JSP Expression Language

Table of Contents

[EL Operators 3](#_Toc100576924)

[EL Identifiers 3](#_Toc100576925)

**Expression Language** (EL) was introduced in JSP 2.0 with the purpose of simplifying the process of accessing data from bean objects and implicit objects. EL provides simple ways to access attribute and parameter values. Note that **JSP EL Expressions** are not the same as JSP Expressions.

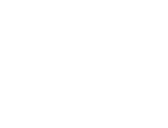
* JSP EL Expression: ${expression}
* JSP Expression: <%=javaExpression%>

EL is yet another attempt to reduce the amount of overlap between Java and HTML code.

Example:

<head>  
 <title>Expression Language Example</title>  
</head>  
<body>  
${1 < 2}  
${1 + 2 + 3}  
</body>

JSP



## EL Operators

There are five types of **operators** we can use inside an EL expressions, arithmetic operators (+, -, \*, /, %), grouping operators (()), logical operators, (&&, ||, !), relational operators (==, !=, <, >, <=, >=), the empty keyword and conditional operators (? :).

${(6 \* 5) + 5} *<%-- = 35 --%>*${(x >= min) && (x <= max)}  
${(empty name)} *<%-- returns true if name is "" or null --%>*

JSP

## EL Identifiers

**Identifiers** are used to refer to objects. User-made objects fall into one of the JSP scopes and are called **scoped variables**. However, there are **11 identifiers** that we cannot use to name our own objects. These are the **implicit objects** and they have fixed scopes.

|  |  |  |
| --- | --- | --- |
| **Category** | **Implicit Object** | **Description** |
| JSP | pageContext | Used to access JSP implicit objects. |
| Scopes | pageScope | A Map object associating names and values of page scoped attributes. |
| requestScope | A Map object associating names and values of request scoped attributes. |
| sessionScope | A Map object associating names and values of session scoped attributes. |
| applicationScope | A Map object associating names and values of application scoped attributes. |
| Request Parameters | param | Maps a request parameter name to a single String parameter value. |
| paramValues | Maps a request parameter name to an array of values. |
| Request Headers | header | Maps a request header name to a single header value. |
| headerValues | Maps a request header name to an array of values. |
| Cookies | cookie | A Map object storing the cookies accompanying the request by name. |
| Initialization Parameters | initParam | A Map object storing the context initialization parameters of the web application by name. |

Note that the JSP implicit objects that we have studied earlier on are not the same as the JSP EL identifier implicit objects.

Examples:

* ${pageContext.response} evaluates to the response JSP implicit object.
* ${param.name} is equivalent to request.getParameter("name")
* ${cookie.name.value} is equivalent to

if (cookie.getName().equals("name")) val = cookie.getValue();

Using both Java Expressions and Java EL Expressions:

*<%-- index.jps --%>*<html>  
<body>  
<%  
 application.setAttribute("author", "someone"); *<%-- JSP Expression --%>*  
 application.setAttribute("site", "www.example.com");  
%>  
</body>  
</html>

JSP

*<%-- display.jps --%>*<html>  
<body>  
Developer Name: ${applicationScope.author} <br> *<%-- JSP EL Expression --%>*  
Website: ${applicationScope.site}  
</body>  
</html>

JSP

Using JavaBeans:

*// employee.java*  
  
import java.io.*Serializable*;  
  
public class Employee implements *Serializable* {  
 private int id;  
 public Employee() {  
 }  
 public int getId() {  
 return id;  
 }  
}

JAVA

*<%-- display.jps --%>*<html>  
<body>  
<c:forEach items="${requestScope.employeeList}" var="emp">  
 <c:out value="${emp.id}" />  
</c:forEach>  
</body>  
</html>

JAVA

The for loop is being implemented using **JSTL**, which we will be studying next.