JSTL (JSP Standard Tag Library)

What is JSTL?

- Standard set of tag libraries
- Encapsulates core functionality common to many JSP applications
  - iteration and conditionals
  - XML
  - database access
  - internationalized formatting
- Likely to evolve to add more commonly used tags in future versions

Why JSTL?

- You don't have to write them yourself
- You learn and use a single standard set of tag libraries that are already provided by compliant Java EE platforms
- Vendors are likely to provide more optimized implementation
- Portability of your applications are enabled

JSTL Tag Libraries

- Core (prefix: c)
  - Variable support, Flow control, URL management
- XML (prefix: x)
  - Core, Flow control, Transformation
- Internationalization (i18n) (prefix: fmt)
  - Locale, Message formatting, Number and date formatting
- Database (prefix: sql)
  - SQL query and update
- Functions (prefix: fn)
  - Collection length, String manipulation

Declaration of JSTL Tag Libraries

- Core
  - <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
- XML
  - <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
- Internationalization (i18n)
  - <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
- Database (SQL)
  - <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
- Functions
  - <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
- http://download.oracle.com/javaee/5/tutorial/doc/bnakc.html

Expression Language

Expression Language

- The EL provides identifiers, accessors, and operators for retrieving and manipulating data resident in the JSP container.
  - Identifiers are used to reference data objects stored in the data center.
    - The EL has 11 reserved identifiers, corresponding to 11 EL implicit objects.
  - All other identifiers are assumed to refer to *scoped variables*.
  - Accessors are used to retrieve the properties of an object or the elements of a collection.
  - Literals represent fixed values -- numbers, character strings, booleans, or nulls.
  - Operators allow data and literals to be combined and compared.
- It is not a programming language, or even a scripting language.

Expression Language

- When combined with the JSTL tags, it enables complex behavior to be represented using a simple and convenient notation.
- EL expressions are delimited using a leading dollar sign ($) and both leading and trailing braces ({})
    - <c:out value="**${user.firstName}**"/>
- you can combine multiple expressions with static text to construct a dynamic attribute value through string concatenation.

  <c:out value="Hello **${user.firstName}** **${user.lastName}**"/>

Implicit objects

- pageContext
  - The PageContext instance corresponding to the processing of the current page
- pageScope
  - A Map associating the names and values of page-scoped attributes
- requestScope
  - A Map associating the names and values of request-scoped attribute
- sessionScope
  - A Map associating the names and values of session-scoped attributes
- applicationScope
  - A Map associating the names and values of application-scoped attributes
- param
  - A Map storing the primary values of the request parameters by name

Implicit objects

- paramValues
  - A Map storing all values of the request parameters as String arrays
- header
  - A Map storing the primary values of the request headers by name
- headerValues
  - A Map storing all values of the request headers as String arrays
- cookie
  - A Map storing the cookies accompanying the request by name
- initParam
  - A Map storing the context initialization parameters of the Web application by name

Accessors

- The EL provides two different accessors to access the properties of the objects
  - the dot operator (.)
  - the bracket operator ([])
- The dot operator is typically used for accessing the properties of an object.

  ```
  ${user.firstName}
  ```

- When the property being accessed is itself an object, the dot operator can be applied recursively.

  ```
  ${user.address.city}
  ```

Accessors

- The bracket operator is used to retrieve elements of arrays and collections.
- the index of the element to be retrieved appears inside the brackets

```
${urls[3]}
```

- For collections implementing the java.util.Map interface, the bracket operator looks up a value stored in the map using the associated key.

```
${commands["dir"]}
```

Accessors

- the bracket operator can be applied recursively.
  - This allows the EL to retrieve elements from multi-dimensional arrays, nested collections, or any combination of the two.
- the dot operator and the bracket operator are interoperable.

```
${urls[3].protocol}
```

Operators

- the EL also includes several operators to manipulate and compare data accessed by EL expressions.

