IN28MINUTES

# Java Web Application with JSP and Servlets in 25 Steps

Developing your first Java Web Application using JSP and Servlets is fun.

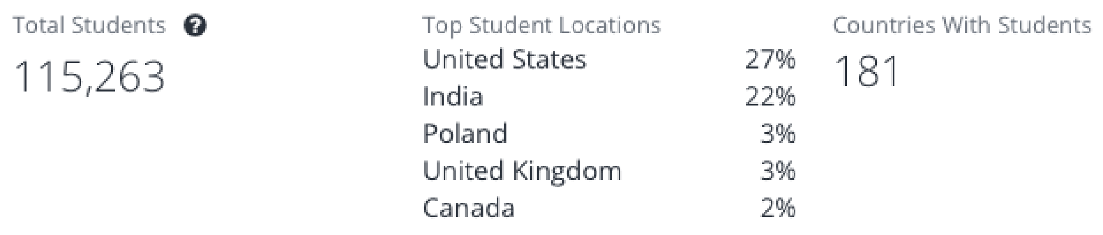
In this course, you will learn the basics developing a Basic Todo Management Application using Java Servlets and JSP with Login and Logout functionalities.

You will build a Dynamic Website using the Core technologies of Java Web Programming. You will understand the basic concepts of Java Web Application Development - HTTP protocol, Request-Response cycle, Java Servlets, JSPs.

## About in28Minutes

Read about what we love, why we create courses and our beliefs - The in28Minutes Way

- 15 Courses
- 100,000 Students



Total Students
115,263

| Top Student Locations | |
|---|---|
| United States | 27% |
| India | 22% |
| Poland | 3% |
| United Kingdom | 3% |
| Canada | 2% |

Countries With Students
181

## Github Repository

- https://github.com/in28minutes/JavaWebApplicationStepByStep

## Wonderful 5-Star Reviews

## Hands-on

> *The best part of it is the hands-on approach which the author maintained throughout the course as he had promised at the beginning of the lecture. He explains the concepts really well and also makes sure that there is not a single line of code you type without understanding what it really does. It was so engaging that it gets you interested in web development and kinda makes you want to learn more about it.*

Check Out our Latest Courses    Learn from the Best

## We don’t teach frameworks. We teach building applications!

> *Simple, yet precise explanation. The course is also enabling in understanding how to use tools like eclipse, maven, tomcat.*

> *You will learn the concepts of Servlets, JSP, Maven, TomCat, and bit of bootstrap and CSS also. A++++*

## Learn quickly

> *I can see that you will be able to learn many advanced concepts very quickly and broadly within a short time frame.*

### You will learn

- To build a Basic Todo Management Application piece by piece in 25 Steps
- Understand Basic Web Application Architecture
- Understand and use Basics of Java EE - Servlets, JSP, Scriptlets, JSTL, web.xml and EL
- Understand Servlet LifeCycle
- Use HttpRequest - GET/POST, Request Parameters
- Understand HTTP Response - Response Status - 404,200,500 etc
- Understand HTML Forms - Method, Action & Form Data
- Understand Basics of using Maven, Tomcat and Eclipse
- Understand Difference between Session and Request Scopes
- Use Maven for Basic Dependency Management
- Run Web Application in Tomcat
- Style web applications with Bootstrap (Basics)
- Use Filters to intercept Request

### Step By Step Overview

- Step 01 : Up and running with a Web Application in Tomcat
- Step 02 : First JSP
- Step 03 : Adding a Get Parameter name
- Step 04 : Adding another Get Parameter Password
- Step 05 : Let’s add a form
- Step 06 : New Form and doPost
- Step 07 : Adding Password and Validation of User Id / Password combination
- Step 08 : Adding a TodoService and Todos to welcome page
- Step 09 : Bit of Refactoring - Packages
- Step 10 : Redirect from One Servlet to another - New TodoServlet.
- Step 11 : First JSTL Tag : Using a Loop around todos
- Step 12 : Difference between Session and Request Scopes
- Step 13 : Add a New Todo
- Step 14 : Delete Todo with equals and hashcode methods
- Step 15 : Adding webjars for jquery and bootstrap
- Step 16 : Missing Step :) We want you to take a break. Nothing in here..
- Step 17 : Updating Bootstrap to 3.3.6
- Step 18 : More Refactoring
- Step 19 : Adding a Filter for More Security.
- Step 20 : Logout
- Step 21 : Theory : Understand Maven and Tomcat
- Step 22 : Theory : Servlet LifeCycle
- Step 23 : Theory : Model 1 and Model 2 MVC Architectures
- Step 24 : Moving Add Functionality to a New Page.

