Session 11

Expression Language (EL)

1

Reading

- Reading
  - Head First - pages 368-401
  - Java EE 6 - Chapter 6 in the Tutorial
  docs.oracle.com/javaee/6/tutorial/doc/gjddd.html
- Reference
  - JSTL Reference (Chapter 3 and Appendix A) - link on CSE336 Web site (References Section)
  - JSP 2.1 Specification - link on CSE336 Web site (References Section) - This document contains a good description of EL (Chapter 2)
  - EL Specification -jsp.java.net/spec/jsp-2_1-fr-spec-el.pdf
  - JSTL
  www-128.ibm.com/developerworks/java/library/j-jstl0211.html
  - JSP 2.0
  www.onjava.com/pub/a/onjava/2003/11/05/jsp.html

Some of the reading material refers to JSF

© Robert Kelly, 2001-2012

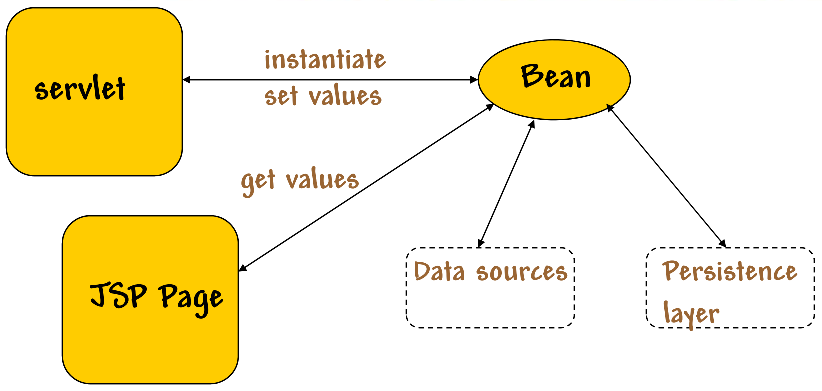
2

Lecture Objectives

- Understand how EL can reduce the use of Java (e.g., scriptlets and JSP Expressions) in JSPs
- Understand how to use EL to reference object properties
- Understand how to combine references so that you can obtain properties of properties
- 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

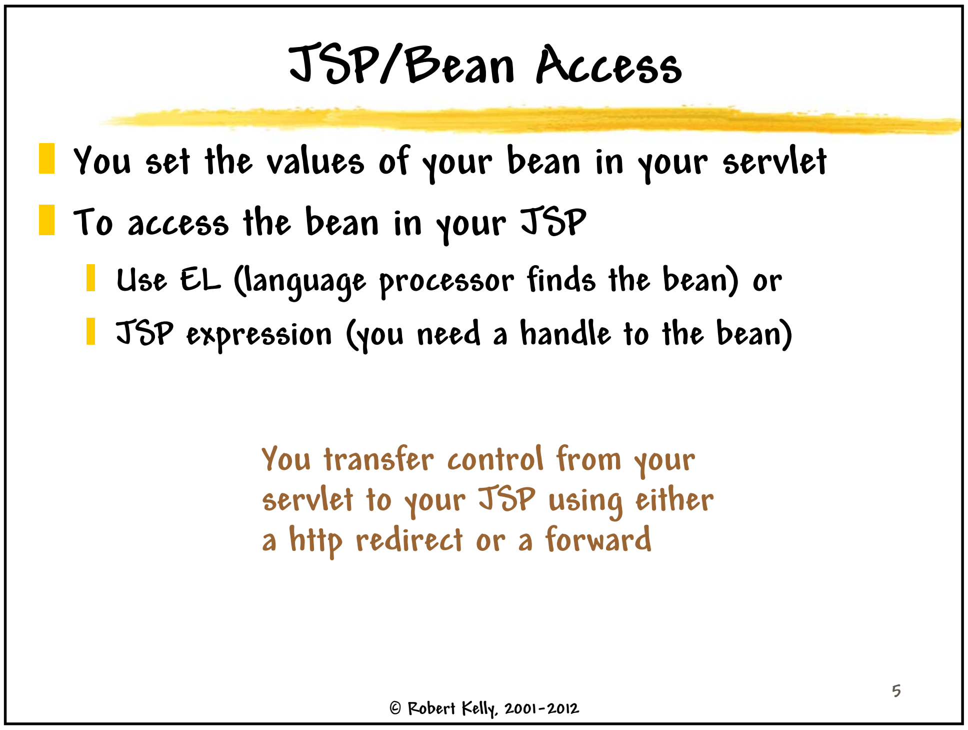
© Robert Kelly, 2001-2012 3

How Do We Access a Bean From a JSP?



