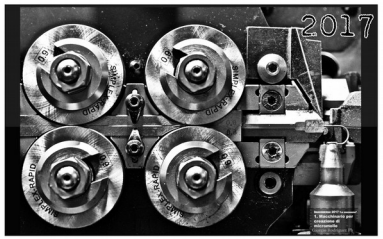
PARTE I
TECNOLOGIA JSP

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Biblioteca Universitaria
Scientifico Tecnologica
"Enzo Ferrari"

IMAGINERIA 2019



Studi ingegneria* e sei appassionato di fotografia?
Partecipa ad Imagineria 2019
Tema della 6. edizione "ingegneria&matematica"
Rendi i tuoi scatti parte del "Wall of fame" alla Biblioteca di Ingegneria!

concorso rivolto a tutti gli afferenti al DIEF

Modulo 1
Overview JSP

Java Server Pages (JSP)

- **Unisce elementi di markup (HTML) con frammenti di codice Java**
  - simile a quanto visto per PHP
  - compete con ASP
  - Pagine HTML con tag aggiuntivi
- **Estensione della tecnologia Servlet**
  - Al momento della prima invocazione, le pagine JSP vengono trasformate automaticamente da un compilatore JSP in servlet (classi di implementazione della pagina)
  - Le classi di implementazione sono conservate in una cache per migliorare le prestazioni

JSP

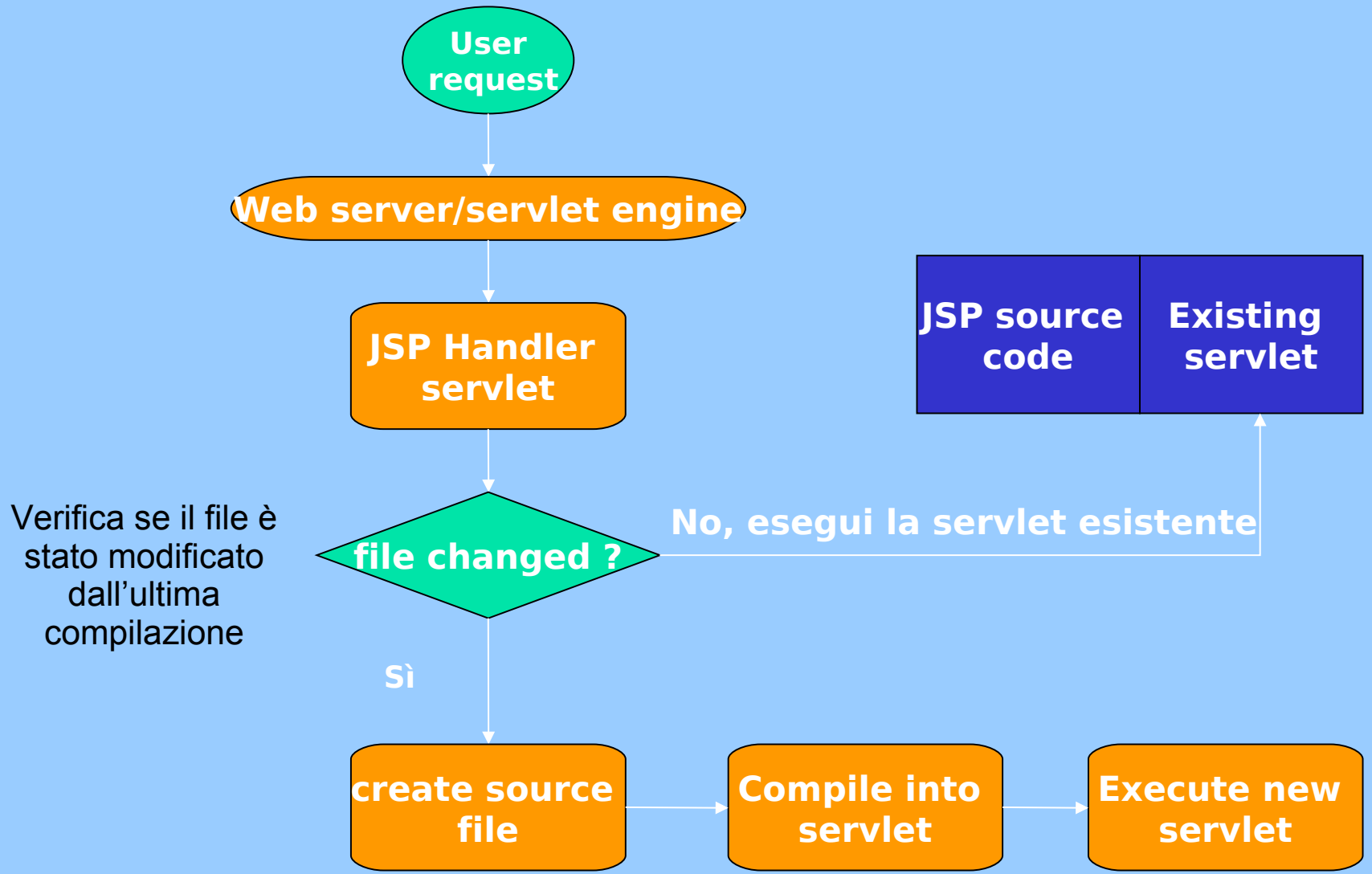
- **Condivide molte caratteristiche delle servlet:**
  - Integrazione con Java
  - Sicurezza
  - Portabilità
- **Maggiore semplicità**
- **Componenti:**
  - Engine
  - Container

