Expression Language - EL

LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

- Introduction to EL
- Scope Variables
- Implicit Objects
- Operators

Expression Language

- Shorthand Language which helps to get data from JavaBeans, arrays, maps and lists that have been stored as attributes
- Simplifies the presentation layer by replacing scripting elements and useBean Action Tags
- EL expressions are within curly braces, and prefixed with the dollar sign

```
${expression}
```

- EL expressions can appear in html text or in JSP tag attributes

```
<UL>
    <LI>Name: ${expression1}
    <LI>Address: ${expression2}
</UL>

<jsp:include page="${expression3}" />
```

Comparison

A JSP that uses EL to access a User object named user that has been stored in the session object

```
<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td align="right">First name:</td>
        <td>${user.firstName}</td>
    </tr>
    <tr>
        <td align="right">Last name:</td>
        <td>${user.lastName}</td>
    </tr>
    <tr>
        <td align="right">Email address:</td>
        <td>${user.emailAddress}</td>
    </tr>
</table>
```

The same JSP using standard JSP tags

```
<jsp:useBean id="user" scope="session" class="business.User"/>
<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td align="right">First name:</td>
        <td><jsp:getProperty name="user" property="firstName"/></td>
    </tr>
    <tr>
        <td align="right">Last name:</td>
        <td><jsp:getProperty name="user" property="lastName"/></td>
    </tr>
    <tr>
        <td align="right">Email address:</td>
        <td><jsp:getProperty name="user" property="emailAddress"/></td>
    </tr>
</table>
```

Expression Language

- Advantages
  - Compact and easy to code and read
  - Easy to access nested properties of beans

    ${employee.address.city}
  - Allows to access collections such as Arrays, maps and lists
  - Handles null values better that standard Action
  - Helps to work with headers, cookies and context initialization parameters
  - Allows to perform calculation and comparison
- Disadvantages
  - It doesn't create a JavaBean
  - Doesn't provide a way to set the properties of bean

Accessing Scoped Variables

- Servlet can store attributes in Request, Session or ServletContext objects
- These scoped variables can be easily accessed using EL
- The sequence used to search the attribute is from smallest scope to largest scope
  page → Request → session → Application

Servlet

```
String userrole  = "admin";
String msg = "Item does not exist";
request.setAttribute("message", msg);
session.setAttribute("message", "test message");
session.setAttribute("role", userrole);
.. .. ..
```

JSP

```
<p> Message from Servlet : ${message}  </p>
<p> Role of User: ${role}  </p>
```

Accessing Bean Property – dot operator

- EL provides a simple dot notation for accessing bean properties

  `${attribute.property}`

  Attribute can be a JavaBean or Map

  property can be JavaBean property or a map-key

- To return the salary property of a scoped variable named employee

  `${employee.salary}`

- Employee bean has a address property. Address has a property named city. To access city

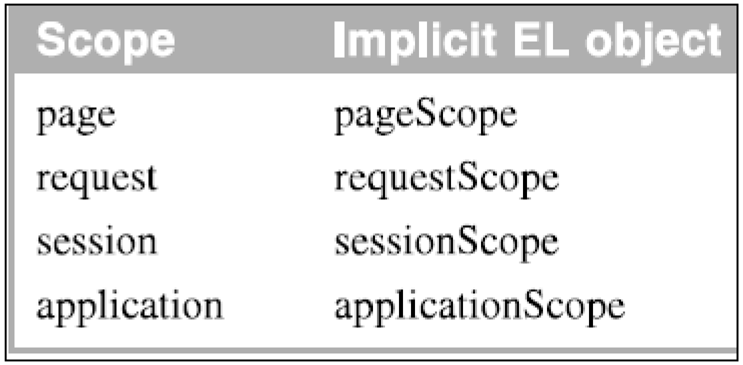
  `${employee.address.city}`

- jsp:useBean does not provide the capability access the property of a property

Implicit EL objects for scope

- Used to access an attribute in a particular scope
- Helpful in case of naming conflicts

```
${scope.attribute}
${scope.attribute.property}
```