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

© Robert Kelly, 2001-2012 4



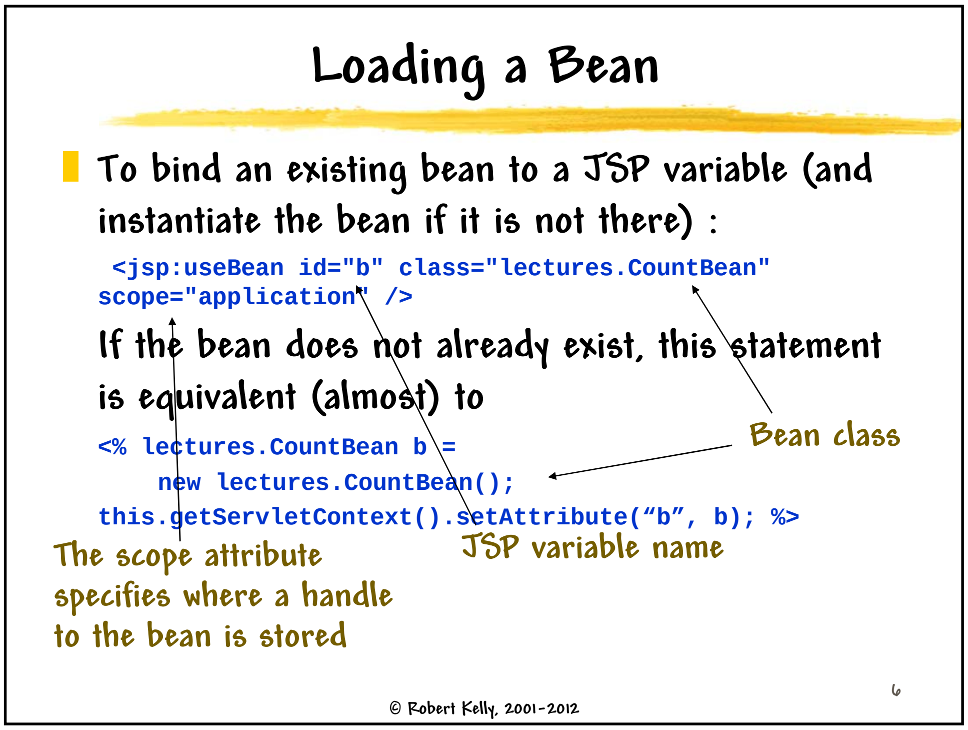
JSP/Bean Access

- You set the values of your bean in your servlet
- To access the bean in your JSP
  - Use EL (language processor finds the bean) or
  - JSP expression (you need a handle to the bean)

You transfer control from your servlet to your JSP using either a http redirect or a forward

© Robert Kelly, 2001-2012

5



Loading a Bean

- To bind an existing bean to a JSP variable (and instantiate the bean if it is not there) :

```
 <jsp:useBean id="b" class="lectures.CountBean"
scope="application" />
```

If the bean does not already exist, this statement is equivalent (almost) to

```
<% lectures.CountBean b =
    new lectures.CountBean();
this.getServletContext().setAttribute("b", b); %>
```

