

COOL:Joe™

JAVA OBJECTS FOR THE ENTERPRISE

---

REVIEWER'S GUIDE



# COOL:Joe Reviewer's Guide

## JAVA OBJECTS FOR THE ENTERPRISE

---

### INTRODUCTION

---

Sterling Software Application Development Group is a leading provider of software solutions used by Fortune 1000 IS organizations to deliver technology solutions that address the needs of their changing businesses. With a heritage of over 20 years of helping large companies deliver run-the-business systems, Sterling Software today helps people get to e-business by extending and leveraging their existing technology assets. Primarily a provider of application development software, Sterling Software has developed a strong ecosystem of partners in the fields of consulting, services, content provisioning, infrastructure and hardware.

COOL:Joe is Sterling Software's newest technology and is an enterprise-scale development environment that allows Java development teams to architect, create, deploy and manage reliable and scalable distributed applications using Enterprise JavaBeans (EJBs). The Reviewer's Guide provides a comprehensive guide to COOL:Joe. Industry analysts and media personnel can use this document to quickly obtain key facts about Sterling Software and COOL:Joe. Reviewers of COOL:Joe will understand how the product works, its features and benefits, and its competitive landscape. The Reviewer's Guide is organized with the following sections:

- A. The Fact Sheet summarizes key facts about Sterling Software and COOL:Joe and provides contact information for sales, media and technical support, the target market for the product, pricing and distribution, and the product's system requirements.
- B. An Executive Summary describes the force behind the development of the product -- the trends driving companies today towards e-business and how COOL:Joe can support robust, scaleable enterprise applications on the web.
- C. The Overview provides an in-depth look at how the COOL:Joe product works by addressing the major functional areas; Integrated Component Modeling, Advanced Wizards, and Creating Business Logic. By using sample screen shots and illustrations, the Overview section guides the reader through a typical workflow allowing you to envision how the product will be used. A typical workflow is described in the following steps:
  - 1. Model business processes and relationships to build a component architecture, which includes the individual component specifications, needed to meet new systems requirements.
  - 2. Use the Specification to Implementation Wizard to transform a component specification into Java classes.
  - 3. Generate a set of persistence support classes to access a relational database with the Persistence Generation Wizard. Use the DDL Generation Wizard to generate the database definition language. Or reverse engineer an existing database.
  - 4. Use the Editor to specify business logic and the Class Diagram to model class structures.
  - 5. The EJB Generation Wizard generates the code to implement the component as an Enterprise JavaBean.
  - 6. Compile component classes and create an executable JAR file with the Build Wizard.
  - 7. The Deployment Wizard deploys the EJB JAR file to the Enterprise JavaBean application server.
  - 8. Use the Test Harness Wizard to generate the code necessary to test the component with a Java desktop application and/or HTML servlets.
  - 9. Test the EJB.
  - 10. EJB implemented successfully.

The Reviewer's Guide continues with:

- D. The Features and Benefits section highlights the major features and benefits in an easy to read, one page format for readers to quickly ascertain the advantages of COOL:Joe.
- E. The Competitive Landscape section provides a high level view of the current competition in the Java development tools market. COOL:Joe's major competitors and competitive advantages are described.
- F. Attachment A is an introduction to Enterprise JavaBeans, which provides a brief historical perspective and the importance of EJBs to distributed computing today.
- G. Attachment B is a step by step guide to using the COOL:Joe evaluation edition, which is downloadable from the Sterling Software web site. The guide shows you how to create an EJB in 15 minutes and upon completion of the exercises, further enhances the reviewer's understanding of COOL:Joe.

The COOL:Joe Reviewer's Guide is designed to provide an overview of the product and answer the reviewer's most commonly asked questions. If you have additional questions, simply use the fact sheet to identify the appropriate contact information you need.

---

## FACT SHEET

---

Description	COOL:Joe is an integrated component architecture, modeling and EJB development environment that allows Java development teams to design, create, deploy and manage enterprise-scale applications.
Developer	<p>Sterling Software is a leading provider of software and services for the application development, business intelligence, information management, storage management, network management, VM systems management, and federal systems markets. The company is one of the 20 largest independent software companies in the world. Headquartered in Dallas, Sterling Software has a worldwide installed base of more than 20,000 customer sites and 3,700 employees in 90 offices worldwide. For more information on Sterling Software, visit the company's Web site at <a href="http://www.sterling.com">www.sterling.com</a>.</p> <p>5800 Tennyson Parkway Plano TX, 75024</p> <p>1.877.SSW.COOL or 1.972.801.6000 1.972.801.6051 Fax</p>
Target Market	IT architects who need to design large-scale distributed e-business systems. Development teams who want to create and deploy server-side components with Enterprise JavaBeans technology. Java specialists that want to be insulated from the underlying technology while taking advantage of EJBs.
Distribution	Direct from Sterling Software. COOL:Joe Evaluation copies are available to download at <a href="http://www.sterling.com/cooljoe">www.sterling.com/cooljoe</a>
Price	Pricing information will be made available at commercial release in January 2000.
Technical Support	Full technical support for COOL:Joe is provided world-wide. For additional details on technical support visit <a href="http://www.sterling.com">www.sterling.com</a> .
System Requirements	Pentium II 300 MHz microprocessor or higher recommended • CD-ROM drive • 128 MB RAM minimum, 256 MB RAM recommended • SVGA monitor resolution or better • 100 MB storage. Operating System: MS Windows NT 4.0 with Service Pack 4 with Year 2000 updates. MS Network services • Mouse • MS Word 97 or later. Relational database such as MS SQL Server 7.0, DB2/2 5.0, or Oracle 8i /8.1.5 or higher with JDBC drivers. <b>Provided:</b> Enterprise JavaBean specification classes 1.0 • JavaServer Web Development Kit 1.0 (JSWDK) • ObjectStore 5.1 and ObjectStore Java Interface • Java 2 SDK, Standard Edition (includes Java 2 Runtime environment) 1.2.2 • Java 2 SDK documentation 1.2.2 • Java Naming and Directory Interface 1.1.2 (JNDI). <b>Other:</b> Web browser • Java Activator Plug-ins • EJB Application Server.
Media Contacts	<p>Maysoon Al-Hasso Sterling Software 1.972.801.6601 <a href="mailto:maysoon.al-hasso@sterling.com">maysoon.al-hasso@sterling.com</a></p> <p>Laura Sellers Sterling Software 1.972.801.6652 <a href="mailto:laura.sellers@sterling.com">laura.sellers@sterling.com</a></p>

---

## EXECUTIVE SUMMARY

---

COOL:Joe is an EJB development environment seamlessly integrated with architecture design and component modeling capabilities. Java development teams will now be able to architect, create, deploy and manage reliable and scalable enterprise applications using EJBs. Using COOL:Joe's built-in smart features, developers can generate EJBs automatically from component specifications, create, test and debug business logic, and deploy to an application server. Developing distributed systems is easy because COOL:Joe automatically generates the EJB infrastructure code and frees the developers to concentrate on business logic. COOL:Joe makes developing server-side Java Objects for the Enterprise faster than ever before.

### TRENDS DRIVING E-BUSINESS

- The Internet is having a major impact on the way business is conducted around the world. New businesses are being created and existing businesses are being transformed. E-commerce is driving lower cost of distribution, tighter inventory control, increased productivity and improved customer service. Organizations lack the knowledge, technology and skills to take advantage of emerging web-based technologies.
- Organizations that succeed in taking advantage of web-based technologies will be rewarded with efficient services meeting the needs of a growing customer base, the flexibility to move into new markets, and inflated stock valuations based on their future potential to dominate their chosen domains.
- Organizations that do not embrace web-based technologies will be sidelined as niche players, or at worst, fail to survive at all.
- With the rapid pace of change of web-based technologies, organizations find it difficult to define a long-term strategy for the backbone of their future distributed systems. Software architects must select from a myriad of technical choices to adopt a set of core web-technologies that will meet the organization's immediate needs without resulting in wasted efforts in the future.