Elementi di JSP

- **Contenuti statici**
  - Porzioni della pagina JSP che devono essere mantenute nella pagina generata dinamicamente
  - Scritte in HTML o XML
- **Direttive JSP**
  - Comandi rivolti al JSP engine
  - Sono eseguiti in fase di pre-processing prima dell'esecuzione degli script contenuti nella pagina
- **Scripting**
  - Frammenti di codice scritto (tipicamente) in Java e eseguiti dalla JVM

Architettura JSP

La traduzione (che produce un servlet) richiede un parsing del sorgente JSP allo scopo di individuare se vi sono direttive, dichiarazioni Java e scriptlet,...



Traduzione e compilazione

- **Parsing codice della pagina alla ricerca di elementi JSP**
- **Viene costruita una servlet**
- **In codice della servlet viene messo in:**

J2EE_HOME/repository/host/web/contextRoot/ jspfilename_jsp.java

- **JSPPage estende l'interfaccia Servlet**
  - jspInit()
  - jspDestroy()
  - _jspService(HttpServletRequest req, HttpServletResponse res)
  - I primi due metodi sono definiti dall'autore della pagina, il resto è prodotto dell'engine JSP

Traduzione e compilazione

- **Entrambe le fasi possono causare errori:**
  - In caso di errori nel codice JSP si solleva ParseException
    - La servlet generata resta incompleta
  - Se ci sono errori durante la compilazione si genera una JasperException (caso tipico di errori nel codice Java)

- **Ciclo di vita di una pagina JSP**
  - Caricamento della classe servlet generata
  - la classe viene istanziata
  - Viene chiamato jspInit()
  - Viene chiamato _jspService()

Elementi di una pagina JSP

- **Direttive**
  - Informazioni globali sulla pagina, statement di import, gestione di sessione
- **Dichiarazioni**
  - Variabili “di pagina” e dichiarazione di metodi
- **Scriptlets**
  - Frammenti di codice Java che vanno messi nel metodo _jspService
- **Espressioni**
  - Espressioni che definiscono l'output di una pagina

Direttive JSP <%@ page …%>

- **language=“java”**
- **extends=“package.class”**
  - Classe genitore della servlet implementata
- **session=“true | false”**
  - Usa informazioni di sessione nella pagina
- **import=“package.*, package.class”**
  - Importa un package nella servlet generata dalla pagina JSP

Direttive JSP <%@ page …%>

- **buffer=“none|sizekb”**
  - Dimensione dell'output stream buffer
- **errorPage=“filename”**
  - Pagina da mostrare in caso di errore
- **isErrorPage=“true|false”**
  - Indica se la pagina è una pagina di errore
- **isThreadSafe=“true|false”**
  - Indica se la pagina e' thread safe