- Step 25 : Add Category Field
- Step 26 : Format the JSPs better.
- Step 27 : JSP Fragments

## Installation Guide

## Installing Java, Eclipse & Embedded Maven

- Installation Video
- GIT Repository For Installation
- PDF

## Troubleshooting Guide

- A 50 page troubleshooting guide with more than 200 Errors and Questions answered

## Step By Step Details

## Step 01 : Up and running with a Web Application in Tomcat

In this step, we will quickly setup a running web application and run it in tomcat.

> *Tip : This is one of the rare steps where we copy code in! We want to ensure that you have a running web application without any mistakes. We will understand every line of code in the next 2 steps.*

Create a Simple Maven Project and copy the three files in.

You can copy code from

- Step 01 on Github Repository

You can run the project using Run as > Maven build > tomcat7:run.

\pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apac
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.in28minutes</groupId>
        <artifactId>in28Minutes-first-webapp</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>
        <dependencies>
                <dependency>
                        <groupId>javax</groupId>
                        <artifactId>javaee-web-api</artifactId>
                        <version>6.0</version>
                        <scope>provided</scope>
                </dependency>
        </dependencies>
        <build>
                <pluginManagement>
                        <plugins>
                                <plugin>
                                        <groupId>org.apache.maven.plugins</grou
                                        <artifactId>maven-compiler-plugin</arti
                                        <version>3.2</version>
                                        <configuration>
```

```
                                                        <verbose>true</verbose>
                                                        <source>1.7</source>
                                                        <target>1.7</target>
                                                        <showWarnings>true</showWarning
                                                </configuration>
                                        </plugin>
                                        <plugin>
                                                <groupId>org.apache.tomcat.maven</grouq
                                                <artifactId>tomcat7-maven-plugin</artif
                                                <version>2.2</version>
                                                <configuration>
                                                        <path>/</path>
                                                        <contextReloadable>true</conte>
                                                </configuration>
                                        </plugin>
                                </plugins>
                        </pluginManagement>
                </build>
</project>
```

\src\main\java\webapp\LoginServlet.java

```
package webapp;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/*
 * Browser sends Http Request to Web Server
 *
 * Code in Web Server => Input:HttpRequest, Output: HttpResponse
 * JEE with Servlets
 *
 * Web Server responds with Http Response
 */


@WebServlet(urlPatterns = "/login.do")
public class LoginServlet extends HttpServlet {

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse re
                PrintWriter out = response.getWriter();
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Yahoo!!!!!!!!</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("My First Servlet");
                out.println("</body>");
                out.println("</html>");

        }

}
```

\src\main\webapp\WEB-INF\web.xml

```
<!-- webapp/WEB-INF/web.xml -->
<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.c
        version="3.0">

        <display-name>To do List</display-name>

        <welcome-file-list>
                <welcome-file>login.do</welcome-file>
        </welcome-file-list>

</web-app>
```

## Trouble Shooting

Go to http://www.in28minutes.com/spring-boot-maven-eclipse-troubleshooting-guide-and-faq for any issues that you face!

## Step 01 : Theory

Let's quickly look at some of the important things, we had already configured in the practical part of Step 01.

Maven Tomcat Plugin (configured in pom.xml) helps us to download Tomcat and run the web application in Tomcat. `mvn tomcat7:run` is enabled by this plugin.

```
<plugin>
        <groupId>org.apache.tomcat.maven</groupId>
        <artifactId>tomcat7-maven-plugin</artifactId>
        <version>2.2</version>
        <configuration>
                <path>/</path>
                <contextReloadable>true</contextReloadable>
        </configuration>
</plugin>
```

Notes

- `<contextReloadable>true</contextReloadable>` - Enables automatic reload when java or jsp files change without needing to restart the server
- `<path>/</path>` - Setting an empty context root. So, the application can be directly accessed at http://localhost:8080 without a context root

To create a servet, we need HttpServlet which is defined in Java EE Web API. We added a dependency in the pom.xml.