### WHAT IS COOL:Joe?

COOL:Joe is used by Java development teams to architect, create and deploy scalable distributed enterprise-scale applications using Enterprise JavaBeans (EJB). There are many alternative methods of developing distributed systems using EJBs – COOL:Joe is different because it is a Java development environment that is seamlessly integrated with architecture design and component modeling capabilities. Building a component architecture that can be visualized provides a better understanding of how components meet business requirements and creates a structured environment in which they will exist.

COOL:Joe enables companies to deliver complex, distributed systems while taking advantage of the latest Java technology without having to recruit armies of Java specialists. COOL:Joe allows EJB developers to rapidly deliver a solution by abstracting problems by modeling the solution and while being insulated from the underlying complexities of the target application environment, leverage code generation capabilities. Using repository technologies, COOL:Joe supports team based development of EJB components to deliver flexible, robust solutions that are portable across platforms.

---

## COOL:Joe OVERVIEW

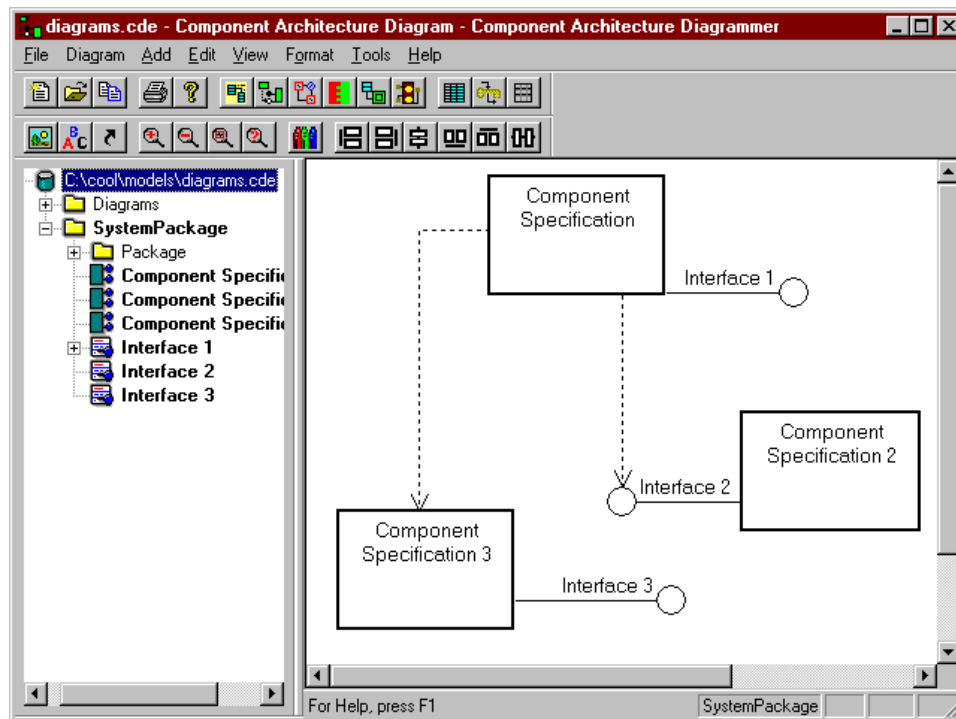
---

### INTEGRATED COMPONENT MODELING

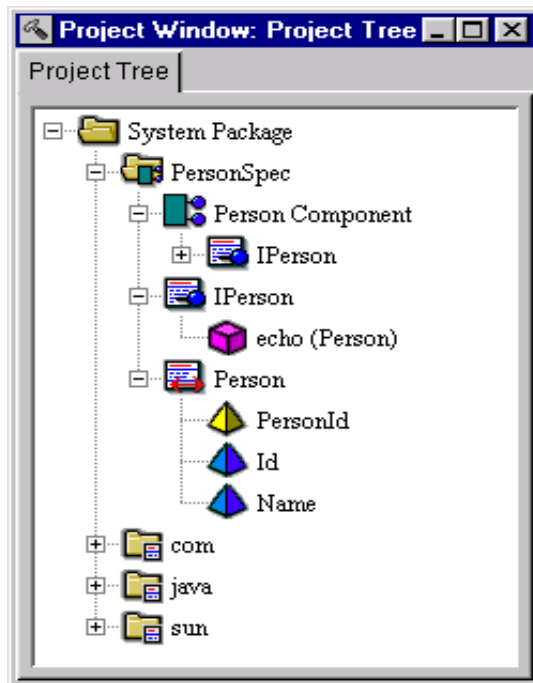
#### *COOL:Joe Provides Component Modeling and Management*

Before an EJB component can be built and deployed, it must be defined – what is often referred to as creating a component specification. A component specification defines the component's behavior and its interfaces. For any application, a defined set of inter-dependent components can be modeled in COOL:Joe, creating a component architecture for that application.

**Step 1** Create a Component Architecture. In COOL:Joe, a component architecture diagram is used to model the dependencies between components. A component architecture diagram shows how various components are related and provides an overview of the structure of the software. The diagram consists of component specifications, interfaces, and the usage relationships, or dependencies, between these objects. In the diagram below, component specifications are represented as rectangles, while interfaces are represented as lollipops (or nodes) extending from the component specification in the diagram. Usage relationships are represented as a dashed-line arrow between two boxes or as a dashed-line arrow between a box and a lollipop in the diagram.



Individual component specifications can then be transformed automatically to component implementations as EJBs. COOL:Joe's model-driven approach to development allows you to visualize the solution, architect the solution and build a component architecture. COOL:Joe's models are based on an object-oriented repository, which allows the development team to make dynamic changes to the component architecture as it evolves.



The COOL:Joe Project Tree view shows a model with a complete component specification that contains an interface type, a specification type, and a component specification object. The Person component specification object makes the IPerson interface type available as a means of accessing the echo (Person) operation defined in the IPerson interface type. The echo (Person) operation returns a Person specification type with the appropriate data in the attributes; PersonId, Id or Name. The component specifications and types can be modeled with a Type Diagram.

