## Chapter 8

# How to use EL (Expression Language)

## Objectives

### Applied

1. Use EL in your JSPs to access the attributes and properties of JavaBeans, maps, arrays, and lists.
2. Use EL with the implicit EL objects to work with request parameters, request headers, cookies, context initialization parameters, and pageContext objects.
3. Disable or enable EL.
4. Disable or enable scripting.

### Knowledge

1. Describe the advantages and disadvantages of using EL.
2. Explain why you might want to disable EL or scripting.

## Code that accesses a User object stored in the session object

### EL

```
<label>Email:</label>
<span>${user.email}</span><br>
<label>First Name:</label>
<span>${user.firstName}</span><br>
<label>Last Name:</label>
<span>${user.lastName}</span><br>
```

### Standard JSP tags

```
<jsp:useBean id="user" scope="session" class="murach.business.User"/>
<label>Email:</label>
<span><jsp:getProperty name="user" property="email"/></span><br>
<label>First Name:</label>
<span><jsp:getProperty name="user" property="firstName"/></span><br>
<label>Last Name:</label>
<span><jsp:getProperty name="user" property="lastName"/></span><br>
```

## Advantages of EL

- EL has a more elegant and compact syntax than standard JSP tags.
- EL lets you access nested properties.
- EL lets you access collections such as maps, arrays, and lists.
- EL does a better job of handling null values.
- EL provides more functionality.

## Disadvantages of EL

- EL doesn't create a JavaBean if it doesn't already exist.
- EL doesn't provide a way to set properties.

## How to access an attribute

### Syntax

```
${attribute}
```

### Servlet code

```
Date currentDate = new Date();
request.setAttribute("currentDate", currentDate);
```

### JSP code

```
<p>The current date is ${currentDate}</p>
```

## How to access a property of an attribute

### Syntax

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

### Servlet code

```
User user = new User(firstName, lastName, email);
session.setAttribute("user", user);
```

### JSP code

```
<p>Hello ${user.firstName}</p>
```

## Sequence of scopes searched to find the attribute

| Scope | Description |
|---|---|
| `page` | The bean is stored in the implicit PageContext object. |
| `request` | The bean is stored in the HttpServletRequest object. |
| `session` | The bean is stored in the HttpSession object. |
| `application` | The bean is stored in the ServletContext object. |

## The dot operator with JavaBeans and maps

- A JavaBean is a special type of object that provides a standard way to access its *properties*.
- A *map* is a special type of collection that's used to store key/value pairs. A HashMap collection is a map.
- When you use the dot operator, code to the left of the operator must specify a JavaBean or a map. Code to the right of the operator must specify a JavaBean property or a map key.
- When you use this syntax, EL looks up the attribute starting with the smallest scope (page scope) and moving towards the largest scope (application scope).

## Implicit EL objects for specifying scope

| Scope | Object |
|---|---|
| `page` | `pageScope` |
| `request` | `requestScope` |
| `session` | `sessionScope` |
| `application` | `applicationScope` |

## How to explicitly specify request scope

### Syntax

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

### Servlet code

```
Date currentDate = new Date();
request.setAttribute("currentDate", currentDate);
```

### JSP code

```
<p>The current date is ${requestScope.currentDate}</p>
```

## How to explicitly specify session scope

### Syntax

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

### Servlet code

```
User user = new User(firstName, lastName, email);
session.setAttribute("user", user);
```

### JSP code

```
<p>Hello ${sessionScope.user.firstName}</p>
```

## Implicit EL objects for scope

- If you have a naming conflict, use the *implicit EL objects* to specify scope.
- Because all of the implicit EL objects for specifying scope are maps, you can use the dot operator to specify a key when you want to return the object for that key.

## Syntax for the [ ] operator

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

## [ ] operator with a JavaBean

### Servlet code

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

### JSP code

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

## How to use the [ ] operator 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>
```

## How to use the [ ] operator 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].email}<br>
   The second address on our list is ${users[1].email}
</p>
```

### Another way to write the JSP code

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

## [ ] operator (part 1)

- A *list* is a special type of collection such as an ArrayList<> that uses an index to retrieve an object stored in the collection.
- Although the [ ] operator can be used to work with JavaBeans and maps, it is commonly used to work with arrays and lists.
- With EL, quotation marks are required for specifying a property in a JavaBean or a key in a map, but quotation marks are optional when specifying an index of an array or a list.