Bean class

JSP variable name

The scope attribute specifies where a handle to the bean is stored

© Robert Kelly, 2001-2012

6

CountBean

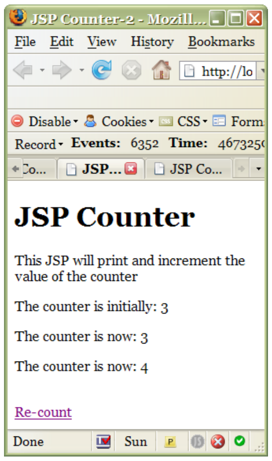
```
public class CountBean implements Serializable
   {
  private int count = 0;

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

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

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

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


JSP Counter-2 - Mozill...
File Edit View History Bookmarks
http://lo
Disable Cookies CSS Form
Record Events: 6352 Time: 467325
Co... JSP... JSP Co...
Done Sun

JSP Counter

This JSP will print and increment the value of the counter

The counter is initially: 3

The counter is now: 3

The counter is now: 4

Re-count

7

© Robert Kelly, 2001-2012

Example-Counter

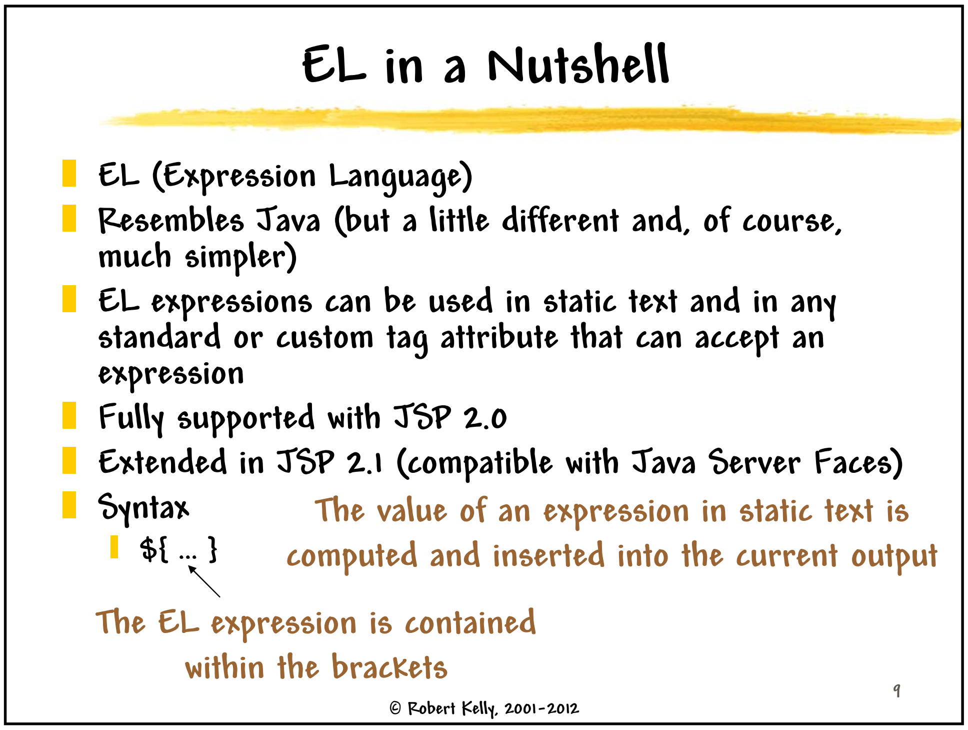
```
<%@ page language="java" contentType="text/html" %>
<html><head>
<title>JSP Counter</title>
<%@ page import="lecturecode.CountBean" %>
</head>
<body>