```
                <dependency>
                        <groupId>javax</groupId>
                        <artifactId>javaee-web-api</artifactId>
                        <version>6.0</version>
                        <scope>provided</scope>
                </dependency>
```

> *Servlet takes in a request and gives a response as output. HTTP is the protocol of the internet. Thats how all web applications work. For creating web applications, we extend HttpServlet which takes a HttpRequest as input and give HttpResponse as output.*

**How do web applications work?**

Once you have everything setup and running you would see the following page render at http://localhost:8080 JSP-Servlets-Step-1-Theory-1

How does it really work? Let's take a quick look at what happens in the background.

When you type in in the browser url, the browser creates a HTTP GET Request. Recommended Reading - GET vs POST JSP-Servlets-Step-1-Theory-2

The HTTP GET Request is received by the web application deployed on Tomcat server.

We have configured a welcome-file of login.do.

```
<welcome-file-list>
        <welcome-file>login.do</welcome-file>
</welcome-file-list>
```

urlPattern of LoginServlet is "/login.do". We defined a `doGet(HttpServletRequest request, HttpServletResponse response)` method in LoginServlet. The doGet method handles the GET requests to the url pattern "/login.do".

```
@WebServlet(urlPatterns = "/login.do")
public class LoginServlet extends HttpServlet {

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse re
```

In the doGet method, we are writing HTML content into the response.

```
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Yahoo!!!!!!!!</title>");
out.println("</head>");
out.println("<body>");
out.println("My First Servlet");
out.println("</body>");
out.println("</html>");
```

This response is sent out to the browser.

JSP-Servlets-Step-1-Theory-3

The browser understands the HTML and renders a beautiful page.

**Notes**

Java Platform, Enterprise Edition (Java EE) JEE6

Servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.

```
@WebServlet(urlPatterns = "/login.do")
public class LoginServlet extends HttpServlet {
        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse re
```

Servlet

- `extends javax.servlet.http.HttpServlet` - All servlets should extend HttpServlet class
- `@WebServlet(urlPatterns = "/login.do")` - Provide the url pattern to access the servlet
- `doGet(HttpServletRequest request, HttpServletResponse response)` - To handle the RequestMethod GET we need to implement doGet method.

## Step 02 : First JSP

Complete code

**What we will learn**

- How to create a JSP?
- How to redirect to a JSP?

**Code Snippets and Examples**

Creating a JSP

\src\main\webapp\WEB-INF\views\login.jsp

```
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
My First JSP!!!
</body>
</html>
```

Redirect to a view - JSP

\src\main\java\webapp\LoginServlet.java

```
request
 .getRequestDispatcher("/WEB-INF/views/login.jsp")
 .forward(request, response);
```

## Step 03 : Adding a Get Parameter name

Complete code

**What we will learn**

- Passing a Request Parameter “name”

**Code Snippets and Examples**

We read the parameter from the HTTP Request.

We would want to show the value on the JSP. We set it as a request attribute. Request attributes can be accessed from the view (jsp).

\src\main\java\webapp\LoginServlet.java

```
request.setAttribute("name",
                request.getParameter("name"));
```

\src\main\webapp\WEB-INF\views\login.jsp

```
My First JSP!!! My name is ${name}
```

## Step 04 : Adding another Get Parameter Password

Complete code

**Code Snippets and Examples**

\src\main\java\webapp\LoginServlet.java

```
request.setAttribute("password",
                  request.getParameter("password"));
```

\src\main\webapp\WEB-INF\views\login.jsp

```
My First JSP!!! My name is ${name} and password is ${password}
```

## Step 05 : Let’s add a form

Complete code

**Code Snippets and Examples**

\src\main\java\webapp\LoginServlet.java

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException {
        request
                .getRequestDispatcher("/WEB-INF/views/login.jsp")
                .forward(request, response);
}
```

\src\main\webapp\WEB-INF\views\login.jsp

```
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
        <form action="/login.do" method="POST">
                Name : <input type="text" /> <input type="submit" />
        </form>
</body>
</html>
```

\src\main\webapp\WEB-INF\views\welcome.jsp

```
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
Welcome ${name}
</body>
</html>
```

## Step 06 : New Form and doPost

Complete code

\src\main\java\webapp\LoginServlet.java

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException {
        request.setAttribute("name", request.getParameter("name"));
        request.getRequestDispatcher("/WEB-INF/views/welcome.jsp").forward(requ
}
```

\src\main\webapp\WEB-INF\views\welcome.jsp