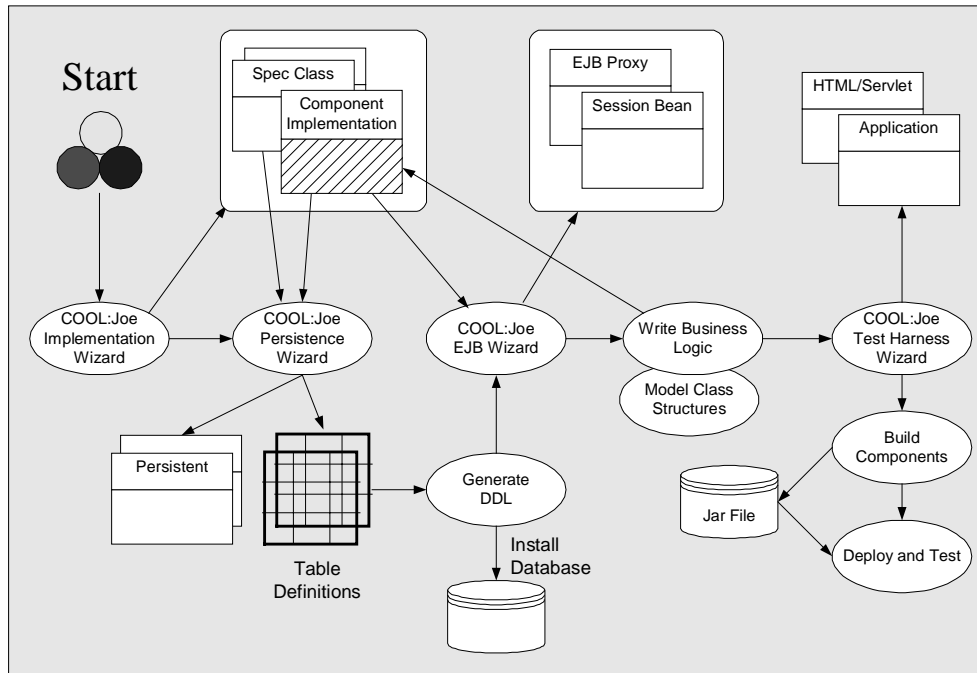
## ADVANCED WIZARDS

### *COOL:Joe Wizards Automate the Workflow for EJB Development*

COOL:Joe uses advanced wizards to automate the development effort from transforming a component specification to creating the Java classes and EJB framework, from creating and editing business logic to testing, debugging, building and deploying Enterprise JavaBeans. In fact, most of the effort in creating an EJB component is accomplished using wizards. The workflow steps are:

1. Model business processes and relationships to build a component architecture, which includes the individual component specifications, needed to meet new systems requirements.
2. Use the Specification to Implementation Wizard to transform a component specification into Java classes.
3. Generate a set of persistence support classes to access a relational database with the Persistence Generation Wizard. Use the DDL Generation Wizard to generate the database definition language. Or reverse engineer an existing database.
4. Use the Editor to specify business logic and Class Diagram to model class structures.
5. The EJB Generation Wizard generates the code to implement the component as an Enterprise JavaBean.
6. Compile component classes and create an executable JAR file with the Build Wizard.
7. The Deployment Wizard deploys the EJB JAR file to the Enterprise JavaBean application server.
8. Use the Test Harness Wizard to generate all the code necessary to test the component with a Java desktop application and/or HTML servlets.
9. Test the EJB.
10. EJB implemented successfully.

The following describes the COOL:Joe workflow step by step and illustrates why COOL:Joe's advanced wizards' make generating EJBs faster and easier than ever before.



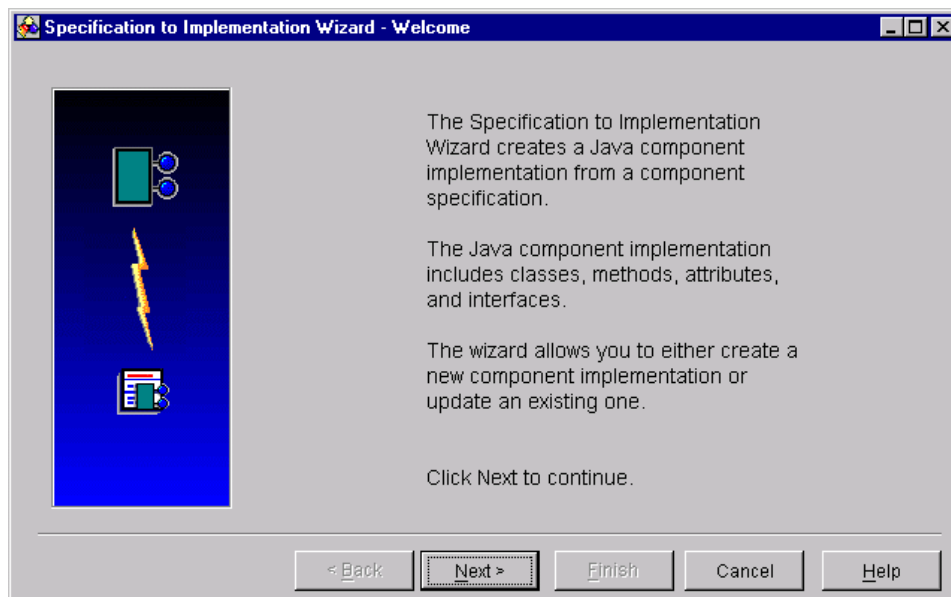
### *Specification to Implementation*

**Step 2** Implement a Component specification.

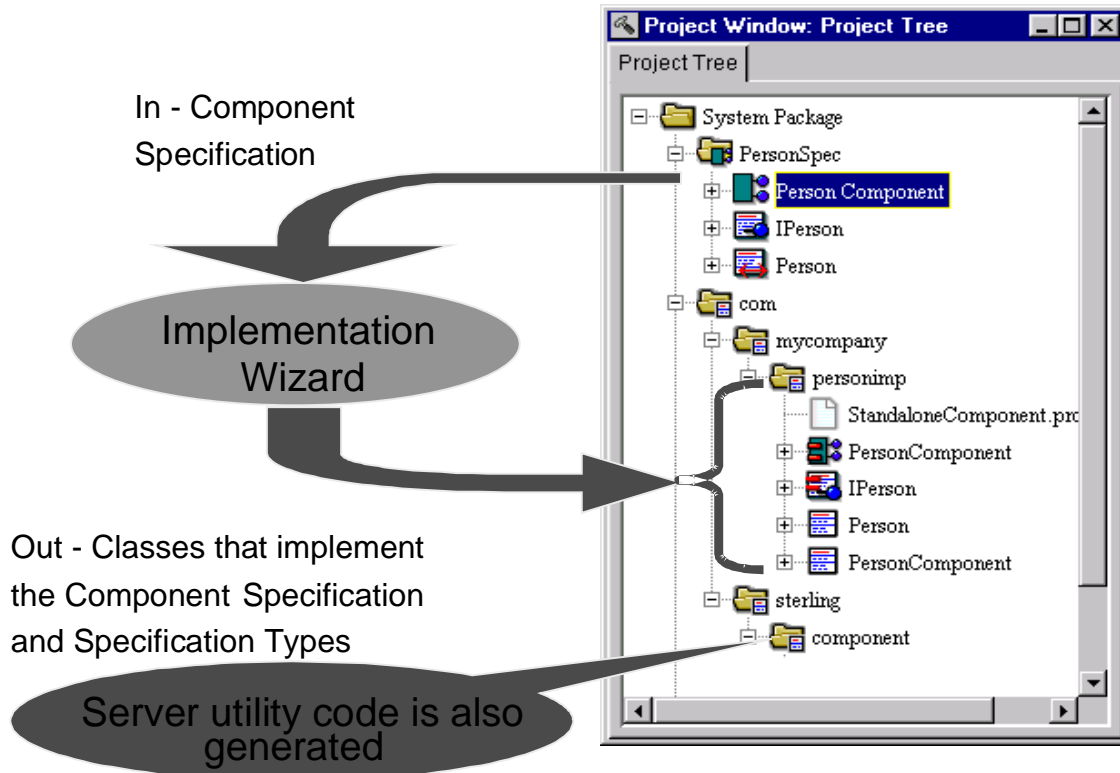
The Specification to Implementation Wizard creates a Java component implementation from a component specification.



The Specification to Implementation Wizard transforms component specification objects to component implementation objects. This wizard generates the initial component implementation class and interface used to write business logic. Next, create or select the package where Java classes and interfaces will reside. The component specification is transformed.







