IS502052: Enterprise Systems Development Concepts

Lab 10: Java Web Application

I. Introduction

In this lab tutorial, we will create a Java Web Application, which interacts with the Enterprise JavaBeans server application.

The sample Web Application bases on the ***SampleEJB-04*** project and ***SampleDB3*** database, as explained in **Lab 8**.

II. Create Java Web Application

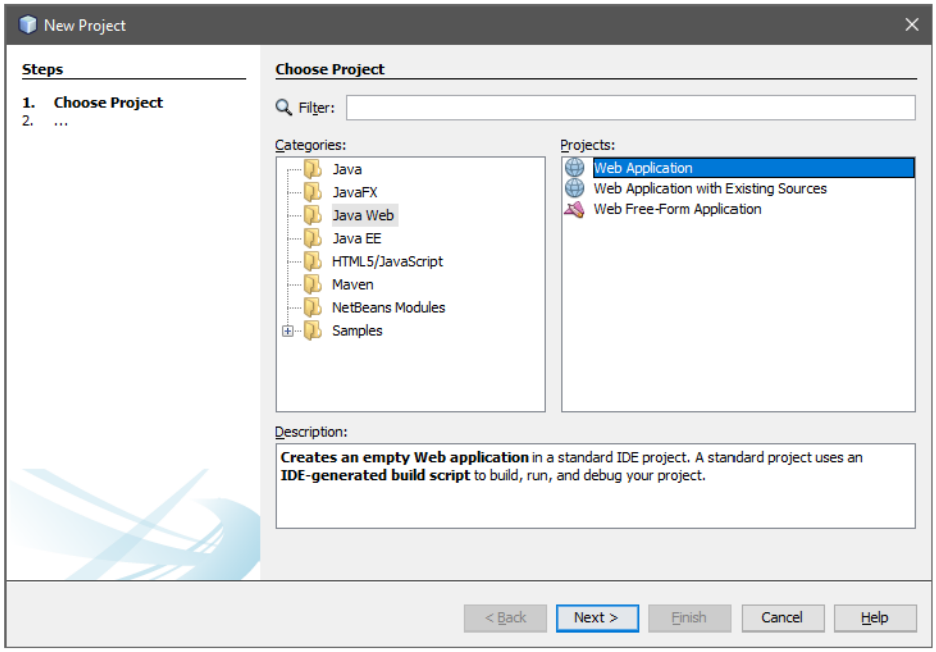
To create a Java Web Application, you should do the following steps:

- Create new Java Web Application project;
- Modify the JSP web page;
- Create controller to manipulate the JSP web page.