```
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
Welcome ${name}
</body>
</html>
```

## Step 07 : Adding Password and Validation of User Id

Complete code

**Code Snippets and Examples**

\src\main\java\webapp\LoginService.java

```
public class LoginService {
        public boolean validateUser(String user, String password) {
                return user.equalsIgnoreCase("in28Minutes") && password.equals(
        }

}
```

\src\main\java\webapp\LoginServlet.java

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException {
```

```
        String name = request.getParameter("name");
        String password = request.getParameter("password");

        boolean isValidUser = service.validateUser(name, password);

        if (isValidUser) {
                request.setAttribute("name", name);
                request.getRequestDispatcher("/WEB-INF/views/welcome.jsp")
                                .forward(request, response);
        } else {
                request.setAttribute("errorMessage", "Invalid Credentials!!");
                request.getRequestDispatcher("/WEB-INF/views/login.jsp")
                                .forward(request, response);
        }
}
```

\src\main\webapp\WEB-INF\views\login.jsp

```
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
        <p><font color="red">${errorMessage}</font></p>
        <form action="/login.do" method="POST">
                Name : <input name="name" type="text" /> Password : <input name
        </form>
</body>
</html>
```

## Step 08 : Adding a TodoService and Todos to welcome page

Complete code

### Code Snippets and Examples

\src\main\java\webapp\LoginServlet.java

```
private LoginService service = new LoginService();
private TodoService todoService = new TodoService();

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException {
        String name = request.getParameter("name");
        String password = request.getParameter("password");

        boolean isValidUser = service.validateUser(name, password);

        if (isValidUser) {
                request.setAttribute("name", name);
                request.setAttribute("todos",
                                todoService.retrieveTodos());
                request.getRequestDispatcher("/WEB-INF/views/welcome.jsp").forw
        } else {
                request.setAttribute("errorMessage", "Invalid Credentials!!");
                request.getRequestDispatcher("/WEB-INF/views/login.jsp").forwar
        }
}
```

\src\main\java\webapp\todo\Todo.java

```
public class Todo {

        public Todo(String name) {
                super();
                this.name = name;
        }

        private String name;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        @Override
        public String toString() {
                return "Todo [name=" + name + "]";
        }
}
```

\src\main\java\webapp\todo\TodoService.java

```
import java.util.ArrayList;
import java.util.List;

public class TodoService {
        private static List<Todo> todos = new ArrayList();

        static {
                todos.add(new Todo("Learn Web Application"));
                todos.add(new Todo("Learn Spring"));
                todos.add(new Todo("Learn Spring MVC"));
        }

        public List<Todo> retrieveTodos() {
                return todos;
        }
}
```

\src\main\webapp\WEB-INF\views\welcome.jsp

```
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
<H1>Welcome ${name}</H2>
<div>
Your Todos are
${todos}
</div>
</body>
</html>
```

## Step 09 : Bit of Refactoring - Packages

Basic Refactoring

Complete code

## Step 10 : Redirect from One Servlet to another - New TodoServlet.

Complete code

**Code Snippets and Examples**

src\main\java\in28minutes\login\LoginServlet.java

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException {
        String name = request.getParameter("name");
        String password = request.getParameter("password");

        boolean isValidUser = service.validateUser(name, password);

        if (isValidUser) {
                response.sendRedirect("/todo.do");
        } else {
                request.setAttribute("errorMessage", "Invalid Credentials!!");
                request.getRequestDispatcher("/WEB-INF/views/login.jsp").forwar
        }
}
```

src\main\java\in28minutes\todo\TodoServlet.java

```
@WebServlet(urlPatterns = "/todo.do")
public class TodoServlet extends HttpServlet {

        private static final long serialVersionUID = 1L;
        private TodoService todoService = new TodoService();

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse re
                        throws IOException, ServletException {
                request.setAttribute("todos", todoService.retrieveTodos());
                request.getRequestDispatcher("/WEB-INF/views/todo.jsp").forwarc
        }
}
```

src\main\webapp\WEB-INF\views\todo.jsp

```
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
<H1>Welcome ${name}</H2>
<div>
Your Todos are
${todos}
</div>
```

```
</body>
</html>
```

## Step 11 : First JSTL Tag : Using a Loop around todos

Complete code

**Code Snippets and Examples**

pom.xml