Scope	Implicit EL object
page	pageScope
request	requestScope
session	sessionScope
application	applicationScope

Servlet Code

```
Date d1 = new Date();
request.setAttribute("date",d1);
User user1 = new User("Charles","ch@test.com");
session.setAttribute("user", user1);
```

JSP Code

```
<p> Current date is ${requestScope.date} <br>
        Hello ${sessionScope.user.name}
```

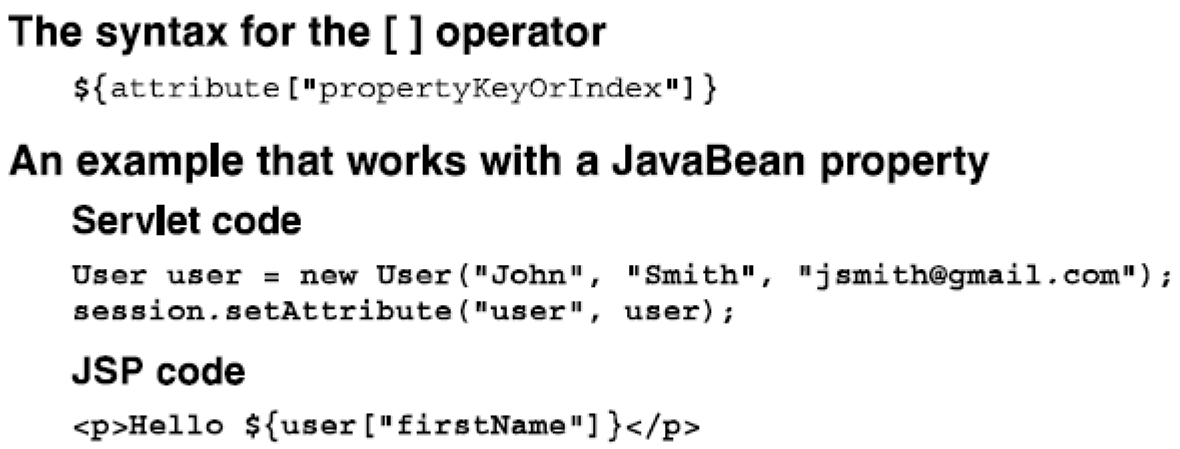
[ ] operator

- EL allows to replace dot notation with [] notation

  ${employee.salary} Can be replaced with

  ${employee[**“salary”**]}

  - computes the name of the property at request time
  - value inside the brackets can be a variable
  - very useful when accessing collections



The syntax for the [ ] operator

```
${attribute["propertyKeyOrIndex"]}
```

An example that works with a JavaBean property

Servlet code

```
User user = new User("John", "Smith", "jsmith@gmail.com");
session.setAttribute("user", user);
```

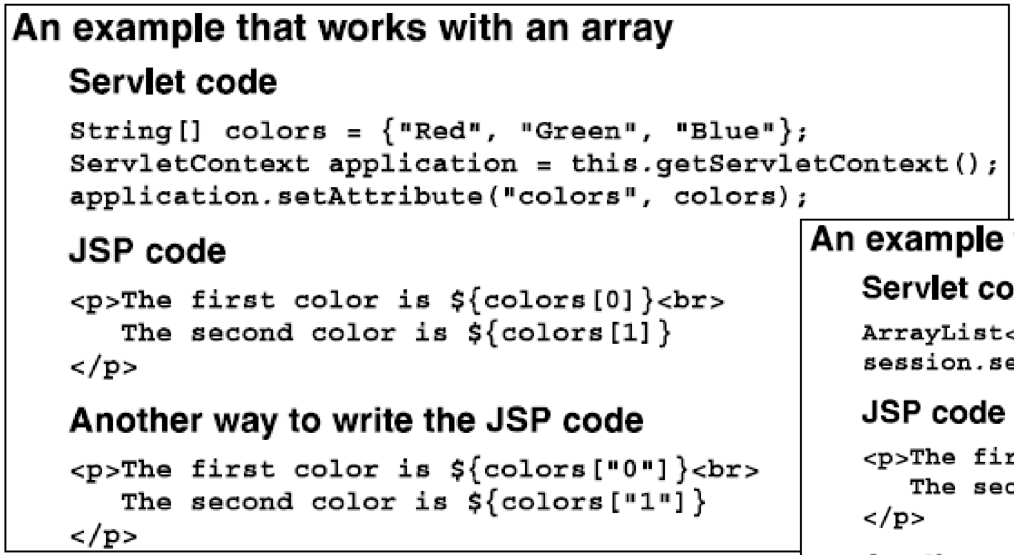
JSP code

```
<p>Hello ${user["firstName"]}</p>
```