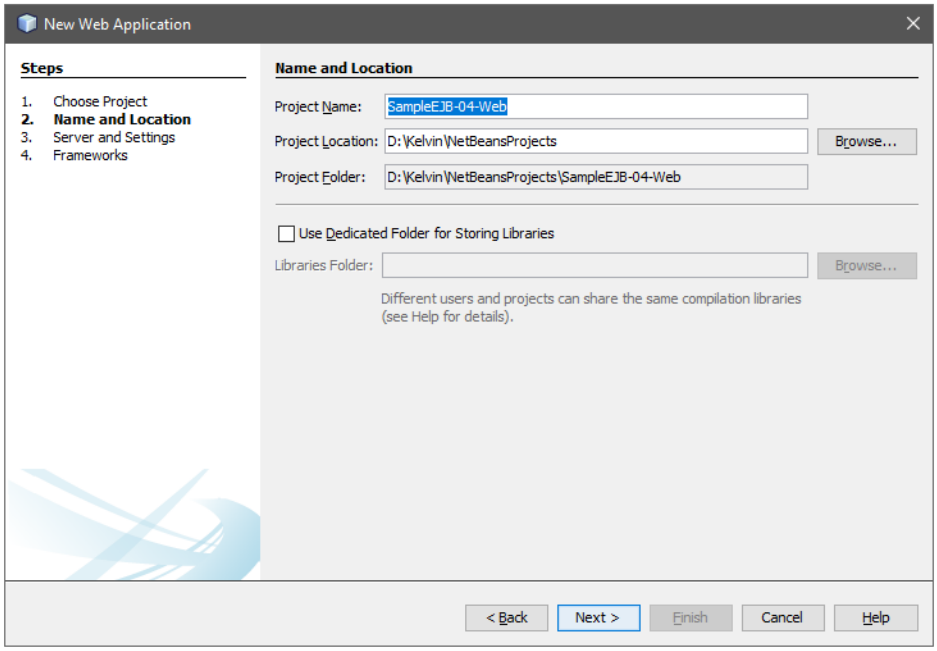
1. Create new project

Do the following steps to create new project:

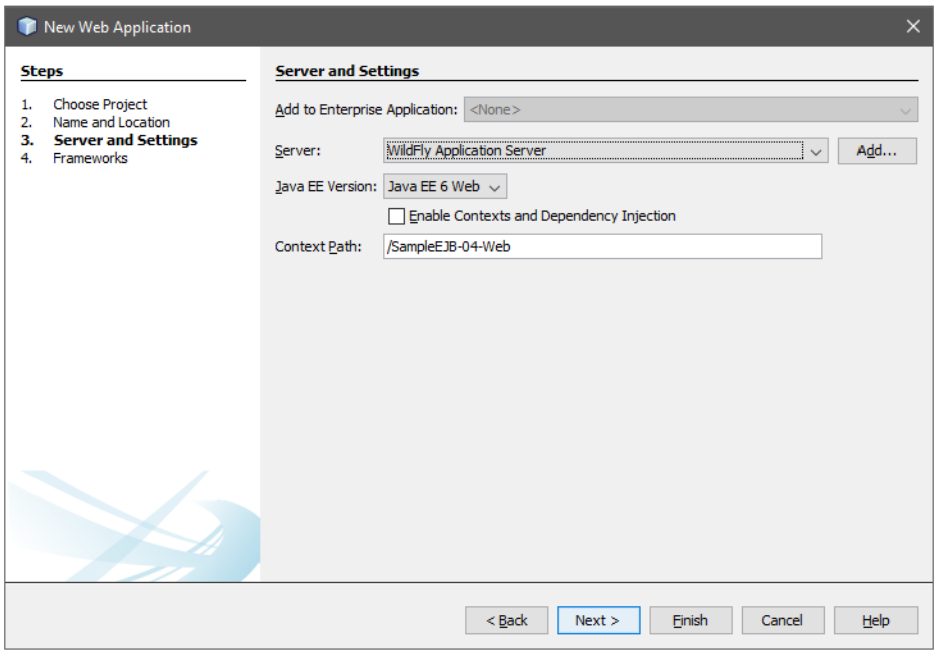
- In NetBeans, open **New Project** wizard, then select **Web Application** in **Java Web** category, click **Next**;