Dichiarazioni JSP <%!...%>

- **Variabili e metodi utilizzabili in tutta la pagina**
- **I metodi *jspInit* e *jspDestroy* sono definiti in questi elementi**

JSP Scriptlets <%…%>

- **Qualsiasi blocco valido di codice java**
- **Il codice viene inserito nel metodo _jspService()**
- **Alcuni oggetti sono implicitamente definiti:**
  - request/response: oggetti di richiesta e risposta
  - pageContext: attributi della pagina e del contesto
  - session: sessione http
  - …

JSP Scriptlets <%…%>

- **Alcuni oggetti sono implicitamente definiti:**
  - …
  - application:il contesto della servlet, come ritornato da getServletConfig().getContext()
  - out: output stream (jspWriter)
  - config: configurazione della servlet
  - page: simile a “this”, fa riferimento alla servlet della pagina

Espressioni JSP <%=…%>

- **Strumento per inserire valori nel codice HTML**
- **Tutto quello che sta entro i delimitatori viene valutato, trasformato in stringa e mandato in output**

Esempio di elementi di scripting

```
<%! int count=0; %>
Il contatore vale: <%= ++count; %>
<% if (count >0) { %>
Il suo valore è maggiore di 0
<% } else { %>
Il suo valore è minore o uguale a 0
<% } %>
```

JSP vs Servlet: Esempio di servlet

```
public class HelloWorldServlet implements Servlet {
    public void service(ServletRequest req,
        ServletResponse resp) throws Exception {
        resp.defContentType(''text/html'');
        PrintWriter out = resp.getWriter();
        out.println(''<html><head><title>'');
        out.println(''Hello'');
        out.println(''</title></head>'');
        out.println(''<body><h1>Hello</h1>'');
        out.println(''It's'' + (new java.util.Date()).toString());
        out.println(''</body></html>);
    }
}
```

JSP vs. Servlet: Esempio di JSP

```
<html>
  <head><title>Hello</title></head>
  <body>
    <h1>Hello</h1>
It's <%= new java.util.Date().toString() %>
  </body>
</html>
```

JSP vs. ASP

- **Piattaforme supportate**
  - JSP: tutte le principali piattaforme
  - ASP: solo Microsoft
- **Linguaggio di base:**
  - JSP: Java
  - ASP: Jscript/VBScript
- **Supporto per oggetti distribiti**
  - JSP: J2EE
  - ASP: COM/DCOM → .Net
- **Interpretazione del codice:**
  - JSP: una volta sola (caching della servlet)
  - ASP: ad ogni chiamata (meccanismi di caching sono stati aggiunti nel passaggio a .Net)

Modulo 2
Hands on JSP

Esercitazioni JSP

- **Trasformare gli esempi di codice visti con la tecnologia servlet in pagine JSP**
- **Tre applicazioni:**
  - Data aggiornata ogni secondo
  - Calcolatrice
  - Contatore gestito a livello di sessione utente

Modulo 3
JSP Tag Library

Estensioni JSP

- **JSP prevede una serie di tag aggiuntivi per semplificare il lavoro del programmatore**
- **Tali tag sono racchiusi in librerie che possono essere inserite nelle pagine**
- **JSTL: JavaServer Pages Standard Tag Library**

Tipologie di tag

- **Core tag**
- **Formatting tag**
- **SQL tag**
- **XML tag**
- **JSTL functions**

Core tag

- **Per includere il set di core tag:**

```
<%@ taglib prefix="c"
  uri="http://java.sun.com/jsp/jstl/core" %>
```

Overview core tag

Tag	Description
<c:out >	Like <%= ... >, but for expressions.
<c:set >	Sets the result of an expression evaluation in a 'scope'
<c:remove >	Removes a scoped variable (from a particular scope, if specified)
<c:catch>	Catches any Throwable that occurs in its body and optionally exposes it
<c:if>	Simple conditional tag which evalutes its body if the supplied condition is true
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<c:when>	Subtag of <choose> that includes its body if its condition evalutes to 'true'.