## How to access nested properties

### Syntax

```
${attribute.property1.property2}
```

### Servlet code

```
Product p = new Product();
p.setCode("pf01");
LineItem lineItem = new LineItem(p, 10);
session.setAttribute("item", lineItem);
```

### JSP code

```
<p>Product code: ${item.product.code}</p>
```

## Another way to access nested properties

### Syntax

```
${attribute["property1"].property2}
```

### Servlet code

```
Product p = new Product();
p.setCode("pf01");
LineItem lineItem = new LineItem(p, 10);
session.setAttribute("item", lineItem);
```

### JSP code

```
<p>Product code: ${item["product"].code}</p>
```

## Nested properties

- If a JavaBean has a property that returns another JavaBean, you can use the dot operator to access nested properties.
- There is no limit to the number of nested properties that you can access with the dot operator.

## How to access an attribute within the [ ] operator

### Syntax

```
${attribute[attribute].property}
```

### Servlet code

```
HashMap<String, User> usersMap = UserIO.getUsersMap(path);
session.setAttribute("usersMap", usersMap);

String email = request.getParameter("email");
session.setAttribute("email", email);
```

### JSP code

```
<p>First name: ${usersMap[email].firstName}</p>
```

### JSP code that returns an empty string

```
<!-- this doesn't work because the attribute is enclosed in quotes -->
<p>First name: ${usersMap["email"].firstName}</p>
```

## How to use multiple [ ] operators

### Syntax

```
${attribute[attribute[index]].property}
```

### Servlet code

```
HashMap<String, User> usersMap = UserIO.getUsersMap(path);
session.setAttribute("usersMap", usersMap);

String[] emails = {"jsmith@gmail.com", "joel@murach.com"};
session.setAttribute("emails", emails);
```

### JSP code

```
<p>First name: ${usersMap[emails[0]].firstName}</p>
```

### JSP code that returns an empty string

```
<!-- this doesn't work because the attribute is enclosed in quotes -->
<p>First name: ${usersMap["emails[0]"].firstName}</p>
```

## [ ] operator (part 2)

- If the expression within the [ ] operator isn't enclosed within quotes, EL evaluates the expression. To start, EL checks to see if the expression is an attribute. Then, it attempts to evaluate the expression.
- If multiple [ ] operators exist, the expression is evaluated from the innermost [ ] operator to the outermost [ ] operator. As a result, you can nest as many [ ] operators as necessary.

## Other implicit EL objects

| Object | Description |
|---|---|
| `param` | Map that returns a value for specified request parameter name. |
| `paramValues` | Map that returns an array of values for specified request parameter name. |
| `header` | Map that returns the value for specified HTTP request header. |
| `cookie` | Map that returns the Cookie object for specified cookie. |
| `initParam` | Map that returns the value for specified parameter name in context-param element of web.xml file. |
| `pageContext` | Reference to the implicit pageContext object available from any JSP. |

## Implicit EL objects

- All of the implicit objects used by EL are maps, except for the pageContext object, which is a JavaBean.

## How to get parameter values from the request

### HTML with two parameters with the same name

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

### JSP code

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

## How to get an HTTP header

### JSP code

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

### The text a browser might display

```
MIME types: text/html,application/xml;q=0.9,image/webp,*/*;q=0.8
Compression types: gzip,deflate,sdch
```

## How to work with cookies

### Servlet code

```
Cookie c = new Cookie("emailCookie", email);
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>
```

## How to use the pageContext object

### JSP code

```
<p>HTTP request method: ${pageContext.request.method}<br>
   HTTP response type: ${pageContext.response.contentType}<br>
   HTTP session ID: ${pageContext.session.id}<br>
   HTTP contextPath: ${pageContext.servletContext.contextPath}<br>
</p>
```

### The text that a browser might display

```
HTTP request method: POST
HTTP response type: text/html
HTTP session ID: 4C1CFDB54B0339B53BE3AC8E9BADC0F5
HTTP servletContext path: /ch8email
```