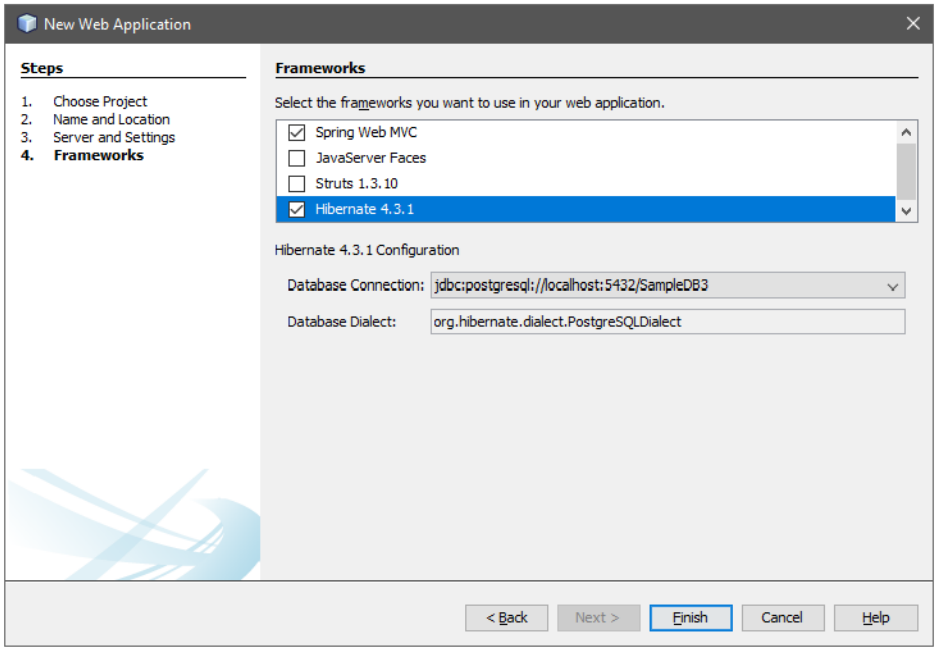
- Name the project name as ***SampleEJB-04-Web***, click **Next**;



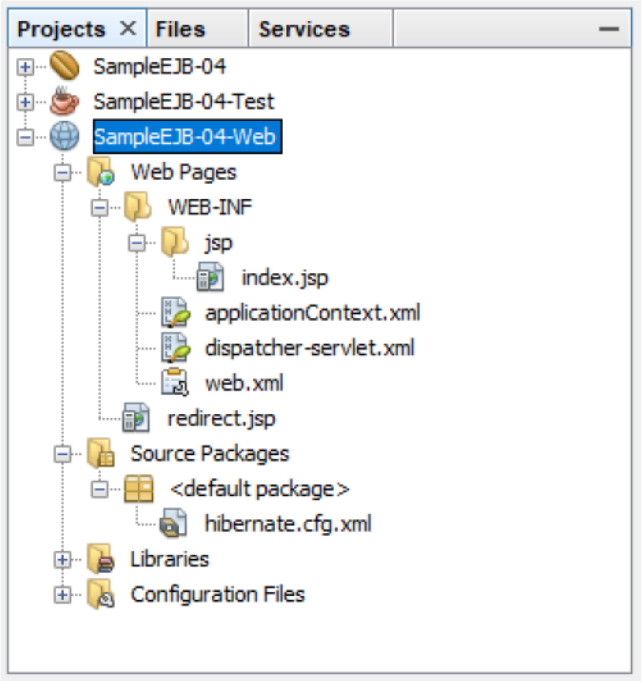
- Select **WildFly** as application server, and you can modify the **Context Path** *(URL of Web App)*, click **Next**;



- In this final step, select both **Spring Web MVC** and **Hibernate 4.3.1** as web framework. Note that, when select **Hibernate 4.3.1**, you need to point the data connection to the ***SampleDB3*** database. Then, click **Finish**.



After NetBeans completely create the new project, the project structure will look like follows:



2. Configure the project

Before proceeding to other steps, we must configure the project settings as follows:

- Add ***SampleEJB-04*** to class path of ***SampleEJB-04-Web***;
- Add JBoss client API *(jboss-client.jar)* to class path;
- Add ***jndi.properties*** and ***jboss-ejb-client.properties***, as explained in previous tutorial.

3. Modify the JSP web page

Do the following steps to modify the JSP web page:

