Session 12

JSP Tag Library (JSTL)

1

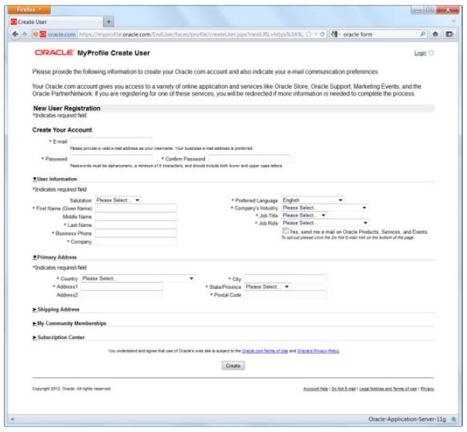
Reading & Reference

- Reading
  - Head First - Chap 9, pages 439-474
- Reference (skip internationalization and sql sections)
  - Java EE 5 Tutorial (Chapter 7) - link on CSE336 Web site (References Section)
  - JavaWorld - www.javaworld.com/javaworld/jw-02-2003/jw-0228-jstl.html
- Collections
  java.sun.com/docs/books/tutorial/collections/

2

HW 4 - Project Form

- HW 4 requires
  - a clear approach to testing and development
  - an understanding of objects that are passed and objects that are shared
- Your debugging should test your assumptions about the passed and shared objects
- We will build a JSP using JSTL that may help you in your debugging for HW#4

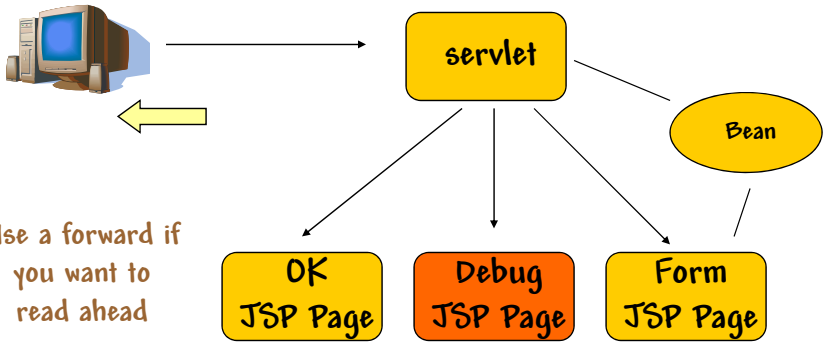


3

HW 4 Approach

For now, you can use a redirect to send control from the servlet to a JSP

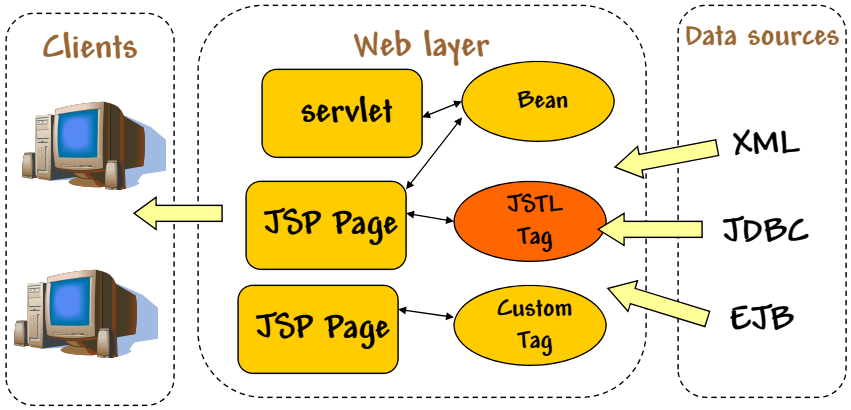
Remember that redirect is an http header that you set in your response (with a URL)



Use a forward if you want to read ahead

4

Web Architecture



5

JSTL Background

- JSTL tags are valid XML
- JSTL contains actions for common tasks
  - Iteration
  - Session tracking
  - Redirect
  - XML
  - Etc.
- You will use EL expressions in your JSTL tags

Remember - the development theme is "no scriptlets"

6

Libraries

- JSTL has 5 standard tag libraries
  - Core
  - Internationalization/format
  - XML
  - SQL
  - Functions (primarily string manipulations)
- We refer to JSTL tags as actions (to distinguish them from static tags)