## Arithmetic EL operators

| Operator | Alternative | Description |
|---|---|---|
| `+` | | Addition |
| `-` | | Subtraction |
| `*` | | Multiplication |
| `/` | `div` | Division |
| `%` | `mod` | Modulus (remainder) |

## Arithmetic EL operators (continued)

| Example | Result |
|---|---|
| `${1+1}` | `2` |
| `${17.5+10}` | `27.5` |
| `${2.5E3}` | `2500.0` |
| `${2.5E3+10.4}` | `2510.4` |
| `${2-1}` | `1` |
| `${7*3}` | `21` |
| `${1 / 4}` | `0.25` |
| `${1 div 4}` | `0.25` |
| `${10 % 8}` | `2` |
| `${10 mod 8}` | `2` |
| `${1 + 2 * 4}` | `9` |
| `${(1 + 2) * 4}` | `12` |
| `${userID + 1}` | `9 if userID equals 8; 1 if userID equals 0` |

## Relational operators

| Operator | Alternative | Description |
|---|---|---|
| `==` | `eq` | Equal to |
| `!=` | `ne` | Not equal to |
| `<` | `lt` | Less than |
| `>` | `gt` | Greater than |
| `<=` | `le` | Less than or equal to |
| `>=` | `ge` | Greater than or equal to |

## Relational operators (continued)

| Example | Result |
|---|---|
| `${"s1" == "s1"}` | `true` |
| `${"s1" eq "s1"}` | `true` |
| `${1 == 1}` | `true` |
| `${1 != 1}` | `false` |
| `${1 ne 1}` | `false` |
| `${3 < 4}` | `true` |
| `${3 lt 4}` | `true` |
| `${3 > 4}` | `false` |
| `${3 gt 4}` | `false` |
| `${3 <= 4}` | `true` |
| `${3 >= 4}` | `false` |
| `${user.firstName == null}` | `true if firstName returns a null value` |
| `${user.firstName == ""}` | `true if firstName returns an empty string` |
| `${isDirty == true}` | `true if isDirty is true,`<br>`false if isDirty is false,`<br>`false if isDirty is null` |

## Logical operators

| Operator | Alternative | Description |
|---|---|---|
| `&&` | `and` | And |
| `\|\|` | `or` | Or |
| `!` | `not` | Not |

### Example / Result

```
${"s1" == "s1" && 4 > 3}        true
${"s1" == "s1" and 4 > 3}       true
${"s1" == "s1" && 4 < 3}        false
${"s1" == "s1" || 4 < 3}        true
${"s1" != "s1" || 4 < 3}        false
${"s1" != "s1" or 4 < 3}        false
${!true}                        false
${not true}                     false
```

## Other operators

| Syntax | Description |
|---|---|
| `empty x` | Returns true if the value of x is null or equal to an empty string. |
| `x ? y : z` | If x evaluates to true, returns y. Otherwise, returns z. |

| Example | Result |
|---|---|
| `${empty firstName}` | `true if firstName returns a null value or an empty string` |
| `${true ? "s1" : "s2"}` | `s1` |
| `${false ? "s1" : "s2"}` | `s2` |

## Keywords you can use in expressions

| Keyword | Description |
|---|---|
| `null` | A null value |
| `true` | A true value |
| `false` | A false value |

## EL expressions

- For arithmetic expressions, you can use parentheses to control or clarify the order of precedence.
- In arithmetic expressions, EL treats a null value as a zero.
- In logical expressions, EL treats a null value as a false value.

## How to disable EL

### For a single page (a page directive)

```
<%@ page isELIgnored ="true" %>
```

### For the entire application (the web.xml file)

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

## How to disable scripting

### For the entire application (the web.xml file)

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

## How to disable EL or scripting

- For JSP 2.0 and later, the servlet container evaluates any text that starts with ${ and ends with } as an EL expression. Most of the time, this is what you want. If it isn't, you can ignore EL for a single JSP or for all JSPs in the entire application.
- If you want to remove all scripting from all JSPs in your application, you can modify the web.xml file so it doesn't allow scripting. Then, if you request a JSP that contains scripting, the servlet container displays an error page.