Web Development in Java – Live Demonstrations

(Live demonstrations done using Eclipse for Java EE 4.8 and WildFly 13)[1]

Java Servlets:

1. Switch to the Java EE Perspective (if not there already);
2. File > New > Dynamic Web Project;
3. Follow the wizard, give the project a name (e.g., LiveDemo) and choose to generate `web.xml`;
4. Go to Java Resources/src, right-click > New > Class;
5. Create the `livedemo.HelloServlet` class, extending `HttpServlet`:

```
public class HelloServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
     resp.setContentType("text/html");
     try (PrintWriter out = resp.getWriter()) {
       out.write("<html><head><title>Hello, Servlet!</title></head><body>");
       out.write("<h1>Hello, World!</h1>");
       out.write("<p>The time now is " + new Date() + "</p>");
       out.write("</body></html>");
     }
  }
}
```

6. Edit `web.xml`:

```
<servlet>
  <servlet-name>HelloServlet</servlet-name>
  <servlet-class>livedemo.HelloServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>HelloServlet</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

7. Right-click the project > Run As > Run on Server;
8. Open http://localhost:8080/LiveDemo/hello

Completing the WebApp:

1. Right-click WebContent > New > HTML File > "index.html";
2. Body of the page:

```
<form action="hello" method="get">
  <p>Please tell me your name:
    <input type="text" name="visitor" size="20" /></p>
  <p><input type="submit" value="Enter" /></p>
</form>
```

3. Change the Servlet:

[1] See https://github.com/dwws-ufes/jbutler/wiki for tool installation instructions.

```
out.write("<h1>Hello, " + req.getParameter("visitor") + "!</h1>");
```

GET and POST requests:

1. Change method="get" to method="post" in index.html;
2. Try it, see that it doesn't work;
3. Implement the doPost() method in the servlet, calling doGet(). Mention common pattern of both methods calling a third one, doService() or something;
4. Try again, showing that the query string is gone.

Annotations for URL mapping (Servlets 3):

1. Delete the Servlet mapping from web.xml;
2. Add annotation to the Servlet:

```
@WebServlet(name = "HelloServlet", urlPatterns = {"/hello"})
```

JavaServer Pages:

1. Right-click WebContent > New > JSP File, File name: hello.jsp, Finish;

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
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>Hello, JSP!</title>
</head>
<body>

<%
String visitor = request.getParameter("visitor");
visitor = ((visitor == null) || (visitor.length() == 0)) ? "visitor" : visitor;
%>

<h1>Hello, <%= visitor %>!</h1>

<p>The time now is <%= new java.util.Date() %></p>

</body>
</html>
```

2. Note the three types of JSP tags (<%@ @>, <% %> and <%= %>);
3. Change action="hello" to action="hello.jsp" in index.html and test it.

JavaServer Faces:

1. Right-click the project > Properties > Project Facets;
2. Select JavaServer Faces and configure it to *.faces;
3. Open web.xml and show the JSF configuration;

4. Also show there's a faces-config.xml, but it's empty for now;
5. Right-click WebContent > New > HTML File > "index.xhtml" > Next;
6. Select New XHTML File (1.0 strict) > Finish;

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
<h:head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>Hello, JSF!</title>
</h:head>
<h:body>
   <h1>Hello, <h:outputText value="#{helloController.visitor}" />!</h1>
   <p>The time now is <h:outputText value="#{helloController.time}" /></p>
   <h:form id="helloForm">
     <p>Please tell me your name: <h:inputText
value="#{helloController.visitor}" /></p>
     <p><h:commandButton value="Enter" /></p>
   </h:form>
</h:body>
</html>
```

7. Create a new class for the controller:

```
package livedemo;

import java.util.Date;
import javax.enterprise.inject.Model;

@Model
public class HelloController {
   private String visitor = "visitor";
   public String getVisitor() { return visitor; }
   public void setVisitor(String visitor) { this.visitor = visitor; }
   public Date getTime() { return new Date(); }
}
```

8. Try it, then back to Eclipse to AJAXify it:

```
   <h:form id="helloForm">
     <h:panelGroup id="helloPanel">
       <h1>Hello, <h:outputText value="#{helloController.visitor}" />!</h1>
       <p>The time now is <h:outputText value="#{helloController.time}" /></p>
     </h:panelGroup>
     <p>Please tell me your name:
       <h:inputText value="#{helloController.visitor}">
         <f:ajax event="blur" render="helloPanel" execute="@this" />
       </h:inputText>
     </p>
   </h:form>
   </h:body>
```

9. Note that he panel must be inside the form, or else JSF won't find it;
10. Run Marvin (http://dev.nemo.inf.ufes.br:8180/Marvin) and show some screens.

Java Hostel

Create the project:

1. File > New > Dynamic Web Project:, Project name: JavaHostel, Target runtime: WildFly ## Runtime (## is the version number, whatever it is), Next > Next;
2. Select Generate web.xml deployment descriptor > Finish.

Add facets:

1. Right-click JavaHostel > Properties > Project Facets. Select:
    a. JavaServer Faces;
    b. JPA;
2. Open JavaHostel/JPA Content/persistence.xml. Inside <persistence-unit> Try to use the graphical editor:
    a. General / Persistence provider: org.hibernate.ejb.HibernatePersistence;
    b. Connection / JTA data source: java:/jboss/datasources/JavaHostel;
    c. Properties / Add... / Name: hibernate.hbm2ddl.auto / Value: update;
3. XML result should be:

```
<provider>org.hibernate.ejb.HibernatePersistence</provider>
<jta-data-source>java:/jboss/datasources/JavaHostel</jta-data-source>
<properties>
 <property name="hibernate.hbm2ddl.auto" value="update" />
</properties>
```

Apply web template as decorator:

1. Download the Elements Responsive template from https://www.html5webtemplates.co.uk/templates.html;
2. Create new folder: JavaHostel/WebContent/resources;[2]
3. Copy the folders (css, font, images, js, sass) from the template to JavaHostel/WebContent/resources;
4. Create new folder: JavaHostel/WebContent/WEB-INF/templates;
5. Right-click JavaHostel/WebContent/WEB-INF/templates > New > HTML File;
6. File name: decorator.xhtml > Next;
7. Templates: select New XHTML File (1.0 strict) > Finish;
8. Change decorator.xhtml. On <html> add:

```
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:h="http://java.sun.com/jsf/html"
```

[2] Do not change the name of this folder. See: https://stackoverflow.com/questions/8367421/how-to-reference-css-js-image-resource-in-facelets-template

9. Replace <body> with <h:body>, <head> with <h:head>
10. Inside <title> add:

```
<h:outputText value="JavaHostel :: " /><ui:insert name="title" />
```

11. Copy the contents of the <head> (except title and content-type) and <body> of no-sidebar.html to the <h:head> and <h:body> of decorator.xhtml;
12. Change relative references to css/ and js/ to absolute references using JSF tags by performing the following search + replace tasks:
    a. <script src= → <h:outputScript name=
    b. ></script> → />
    c. <link rel="stylesheet" href= → <h:outputStylesheet name=
13. Look for "<!-- Main -->" and change the contents of the <header> just below it:

```
<header class="major">
   <h2>JavaHostel</h2>
   <p>Java EE Example WebApp</p>
</header>
```

14. Look for "<!-- Content -->" and delete all the contents in <section id="content">, just below it:

```
<section id="content">
   <ui:insert name="contents">Blank page.</ui:insert>
</section>
```

15. Right-click JavaHostel/WebContent > New > HTML File, File name: index.xhtml, Template: New XHTML File (1.0 strict);
16. Change the file, deleting the entire <html> element (all the way until </html>). Replace it with:

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   template="/WEB-INF/templates/decorator.xhtml">

   <ui:define name="title">Welcome</ui:define>
   <ui:define name="contents">
      <h1>Welcome to JavaHostel</h1>
      <p>Under development.</p>
   </ui:define>
</ui:composition>
```

17. Right-click JavaHostel/WebContent > New > File, File name: index.html. Contents of this file:

```
<html>
<head><meta http-equiv="refresh" content="0;url=index.faces"></head>
</html>
```

18. Open JavaHostel/WebContent/WEB-INF/web.xml. Edit welcome-file-list to include index.html;

19. Deploy and test.

Database creation and set-up:

1. Follow the instructions for "Database creation and set-up" and "Datasource configuration in WildFly" from JButler's wiki;[3]
    a. The datasource should be: java:/jboss/datasources/JavaHostel
    b. The pool name should be: JavaHostelPool
    c. The database and user names should be: javahostel

Implement domain classes:

1. Create package br.ufes.inf.nemo.javahostel.domain;
2. Create classes: Guest, Booking, Bed, Room;
3. Add class attributes and JPA mappings. For all classes, add class annotation @Entity and ID attribute:

```
@Id @GeneratedValue(strategy = GenerationType.AUTO)
private Long id;
```

4. For Guest:

```
private String name;
private String email;
private String password;

@Temporal(TemporalType.DATE)
private Date birthDate;
```