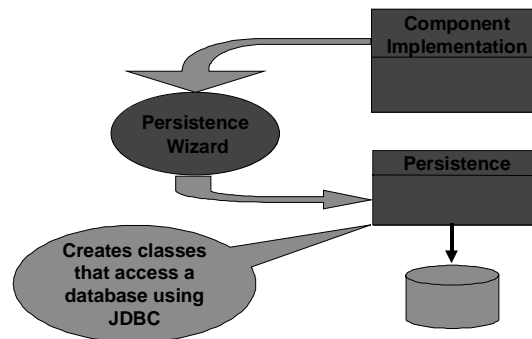
### Defining Database Structure

COOL:Joe addresses persistence implementation, maintenance, and mapping problems found in traditional implementations by providing an object layer between the business logic and the database. The object layer insulates the developer from having to know the details of how to access a database, and consists of generated persistent support classes and component framework classes. The persistent support classes allow the developer to use class specific methods to locate and maintain instances of persistent objects. The component framework classes interact with a Java Database Connectivity (JDBC) driver to access a relational database.

The object layer also ensures that the business logic is separate from the persistence logic, making the task of changing the underlying database easier.

### Step 3 Generate Persistent Classes

The Persistence Wizard generates the Java code necessary to store one or more Java classes in a relational database. These classes called persistent support classes enable the Enterprise JavaBeans to communicate with the database. The wizard creates database table objects based on the classes specified to be persistent. A database table object is an abstract representation of a database table in a COOL:Joe model.



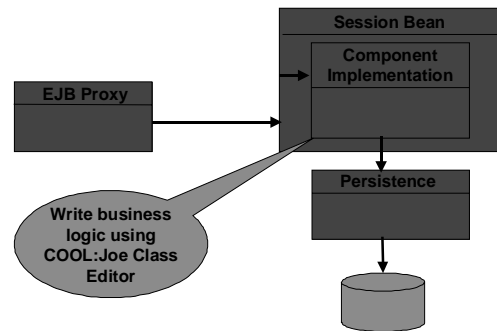
Then, if needed, the DDL Generation Wizard can use the generated database table objects to generate a DDL file. The DDL file contains SQL commands to create the physical database tables, including the commands to create the primary key. Then, a database may be created and populated.

## CREATING BUSINESS LOGIC

### *COOL:Joe Provides Developer-Friendly Java Editor*

#### Step 4 Write the Business Logic

The COOL:Joe Editor will be used to create the Java code that will implement the methods of the component. The Editor lets the developer quickly create and edit business logic. The Editor includes a syntax parser, which makes coding much faster.



By selecting the menu list (with a right mouse click) on the PersonComponent component implementation icon in the Project Tree View, the Editor can be invoked. COOL:Joe's Editor is illustrated below.

```

    package com.mycompany.personimp;
    import java.io.Serializable;
    import com.sterling.component.IComponent;
    import com.sterling.component.IComponentContext;
    public class PersonComponent implements IPerson, Cloneable, Serializable, IComponent
    {
        private transient IComponentContext componentContext = null;

        public String change (Person inPerson)
        {
            return "";
        }

        public int add (String nameToAdd)
        {
            return 0;
        }
        public Person get (int inPersonId)
        {
            return null;
        }
    }
  
```

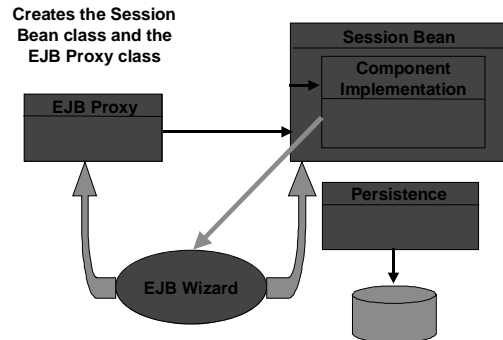


## MORE ADVANCED WIZARDS

### *EJB Generation*

#### Step 5 Create the Enterprise JavaBean

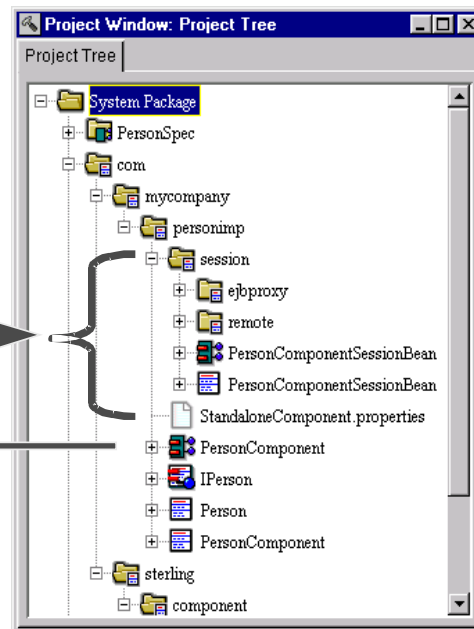
The illustration shows how the Enterprise Java Beans wizard is used to create the classes that implement the Session Bean and the EJB proxy. EJB components are either derived from predefined component specifications or imported Java classes that are converted into an EJB implementation objects.



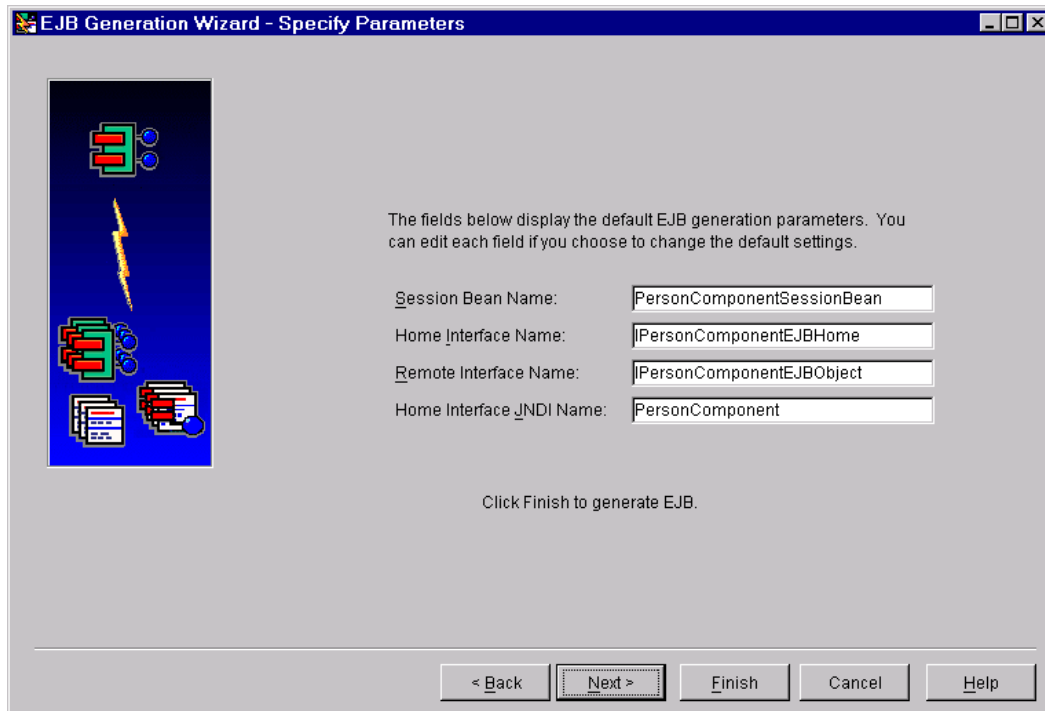
Out - 'session' package that contains classes implementing the Session Bean and EJB Proxy

EJB Wizard

In - Component Implementation



The Specify Parameters panel for the EJB Generation Wizard is shown below. The fields in the panel display the default EJB generation parameters for creating the Session Bean, the Home and Remote Interfaces. Upon executing the wizard, the entire EJB framework is generated as well as all the code necessary to implement a component as an Enterprise Java Bean.