7

Example - Form Debugger

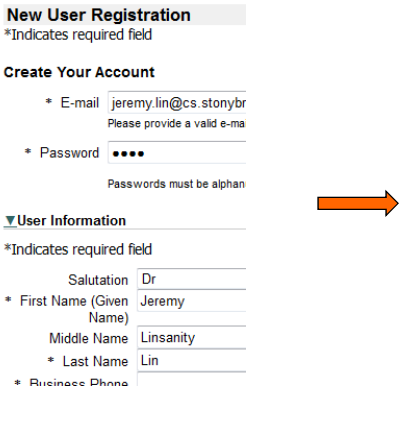
- A JSP that you can use during testing to verify the state of your objects when you submit a request from your browser

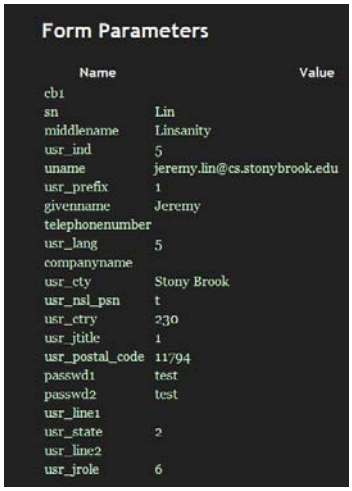
To use the form debugger, you can send your form request to this JSP instead of to your servlet

8

Example - Form Debugger Output

- The JSP will output the collection of parameter name/value pairs in a table





Form Parameters

Name	Value
cb1	
sn	Lin
middlename	Linsanity
usr_ind	5
uname	jeremy.lin@cs.stonybrook.edu
usr_prefix	1
givenname	Jeremy
telephonenumber	
usr_lang	5
companyname	
usr_cty	Stony Brook
usr_nsl_psn	t
usr_ctry	230
usr_jtitle	1
usr_postal_code	11794
passwd1	test
passwd2	test
usr_line1	
usr_state	2
usr_line2	
usr_jrole	6

9

Example - Form Debug JSP

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
   prefix="c" %>
<html>
<head>
  <title>FormTester</title>
  <link rel="stylesheet" type="text/css“
    href="http://www.cs.sunysb.edu/~cse336/Cse336.css"/>
</head>
<body>
  <h2>Testing the Input Form</h2>
  <table>
    <tr>
    <th>Parameter Name</th>
    <th>Parameter Value</th></tr>
     <c:forEach var="p" items="${param}">
       <tr>
         <td> <c:out value="${p.key}"/> </td>
         <td> <c:out value="${p.value}"/> </td>
       </tr>
     </c:forEach>
  </table>
</body>
</html>
```

Declares the JSTL core tag library

Variable is a named reference to an object (explicit or implicit)

EL expression

10

c:out

- Use the <c:out> tag to evaluate an expression and include the expression's output in the page

```
<c:out value="${p.value}" escapeXML="true"
       default="No value for parameter" />
```

- Attributes
  - value - expression whose value is sent to the output
  - escapeXML - specifies whether the html characters (e.g., <) should be converted to the corresponding character entity code
  - default - default value if EL cannot evaluate the expression or if the expression evaluates to null

Is `<c:out value="${p.key}"/>` the same as `${p.key}` ?

11

JSP 2.0

- You can use pure EL instead of c:out

```
<c:out value="${applicationScope[x]}" />
```

```
<c:out value="${x}" />
```

```
${x}
```

Remember, these tags are equivalent
(if x is a key in only the ServletContext)

12

c:forEach

- Use the <c:foreach> tag to iterate through a collection's elements

```
<c:forEach var="p" items="${param}">
```

- If the items attribute is a map, the iteration variable is a Map.Entry with properties of key and value
- Allows partial collection iteration through attributes
  - begin
  - end
  - step

```
<c:forEach var="x" begin="0" end="10" step="2">
```

13

Java For Each Loop

- With Java 5.0, there is a new loop construct

Equivalent Code

```
Collection<String> c = …;
Iterator<String> iter
   = c.iterator();
while (iter.hasNext())
   {
   String elem = iter.next ();
   loop content
   }
```

Both constructs iterate over c, a collection of objects.

Java 5.0 forEach loop Code

