Session

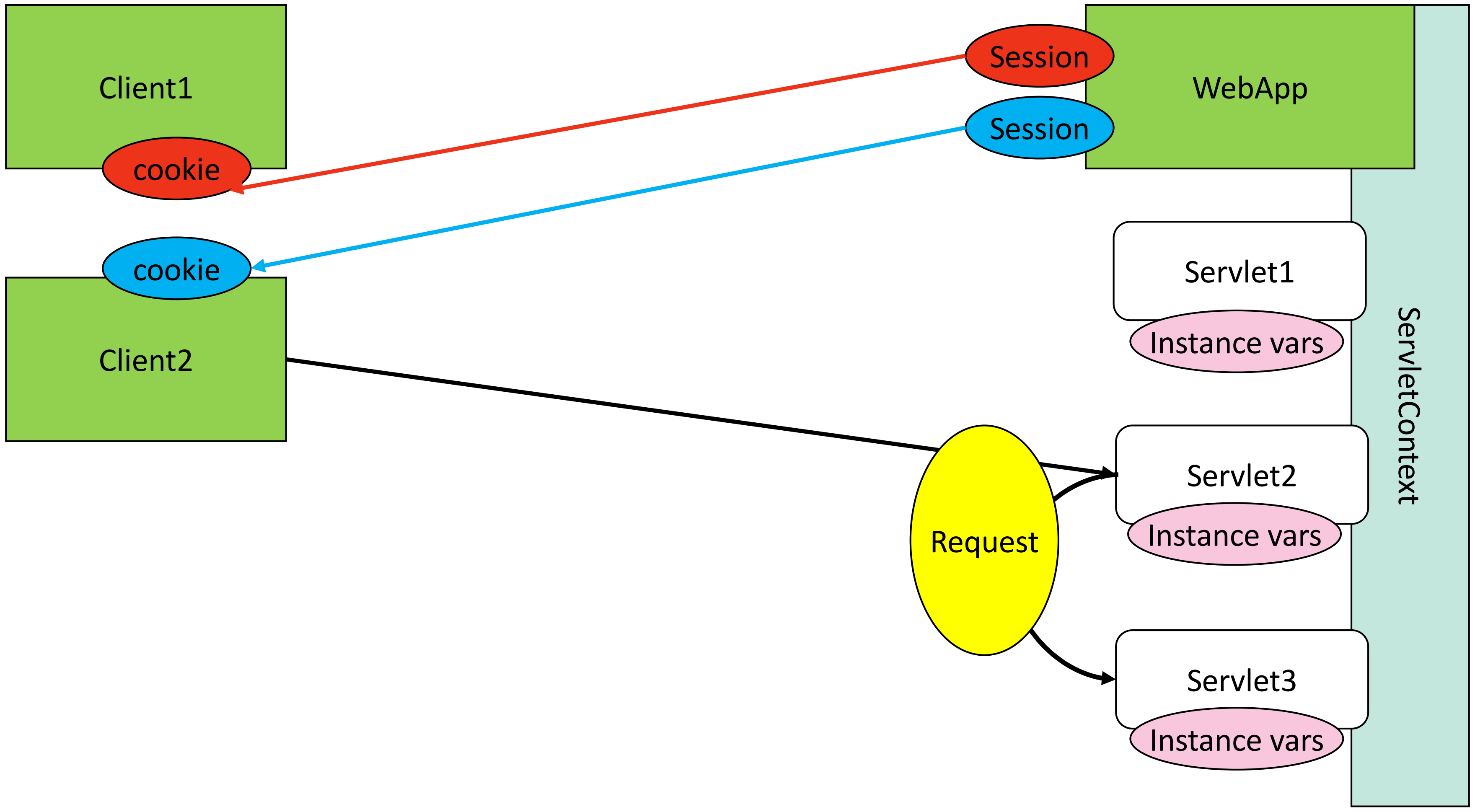
Final notes

Advanced: associating events with session objects

- Your application can notify web context and session listener objects of servlet lifecycle events (Handling Servlet Lifecycle Events). You can also notify objects of certain events related to their association with a session, such as the following:
  - When the object is added to or removed from a session. To receive this notification, your object must implement the javax.servlet.http.HttpSessionBindingListener interface.
  - When the session to which the object is attached will be passivated or activated. A session will be passivated or activated when it is moved between virtual machines or saved to and restored from persistent storage. To receive this notification, your object must implement the javax.servlet.http.HttpSessionActivationListener interface.

Q

How can we keep global information in a webApp?