Overview core tag

Tag	Description
<c:otherwise >	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'
<c:import>	Retrieves an absolute or relative URL and exposes its contents to either the page, a String in 'var', or a Reader in 'varReader'
<c:forEach >	The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality
<c:forTokens>	Iterates over tokens, separated by the supplied delimeters
<c:param>	Adds a parameter to a containing 'import' tag's URL
<c:redirect >	Redirects to a new URL
<c:url>	Creates a URL with optional query parameters

<c:out>

- **The <c:out> tag displays the result of an expression, similar to the way <%= %> works with a difference that <c:out> tag lets you use the simpler "." notation to access properties. For example, to access customer.address.street just use tag is <c:out value="customer.address.street"/>.**
- **The <c:out> tag can automatically escape XML tags so they aren't evaluated as actual tags.**

<c:out> Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
<head>
<title><c:out> Tag Example</title>
</head>
<body>
<c:out value="${'<tag> , &'}"/>
</body>
</html>
```

→ <tag> , &

<c:set>

- **The <c:set> tag is JSTL-friendly version of the setProperty action. The tag is helpful because it evaluates an expression and uses the results to set a value of a JavaBean or a java.util.Map object.**

<c:set> Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
   %>
<html>
<head>
<title><c:set> Tag Example</title>
</head>
<body>
<c:set var="salary" scope="session" value="${2000*2}"/>
<c:out value="${salary}"/>
</body>
</html>
```

→ 4000

<c:remove>

- **The <c:remove> tag removes a variable from either a specified scope or the first scope where the variable is found (if no scope is specified). This action is not normally particularly helpful, but it can aid in ensuring that a JSP cleans up any scoped resources it is responsible for.**

<c:remove> Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>c:remove Tag Example</title>
</head>
<body>
<c:set var="salary" scope="session" value="${2000*2}"/>
<p>Before Remove Value: <c:out value="${salary}"/></p>
<c:remove var="salary"/>
<p>After Remove Value: <c:out value="${salary}"/></p>
</body>
</html>
```

→ Before Remove Value: 4000
  After Remove Value:

<c:catch>

- **The <c:catch> tag catches any Throwable that occurs in its body and optionally exposes it. Simply it is used for error handling and to deal more gracefully with the problem.**

Esempio:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>c:catch Tag Example</title>
</head>
<body>
<c:catch var ="catchException">
   <% int x = 5/0;%>
</c:catch>
```

<c:catch>

Esempio:

```
<c:if test = "${catchException != null}">
   <p>The exception is : ${catchException} <br />
   There is an exception: ${catchException.message}</p>
</c:if>

</body>
</html>
```

→ The exception is : java.lang.ArithmaticException: / by zero
There is an exception: / by zero

Tag condizionali

- **The <c:if> tag evaluates an expression and displays its body content only if the expression evaluates to true.**
- **The <c:choose> works like a Java switch statement in that it lets you choose between a number of alternatives. Where the switch statement has case statements, the <c:choose> tag has <c:when> tags. A a switch statement has default clause to specify a default action and similar way <c:choose> has <c:otherwise> as default clause.**

<c:if> Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>c:if Tag Example</title>
</head>
<body>
<c:set var="salary" scope="session" value="${2000*2}"/>
<c:if test="${salary > 2000}">
  <p>My salary is: <c:out value="${salary}"/><p>
</c:if>
</body>
</html>
```

→ My salary is: 4000

<c:choose> Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title><c:choose> Tag Example</title>
</head>
<body>
<c:set var="salary" scope="session" value="${2000*2}"/>
<p>Your salary is : <c:out value="${salary}"/></p>
<c:choose>
   <c:when test="${salary <= 0}"> Salary is very low to survive. </c:when>
   <c:when test="${salary > 1000}"> Salary is very good. </c:when>
   <c:otherwise> No comment sir... </c:otherwise>
</c:choose>
</body>
</html>
```

Iteratori

- **These tags exist as a good alternative to embedding a Java for, while, or do-while loop via a scriptlet. The <c:forEach> tag is the more commonly used tag because it iterates over a collection of objects. The <c:forTokens> tag is used to break a string into tokens and iterate through each of the tokens.**