### ***Building Components***

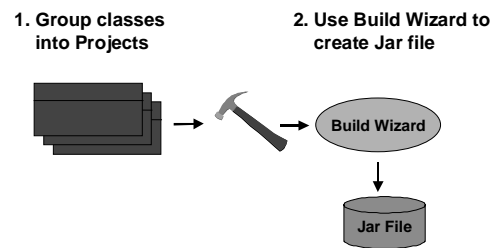
In this step, two things will be accomplished:

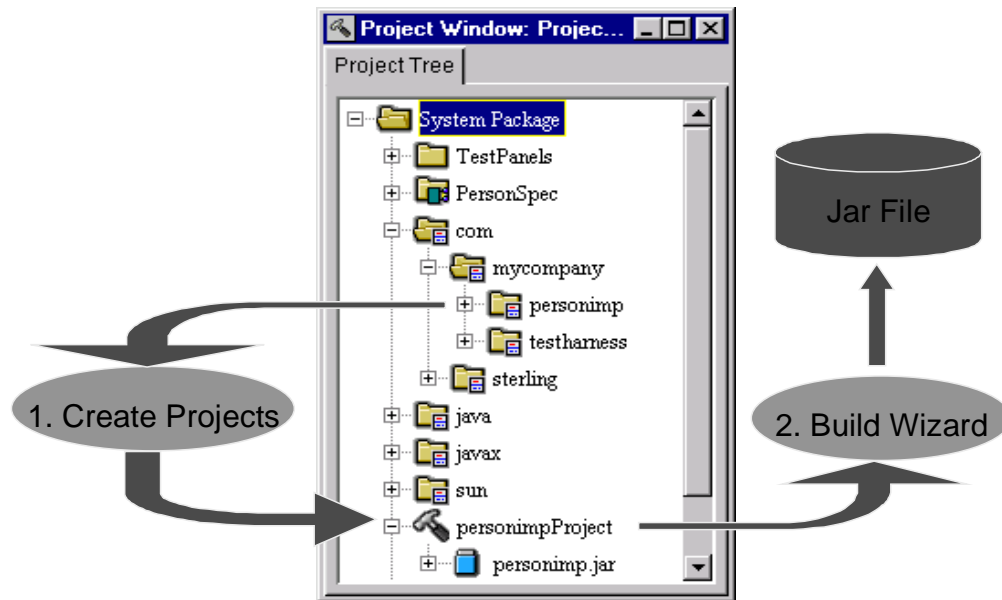
1. One or more projects will be created. A Project is a collection of files, Java files, property files, etc that will be compiled and included in a Jar file. A Jar file can be thought of as a load module and will represent a part of the system being built.

2. Execute the COOL:Joe Build Wizard. The Build Wizard takes, as input, a Project and produces a Jar file as output. The Build Wizard compiles all Java files in the Project and adds them along with any other files in the Project to the resulting Jar file.

### **Step 6 Building Components**

The Create Projects activity groups classes and uses the Build Wizard to create JAR files. JAR files contain the component framework classes for runtime and the component classes.



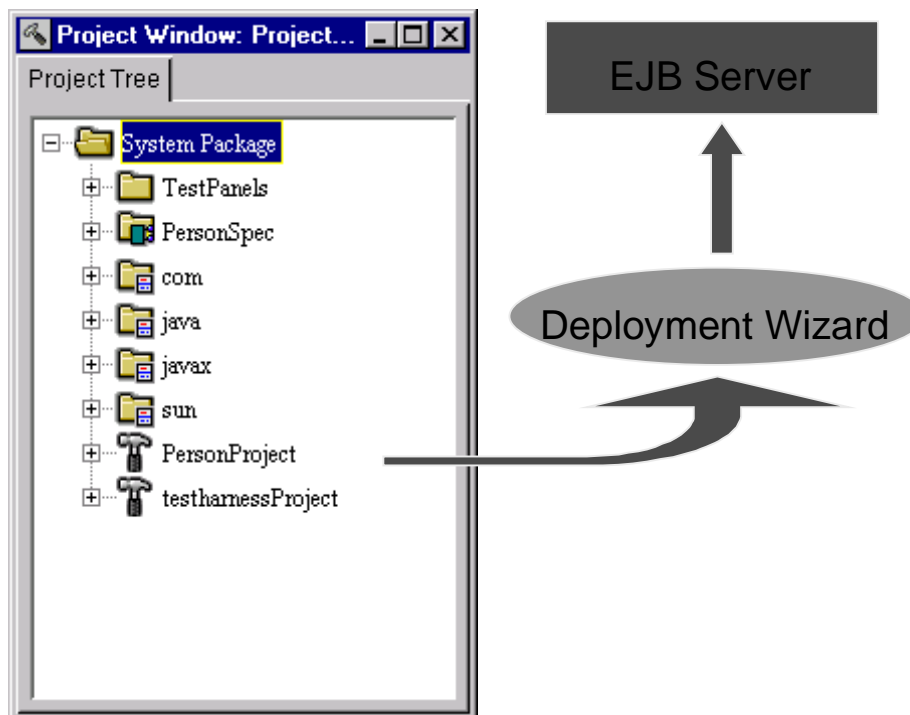


The Tree View above shows the project file using the hammer as the representative icon.

### *Deploying and Testing the EJB*

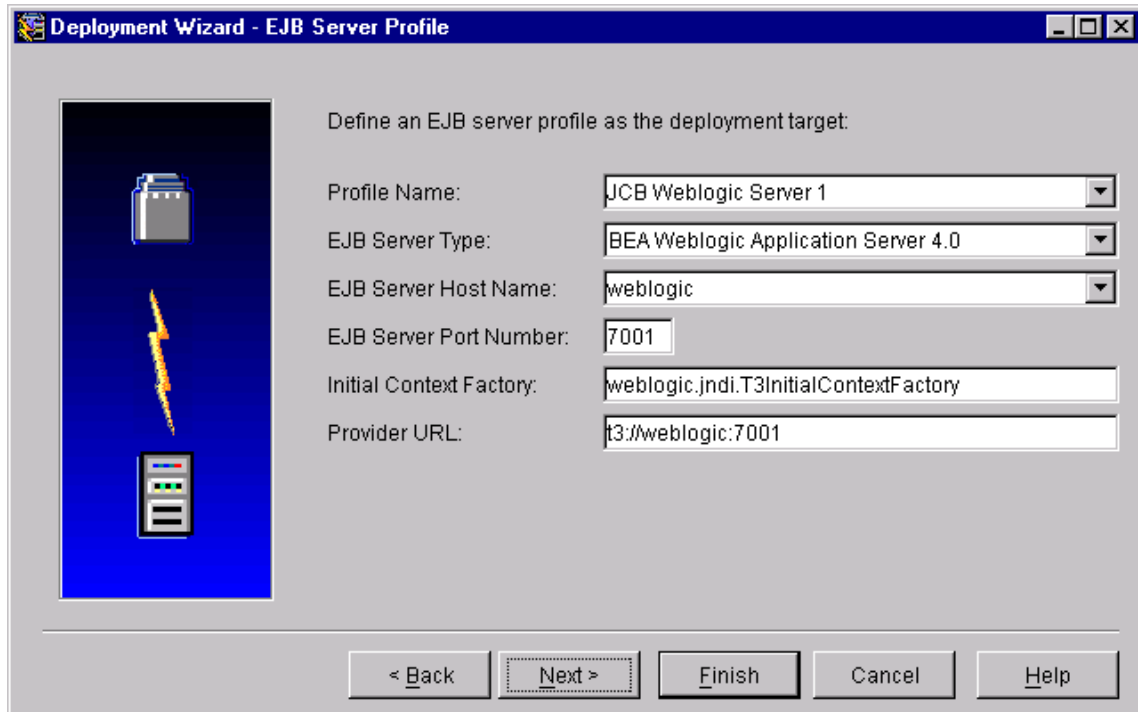
Now, there remains the final steps, which are to deploy and test the EJB server component. COOL:Joe provides a Deployment Wizard to assist the developer in preparing the EJB for testing and production. Once the EJB is generated and built, the Deployment Wizard is run with a correctly configured EJB application server environment. Doing this creates the EJB server container and enables communications with the EJB server machine and database.