5. For Booking:

```
@ManyToOne
private Guest guest;
@OneToMany
private Set<Bed> beds;
@Temporal(TemporalType.DATE)
private Date startDate;
@Temporal(TemporalType.DATE)
private Date endDate;
```

6. For Bed:

```
@ManyToOne
private Room room;
private int number;
private double pricePerNight;
```

7. For Room:

```
private int number;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "room")
private Set<Bed> beds;
```

[3] https://github.com/dwws-ufes/jbutler/wiki/Tutorial%3A-a-Java-EE-Web-Profile-application-with-JButler%2C-part-2

8. Generate getters and setters for all attributes.

Automatic DB schema creation:

1. Deploy/restart the application;
2. Go to MySQL Workbench and check that tables were created in javahostel DB.

Guest registration:

1. Open JavaHostel/WebContent/WEB-INF/templates/decorator.xhtml, add a link to the <nav id="nav"> in the <!-- Header -->:

```
<li><a href="/registration/index.faces">Registration</a></li>
```

2. Create folder JavaHostel/WebContent/registration;
3. Copy: JavaHostel/WebContent/index.xhtml to this new folder;
4. Change the title to Registration and use the following contents:

```
<h1>Registration</h1>
<p>Fill in your data to become a guest at Java Hostel.</p>

<h:form id="regForm">
 <p>Name: <h:inputText id="name" value="#{registrationController.guest.name}"
/></p>

 <p>Birthdate: <h:inputText id="birthDate"
value="#{registrationController.guest.birthDate}">
 <f:convertDateTime pattern="dd/MM/yyyy" />
 </h:inputText></p>

 <p>E-mail: <h:inputText id="email"
value="#{registrationController.guest.email}" /></p>

 <p>Password: <h:inputSecret id="password"
value="#{registrationController.guest.password}" /></p>

 <p><h:commandButton action="#{registrationController.register}"
value="Register" /></p>
</h:form>
```

5. Implement the controller class in package br.ufes.inf.nemo.javahostel.control:

```
@Model
public class RegistrationController implements Serializable {
   @EJB
   private RegistrationService registrationService;
   private Guest guest = new Guest();
   private int age;

   public Guest getGuest() { return guest; }
   public int getAge() { return age; }

   public String register() {
      try {
         registrationService.register(guest);
      }
      catch (UnderAgeGuestException e) {
         age = e.getAge();
         return "/registration/underage.xhtml";
```

```
    }
    return "/registration/success.xhtml";
  }
}
```

6. Quick fix "RegistrationService cannot be resolved to a type" to create the class, in package br.ufes.inf.nemo.javahostel.application (note: for simplicity, JavaHostel is a 2-tier application. The persistence code is in the application layer):

```
@Stateless
@LocalBean
public class RegistrationService implements Serializable {
  @PersistenceContext
  private EntityManager entityManager;

  public void register(Guest guest) throws UnderAgeGuestException {
    int age = calculateAge(guest.getBirthDate());
    if (age < 18) throw new UnderAgeGuestException(age);
    entityManager.persist(guest);
  }

  private static int calculateAge(Date birthDate) {
    if (birthDate == null) return 0;
    Calendar birth = Calendar.getInstance();
    birth.setTime(birthDate);
    Calendar today = Calendar.getInstance();
    today.setTime(new Date(System.currentTimeMillis()));
    int age = today.get(Calendar.YEAR) - birth.get(Calendar.YEAR);
    birth.add(Calendar.YEAR, age);
    if (birth.after(today)) age--;
    return age;
  }
}
```

7. Same thing for UnderAgeGuestException (also in application):

```
public class UnderAgeGuestException extends Exception {
  private int age;

  public UnderAgeGuestException(int age) { this. age = age; }

  public int getAge() { return age; }
}
```

8. Finally, the result pages. First, JavaHostel/WebContent/registration/underage.xhtml (as before, copy from index.xhtml):

```
<p>Dear <h:outputText value="#{registrationController.guest.name}" />,
unfortunately underage people are not allowed to register as guests and,
according to your birth date, you have only <h:outputText
value="#{registrationController.age}" /> years.</p>
```

9. Then, JavaHostel/WebContent/registration/success.xhtml :

```
<p>Dear <h:outputText value="#{registrationController.guest.name}" />, welcome
to JavaHostel.</p>
```

10. Redeploy and test both cases, show that the guest data ends up in the database.
    a. If the date is registered in the database as the day before, add the following to JavaHostel/WebContent/WEB-INF/web.xml

```
<context-param>
  <param-name>javax.faces.DATETIMECONVERTER_DEFAULT_TIMEZONE_IS_SYSTEM_TIMEZONE</param-name>
```

```
    <param-value>true</param-value>
</context-param>
```