<c:forEach> Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
   %>
<html>
<head>
<title><c:forEach> Tag Example</title>
</head>
<body>
<c:forEach var="i" begin="1" end="5">
   Item <c:out value="${i}"/><p>
</c:forEach>
</body>
</html>
```

<c:forTokens> Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title><c:forTokens> Tag Example</title>
</head>
<body>
<c:forTokens items="Zara,nuha,roshy" delims="," var="name">
  <c:out value="${name}"/><p>
</c:forTokens>
</body>
</html>
```

Gestione URL

- **The <c:url> tag formats a URL into a string and stores it into a variable. This tag automatically performs URL rewriting when necessary. The var attribute specifies the variable that will contain the formatted URL.**
- **The JSTL url tag is just an alternative method of writing the call to the response.encodeURL() method. The only real advantage the url tag provides is proper URL encoding, including any parameters specified by children param tag.**

```
<c:url value="/index.jsp" var="myURL">
  <c:param name="trackingId" value="1234"/>
  <c:param name="reportType" value="summary"/
  >
</c:url>
<c:import url="${myURL}"/>
```

→ "/index.jsp?
trackingId=1234;reportType=summary"

<c:redirect>

- **The <c:redirect> tag redirects the browser to an alternate URL by providing automatically URL rewriting, it supports context-relative URLs, and it supports the <c:param> tag.**

Esempio

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title><c:redirect> Tag Example</title>
</head>
<body>
<c:redirect url="http://www.photofuntoos.com"/>
</body>
</html>
```

Formatting tag

- **Per includere il seti di formatting tag:**

```
<%@ taglib prefix="fmt"
          uri="http://java.sun.com/jsp/jstl/fmt"
  %>
```

Overview formatting tag

Tag	Description
<fmt:formatNumber>	To render numerical value with specific precision or format
<fmt:parseNumber>	Parses the string representation of a number, currency, or percentage
<fmt:formatDate>	Formats a date and/or time using the supplied styles and pattern
<fmt:parseDate>	Parses the string representation of a date and/or time
<fmt:bundle>	Loads a resource bundle to be used by its tag body
<fmt:setLocale>	Stores the given locale in the locale configuration variable
<fmt:setBundle>	Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable
<fmt:timeZone>	Specifies the time zone for any time formatting or parsing actions nested in its body
<fmt:setTimeZone>	Stores the given time zone in the time zone configuration variable
<fmt:message>	To display an internationalized message
<fmt:requestEncoding>	Sets the request character encoding

<fmt:formatNumber>

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<head>
  <title>JSTL fmt:formatNumber Tag</title>
</head>
<body>
<h3>Number Format:</h3>
<c:set var="balance" value="120000.2309" />
<p>Formatted Number (1): <fmt:formatNumber value="${balance}" type="currency"/></p>
<p>Formatted Number (2): <fmt:formatNumber type="number"  maxIntegerDigits="3" value="${balance}" /></p>
<p>Formatted Number (3): <fmt:formatNumber type="number" maxFractionDigits="3" value="${balance}" /></p>
<p>Formatted Number (4): <fmt:formatNumber type="number" pattern="###.###E0" value="${balance}" /></p>
<p>Currency in USA :
<fmt:setLocale value="en_US"/>
<fmt:formatNumber value="${balance}" type="currency"/></p>
</body></html>
```

```
Formatted Number (1): £120,000.23
Formatted Number (2): 000.231
Formatted Number (3): 120,000.231
Formatted Number (4): 120E3
Currency in USA : $120,000.23
```

<fmt:parseNumber>

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<head>
  <title>JSTL fmt:parseNumber Tag</title>
</head>
<body>
<h3>Number Parsing:</h3>
<c:set var="balance" value="1250003.350" />

<fmt:parseNumber var="i" type="number" value="${balance}" />
<p>Parsed Number (1) : <c:out value="${i}" /></p>
<fmt:parseNumber var="i" integerOnly="true" type="number" value="${balance}" />
<p>Parsed Number (2) : <c:out value="${i}" /></p>
</body>
</html>
```