  <jsp:useBean id="b" class="lecturecode.CountBean"
   scope="application" />
<h1>JSP Counter</h1>
<p>This JSP will print and increment the value of 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/CodeCSE336/JSPs/JSPCounter4.jsp">
   Re-count</a>
</body></html>
```

Since we set up a reference to "b" in the useBean tag, this has the same effect as `<%= b.getCount() %>`

8

© Robert Kelly, 2001-2012



EL in a Nutshell

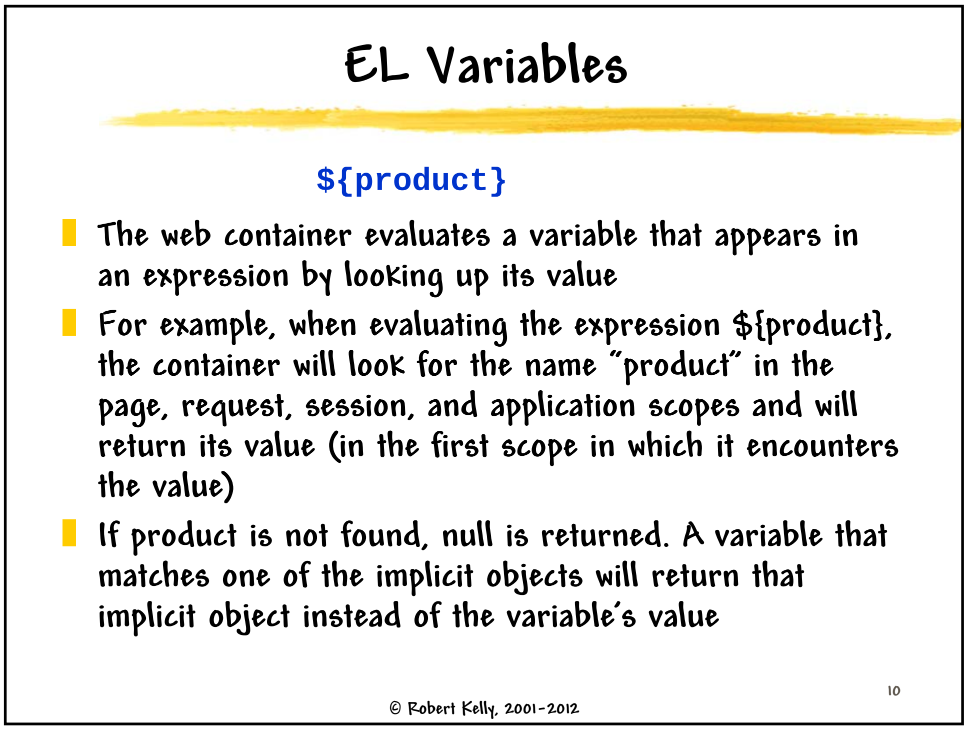
- EL (Expression Language)
- Resembles Java (but a little different and, of course, much simpler)
- EL expressions can be used in static text and in any standard or custom tag attribute that can accept an expression
- Fully supported with JSP 2.0
- Extended in JSP 2.1 (compatible with Java Server Faces)
- 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

9

© Robert Kelly, 2001-2012



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

10

© Robert Kelly, 2001-2012

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?

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;  }
}
```