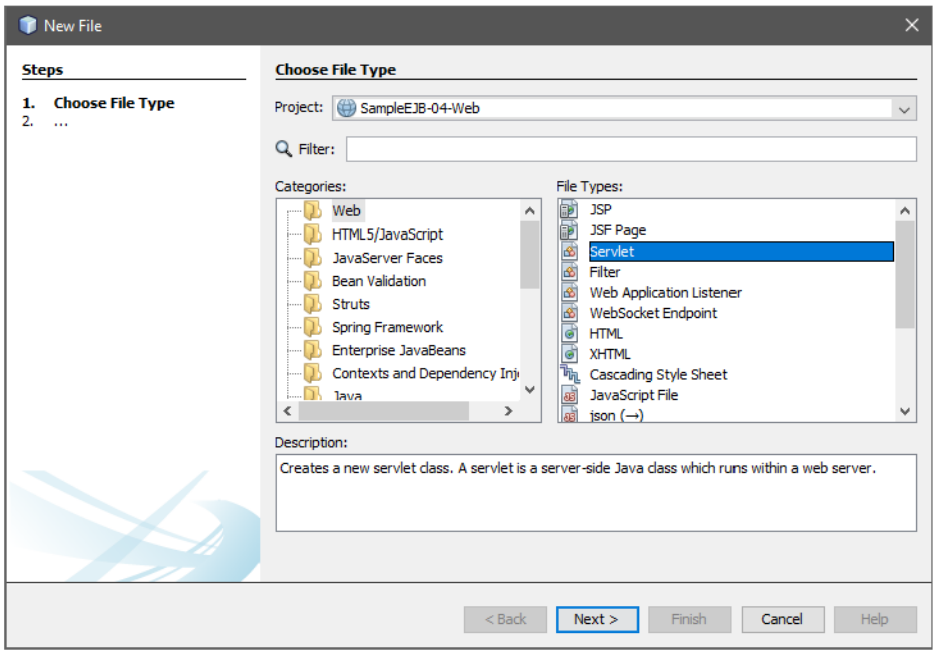
- Open ***index.jsp*** file and modify the content as follows:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Home | TDT System Management</title>
    </head>

    <body>
        <h1>TDT System Management</h1>
        <form action="${pageContext.request.contextPath}/Controller" method="POST">
            Username <input type="text" name="tbUsername" value="" /><br/>
            Password <input type="password" name="tbPassword" value="" /><br/>
            <input type="submit" value="LogIn" name="action" />
        </form>

    </body>
</html>
```

4. Create Controller

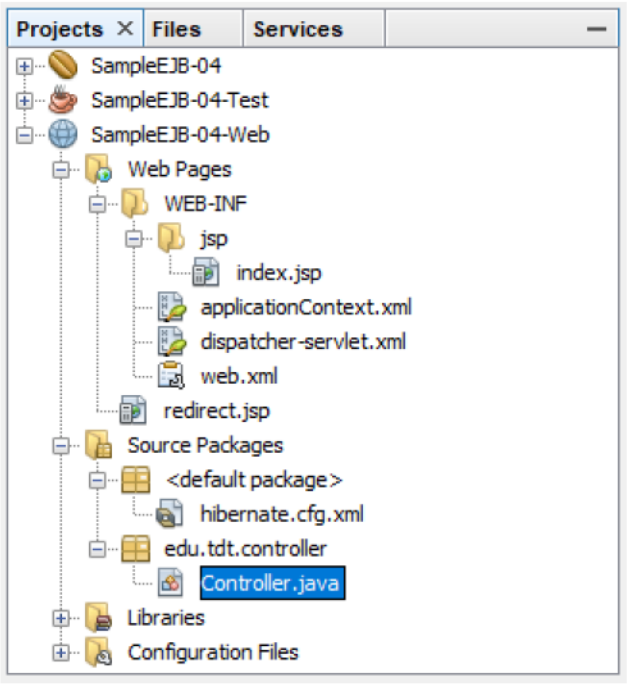
To manipulate the JSP web page, we need to have controller. Do the following steps to create a sample controller:

- Open **New File** wizard and select **Servlet** in **Web** category, then, click **Next**;



- Name this **Servlet** as ***Controller***, and the package is ***edu.tdt.controller***, click **Next** or **Finish**;
- In the **Configure Servlet Deployment** step, we can keep all settings as default.

The project structure is now having two important directories: (1) **Web Pages** and (2) **Source Packages**.



Modify the Controller

In ***Controller.java***, we add the following methods and modify the ***processRequest()*** method:

