Session 21

Expression Languages

1

Reading

- Java EE 7- Chapter 9 in the Tutorial

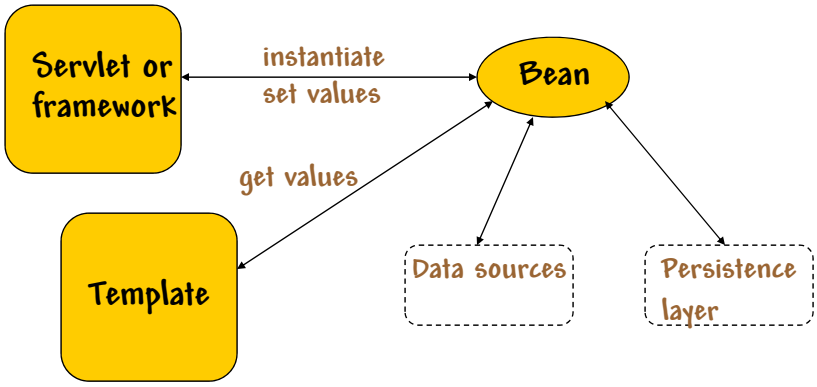
2

Lecture Objectives

- Understand how Expression Languages can simplify the integration of data with a view
- Know that an EL reference will look for the matching property in one of the shared objects
- Know the implicit objects available to you in EL
- Understand the type structure of Map objects
- Understand that objects containing matching get and set methods are considered to have properties

3

How Do We Access a Bean From a Template?



- Before we do anything we need to get the reference to the bean

4

Expression Languages

- Used in Java Web applications to embed expressions into template pages
- Stages
  - Expression Language (EL) - JSP 2.0
  - Expression Language 3.0 - compatible with JSF
  - Various derivative languages

Simple expressions:

- Variable Expressions: ${...}
- Selection Variable Expressions: *{...}
- Message Expressions: #{...}
- Link URL Expressions: @{...}

We'll start with JSP EL, and address extensions when we cover ThymeLeaf

5

JSP Expression Example-Counter

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>  <head>
...
  </head>
  <body>
    <jsp:useBean id="b" class="lectures.CountBean" scope="application" />
    <h1>JSP Counter</h1>
    <p>This JSP will print and increment the value of the counter</p>
    <p>The counter is initially: <%= b.getCount() %> </p>
    <p>The counter is now: <%= b.fetchAndAdd() %> </p>
    <p>The counter is now: <%= b.getCount() %> </p>
    <p>The counter is still:
       <jsp:getProperty name="b" property="count"/></p>
    <br/>
    <a href="http://localhost:8080/CSE336-
   2017/JSPs/lectures/JSPCounter.jsp">Re-count</a>
  </body>
</html>
```

But there is a way to do this without using Java in the JSP

6

EL in a Nutshell

- EL (Expression Language)
- Resembles JavaScript syntax
- EL expressions can be used in static text and in any tag attribute that can accept an expression
- Fully supported with JSP 2.0
- Syntax
  - ${ ... }

The value of an expression in static text is computed and inserted into the current output

The EL expression is contained within the brackets

7

EL Variables

```
${product}
```

- The web container evaluates a variable that appears in an expression by looking up its value
- For example, when evaluating the expression ${product}, the container will look for the name "product" in the page, request, session, and application scopes and will return its value (in the first scope in which it encounters the value)
- If "product" is not found, null is returned
- A variable that matches one of the implicit objects will return that implicit object instead of the variable's value

8

Remember Your Scopes

- Scopes are the Maps contained in various server objects
  - page - lasts as long as the page is active
  - request - lasts as long as the request is active
  - session - lasts as long as the session is active
  - application -contained in ServletContext object
- Order of search in the scopes is important

Scope names and order are one of
the few things you need to memorize

9

Counter Example Revisited