Accessing Arrays and Lists

- For accessing collection
  ${attribute[entryName]}
  attribute is a Collection , entryname is the index

  Consider, nameList is an ArrayList of Customer names. To retrieve the value at the first index
  ${nameList[“0”]} or ${nameList[0]}



An example that works with an array

Servlet code

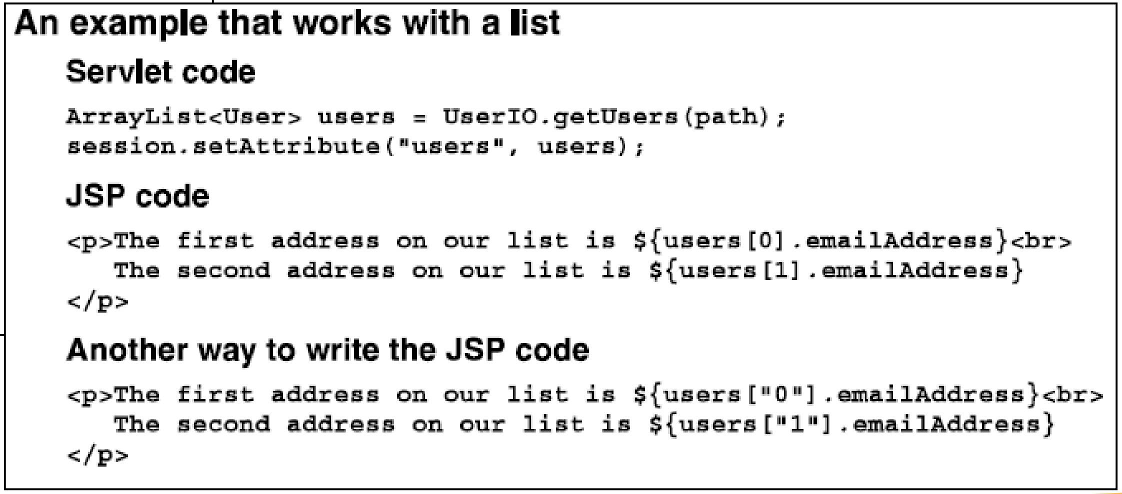
```
String[] colors = {"Red", "Green", "Blue"};
ServletContext application = this.getServletContext();
application.setAttribute("colors", colors);
```

JSP code

```
<p>The first color is ${colors[0]}<br>
   The second color is ${colors[1]}
</p>
```

Another way to write the JSP code

```
<p>The first color is ${colors["0"]}<br>
   The second color is ${colors["1"]}
</p>
```



An example that works with a list

Servlet code

```
ArrayList<User> users = UserIO.getUsers(path);
session.setAttribute("users", users);
```

JSP code

```
<p>The first address on our list is ${users[0].emailAddress}<br>
   The second address on our list is ${users[1].emailAddress}
</p>
```

Another way to write the JSP code

```
<p>The first address on our list is ${users["0"].emailAddress}<br>
   The second address on our list is ${users["1"].emailAddress}
</p>
```