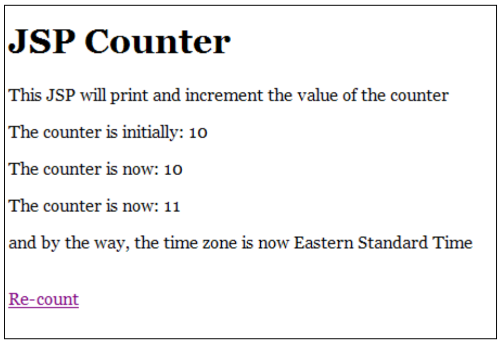
Can your JSP include a reference to the TimeZone property of the calendar?

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



JSP Counter

This JSP will print and increment the value of the counter

The counter is initially: 10

The counter is now: 10

The counter is now: 11

and by the way, the time zone is now Eastern Standard Time

Re-count

© Robert Kelly, 2001-2012

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

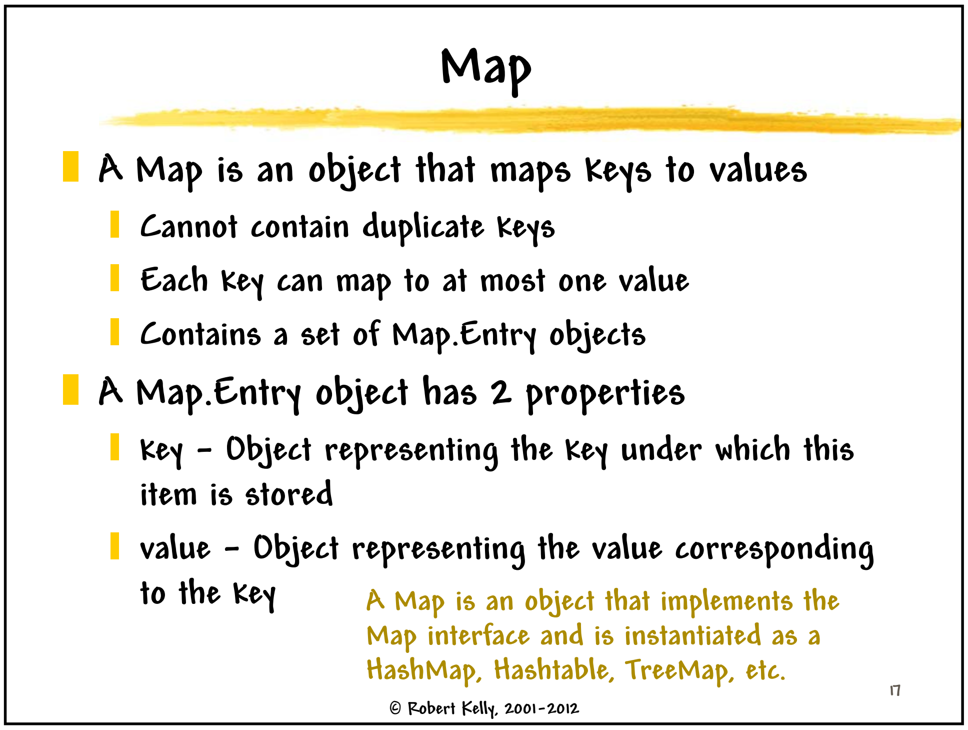
or

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

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)

© Robert Kelly, 2001-2012



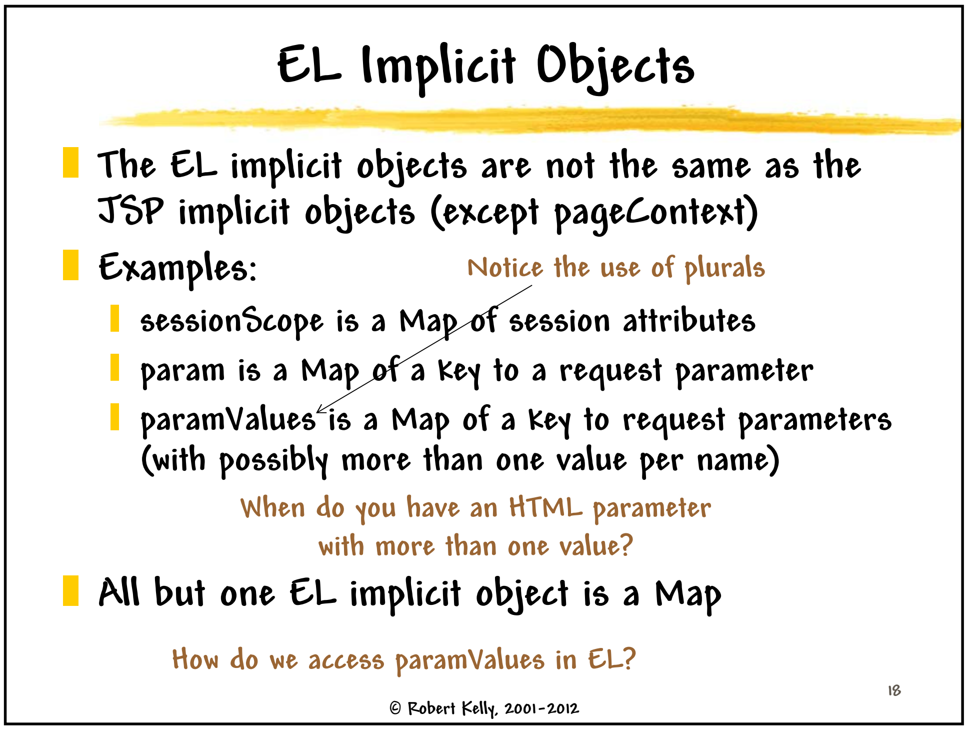
Map

- 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

© Robert Kelly, 2001-2012



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)

