Session 8

JavaBeans

1

Reading & Reference

- Reading
  - Head First - Chapter 3 (MVC)
- Reference
  - JavaBeans Tutorial - docs.oracle.com/javase/tutorial/javabeans/

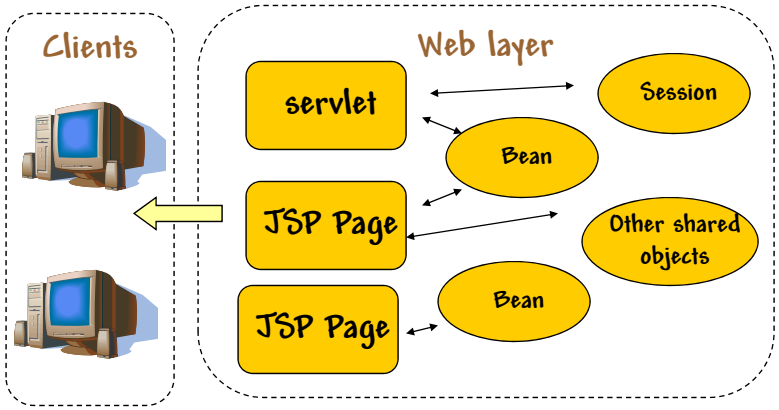
2

Lecture Objectives

- Understand how the Model/View/Controller design pattern applies to Web applications
- Understand how a Java Bean is used to store data in a Web application
- Know the features a Java class must possess to be considered a Java Bean
- Know how to use a servlet to move form data into a Java Bean

3

Web Architecture



Web applications are usually constructed with the Model, View, Controller pattern

4

Model / View / Controller Pattern

- Web systems are usually decomposed into MVC components
  - JSPs - view
  - Servlets - controller
  - Java Beans (and custom tags) - model

For now, just think of a JSP as an easy way to write a servlet that generates HTML

Data and business logic for a Web application are usually stored in objects that are visible to servlets and JSPs

The handle to a bean is usually stored in the relevant shared object (e.g., Session and ServletContext)

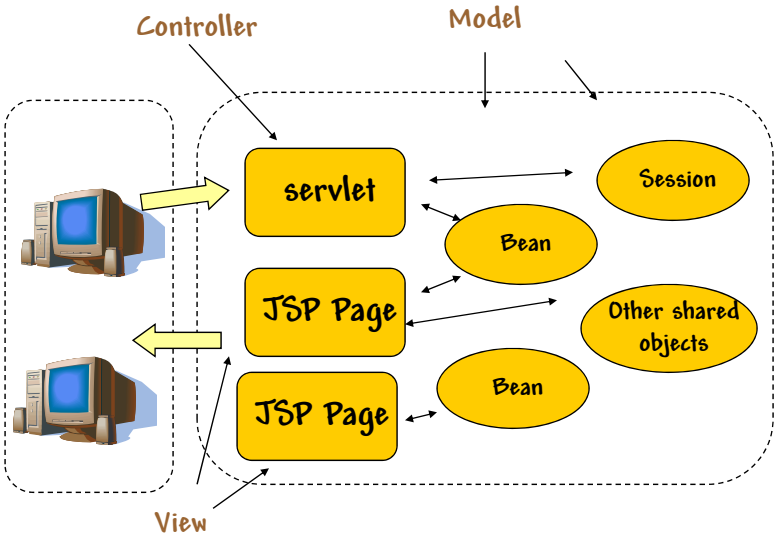
5

JavaBeans and Web Applications

- A bean is an object that you can easily use within your JSP or servlet
- You can create a bean within your JSP or servlet
- You can share a bean with other JSPs and servlets and thereby share data
- You can get and set properties in the bean
- Bean data can be persistent

6

MVC Web Architecture



7

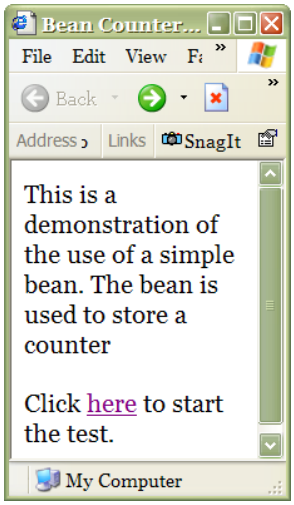
What Makes a Bean a Bean?

