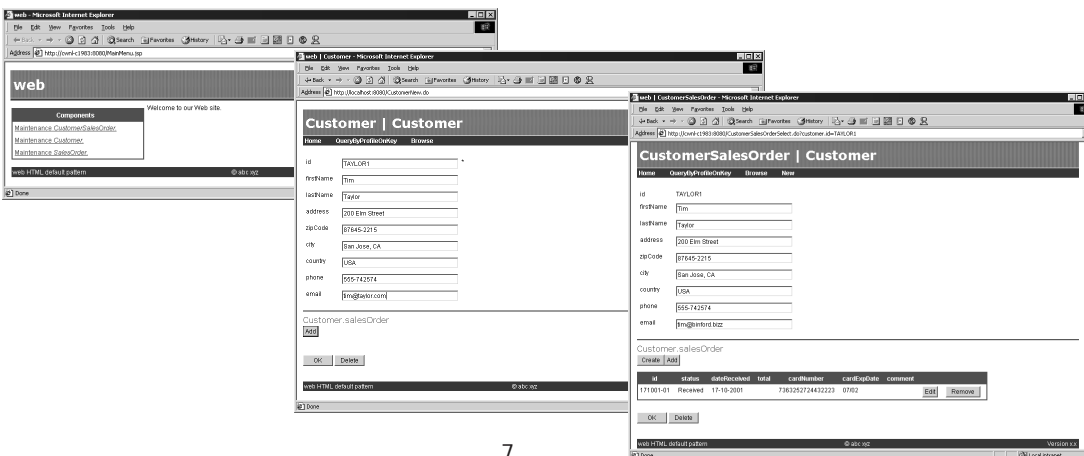
## Model View Controller

There are a number of ways to architect applications using J2EE technologies, including JSPs, servlets and EJBs. If an application development framework is not used to address how developers should create web applications based on the J2EE architecture, the user interface designs tend to group such objects indiscriminately. The Java development community has recognized the need for a framework that is useful in building web applications using J2EE technology. Model View Controller (MVC) offers this framework.

In this scheme, user interfaces are divided into three nearly independent components:

- the model (business logic tier), which holds the data being manipulated and conducts all the computations
- the view (presentation tier), which displays the data to the user
- the controller (servlets), which responds to all user actions and notifies the model and view appropriately.

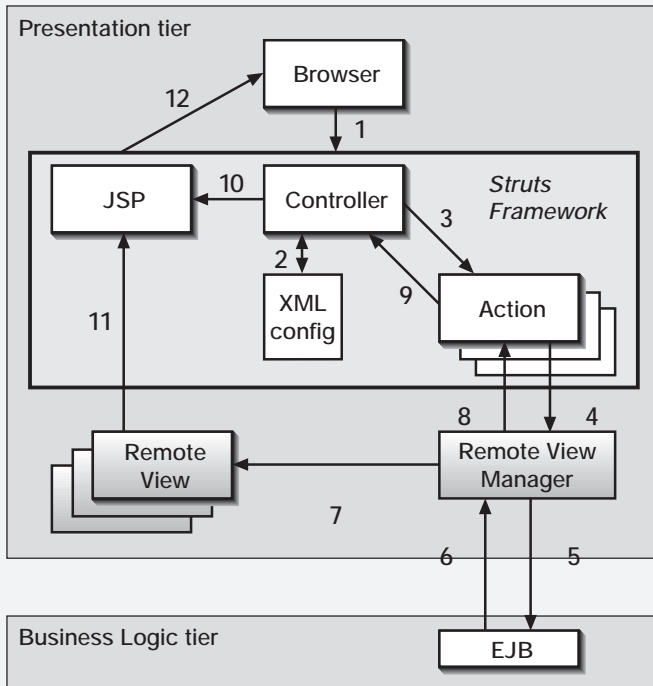
The goal of the MVC is to separate the business logic tier from the way it is represented to the user (presentation tier) and from the way the user controls it (controller). MVC decouples the tiers, thus allowing greater flexibility, scalability and possibility for reuse. MVC also provides a powerful way to organize systems that support multiple presentations of the same information.



Web components generated by OptimalJ

## The Struts Framework

OptimalJ generates JavaServer pages and servlets for use with the Struts Framework. Struts is an Open Source implementation of the MVC concept, and it is part of the Jakarta Project sponsored by Apache Software Foundation. OptimalJ uses Struts as a presentation-tier framework. OptimalJ offers a plug-in by which the Struts tags are recognized by Macromedia Dreamweaver.



Overview of the Struts Framework

The Struts Framework is servlet centric and runs on the servlet engine.

The Struts Framework is used and extended in the following way:

1. The Controller receives a request from the browser.
2. The Controller checks in an XML resource file what the related action is.
3. The Controller makes a call to the related action which is implemented as a JavaBean.
4. The action object calls the Remote View Manager to exchange data.
5. The Remote View Manager communicates with the EJB tier to exchange the requested data.
6. The data is sent over the network to the presentation tier in a coarse-grained way.
7. The received data is cached in memory in the web tier.
8. Remote View Manager returns a reference to the action object, which is processed by the action object.
9. The action object forwards the reference to the Controller, together with a true or false indicator, which depends on the result of the action. Based on the true/false indicator, the Controller determines in the XML Resource file what the next action is.
10. The Controller instantiates the JSP as a servlet and forwards the reference.
11. The servlet fetches the data based on the reference from memory (Remote View).
12. The servlet sends the HTML page with the data to the browser.

Compuware products and professional services—delivering quality applications

Compuware is a leading global provider of software products and professional services which IT organizations use to develop, integrate, test and manage the performance of the applications that drive their businesses. Our software products help optimize every step in the application life cycle—from defining requirements to supporting production service levels—for web, distributed and mainframe platforms. Our services professionals work at customer sites around the world, sharing their real-world perspective and experience to deliver an integrated, reliable solution.

Please contact us to learn more about how our comprehensive products and services can help your organization improve productivity, create higher quality applications and ensure performance in production.



[www.compuware.com](http://www.compuware.com)

All Compuware products and services listed within are trademarks or registered trademarks of Compuware Corporation. Java and all Java-based marks are the trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other company or product names are trademarks of their respective owners.  
© 2001 Compuware Corporation