More EL Implicit Objects

Following implicit objects are available to the EL in a JSP

- pageContext
  - Reference to the PageContext object of the current page
  - has access to request, response, session, out, and servletContext properties

    ${pageContext.session.id}

- param and paramValues
  - Map that returns the request parameter value or an array of request parameter values

    ${param.userID}

- initParam
  - Map that returns value for the specified context initialization parameter

    ${initParam.adminemail}

- cookie
  - Map that returns reference of specified cookie objects

    ${cookie.userCookie.value}

EL Implicit Objects

How to get parameter values from the request

An HTML form that has two parameters with the same name

```
<form action="addToEmailList" method="post">
    <p>First name: <input type="text" name="firstName"></p>
    <p>Email address 1: <input type="text" name="emailAddress"></p>
    <p>Email address 2: <input type="text" name="emailAddress"></p>
</form>
```

JSP code

```
<p>First name: ${param.firstName}<br>
   Email address 1: ${paramValues.emailAddress[0]}<br>
   Email address 2: ${paramValues.emailAddress[1]}
</p>
```

How to get an HTTP header

JSP code

```
<p>Browser MIME types: ${header.accept}<br><br>
   Browser compression types: ${header["accept-encoding"]}
</p>
```

EL Implicit Objects

How to work with cookies

Servlet code

```
Cookie c = new Cookie("emailCookie", emailAddress);
c.setMaxAge(60*60); //set its age to 1 hour
c.setPath("/"); //allow the entire application to access it
response.addCookie(c);
```

JSP code

```
<p>The email cookie: ${cookie.emailCookie.value}</p>
```

How to get a context initialization parameter

XML in the web.xml file

```
<context-param>
    <param-name>custServEmail</param-name>
    <param-value>custserv@murach.com</param-value>
</context-param>
```

JSP code

```
<p>The context init param: ${initParam.custServEmail}</p>
```

EL Operators

- EL provides operators for performing calculations and logic
- JSP is the View and its job is to render the response
- Shouldn't be used for calculations and logic

Arithmetic (5)

Addition:	**+**
Subtraction:	**-**
Multiplication:	*****
Division:	**/** *and* **div**
Remainder:	**%** *and* **mod**

Logical (3)

AND:	**&&** *and* **and**
OR:	**||** *and* **or**
NOT:	**!** *and* **not**

Relational (6)

Equals:	**==** *and* **eq**
Not equals:	**!=** *and* **ne**
Less than:	**<** *and* **lt**
Greater than:	**>** *and* **gt**
Less than or equal to:	**<=** *and* **le**
Greater than or equal to:	**>=** *and* **ge**

```
<p> PF : ${salary * 0.12} </p>    //salary is a request attribute
```

Examples of Operators

EL Operator	Result
${1 > (4/2)}	False
${3 <= 4.0}	true
${(10*10) ne 100}	false
${3 div 4}	0.75
${10 mod 4}	2
${emp.empName == null}	true if empName returns null
${cust.isActive == true}	True if isActive(boolean) is true
${empty emp.empName}	Returns true if value of empName is null or empty string
${condition?value1:value2}	If condition is true returns value1

Disable Scripting and EL

Disabling Scripting for Entire Application

```
<jsp-config>
        <jsp-property-group>
                <url-pattern>*.jsp</url-pattern>
                <scripting-invalid>true</scripting-invalid>
        </jsp-property-group>
</jsp-config>
```

Disabling EL for Entire Application

```
<jsp-config>
        <jsp-property-group>
                <url-pattern>*.jsp</url-pattern>
                <el-ignored>true</el-ignored>
        </jsp-property-group
</jsp-config>
```

Disabling EL for Single Page

```
<%@ page isELIgnored = “true” %>
```

SUMMARY

Expression Language - EL

SUMMARY

In this lesson, you've learned to:

- Introduction to EL
- Scope Variables
- Implicit Objects
- Operators