- A bean is an instance of a Java class that:
  - Must have a zero argument constructor
  - Should have no public instance variables
  - Should have (properly named) get and set methods for any instance variables that are to be accessed (setter argument type and getter return type must be identical)
  - Must support persistence (the bean is serializable)
- A bean usually supports events either by firing events when some properties change or listening for events (although we usually do not use this feature)

8

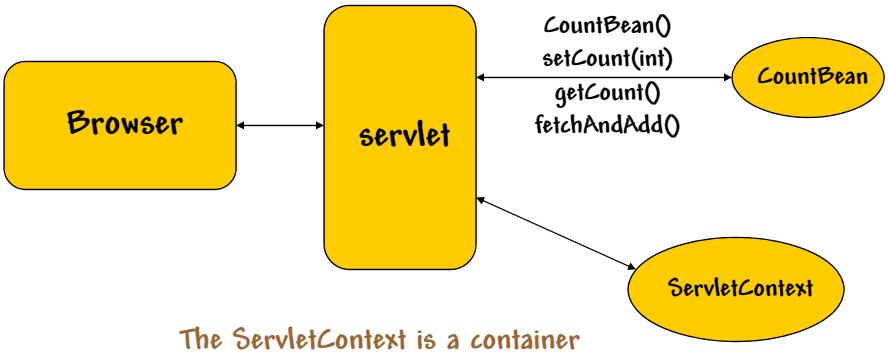
Example: Counter

- The counter value is stored in a bean - along with methods to increment, get, and set the counter



9

Bean Counter



The ServletContext is a container level object that is used to store a handle to the CountBean

10

ServletContext

- Actually, an interface to the Web container, but you can think of it as a shared object used to store data available to all servlets in the Web application
- There is one ServletContext per Web application
- You can obtain a reference (i.e., handle) to the ServletContext object in many ways (e.g., through a call to the getServletContext method of HttpServlet)

11

Using the ServletContext

- Includes methods for getting and setting an attribute (name/value pair) as if the ServletContext were a Map
  - Object getAttribute(String)
  - void setAttribute(String, Object)
  - Enumeration getAttributeNames()

The name/value pairs are of type String/Object

12

BeanCounter Generated Page

13

BeanCounter Servlet ...

```
public class BeanCounter
   extends HttpServlet {
   private static final String CONTENT_TYPE = "text/html";
   private static final String DOC_TYPE =
     "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
   Strict//EN\"\n" +
     "  \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">";
  /**Initialize global variables*/
  public void init() throws ServletException {
  }

   public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws
        ServletException, IOException {
    int bCount = 0;
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println(
       "<head><title>Bean Counter</title></head>");
    out.println("<body >");
    out.println("<h2>Bean Counter</h2>");
```

14

... BeanCounter Servlet

```
    ServletContext sc = this.getServletContext();
    CountBean b = (CountBean) sc.getAttribute("b");
    if (b == null) {
      b = new CountBean();
      sc.setAttribute("b", b);
    }
    bCount = b.fetchAndAdd();
    out.println("<p>Initial value of counter in the bean - ");
    out.println(bCount + "</p>");
    bCount = b.getCount();
       out.println("<p>Incremented value of counter in the bean -
  ");
    out.println(bCount + "</p>");
    out.println("<p>Return to");
    out.println("<a
  href=\"http://localhost:8080/CodeCSE336/counter.htm\">");
    out.println("Return to the bean counter servlet</a>");
    out.println("</body></html>");
  }
}
```

The servlet gets the value of the counter from the CountBean bean

Shows that a bean can have methods other than getters and setters

15

CountBean

Notice the setter and getter naming conventions

```
public class CountBean implements Serializable
  {
  private int count = 0;

  public int getCount() {
    return (count);
  }

  public int fetchAndAdd() {
    int temp=count;
    count++;
    return (temp);
  }

  public void setCount(int newCount) {
    this.count = newCount;
  }
}
```