```
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
```

src\main\webapp\WEB-INF\views\todo.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
<H1>Welcome ${name}</H2>
<div>
Your Todos are
<ol>
<c:forEach items="${todos}" var="todo">
   <li>${todo.name}</li>
</c:forEach>
</ol>
</div>
</body>
</html>
```

## Step 12 : Difference between Session and Request Scopes

- Session Scope
  - Valid for the entire session of a user.
  - Data in session can be shared between multiple requests from the same user.
- Request Scope
  - Only valid for the duration of the HTTPRequest

> *Recommendation : Keep number of objects in a session to a bare minimum.*

> *Recommendation : Clean up objects in session when they are not needed anymore.*

## Step 13 : Add a New Todo

Complete code

**Code Snippets and Examples**

src\main\java\in28minutes\todo\TodoService.java

```
public void addTodo(String todo) {
        todos.add(new Todo(todo));
}
```

src\main\java\in28minutes\todo\TodoServlet.java

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException {
        String todo = request.getParameter("todo");
        if ("".equals(todo)) {
                request.setAttribute("errorMessage", "Enter a valid todo");
        } else {
                todoService.addTodo(todo);
        }
        request.setAttribute("todos", todoService.retrieveTodos());
        request.getRequestDispatcher("/WEB-INF/views/todo.jsp").forward(request
}
```

src\main\webapp\WEB-INF\views\todo.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
<H1>Welcome ${name}</H2>
<div>
Your Todos are
<ol>
<c:forEach items="${todos}" var="todo">
   <li>${todo.name}</li>
</c:forEach>
</ol>

<p><font color="red">${errorMessage}</font></p>
<form method="POST" action="/todo.do">
New Todo : <input name="todo" type="text" /> <input name="add" type="submit" /
</form>

</div>
</body>
</html>
```

## Step 14 : Delete Todo with equals and hashcode methods

Complete code

**Code Snippets and Examples**

src\main\java\in28minutes\todo\DeleteTodoServlet.java

```
@WebServlet(urlPatterns = "/delete-todo.do")
public class DeleteTodoServlet extends HttpServlet {

        private static final long serialVersionUID = 1L;
        private TodoService todoService = new TodoService();

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse re
                        throws IOException, ServletException {
                todoService.deleteTodo(request.getParameter("todo"));
                response.sendRedirect("/todo.do");
        }
}
```

src\main\java\in28minutes\todo\TodoService.java

```
public void deleteTodo(String todo) {
        todos.remove(new Todo(todo));
}
```

src\main\webapp\WEB-INF\views\todo.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Yahoo!!</title>
</head>
<body>
<H1>Welcome ${name}</H2>
<div>
Your Todos are
<ol>
<c:forEach items="${todos}" var="todo">
   <li>${todo.name} <a href="/deletetodo.do?todo=${todo.name}">Delete</a></li>
</c:forEach>
</ol>

<p><font color="red">${errorMessage}</font></p>
<form method="POST" action="/todo.do">
New Todo : <input name="todo" type="text" /> <input name="add" type="submit" /:
</form>
</div>
</body>
</html>
```

## Step 15 : Adding webjars for jquery and bootstrap

Complete code

**Code Snippets and Examples**

pom.xml

```
<dependency>
        <groupId>org.webjars</groupId>
        <artifactId>bootstrap</artifactId>
        <version>3.3.6</version>
</dependency>
<dependency>
        <groupId>org.webjars</groupId>
        <artifactId>jquery</artifactId>
        <version>1.9.1</version>
</dependency>
```

src\main\webapp\WEB-INF\views\todo.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<title>Yahoo!!</title>
<!-- Bootstrap core CSS -->
<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"
        rel="stylesheet">
</head>
<body>
        <H1>Welcome ${name}</H1>
        <div>
                Your Todos are
                <ol>
                        <c:forEach items="${todos}" var="todo">
                                <li>${todo.name} <a href="/deletetodo.do?todo=$
                        </c:forEach>
                </ol>

                <p>
                        <font color="red">${errorMessage}</font>
                </p>
                <form method="POST" action="/todo.do">
                        New Todo : <input name="todo" type="text" /> <input nam
                                type="submit" />
                </form>
        </div>
        <script src="webjars/jquery/1.9.1/jquery.min.js"></script>
        <script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>
</body>
</html>
```

## Step 16 : Missing Step :)

We want you to take a break. Nothing in here..