```
...
<p>The counter is initially:
      ${b.count} </p>
```

- Let's look at the example again
- How does EL find the bean?
- How does EL get the value of count?

10

EL Example-Counter

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Counter 2 (EL)</title>
    </head>
    <body>
    <jsp:useBean id="b" class="lectures.CountBean" scope="application" />
    <h1>JSP Counter</h1>
    <p>This JSP will print and increment the value of the counter</p>
    <p>The counter is initially: ${b.count} </p>
    <p>The counter is now: <%= b.fetchAndAdd() %></p>
    <p>The counter is now: ${b.count} </p>
    <br/>
    <a href="http://localhost:8080/CSE336-
  2017/JSPs/lectures/JSPCounter2.jsp">Re-count</a>
    </body>
</html>
```

11

What if Your Bean Contains Objects?

```
public class CountBean2 implements java.io.Serializable  {
  private int count = 0;
  private Calendar calendar = new GregorianCalendar();

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

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

  public void setCount(int newCount) {
    this.count = count;  }
  public Calendar getCalendar() {
    return (calendar);  }
  public void setCalendar(Calendar newCalendar) {
    this.calendar = newCalendar;  }
}
```

Can your JSP include a reference to the TimeZone property of the calendar?

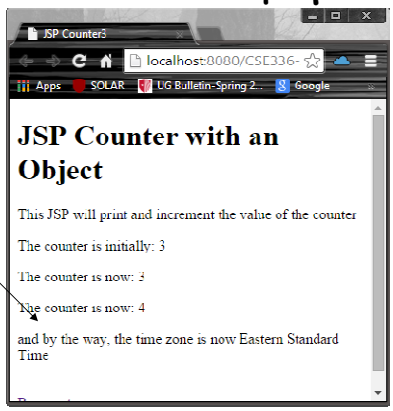
12

With EL You Can

```
<p>and by the way, the time zone is now
    ${b.calendar.timeZone.displayName} </p>
```

- The above JSP code will cause the browser to display:

More on the EL syntax later



13

EL Syntax

This is the dot (.) operator

```
Today is: ${b.d.hours}
```

Either a map key or a bean property (depending on whether b is a Map or a bean)

- EL Implicit Object
  - pageScope
  - requestScope
  - sessionScope
  - applicationScope
  - param
  - paramValues
  - header
  - headerValues
  - cookie
  - initParam
  - pageContext

or

- Attribute name
  - In pageScope
  - In requestScope
  - In sessionScope
  - In applicationScope

This works if an object exhibits "bean-like" behavior

A variable on the left side of a dot is either a Map (something with keys) or a bean (something with properties)

b.d evaluates to either a Map value or a bean property value (d is either a Map key or a bean property name)

14

Are We on Track?

- Code a JSP that displays the http method name used to call the JSP
- Steps
  - Create a default JSP in NetBeans
  - Use a scriptlet to store your request object in the session as an attribute
  - Use EL to display the result
  - Run the JSP

Remember to use your JSP implicit names in your scriptlet

Scriptlet syntax is <% ... %>

Remember, your request object exhibits "bean-like" behavior

Are We on Track for EL?

GET

15

Were We on Track?

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Are We on Track for EL?</h1>
        <% session.setAttribute("r", request); %>
        <p>
            ${r.method}
        </p>
    </body>
</html>
```

Without setting the session attribute:

```
${pageContext.request.method}
```


16

Map

- A Map is an object that maps keys to values
  - Cannot contain duplicate keys
  - Each key can map to at most one value
  - Contains a set of Map.Entry objects
- A Map.Entry object has 2 properties
  - key - Object representing the key under which this item is stored
  - value - Object representing the value corresponding to the key

A Map is an object that implements the Map interface and is instantiated as a HashMap, Hashtable, TreeMap, etc.

17

EL Implicit Objects

- The EL implicit objects are not the same as the JSP implicit objects (except pageContext)
- Examples:

Notice the use of plurals

  - sessionScope is a Map of session attributes
  - param is a Map of a key to a request parameter
  - paramValues is a Map of a key to request parameters (with possibly more than one value per name)

When do you have an HTML parameter with more than one value?

- All but one EL implicit object is a Map

How do we access paramValues in EL?

18

EL [] Operator

- The [] operator is more powerful than the dot operator
- These EL expressions are equivalent:

```
${header.host}  <⟺>  ${header["host"]}
```

- Are these EL expressions equivalent?