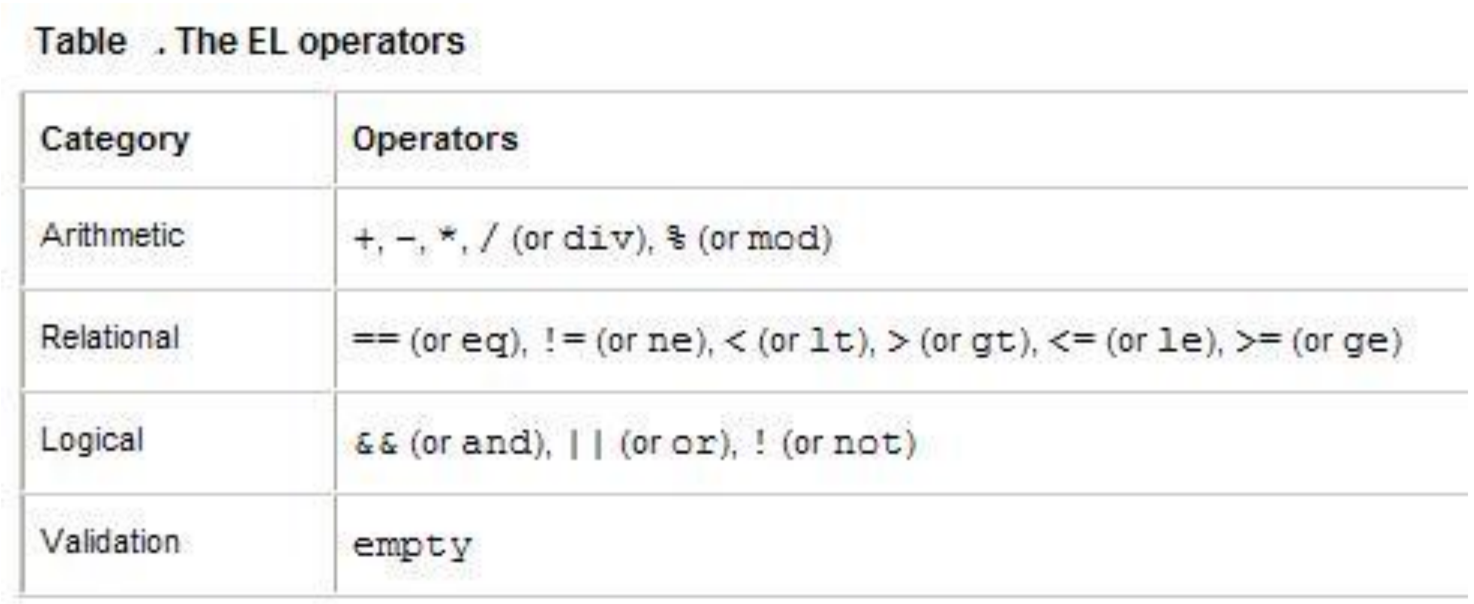

Table . The EL operators

Category	Operators
Arithmetic	+, −, *, / (or div), % (or mod)
Relational	== (or eq), != (or ne), < (or lt), > (or gt), <= (or le), >= (or ge)
Logical	&& (or and), || (or or), ! (or not)
Validation	empty

```
${item.price * (1 + taxRate[user.address.zipcode])}
${(x >= min) && (x <= max)}
```

- The final EL operator empty is particularly useful for validating data.
- The empty operator takes a single expression as its argument

```
${empty input}
```

EL operator precedence

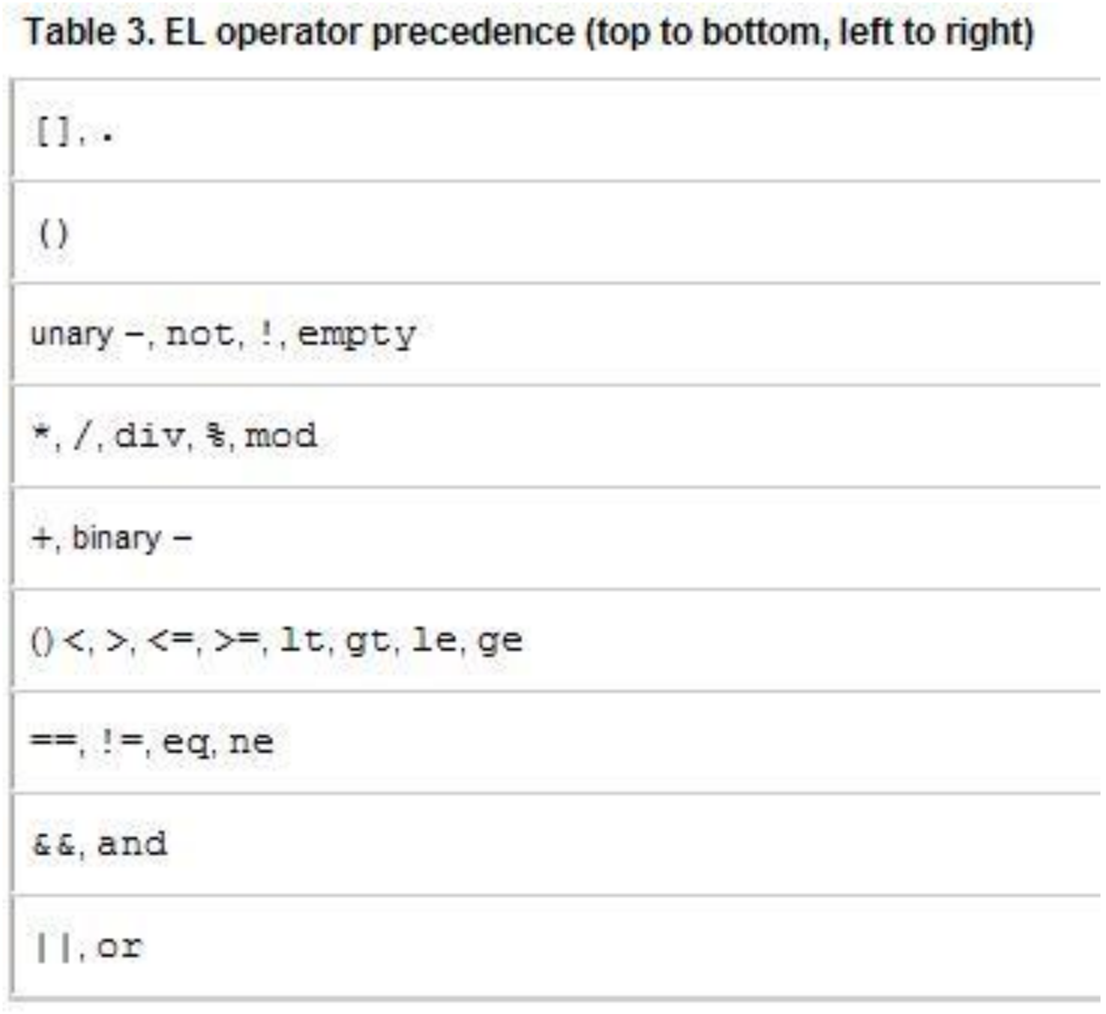

Table 3. EL operator precedence (top to bottom, left to right)

[], .
()
unary -, not, !, empty
*, /, div, %, mod
+, binary -
() <, >, <=, >=, lt, gt, le, ge
==, !=, eq, ne
&&, and
||, or