## Step 17 : Updating Bootstrap to 3.3.6

- Let's use Bootstrap to make our todo's look better.
- We will make improvements to reduce duplication in next steps.

Complete code

### Code Snippets and Examples

Bootstrap Sample Page

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<title>Todos</title>
<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"
        rel="stylesheet">

<style>
        .footer {
                position: absolute;
                bottom: 0;
                width: 100%;
                height: 60px;
                background-color: #f5f5f5;
        }
</style>
</head>

<body>

        <nav class="navbar navbar-default">

                <a href="/" class="navbar-brand">Brand</a>

                <ul class="nav navbar-nav">
                        <li class="active"><a href="#">Home</a></li>
                        <li><a href="/todo.do">Todos</a></li>
                        <li><a href="http://www.in28minutes.com">In28Minutes</a
                </ul>

                <ul class="nav navbar-nav navbar-right">
                        <li><a href="/login.do">Login</a></li>
                </ul>

        </nav>

        <div class="container">
                <H1>Heading</H1>
                Body of the Page
        </div>

        <footer class="footer">
                <p>footer content</p>
        </footer>

        <script src="webjars/jquery/1.9.1/jquery.min.js"></script>
        <script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

</body>

</html>
```

src\main\webapp\WEB-INF\views\todo.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>
<html>

<head>
<title>Yahoo!!</title>
<!-- Bootstrap core CSS -->
<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"
        rel="stylesheet">

<style>
.footer {
        position: absolute;
        bottom: 0;
        width: 100%;
        height: 60px;
        background-color: #f5f5f5;
}

.footer .container {
width: auto;
max-width: 680px;
padding: 0 15px;
}
</style>
```

```
</head>

<body>

        <nav role="navigation" class="navbar navbar-default">

                <div class="">
                        <a href="/" class="navbar-brand">Brand</a>
                </div>

                <div class="navbar-collapse">
                        <ul class="nav navbar-nav">
                                <li class="active"><a href="#">Home</a></li>
                                <li><a href="/todo.do">Todos</a></li>
                                <li><a href="http://www.in28minutes.com">In28Mi
                        </ul>
                        <ul class="nav navbar-nav navbar-right">
                                <li><a href="/login.do">Login</a></li>
                        </ul>
                </div>

        </nav>

        <div class="container">
                <H1>Welcome ${name}</H1>

                Your Todos are
                <ol>
                        <c:forEach items="${todos}" var="todo">
                                <li>${todo.name}&nbsp;<a
                                        href="/deletetodo.do?todo=${todo.name}"
                        </c:forEach>
                </ol>

                <p>
                        <font color="red">${errorMessage}</font>
                </p>
                <form method="POST" action="/todo.do">
                        New Todo : <input name="todo" type="text" /> <input nam
                                type="submit" />
                </form>
        </div>

        <footer class="footer">
                <div class="container">
                        <p>footer content</p>
                </div>
        </footer>

        <script src="webjars/jquery/1.9.1/jquery.min.js"></script>
        <script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

</body>

</html>
```

## Step 18 : More Refactoring

> *The Boy Scouts have a rule: “Always leave the campground cleaner than you found it.”*

Let’s follow the boy scout rule by refactoring a little bit.

## Step 19 : Adding a Filter for More Security.

Lets ensure that a user cannot get to todo's page without logging in!

We do that by adding in a Filter to intercept all requests matching the pattern *.do.

**Code Snippets and Examples**

## src/main/java/com/in28minutes/filter/LoginRequiredFilter.java

```
package com.in28minutes.filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;

@WebFilter(urlPatterns = "*.do")
public class LoginRequiredFilter implements Filter {

        @Override
        public void destroy() {

        }

        @Override
        public void doFilter(ServletRequest servletRequest,
                        ServletResponse servletResponse, FilterChain chain)
                        throws IOException, ServletException {
                HttpServletRequest request = (HttpServletRequest) servletReque

                if (request.getSession().getAttribute("name") != null) {
                        chain.doFilter(servletRequest, servletResponse);
                } else {
                        request.getRequestDispatcher("/login.do").forward(serv
                                        servletResponse);
                }
        }

        @Override
        public void init(FilterConfig arg0) throws ServletException {
        }

}
```

## Step 20 : Logout

- What's the use of an application without a logout?
    - Add logout link to the todo page
    - Add a Logout Servlet to handle the request