Client1
cookie
cookie
Client2
Session
Session
WebApp
Servlet1
Instance vars
Request
Servlet2
Instance vars
Servlet3
Instance vars
ServletContext

Sharing information

- Within a request:

multiple servlets cooperating through *forward* and *include* mechanisms:

- pass request and response, add information to the **request** (request in jsp)

- Among different requests by the same user:

- use the **Session** object

getSession() in servlets

session in jsp

Sharing information

- Among different invocations of the same servlet:

- use instance variables or static variables

- Among different servlets/jsps of the same WebApp:

- use ServletContext
  - servlets: getServletConfig().getServletContext
  - jsps: application (a special, predefined object)

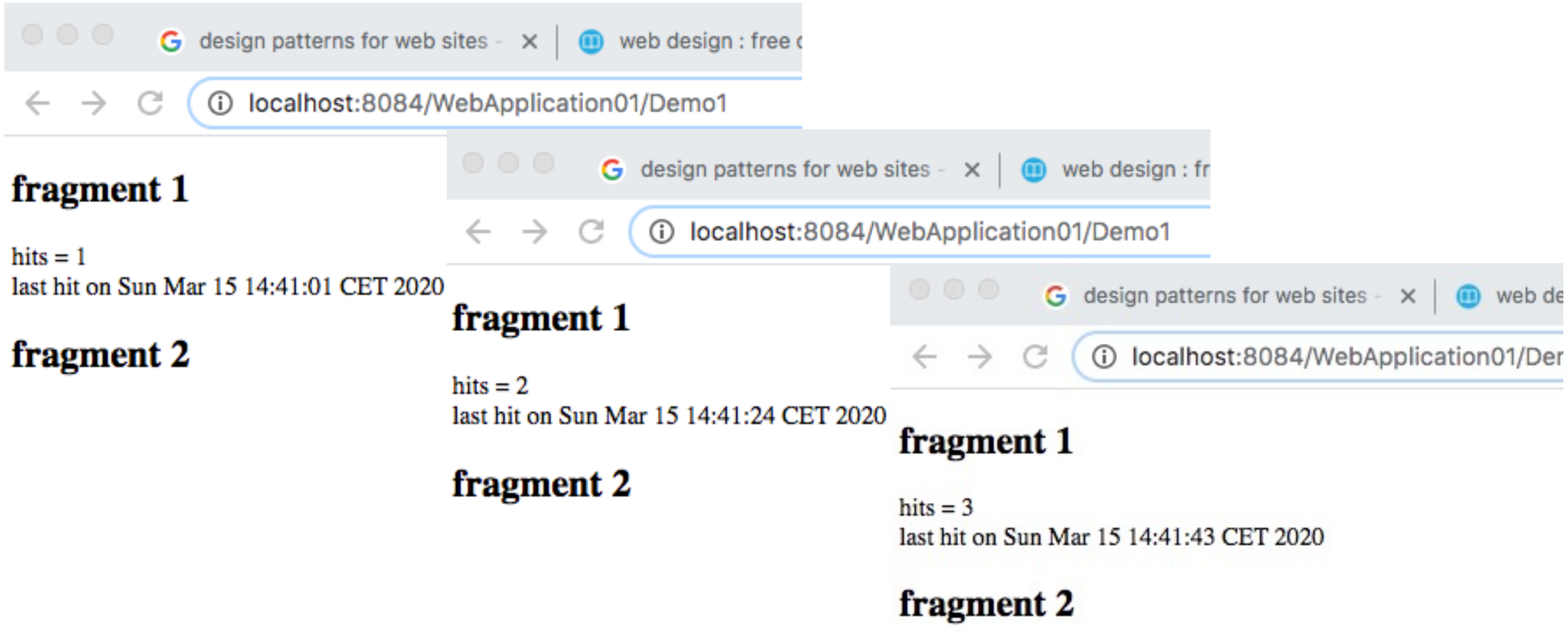
Let us build a hit counter - 1

```
public class Counter {
    int count = 0;
    Calendar timeStamp = Calendar.getInstance();
    public void increase(){
        count++;
        timeStamp = Calendar.getInstance();
    }
    @Override
    public String toString() {
        StringBuffer s = null;
        if (count == 0)
            s = new StringBuffer("<p>no hits yet</p>");
        else {
            s = new StringBuffer("<p>hits = ");
            s.append(count)
                    .append("<br>last hit on ")
                    .append(timeStamp.getTime().toString());
        }
        return s.toString();
    }
}
```

Let us build a hit counter - 2

```
@WebServlet(name = "Demo1", urlPatterns = {"/Demo1"})
public class Demo1 extends HttpServlet {
    Counter counter=new Counter();
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            request.getRequestDispatcher("/fragment1.html")
                   .include(request, response);
            counter.increase();
            out.println(counter);
            request.getRequestDispatcher("/fragment2.html")
                    .include(request, response);
        }
    }
}
```

Output


design patterns for web sites -
web design : free
localhost:8084/WebApplication01/Demo1

fragment 1

hits = 1
last hit on Sun Mar 15 14:41:01 CET 2020

fragment 2

fragment 1

hits = 2
last hit on Sun Mar 15 14:41:24 CET 2020

fragment 2

fragment 1

hits = 3
last hit on Sun Mar 15 14:41:43 CET 2020

fragment 2

Even if anyone accesses from a different client!
But if we restart the server, counter restarts from 1!

!

Wait – our counter solution is not thread-safe!

when sharing info, always think about thread-safety!