Notice that fetchAndAdd returns the pre-incremented value of the counter

Notice that the bean is a standard Java class, but has the features of a bean (constructor, persistence, private instance variable, and properly named methods)

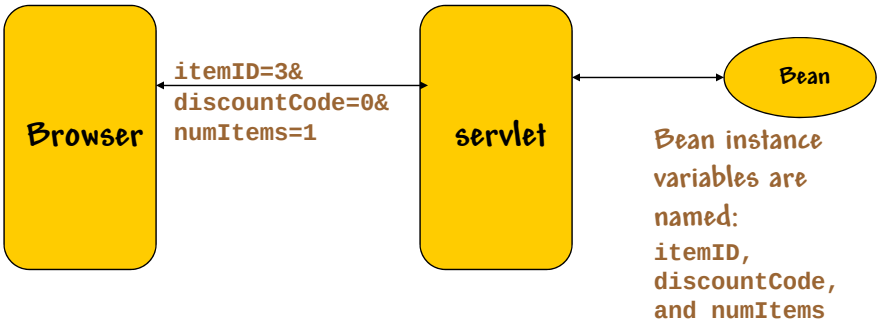
16

Using a Bean to Store Form Data

- Frequently, you will use a bean to store the values of a form dataset
- The form values in the bean are available to other modules in your application (e.g., JSPs)
- You may need to add type conversion
- You may need to add validation to the bean

17

Setting all Bean Values From the Form



- A Web module (e.g., servlet) will usually read the form data set and set the values of the form in a bean so that they can be used by other Web modules

18

Typical Bean Setup

```
public void doGet(HttpServletRequest request,
  HttpServletResponse response) throws
    ServletException, IOException {
      ServletContext sc = this.getServletContext();
      FormBean b = (FormBean) sc.getAttribute("OracleForm");
      if (b == null) {
             b = new FormBean();
             sc.setAttribute("OracleForm", b);
             }
      b.setEmail(request.getParameter("email"));
      b.setFirstName(request.getParameter("firstName"));
      ...
      if (b.isValid()) {
     display the JSP associated with the data
      }
      else {
     redisplay the form with the fields filled in and
      with the error messages
      }
}
```

Note the names

This is the logic of the class project

19

Form Bean Value Setting

- Typically, you will set the value of a bean to the value of the associated form element (in the form data set)
  - Allows the form data set to persist
  - Allows the form data set to be shared among a collection of servlets and JSPs

```
b.setDate(request.getParameter("date"));
```

Notice how the same name is used for the bean attribute and the form element

```
<input name="date" size="10" class="nav" type="text" />
```

20

Explicit Type Conversion

- In the previous example, if the last name parameter is of type String - it works OK
- But, there could be some type considerations
  - Remember that each item in a form data set is of type String/String
- If your bean instance variable is not a String:
  - The value should be explicitly cast to the correct type

```
b.setNumItems(
    Integer.parseInt(request.getParameter("numItems")));
```

Explicit type conversion may not always be needed (Java 5 and above)

21

General Forms Processing

- Define form in HTML (or JSP)
  - Specify initial values of elements
- Transmit form to server
- Validate form content (within a bean method)
- Select page to be displayed, based on validation
  - If form content contains errors
    - Repopulate form with already entered values
    - Include error messages
    - Send form (as HTML) to browser
  - If no errors, display next page in sequence

22

Have You Satisfied the Lecture Objectives?

- Understand how the Model/View/Controller design pattern applies to Web applications
- Understand how a Java Bean is used to store data in a Web application
- Know the features a Java class must possess to be considered a Java Bean
- Know how to use a servlet to move form data into a Java Bean

23

Assignment 3

- Write a servlet and form bean to store the form data
- The form bean should contain variables for all the form parameters in the project page
- Add a validation method to the bean that will validate
  - All required fields contain data (* on page denotes required)
  - E-mail address is in correct format (i.e., xxxx@yyyyy)
- Your servlet should
  - store the data in the bean and
  - display (in a simple HTML page) an error message(s) for any field that is not valid or a message stating that the fields are correct

24