**Code Snippets and Examples**

## src/main/java/com/in28minutes/logout/LogoutServlet.java

```
package com.in28minutes.logout;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = "/logout.do")
public class LogoutServlet extends HttpServlet {
        protected void doGet(HttpServletRequest request,
                        HttpServletResponse response) throws ServletException,
                request.getSession().invalidate();
                request.getRequestDispatcher("/WEB-INF/views/login.jsp").forwa
                                request, response);
        }
}
```

## src/main/webapp/WEB-INF/views/todo.jsp

```
<nav class="navbar navbar-default">

        <a href="/" class="navbar-brand">Brand</a>

        <ul class="nav navbar-nav">
                <li class="active"><a href="#">Home</a></li>
                <li><a href="/todo.do">Todos</a></li>
                <li><a href="http://www.in28minutes.com">In28Minutes</a></li>
        </ul>

        <ul class="nav navbar-nav navbar-right">
                <li><a href="/logout.do">Logout</a></li>
        </ul>

</nav>
```

## Step 21 : Theory : Understand Maven and Tomcat

Maven is used to manage libraries/dependencies. You don't need to manually download the jars and build the project. Maven does that for you

- [You can read more about Maven here] (https://github.com/in28minutes/in28minutes-initiatives/tree/master/The-in28Minutes-TroubleshootingGuide-And-FAQ#maven)

We download Tomcat server using a Maven plugin and deploy the application war to the tomcat server. So, you would not need to install tomcat manually!

## Step 22 : Theory : Servlet LifeCycle

HttpServlet methods

- init() method
  - called only once.
- service() method
  - called once per servlet invocation
  - looks at the request method and call the appropriate method - doPost, doGet etc

- destroy() method
    - called only once
    - when the application is brought down

JSPs are also servlets. They have all the 3 methods shown above.

## Step 23 : Theory : Model 1 and Model 2 MVC Architectures

Model 1 Architecture

- JSP
    - JSP do everything.
    - Very complex to maintain.

Model 2 Architecture

- Servlet
    - Controller
        - Control the flow.
        - Decides which view to show and what model is needed to display the view.
- JSP
    - View - Displays the model
- Model
    - Todo

Front Controller

- One Controller handles all the requests
    - It would route the requests to other servlets
    - Spring MVC
        - Dispatcher Servlet

## Step 24 : Moving Add Functionality to a New Page.

> *The Boy Scouts have a rule: "Always leave the campground cleaner than you found it."*

Let's follow the boy scout rule by refactoring a little bit.

## Step 25 : Add Category Field

Let's add a category field to the todo page

- We will start with add category to model Todo
- We would need to add it to servlets and jsp pages

Complete code

**Code Snippets and Examples**

## src/main/java/com/in28minutes/todo/AddTodoServlet.java

```
protected void doPost(HttpServletRequest request,
                HttpServletResponse response) throws ServletException, IOExcept
        String newTodo = request.getParameter("todo");
        String category = request.getParameter("category");
        todoService.addTodo(new Todo(newTodo, category));
        response.sendRedirect("/list-todos.do");
}
```

## src/main/java/com/in28minutes/todo/DeleteTodoServlet.java

```
protected void doGet(HttpServletRequest request,
                HttpServletResponse response) throws ServletException, IOExcept
        todoService.deleteTodo(new Todo(request.getParameter("todo"), request
                        .getParameter("category")));
        response.sendRedirect("/list-todos.do");
}
```

## src/main/java/com/in28minutes/todo/Todo.java

```
package com.in28minutes.todo;

public class Todo {
        private String name;
        private String category;

        public Todo(String name, String category) {
                super();
                this.name = name;
                this.category = category;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getCategory() {
                return category;
        }

        public void setCategory(String category) {
                this.category = category;
        }

        @Override
        public String toString() {
                return String.format("Todo [name=%s, category=%s]", name, cate
        }

        //hashcode and equals

}
```

## src/main/java/com/in28minutes/todo/TodoService.java

```
package com.in28minutes.todo;

import java.util.ArrayList;
import java.util.List;
```

```
public class TodoService {
        private static List<Todo> todos = new ArrayList<Todo>();
        static {
                todos.add(new Todo("Learn Web Application Development", "Study
                todos.add(new Todo("Learn Spring MVC", "Study"));
                todos.add(new Todo("Learn Spring Rest Services", "Study"));
        }

        public List<Todo> retrieveTodos() {
                return todos;
        }

        public void addTodo(Todo todo) {
                todos.add(todo);
        }

        public void deleteTodo(Todo todo) {
                todos.remove(todo);
        }

}
```