```
Collection<String> c = …;

for (String elem : c)
{
   loop content
}
```

For each iteration, elem is the current object

14

Iteration Over a Collection

- In JSTL, you can iterate over a collection or a range - similar to Java

Java

```
Collection<String> t = …;

for (String i : t)
{
   loop content
}
```

JSTL

```
<c:forEach var="i" items="${t}" >
   loop content
</c:forEach>
```

For each iteration, i is the current object

Both constructs iterate over t, a collection of objects.

It is helpful to understand the type of the iterator

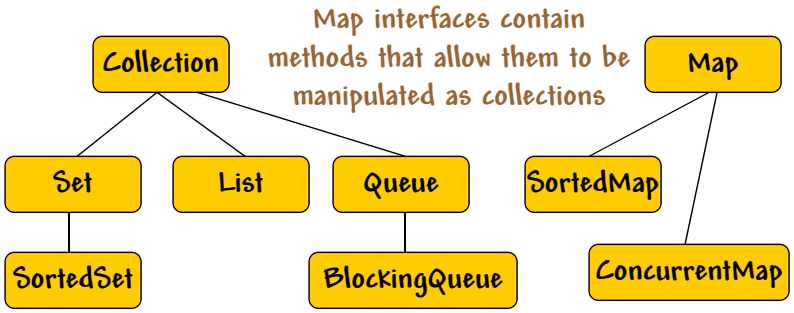
15

What is a Collection?

- A collection is a Java object that groups multiple elements into a single unit
- Java uses collections in a collection framework - a unified architecture for representing and manipulating collections, allowing them to be manipulated independently of the details of their representation
- The Collection Framework includes
  - Interfaces - abstract data types
  - Implementations - concrete implementations
  - Algorithms - methods that perform useful computations
  - Array utilities

16

Collections Interfaces



- Some collections are ordered, some not
- Some allow duplicate elements, some do not
- Collection Interface includes an iterator method - and Map interface includes methods to obtain a collection of values and a set of keys

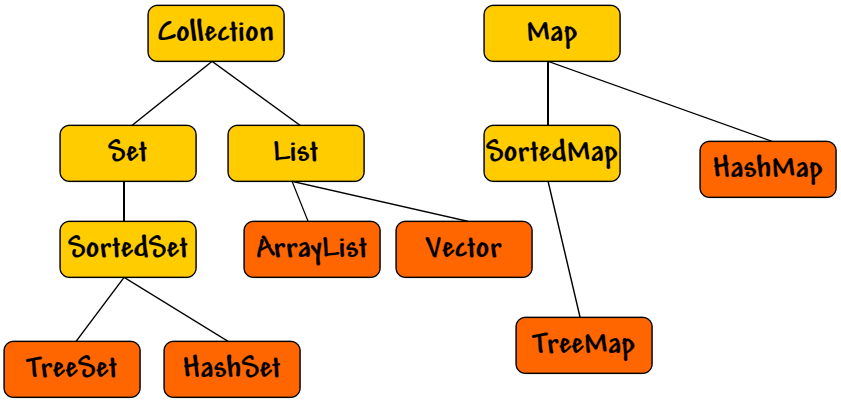
17

Interfaces

- Set - an unordered collection that does not allow duplicate entries
- List - an ordered collection that allows duplicate entries
- Map - an object that maps keys to values and does not allow duplicate keys
- SortedSet - set sorted in ascending order
- SortedMap - map sorted in ascending key order

18

Implementations (partial list)



Important to separate the declaration from the implementation