```
NUMBER PARSING:
Parsed Number (1) : 1250003.35
Parsed Number (2) : 1250003
```

<fmt:message>

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<html>
<head>
<title>JSTL fmt:message Tag</title>
</head>
<body>
<fmt:setLocale value="en"/>
<fmt:setBundle basename="it.unimore.ing.weblab.Example" var="lang"/>
<fmt:message key="count.one" bundle="${lang}"/><br/>
<fmt:message key="count.two" bundle="${lang}"/><br/>
<fmt:message key="count.three" bundle="${lang}"/><br/>
</body>
</html>
```

```
package it.unimore.ing.weblab.Example;
import java.util.ListResourceBundle;
public class Example_En extends ListResourceBundle {
  public Object[][] getContents() {return contents; }
  static final Object[][] contents = {{"count.one", "One"},
  {"count.two", "Two"},
  {"count.three", "Three"},};
}
```

SQL tag

- **Per includere il set di sql tag:**

```
<%@  taglib
  uri="http://java.sun.com/jsp/jstl/sql"
  prefix="sql" %>
```

Overview sql tag

Tag	Description
<sql:setDataSource>	Creates a simple DataSource suitable only for prototyping
<sql:query>	Executes the SQL query defined in its body or through the sql attribute
<sql:update>	Executes the SQL update defined in its body or through the sql attribute
<sql:param>	Sets a parameter in an SQL statement to the specified value
<sql:dateParam>	Sets a parameter in an SQL statement to the specified java.util.Date value
<sql:transaction>	Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction

<sql:setDataSource>

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>JSTL sql:setDataSource Tag</title>
</head>
<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/TEST"
    user="user_id"  password="mypassword"/>

<sql:query dataSource="${snapshot}" sql="..." var="result" />

</body>
</html>
```

<sql:param>

```
<c:set var="empId" value="103"/>

<sql:update dataSource="${snapshot}" var="count">
  DELETE FROM Employees WHERE Id = ?
  <sql:param value="${empId}" />
</sql:update>
```

<sql:setDataSource>

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>JSTL sql:setDataSource Tag</title>
</head>
<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/TEST"
    user="user_id"  password="mypassword"/>

<sql:query dataSource="${snapshot}" sql="..." var="result" />

</body>
</html>
```

Funzioni

- **Per includere le funzioni:**

```
<%@  taglib
  uri="http://java.sun.com/jsp/jstl/functions"
  prefix="fn" %>
```

Overview funzioni

Tag	Description
fn:contains()	Tests if an input string contains the specified substring
fn:containsIgnoreCase()	Tests if an input string contains the specified substring in a case insensitive way
fn:endsWith()	Tests if an input string ends with the specified suffix
fn:escapeXml()	Escapes characters that could be interpreted as XML markup
fn:indexOf()	Returns the index withing a string of the first occurrence of a specified substring
fn:join()	Joins all elements of an array into a string
fn:length()	Returns the number of items in a collection, or the number of characters in a string
fn:replace()	Returns a string resulting from replacing in an input string all occurrences with a given string
fn:split()	Splits a string into an array of substrings
fn:startsWith()	Tests if an input string starts with the specified prefix

Overview funzioni

Tag	Description
fn:substring()	Returns a subset of a string
fn:substringAfter()	Returns a subset of a string following a specific substring
fn:substringBefore()	Returns a subset of a string before a specific substring
fn:toLowerCase()	Converts all of the characters of a string to lower case
fn:toUpperCase()	Converts all of the characters of a string to upper case
fn:trim()	Removes white spaces from both ends of a string

Esempio di funzioni

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<head>
<title>Using JSTL Functions</title>
</head>
<body>

<c:set var="string1" value="This is first String."/>
<c:set var="string2" value="${fn:substring(string1, 5, 15)}" />

<p>Final sub string : ${string2}</p>

</body>
</html>
```

- **Ulteriori informazioni:**
  - http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

Modulo 4
Hands on JSTL

Esercitazioni JSTL

- **Trasformare le pagine JSP precedentemente realizzate in pagine JSP che usano JSTL**
- **Tre applicazioni:**
  - Data aggiornata ogni secondo
  - Calcolatrice
  - Contatore gestito a livello di sessione utente