#### **Step 7 Deploy the EJB**



COOL:Joe session beans can run in any EJB 1.0 compliant server/container. The COOL:Joe Deployment Wizard automatically deploys to leading EJB application servers. In this example, we will deploy to BEA's WebLogic server environment.

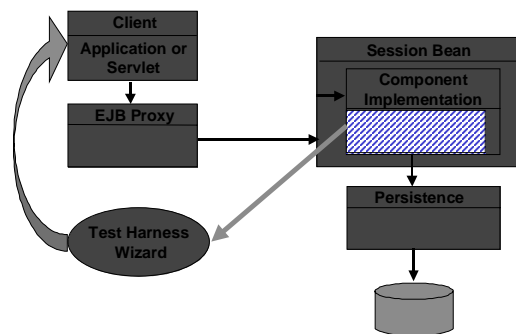
The EJB Server Profile panel in the Deployment Wizard is shown below. When Next is chosen in this panel, the Deployment Wizard contacts the WebLogic server, either locally or remotely across the network and automatically deploys the new session bean.



### *Testing the EJB*

#### **Step 8 Generate the Test Harness**

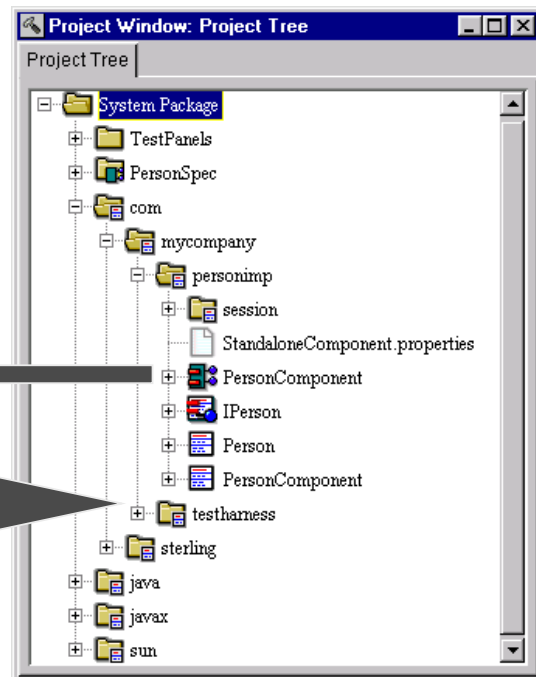
The Test Harness Wizard creates a driver for testing your component. The wizard generates the Java and/or HTML necessary to test individual methods implemented by the component.



In - Component Implementation

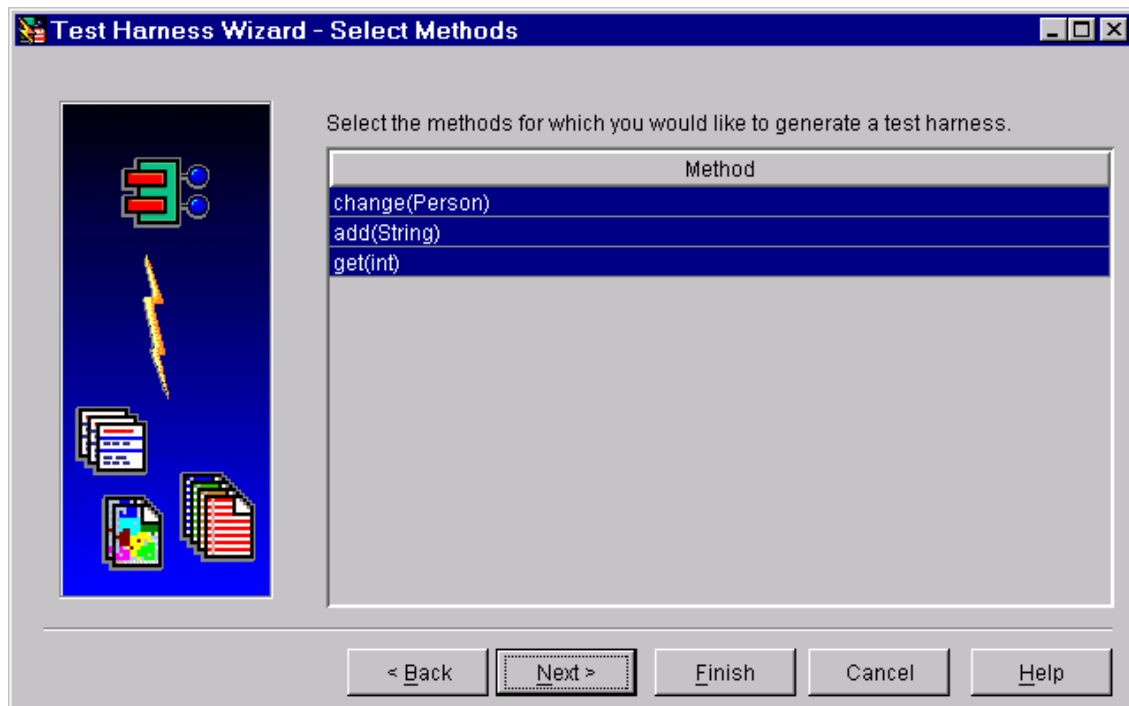


Out - Package containing test application

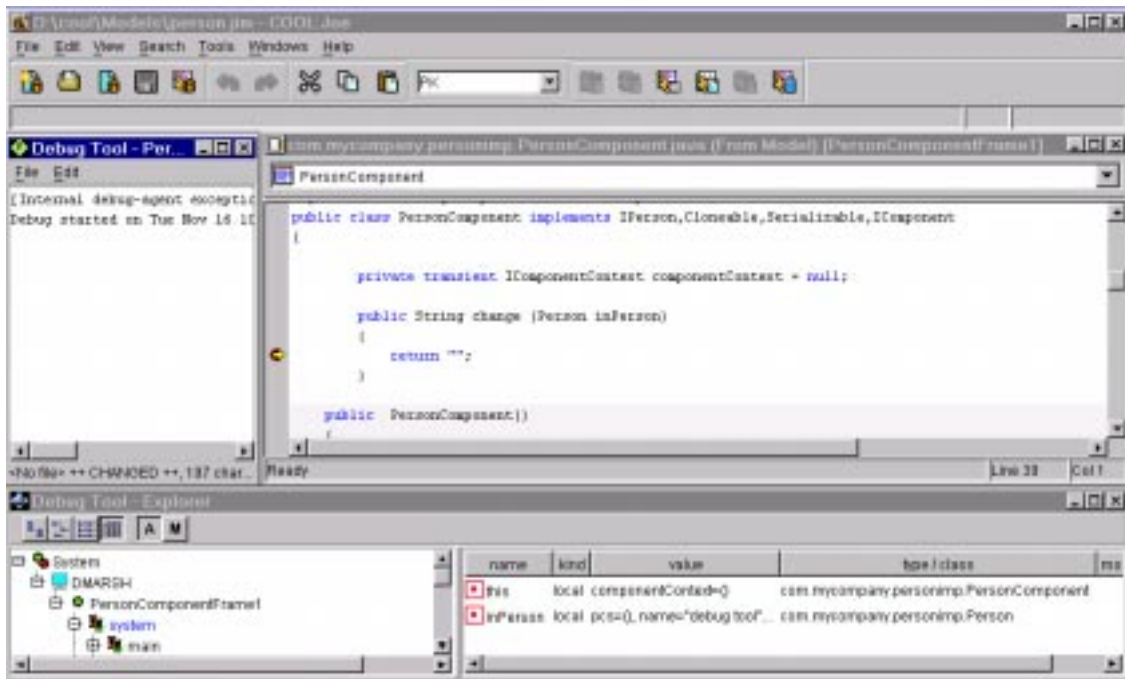


The illustration above shows the component implementation as input and the package containing the test application as the output of the wizard.