```
List l = new ArrayList();
```

19

Deconstruct forEach Code

```
<c:forEach var="p" items="${param}">
  <tr>
    <td> ${p.key} </td>
    <td> ${p.value} </td>
  </tr>
</c:forEach>
```

- What is param? What is the type of param?
- What is p? What is the type of p?
- What is p.key? What is the type of p.key?
- What is p.value? What is the type of p.value?

What if we replace param with paramValues?

20

items Attribute

- The items attribute can be
  - any collection (List, LinkedList, ArrayList, Vector, Stack, Set),
  - any Map (HashMap, Hashtable, Properties, Provider, Attributes),
  - csv String
  - arrays of object, and
  - arrays of primitive.

What is the type of the loop index when you iterate over an array of primitive (e.g., int)?

Hint: the item attribute is a scoped variable?

21

When items Attribute is a Map

- If the items attribute is of type Map, the loop index will be of type Map.Entry
- A Map.Entry object has 2 properties
  - key - key under which this item is stored
  - value - value corresponding to the key

22

Example - Java One Headers

Http Headers

Header Name	Header Value
1 content-type	application/x-www-form-urlencoded
2 cookie	test=123; JSESSIONID=9840796c1e71702f40530e560649
3 connection	keep-alive
4 accept-language	en-us,en;q=0.5
5 host	localhost:8080
6 content-length	304
7 accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
8 user-agent	Mozilla/5.0 (Windows NT 6.1; rv:10.0.2) Gecko/20100101 Firefox/10.0.2
9 accept-encoding	gzip, deflate

- The http headers can be accessed and displayed using an JSTL implicit variable

23

Example - Headers

```
<h2>Http Headers</h2>
<table>
  <tr>
  <th>Header Name</th>
  <th>Header Value</th></tr>
   <c:forEach var="h" items="${header}">
  <tr>
    <td class="name"> ${h.key} </td>
    <td> ${h.value} </td>
  </tr>
   </c:forEach>
</table>
```

Specifies the value property of h (the value of the current http header in the iteration)

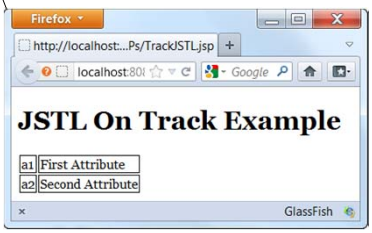
24

Are We on Track?

- Using your laptop, code a JSP that displays the session attribute keys and values
  - Use a scriptlet to store 2 attributes (with values) in your session object
  - Use JSTL and EL to display the result in a table
    Hint: your answer should work for any number of attributes

You can use the following CSS to display a simple cell border

```
td {border: 1px solid black;}
```



25

Were We on Track?

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>    <head>
...
        <style>td {border: 1px solid black;}</style>
    </head>    <body>
        <h1>JSTL On Track Example</h1>
        <% session.setAttribute("a1", "First Attribute");
            session.setAttribute("a2", "Second Attribute");
            %>
            <table>
                <c:forEach var="a" items="${sessionScope}" >
                <tr>
                    <td>${a.key}</td> <td>${a.value}</td>
                </tr>
                </c:forEach>
            </table>
    </body>
</html>
```

26

c:if

- Allows you to conditionally include, or process, a piece of the page
- This example includes a personal greeting if the user is a repeat visitor, as indicated by the presence of a cookie with the user's name

```
<c:if test="${!empty cookie.userName}">
    Welcome back ${cookie.userName.value}
</c:if>
```

27

c:forEach

- You can refer to the iteration count in a foreach iteration

A LoopTagStatus object (contains properties of the loop)

```
<table>
  <tr>
   <th></th>
  <th>Header Name</th>
  <th>Header Value</th></tr>
   <c:forEach var="h" items="${header}" varStatus="i" >
  <tr>
    <td> ${i.count} </td>
       <td class="name">${h.key}</td>
    <td>${h.value}</td>
  </tr>
   </c:forEach>
</table>
```

28

Example

- Display a table of ServletContext attributes (name and value)

29

Example - ServletContext

```
<h2>ServletContext</h2>
<table>
   <thead>
   <tr>
        <th>Parameter Name</th>
        <th>Parameter Value</th></tr></thead>
   <c:forEach var="p" items="${applicationScope}">
   <tr>
        <td class="name">${p.key}</td>
        <td><c:out value="${p.value}"
              default="no value for attribute"/></td>
   </tr>
   </c:forEach>
</table>
```

How would you display a table of request attributes?

30

Example

- Display a table of cookies (names, values, and max age)

A value of -1 (default) for maxAge indicates that the cookie will persist until browser shutdown

Cookies

Cookie Name	Cookie Value	Max Age
JSESSIONID	9840796c1e71702f40530e560649	-1
test	123	-1

31

Example - Cookies Display

```
<h2>Cookies</h2>
<table><tr><th>Cookie Name</th>
<th>Cookie Value</th><th>Max Age</th></tr>
<c:forEach var="cm" items="${cookie}">
  <tr>
    <td class="name">${cm.key}</td>
    <td>${cm.value.value}</td>
    <td>${cm.value.maxAge}</td>
  </tr>
</c:forEach>
</table>
```

What is the type of cm.value?