```
package edu.tdt.controller;

import edu.tdt.entity.SystemManagement;
import edu.tdt.entity.SystemManagementRemote;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.naming.InitialContext;
import javax.naming.NamingException;
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(name = "Controller", urlPatterns =
{
    "/Controller"
})
public class Controller extends HttpServlet {

    private Properties props;
    private InitialContext ctx;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter())
        {
            // Page header
            out.println("<h1>TDT System Management</h1>");
            // Initialize JNDI
            readJNDI();

            // Lookup the SystemManagementRemote
            SystemManagementRemote session = (SystemManagementRemote) ctx.lookup(getJNDI());

            String action = request.getParameter("action");
            if(action.equals("LogIn"))
```

```
            {
                String username = request.getParameter("tbUsername");
                String password = request.getParameter("tbPassword");

                if(password.hashCode() == session.getUserPassword(username))
                {
                    out.println("<p>Log-in Successful!</p>");
                }
                else
                {
                    out.println("Wrong username/password");
                }
            }
        } catch (NamingException ex)
        {
            Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * Read the JNDI properties file
     */
    private void readJNDI()
    {
        props = new Properties();

        try
        {
            props.load(new FileInputStream("jndi.properties"));
        } catch(IOException e)
        {
            e.getMessage();
        }

        try
        {
            ctx = new InitialContext(props);
        } catch (NamingException ex)
        {
            ex.getMessage();
        }
    }

    /**
     * Construct and return the JNDI address of called class
     * @return String
     */
    private String getJNDI()
    {
        String appName = "";
        String moduleName = "SampleEJB-04";
        String distinctName = "";
        String sessionBeanName = SystemManagement.class.getSimpleName();
        String viewClassName = SystemManagementRemote.class.getName() + "?stateful";

        return "ejb:"+appName+"/"+moduleName+"/"+distinctName+"/"+sessionBeanName+"!"+viewClassName;
    }

}
```

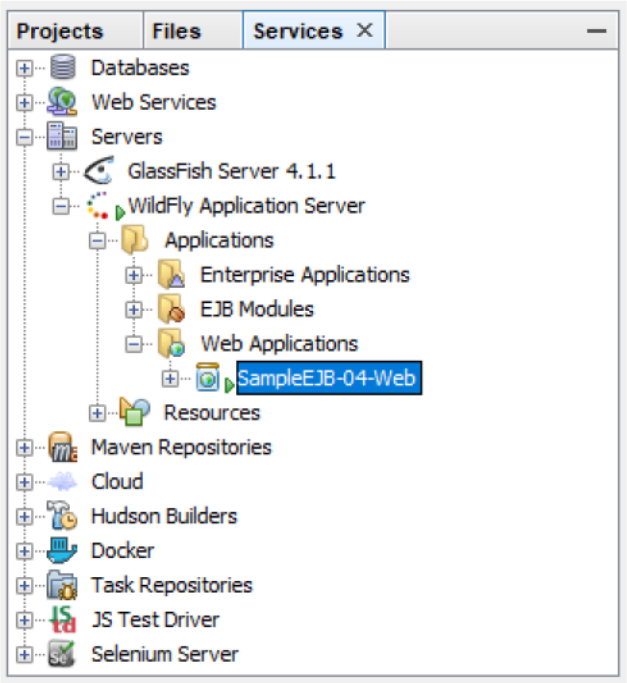
We should notice that we still keep the two methods from all client projects, (1) ***readJNDI()*** and (2) ***getJNDI()***. That means, no matter the client is console application or web application, the configuration is still the same.

5. Deploy and Run the Web Application

To run the web application project, do these steps below:

- Make sure the **WildFly Application Server** is started;
- Right-click on project and select **Deploy**;

- Switch to **Services** tab and expand ***Servers/WildFly Application Server/Applications/Web Applications/***;



- Right-click on web app, supposed ***SampleEJB-04-Web***, and select **Open in Browser**.

III. Exercises

1. Implement more functions for the sample web app.