For the Test Harness; either generate a standalone Java program for each method or generate the HTML servlet for each method or do both. The Test Harness Wizard creates the classes necessary to build a simple user interface and provide connections to the component JAR file. The JAR file is used to test components and is easily exported to 3<sup>rd</sup> party web authoring environments for production UI development.







The Test Harness Wizard Select Methods panel automatically displays the component's methods for which you can generate a test harness. You may choose one or more individually or all of them. COOL:Joe also provides an Interactive Debugger, integrated with the Editor, shown above to help developers quickly identify problems in the code and make corrections, delivering quality solutions much faster.

The end result is complete EJB generation and deployment from component specifications --- all with COOL:Joe -- a visual, integrated development environment that enables developers to use the latest Java standards and technology in a fast and reliable way.

---

## COOL:Joe's FEATURES AND BENEFITS

---

Feature	Benefit
Model-driven development so that all the work done in the tool is captured at the model level.	Allows for flexibility and consistency.
Feature	Benefit
Automated code generation from the model as much of the infrastructure code as possible.	Provides ability to generate EJBs from component specifications, which frees developers to concentrate on business logic.
Feature	Benefit
Enterprise-scale development tool that scales to meet enterprise level application needs.	Provides confidence in building robust “run the business” type applications.
Feature	Benefit
Based on established Component-based-development concepts and techniques.	Promotes reuse of components and leverages existing legacy assets, IT skills and knowledge.
Feature	Benefit
Integrated visual component modeling and management tools to architect and design distributed systems and specify component interfaces.	Lays foundation for understanding and building complex systems that improves implementation success.
Feature	Benefit
Use and share Java expertise built-in to the tool via Smart Options.	Allows the use of the latest Java technology and standards without extensive staff of Java experts.
Feature	Benefit
Advanced wizards automate majority of the development effort.	This allows the development team to work more efficiently and deliver applications faster.
Feature	Benefit
Repository-based, integrated development environment.	Allows for teams to work simultaneously which speeds up application delivery.
Feature	Benefit
Class diagram that utilizes automatic layout placement to model component types and interfaces and class structures.	Provides visualization of complex relationships in an easy to understand format.

---

## THE COMPETITIVE LANDSCAPE

---

### COOL:Joe COMPETITORS

Competitors in the EJB development tools market are just emerging. Currently the competitive landscape ranges from 1<sup>st</sup> generation Java Integrated Development Environment tools (IDEs) to more capable products, which begin to address modeling and server side component development. Based on our analysis of the current market we expect competition from the following vendors:

IBM/Visual Age for Java/Websphere  
Forte for Java (part of Sun Microsystems)  
Oracle/JDeveloper  
Bluestone/Sapphire Web  
Sybase/ PowerJ /Enterprise Application Studio  
BEA /Visual Café  
Inprise/JBuilder

The key advantage to COOL:Joe over competitive tools is the ability to insulate developers from the underlying Java infrastructure and technology requirements. The above mentioned IDEs are focused on providing tools that manage Java source code, binary code, components, version control and configuration management. They are targeted at the experienced Java developer. Most companies have limited access to experienced Java developers and must use the assets they have – both in people and technology. So, the ability to insulate the developers from underlying Java technology and having to understand how to build the infrastructure around EJBs makes COOL:Joe an attractive tool in today's market. COOL:Joe automates the development effort of EJBs via advanced wizards and smart options, which allow the developers to focus on creating business logic.

Another key advantage to COOL:Joe is the integration with component modeling and management to provide the first integrated component architecture design and modeling environment. No other tool automatically generates EJBs from component specifications and provides round-trip engineering at the specification level. And the visualization capabilities to architect the solutions are built in to COOL:Joe. The architecture, the model, the resulting components and their implementations as EJBs are all contained within a single tool. COOL:Joe provides a single development environment; from modeling to testing to debugging and generating code to implementation, providing a complete life cycle for EJB development.

COOL:Joe greatly simplifies creating EJB applications by combining the most recent Java technology advances with proven best practices for component-based development and deployment. An in-depth repository of methods, processes and techniques advice and guidance are documented in the product. COOL:Joe enables organizations to take advantage of quickly emerging e-business opportunities while leveraging legacy systems and preserving ongoing flexibility.

---

## ATTACHMENT A

---

### WHAT IS ENTERPRISE JAVABEANS?<sup>1</sup>

When Java™ was first introduced, most of the IT industry focused on its graphical user interface characteristics and the competitive advantage it offered in terms of distribution and platform independence. Today, the focus has broadened considerably: Java has been recognized as an excellent platform for creating enterprise solutions, specifically for developing distributed server-side applications. This shift has much to do with Java's emerging role as a universal language for producing implementation-independent abstractions for common enterprise technologies.

Enterprise JavaBeans™ is the latest technology abstraction in the Java family, and perhaps the most ambitious. Enterprise JavaBeans (EJB) provides an abstraction for component transaction monitors (CTMs). Component transaction monitors represent the convergence of two technologies: traditional transaction processing monitors, such as CICS, TUXEDO, and Encina, and distributed object services, such as CORBA (Common Object Request Broker Architecture), DCOM, and native Java RMI. Combining the best of both technologies, component transaction monitors provide a robust, component-based environment that simplifies distributed development while automatically managing the most complex aspects of enterprise computing, such as object brokering, transaction management, security, persistence, and concurrency.

Enterprise JavaBeans defines a server-side component model that allows business objects to be developed and moved from one brand of CTM to another. A component (a bean) presents a simple programming model that allows the developer to focus on its business purpose. An EJB server (a CTM that conforms to the Enterprise JavaBeans specification) is responsible for making the component a distributed object and for managing services such as transactions, persistence, concurrency, and security. In addition to defining the bean's business logic, the developer defines the bean's runtime attributes in a way that is similar to choosing the deploy properties of visual widgets. The transactional, persistence and security behaviors of a component can be defined by choosing from a list of properties. The end result is that Enterprise JavaBeans make developing distributed component systems that are managed in a robust transactional environment much easier. For developers and corporate IT shops that have struggled with the complexities of delivering mission-critical, high-performance distributed systems using CORBA, DCOM or Java RMI, Enterprise JavaBeans provides a far simpler and more productive platform on which to base development efforts.

Enterprise JavaBeans has quickly become a de facto industry standard. Enterprise JavaBeans provides a standard distributed component model that greatly simplifies the development process and that allows beans that are developed and deployed on one vendor's EJB server to be easily deployed on a different vendor's EJB server.

---

<sup>1</sup> Excerpt from the Preface of Enterprise JavaBeans by Richard Monson-Haefel Copyright © 1999 O'Reilly & Associates, Inc. All rights reserved.