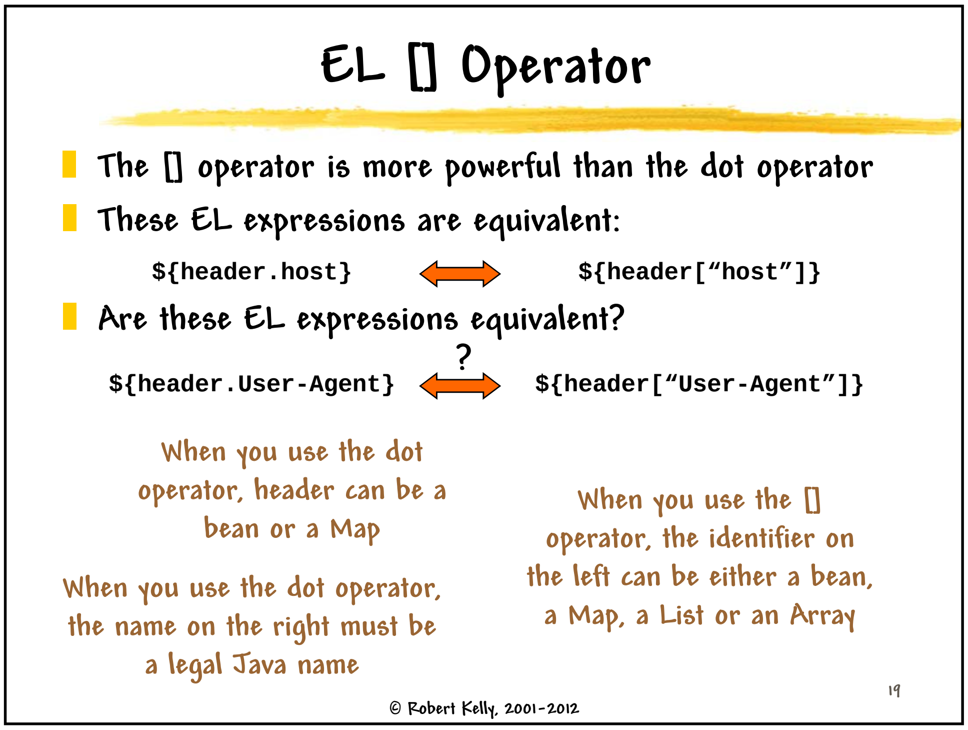
When do you have an HTML parameter with more than one value?

- All but one EL implicit object is a Map

How do we access paramValues in EL?

