# J2EE Programming using Eclipse (Servlets & JSPS)

This course makes extensive use of hands-on examples and exercises to familiarize the student with servlet and JSP programming. While the emphasis of this course is on the programming of servlets and JSPs, participants will also gain expertise in working with many of the features of Eclipse. Participants will develop systems in which web pages, servlets, and JSPs work together consistent with the Model 2 Architecture.

**Audience**
- Java programmers who want to increase their knowledge of the Java language
- Java programmers who will be building enterprise applications using J2EE

**Prerequisites**
- Prior programming experience or training in Java Programming using Eclipse is required
- Prior programming experience or training in HTML and/or JavaScript is recommended but not required

**Course Length**
- Five days

**Teaching Methods**
- Lecture
- Hands-on examples
- Supplemental hands-on exercises

**Learning Objectives**
- understand the servlet life cycle
- develop and test servlets using Eclipse
- know how and when to use the init(), doGet(), doPost(), and destroy() methods
- create HTML/XHTML web pages, especially forms and input controls
- retrieve information from HTML/XHTML forms
- use cookies and session tracking to persist objects
- use JDBC to access relational databases from a servlet
- write servlets for record input, query, and deletion
- understand the JSP life cycle
- develop and test JSPs using ECLIPSE
- create JSPs which use data retrieved from HTML/XHTML forms
- create JSPs which use JavaBeans passed from other servlets
- create JSPs which use JDBC to access relational databases
- write JSPs which work alone and in conjunction with other servlets
- develop systems in which web pages, servlets, and JSPs work together consistent with the Model 2 Architecture

## Course Outline

*QF4*

**Life Cycle of a Servlet**
- Creating a Web Application
- init()
- doGet()
- doPost()
- destroy()

**Parameters**
- Creating servlet parameters
- Accessing servlet parameters
- Creating context parameters
- Accessing context parameters

**Creating a web page**
- Creating JSP init parameters
- HTML/XHTML with ECLIPSE
- Design, Source, and Preview views
- Easy web page creation with Design view
- Form
- Label
- Text field
- List box
- Eclipseio button
- Checkbox
- Submit button
- method = "Get" vs. method = "Post"

**Accessing form parameters**
- getParameter()
- Detecting null input
- Data validation techniques
- Exception handling

**Java Beans**
- Defining entity beans
- Instantiating entity beans with form data

**Object Persistence**
- Data Access Object (DAO) design pattern
- Using cookies
- Using session tracking
- Using JDBC

**Introduction to JSPs**
- What is a JSP?
- JSPs: When and why?
- Components of a JSP

**Our First JSP**
- Creating a JSP
- Including an expression
- Running a JSP on the server

**Scripting**
- Scriplets
- Comments, expressions, and declarations

**Life Cycle of a JSP**
- jspInit()
- jspDestroy()
- <body></body>

**Init Parameters**
- Creating JSP init parameters
- Accessing JSP init parameters
- Creating context init parameters
- Accessing context init parameters

**Simple JSPs**
- More scripting
- Text fields and request getParameter()
- Declaring variables
- Defining methods
- Hidden fields
- Radio buttons
- Submit button
- method = "Get" vs. method = "Post"

**Implicit Objects**
- Implicit objects available to the JSP programmer
- Object scope: application, page, request, session

**Database Query from a JSP**
- Getting the key fields from a FORM
- Accessing the database with JDBC
- Formatting and displaying the results
- <%@ page import %>
- Error processing

**Error Pages**
- Defining exceptions
- Defining an error page JSP
- <%@ page isErrorPage="true" %>
- throwing an Exception from a JSP
- <%@ page errorPage="MyErrorPage.jsp" %>

**Model 2 Architecture: JSP/Servlet Interaction**
- What is Model 2 Architecture?
- Review of JavaBeans
- Passing a bean to the JSP: request.setAttribute
- Forwarding to the JSP: RequestDispatcher.forward
- <jsp:useBean>
- <jsp:getProperty>