32

Using c:set for Scoped Variables

- Allows you to set scoped variables in your JSP
  - var - set attribute value in a shared scope
  - Value - the Object value in the Key/value pair

```
<c:set var="Fido" value="${person.dog}" />
```

You can also specify a scope (and a body)

33

Using c:set for Beans and Maps

- Allows you to do assignments in your JSP
  - target - set attribute values
  - target - set bean properties or map values

```
<c:set target="${PetMap}"
       property="dogName" value="Clover" />
```

Target must not be null

If target is a bean, sets the value of the property "dogName"

If target is a Map, sets the value of a Key named "dogName"

34

c:remove

- You can remove a scoped attribute using the set tag (when value evaluates to null), however it doesn't seem intuitive to remove an object with a set
- Use the c:remove

```
<c:remove var="Fido" scope="request" />
```

Scope is optional, but the attribute will be removed from all scopes if scope is not specified

35

To Include Content in your JSP

- Include directive — Static (translation time)

```
<%@ include file="Header.html" %>
```

- jsp:include action — Dynamic (request time)

```
<jsp:include page="Header.jsp" />
```

- c:import JSTL tag — Dynamic (request time)

```
<c:import url="http://www.cs.sunysb.edu ... " />
```

These are all useful to import html fragments (e.g., headers and footers) common to multiple pages

c:import can access a resource outside the current Web container

36

Other C Library JSTL Tags

- c:choose / c:when / c:otherwise
- c:url - Allows URL rewriting when cookies are disabled

In the exam, you should be able to use JSTL tags you have had no previous experience with

37

Assignment 4b

- Write a JSP that you can use for testing in your project
- The JSP should display
  - The attribute names and values contained in the shared objects (request, session, servlet context)
  - The values of all the http headers
  - The form data set
  - The property values of your bean
- All of the items above should be contained in HTML tables, similar to the examples given in class.
- To display the bean properties you may need to use Java reflection. Using a scriptlet is OK for this assignment

38

Reflection

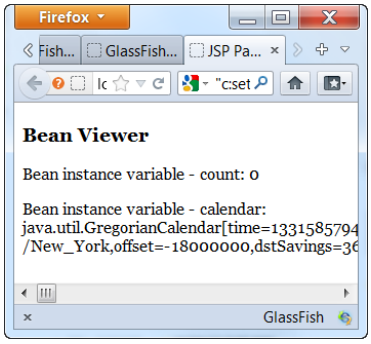
- Mechanism for discovering data about a program at runtime
  - Fields
  - Methods
  - Constructors
- You can
  - Operate on the fields you discover
  - Create an instance of a class whose name is known only at runtime
  - Get and set the value of an object's public fields
  - Invoke a method on an object

You can only directly use the public methods

39

Bean Property Analysis

```
public class CountBean implements java.io.Serializable  {
  private int count = 0;
 private Calendar calendar =
   new GregorianCalendar();
  public int getCount() {
    return (count);
  }
public void setCount(int newCount) {
    this.count = newCount;
  }
 public Calendar getCalendar() {
        return (calendar);
    }
    public void setCalendar(Calendar newCalendar) {
        this.calendar = newCalendar;
    }
}
```



This example will dynamically determine the bean properties and display them

40

Bean Viewer JSP Fragment

```
<h3>Bean Viewer</h3>
  <%@ page import="java.lang.reflect.*" %>
  <jsp:useBean id="bean" class="lectures.CountBean3" />
  <jsp:setProperty name="bean" property="count" value="0" />
  <%
    Class b = Class.forName("lectures.CountBean3");
    Field[] fields = b.getDeclaredFields();
    for (Field f : fields) {
      String name = f.getName();
      out.println("<p>");
      String camelName = "get" +
        (name.substring(0,1).toUpperCase() + name.substring(1));
      Method m = b.getMethod(camelName);
      out.println("Bean instance variable - " + name + ":");
      out.println(m.invoke(bean));
      out.println("</p>");
    }
%>
```

Instantiates the bean

Better done with a custom tag

Gets public and private fields

Note use of the Java 5 for loop

We cannot access the fields directly since they are private

41