18

© Robert Kelly, 2001-2012



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"]}
```

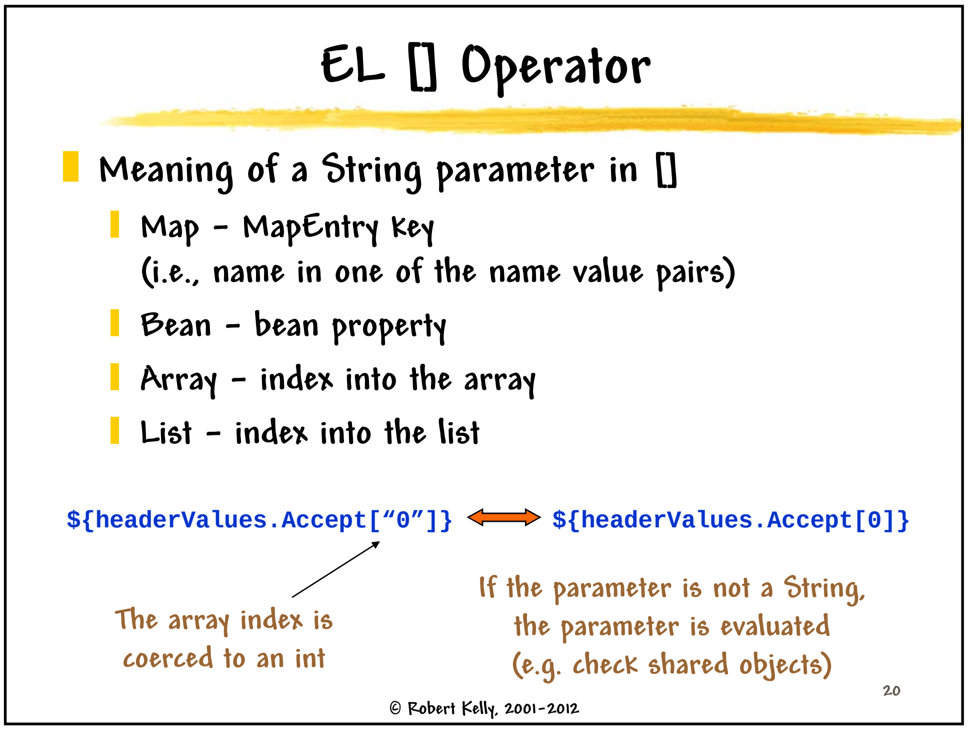
When you use the dot operator, header can be a bean or a Map

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 left can be either a bean, a Map, a List or an Array

19

© Robert Kelly, 2001-2012



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

© Robert Kelly, 2001-2012

Example (from Head First)

- 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"]

© Robert Kelly, 2001-2012

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?

© Robert Kelly, 2001-2012

Example (from Head First)

- 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?)

© Robert Kelly, 2001-2012

EL Literals

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

© Robert Kelly, 2001-2012

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}`

© Robert Kelly, 2001-2012

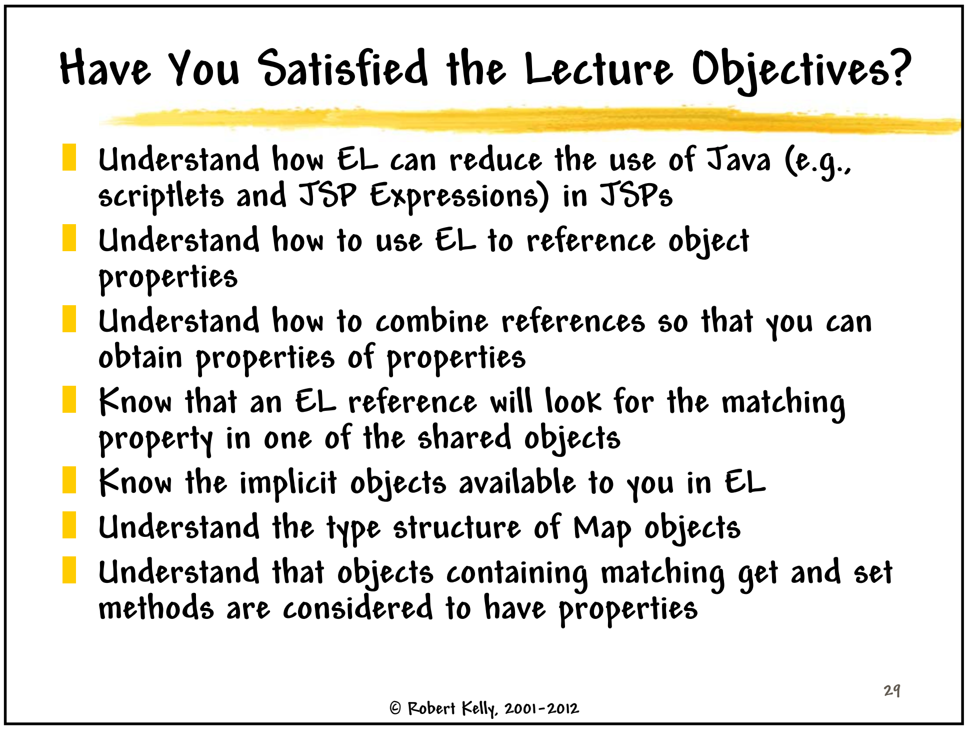
Ternary Operator

- The Ternary operator (x ? Y : z) is useful in initializing radio buttons, check boxes, and selection in drop downs.

Have you used the ternary operator in Java?

```
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
```

© Robert Kelly, 2001-2012



Have You Satisfied the Lecture Objectives?

- Understand how EL can reduce the use of Java (e.g., scriptlets and JSP Expressions) in JSPs
- Understand how to use EL to reference object properties
- Understand how to combine references so that you can obtain properties of properties
- 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

© Robert Kelly, 2001-2012

29