---

## ATTACHMENT B

---

### EJB PROGRAMMING IN 15 MINUTES

Using COOL:Joe™ for a fast-path to EJB

[www.sterling.com/cooljoe](http://www.sterling.com/cooljoe)

#### INTRODUCTION

Thank you for downloading the evaluation edition of Sterling Software's COOL:Joe. The first commercial release of COOL:Joe is now available.

COOL:Joe is an enterprise-scale, EJB development environment that allows Java development teams to architect, create, deploy and manage reliable and scalable enterprise applications using EJBs. Using COOL:Joe's built-in Smart features, developers can generate EJBs automatically from component specifications, create, test and debug business logic, and deploy to an application server. Developing distributed systems is effortless because COOL:Joe automatically generates the EJB infrastructure code freeing the developers to concentrate on writing business logic.














To introduce you to COOL:Joe's capabilities, we've outlined 5 steps for you to follow. In only a few minutes, you will create and implement a component specification, add business logic, generate a test harness, and build and test the component in a test application. By using COOL:Joe's built-in wizards, creating EJB components is fast and easy.








---

#### Quick Evaluation Script

Now it is time to create and implement your first COOL:Joe component! To start COOL:Joe, go to the program group where you installed the evaluation copy. Under the COOL:Joe Evaluation item, select Component Implementation.






##### 1. Create a component specification.





- Create a new COOL:Joe model called: `evaljoe.jim`. Click on the  Create a new model button, type `evaljoe` in the File name field and click the Save button. This creates and opens the new model.
- Right-click on  System Package and select  Create, then  Package. Name the new package `myFirstComponent` and press the Enter key followed by the Escape key to exit the multiple add mode.
- Right-click on  `myFirstComponent` and select  Create, then  Component Specification. Name the component specification `myComponent` and press the Enter key followed by the Escape key to exit the multiple add mode.
- Right-click on  `myFirstComponent` and select  Create, then  Interface Type. Name the interface type `myInterface` and press the Enter key followed by the Escape key to exit the multiple add mode.
- Right-click on  `myInterface` and select  Create, then  Operation. Name the operation `myOperation` and press the Enter key followed by the Escape key to exit the multiple add mode.

- Right-click on  myOperation() and select  Properties. On the Operation Properties dialog, under Result Type pick text and click OK to exit.
- Right-click on  myComponent and select  Properties. On the Component Specification Properties dialog, click the  Insert Row button beside the Offers interface types list. On the Interface Type Selection dialog, expand , select , and click the OK button 2 times to exit.






Congratulations, step 1 is now complete. You have created a component specification (myComponent) that offers one interface (myInterface) with one operation (myOperation). The result of the operation is to return a text string that you will implement in step 3. You are now ready to implement myComponent. Please continue with step 2.

## 2. Implement the component specification wizard.

- Right-click on  myComponent, select  Implement Specification..., and click the Next> button to display the Select Package dialog.
- Select the  System Package, click  New Package, select newPackage, type *myFirstApp*, and press the Enter key to create  myFirstApp. Click the Finish button to run the wizard. Click the Close button when the wizard completes.




Congratulations, step 2 is now complete. You have run the Specification to Implementation wizard to create a component implementation (, MyComponent), Java interface class (, MyInterface), and Java implementation class (, MyComponent). These objects can be found in the  myFirstApp Java package. You are now ready to add your business logic. Please continue with step 3.

## 3. Add business logic.

- Expand , right-click on  MyComponent, and select  Edit... .
- When the  Editor opens with the myFirstApp.MyComponent class displayed, use the drop-down list at the top to locate the  myOperation() method. Select this method and Editor will position you to the beginning of the method.
- Type the words *Hello World* between the quotes on the return “ ” statement. Example: return “Hello World”;
- Close the Class Editor and click the Yes button when the Save changes to MyComponent? dialog is displayed. (Make sure the Save to model radio button is selected.)

Congratulations, step 3 is now complete. You have edited the Java class (MyComponent) to support the component implementation (MyComponent) and added the necessary business logic to return a Java string containing “Hello World”. You are now ready to generate a test harness for your component. The test harness will provide you with a user interface to send data to, and receive data from, your component. Please continue with step 4.

#### 4. Generate a Test Harness.

- Right-click on  MyComponent, select  Generate Test Harness..., and click the Next> button 4 times (taking all of the defaults) to display the Select Package For Java Classes dialog.
- Select  myFirstApp and click the Finish button to run the wizard. Click the Close button when the wizard completes.

Congratulations, step 4 is now complete. You have run the Test Harness wizard to create several HTML and Java files to support the testing of your component using a web browser or Java application. You are now ready to build and test your component running in a test application. Please continue with step 5.



#### 5. Build and test the component in a test application.

- Right-click on  myFirstApp, select  Default Project..., and click the OK button to accept the default project  myFirstAppProject.
- Right-click on  myFirstAppProject, select  Build..., and click the Finish button to accept all defaults and run the wizard. Click the Close button when the wizard completes.
- Expand  myFirstAppProject,  myFirstApp.jar, and  myFirstApp.
- Right-click on  MyComponentFrame.html (to test in web browser) or  MyComponentFrame (to test as a Java application), and select  Test.
- After executing the MyOperation method the returned string result should be “Hello World”.

Congratulations, step 5 is now complete. You have completed the COOL:Joe evaluation script and successfully tested your component in either a Java application or web browser.

### Summary

Developing and delivering robust e-business solutions has never been easier. Your next step may be to deploy your tested component as an EJB into an EJB application server. By using the EJB Generation and Auto Deployment wizards you would repackage your component as an EJB session bean and be ready to retest using your test harness in a matter of minutes. This task is accomplished without writing any additional code.

To illustrate COOL:Joe’s visualization capabilities, we encourage you to look at the Type Diagram and the Class Diagram. Component specifications, which include the interfaces, types and operations, can be modeled with a Type diagram. To view the diagram, right-click on  myComponent and select Type Diagram. After the component has been implemented via the wizards in COOL:Joe, the resulting class structures can be visualized in a Class Diagram. To view the diagram, right-click on  MyComponent and select Class Diagram. Both diagram styles utilize automatic layout placement, which provides visualization of complex relationships in a format that is always easy to understand.

For a more in-depth look at additional COOL:Joe features please refer to the tutorial provided as a part of this evaluation. The tutorial file, EJoeGST.pdf, is located in the product's ...\\exe directory.

### Features of the evaluation edition of COOL:Joe

<b>Type Diagram</b>	Visualize the Specification Type interaction.
<b>Class Diagram</b>	Visualize the Java classes and their interaction.
<b>Smart Options</b>	Provide ways to get the most out of your developer resources. <ul style="list-style-type: none"><li>• Smart Expansions</li><li>• Smart Methods</li><li>• Smart Macros</li><li>• Smart Syntax</li></ul>
<b>Editor</b>	Quickly create and edit business logic in Java, HTML, JSP, etc.
<b>Advanced Wizards</b>	Automate the majority of the development effort so developers can generate server side EJB components. <ul style="list-style-type: none"><li>• Specification to Implementation Wizard</li><li>• EJB Generation Wizard</li><li>• Persistence Generation Wizard</li><li>• Test Harness Wizard</li><li>• Build Wizard</li><li>• Deployment Wizard</li></ul>
<b>Interactive Debugger</b>	Set breakpoints and step through your Java business logic.

For more information on COOL:Joe please email us at [cool.joe@sterling.com](mailto:cool.joe@sterling.com)

©1999 Sterling Software, Inc. All rights reserved. COOL:Joe is a trademark of Sterling Software. All other trademarks belong to their respective owners.