```
${header.User-Agent}  <⟺?>  ${header["User-Agent"]}
```

When you use the dot operator, the identifier on the left can be a bean or a Map

When you use the [] operator, the identifier on the left can be either a bean, a Map, a List or an Array

When you use the dot operator, the name on the right must be a legal Java name

When you use the [] operator, the identifier on the right can be a number or an identifier that does not fit the Java naming rules

19

EL [] Operator

- Meaning of a String parameter in []
  - Map - MapEntry key
    (i.e., name in one of the name value pairs)
  - Bean - bean property
  - Array - index into the array
  - List - index into the list

```
${headerValues.Accept["0"]}  <⟺>  ${headerValues.Accept[0]}
```

The array index is coerced to an int

If the parameter is not a String, the parameter is evaluated (e.g. check shared objects)

20

Example

- Q: What is the text plus EL that is equivalent to:

```
Host is: <%= request.getHeader("host") %>
```

- Answer:

```
Host is: ${header["host"]}
```

or

```
Host is: ${header.host}
```

What is the meaning of

```
Host is: ${header[host]}
```

Hint: header[host] is not the same as header["host"]

21

pageContext Implicit Object

- The pageContext implicit EL object refers to the real pageContext object
- You can treat it as you would a bean ("bean-like" behavior)
- What is the meaning of

```
${pageContext.request.method}
```

What does the above expression evaluate to?

What does `${requestScope.method}` evaluate to?

22

Example

- Display the value of the cookie with name x
- JSP

```
<% Cookie[] cookies = request.getCookies();
   for (Cookie c: cookies)
     if ((c.getName()).equals("x")) {
       out.println(c.getValue());
       }
   }  %>
```

- EL

This is a Map of cookieName/cookieObject

```
${cookie.x.value}
```

This is a Cookie object, which looks like a bean. (Why?)

23

EL Method Expressions

- Used to invoke an arbitrary public method of a bean, which can return a result
- Can use the . and [] operators
- Parameters allowed (values or expressions, separated by commas)
- Example

```
<input name="email" size="23" class="textbox"
  type="text" value="${b.getEmail()}">
```

26

EL Literals

- The JSP expression language defines the following literals:
  - Boolean: true and false
  - Integer: same as in Java
  - Floating point: same as in Java
  - String: with single and double quotes; " is escaped as \", ' is escaped as \', and \ is escaped as \\.
  - Null: null

27

EL Operators

- In addition to the . and [] operators, EL provides:
  - Arithmetic: +, - (binary), *, / and div, % and mod, - (unary)
  - Logical: and, &&, or, ||, not, !
  - Relational: ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le. (comparisons can be made against other values, or against boolean, string, integer, or floating point literals)
  - Empty: The empty operator is a prefix operation that can be used to determine whether a value is null or empty.
  - Conditional: A ? B : C. Evaluate B or C, depending on the result of the evaluation of A.

Example: `${!empty cookie.userName}`

28

Ternary Operator

- x ? y : z
  - *Condition ? value_if_true : value_if_false*
- useful in initializing radio buttons, check boxes, and selection in drop downs.

Useful in a Java method return

```
Do you need hotel reservations? <br />
<input name="ihotel" value="Yes" type="radio"
      ${b.ihotel=="Yes" ? "checked='checked'" : ""} /> Yes
<input name="ihotel" value="No" type="radio"
      ${b.ihotel=="No" ? "checked='checked'" : ""} /> No
```

29

Have You Satisfied the Lecture Objectives?

- Understand how Expression Languages can simplify the integration of data with a view
- Know that an EL reference will look for the matching property in one of the shared objects
- Know the implicit objects available to you in EL
- Understand the type structure of Map objects
- Understand that objects containing matching get and set methods are considered to have properties

30