## src/main/webapp/WEB-INF/views/add-todo.jsp

```
<div class="container">
        Your New Action Item:
        <form method="POST" action="/add-todo.do">
                Description : <input name="todo" type="text" /> <BR/>
                Category : <input name="category" type="text" /> <BR/>
                <input name="add"
                        type="submit" />
        </form>
</div>
```

## src/main/webapp/WEB-INF/views/list-todos.jsp

```
<ol>
        <c:forEach items="${todos}" var="todo">
                <li>${todo.name}&nbsp;${todo.category}&nbsp;<a
                        href="/delete-todo.do?todo=${todo.name}&category=${todo
        </c:forEach>
</ol>
```

## Step 26 : Format the JSPs better.

> *The Boy Scouts have a rule: “Always leave the campground cleaner than you found it.”*

Let’s follow the boy scout rule by refactoring a little bit.

## Step 27 : JSP Fragments

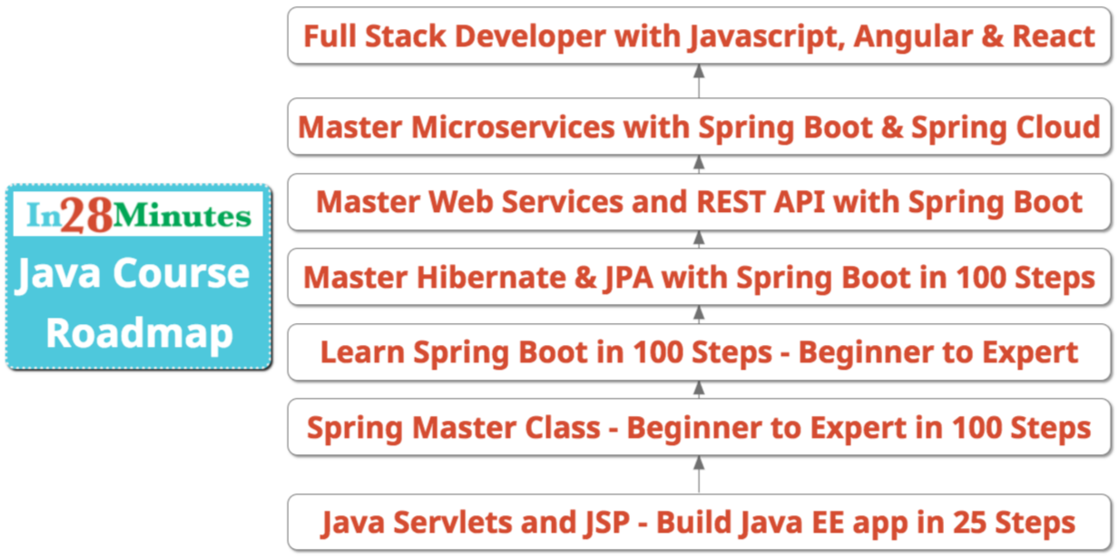
Complete code

**More about in28Minutes**

We would love to hear your thoughts.

**If you loved the course**

Share on Twitter - https://twitter.com/home?
status=Having%20a%20great%20time!%20%0A%20%23in28Minutes%20%23ImLearningIn28Minute


In28Minutes
Java Course Roadmap
Full Stack Developer with Javascript, Angular & React
Master Microservices with Spring Boot & Spring Cloud
Master Web Services and REST API with Spring Boot
Master Hibernate & JPA with Spring Boot in 100 Steps
Learn Spring Boot in 100 Steps - Beginner to Expert
Spring Master Class - Beginner to Expert in 100 Steps
Java Servlets and JSP - Build Java EE app in 25 Steps

Good Luck and Keep Learning in28Minutes

- Linked In : https://www.linkedin.com/in/rangakaranam/
- Facebook : https://www.facebook.com/in28Minutes
- Twitter : https://twitter.com/in28Minutes
- YouTube : https://www.youtube.com/rithustutorials To find out more about our courses and get special offers, visit http://www.in28minutes.com

> *Join our free Spring Boot in 10 Steps Course.*

> *Find out how in28Minutes reached 100,000 Learners on Udemy in 2 years. The in28minutes Way - Our approach to creating awesome learning experiences.*