Servlets are not thread safe!

unless YOU make them so…

- A *thread* is a lightweight process which has its own call stack and accesses shared data of other threads in the same process (shares heap memory).
- A servlet can be invoked simultaneously by multiple threads (i.e., by multiple requests).
- We can fix this problem dealing with concurrency.

- *You should know about concurrency, threads, semaphores and monitors from your bachelor courses. If you do not, see here:*

https://docs.oracle.com/javase/tutorial/essential/concurrency/

(The basics that you need are in the "Concurrency" section)

5 rules to remember

1. Service() , doGet(), doPost() or to be more generic doXXX() methods should not update or modify instance variables as instance variables are shared by all threads of same instance.
2. If you have a requirement which requires modification of instance variable then do it in a synchronized block. (or synchronized method)
3. Above two rules are applicable for static variables also because they are also shared.
4. Local variables are always thread safe (unless they refer to global objects)
5. The request and response objects are thread safe to use because new instance of these are created for every request into your servlet, and thus for every thread executing in your servlet.

Q

Are session objects thread safe?

Sessions and thread safety

- A *session* belongs to a user.
- Hence, when different users activates the same servlet, and this requests a session object, it gets a different object for every user – so no problems with multithreading.

BUT

- if a user opens two windows on the same browser, and accesses the same servlet, then we DO have a thread safety issue!
- that's very unlikely, but yet…

Q

How can we fix our counter making it thread-safe?

Fixing the hit counter - option 1

```
public class Counter {
    int count = 0;
    Calendar timeStamp = Calendar.getInstance();
    public synchronized void increase(){
        count++;
        timeStamp = Calendar.getInstance();
    }
    @Override
    public String toString() {
        StringBuffer s = null;
        if (count == 0)
            s = new StringBuffer("<p>no hits yet</p>");
        else {
            s = new StringBuffer("<p>hits = ");
            s.append(count)
                  .append("<br>last hit on ")
                  .append(timeStamp.getTime().toString());
        }
        return s.toString();
    }
}
```

Fixing the hit counter - option 2

```
@WebServlet(name = "Demo1", urlPatterns = {"/Demo1"})
public class Demo1 extends HttpServlet {
    Counter counter=new Counter();
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            request.getRequestDispatcher("/fragment1.html")
                    .include(request, response);
            synchronized (this) {
                counter.increase();
            }
            out.println(counter);
            request.getRequestDispatcher("/fragment2.html")
                    .include(request, response);
        }
    }
}
```

Q

What are JSPs?

How are they related to servlets?

JSP Technology

A technology somehow similar to PHP or ASP, ASP.net, but Java-based.

Dual to Servlets

Has been the basis for JSP-CustomTags

Has been the basis for JSF

Tutorial
https://www.tutorialspoint.com/jsp/index.htm

Simple.jsp

```
<%@ page import=java.util.*  %>
<html>
 <body>
    <% int x=Calendar.get(Calendar.HOUR_OF_DAY); %>
    <%= x %>
  </body>
</html>
```

A taste of servlet programming-2

```
import java.util.Calendar;
public class SimpleServlet extends HttpServlet {
    public void doGet (HttpServletRequest request,
            HttpServletResponse response)
                                throws ServletException, IOException {
            PrintWriter out=response.getWriter();
            response.setContentType("text/html");
            out.println("<HTML><BODY>");
            x=Calendar.get(Calendar.HOUR_OF_DAY);
            out.println(x);
            out.println("</BODY></HTML>");
            out.close();
    }
}
```

<%@ directives %>

<% scriptlets %>

<%= expressions %>

Equivalent to:
out.println(expression);

A scriptlet is a block of Java code executed during the request-processing time.

In Tomcat all the scriptlets gets put into the service() method of the servlet. They are therefore processed for every request that the servlet receives.

A taste of servlet programming-2

A directive is used as a message mechanism to pass information from the JSP code to the container

Main directives:

page

include (for including other STATIC resources at compilation time)

<%@ directives %>

<%! declarations %>

```
import java.util.Calendar;
public class SimpleServlet extends HttpServlet {
   String nome="pippo"; //instance variable
   final float PI=3.1415926535 // constant
   public void getName() {/* this is my function */}
   public void doGet (HttpServletRequest request,
            …
   }
}
```

A declaration is a block of Java code used to define class-wide variables and methods in the generated servlet.

They are initialized when the JSP page is initialized.

Examples:

<%! String nome="pippo"; %>

<%! public String getName() {return nome;} %>

Directives

<%@ DIRECTIVE{attributo=valore} %>

main attributes:

<%@ page language=java session=true %>

<%@ page import=java.awt.*,java.util.* %>

<%@ page errorPage=URL %>

<%@ page isErrorPage=true %>

JSP nuts and bolts

Syntactic elements:

<%@ directives %> → Interaction with the CONTAINER

<%! declarations %> → In the initialization of the JSP

<% scriptlets %> → In the service method

<%= expression %> → (Syntactic sugar)
  same as scriptlet: <% out.println(expression %)>

<jsp:actions/>

JSP Standard actions

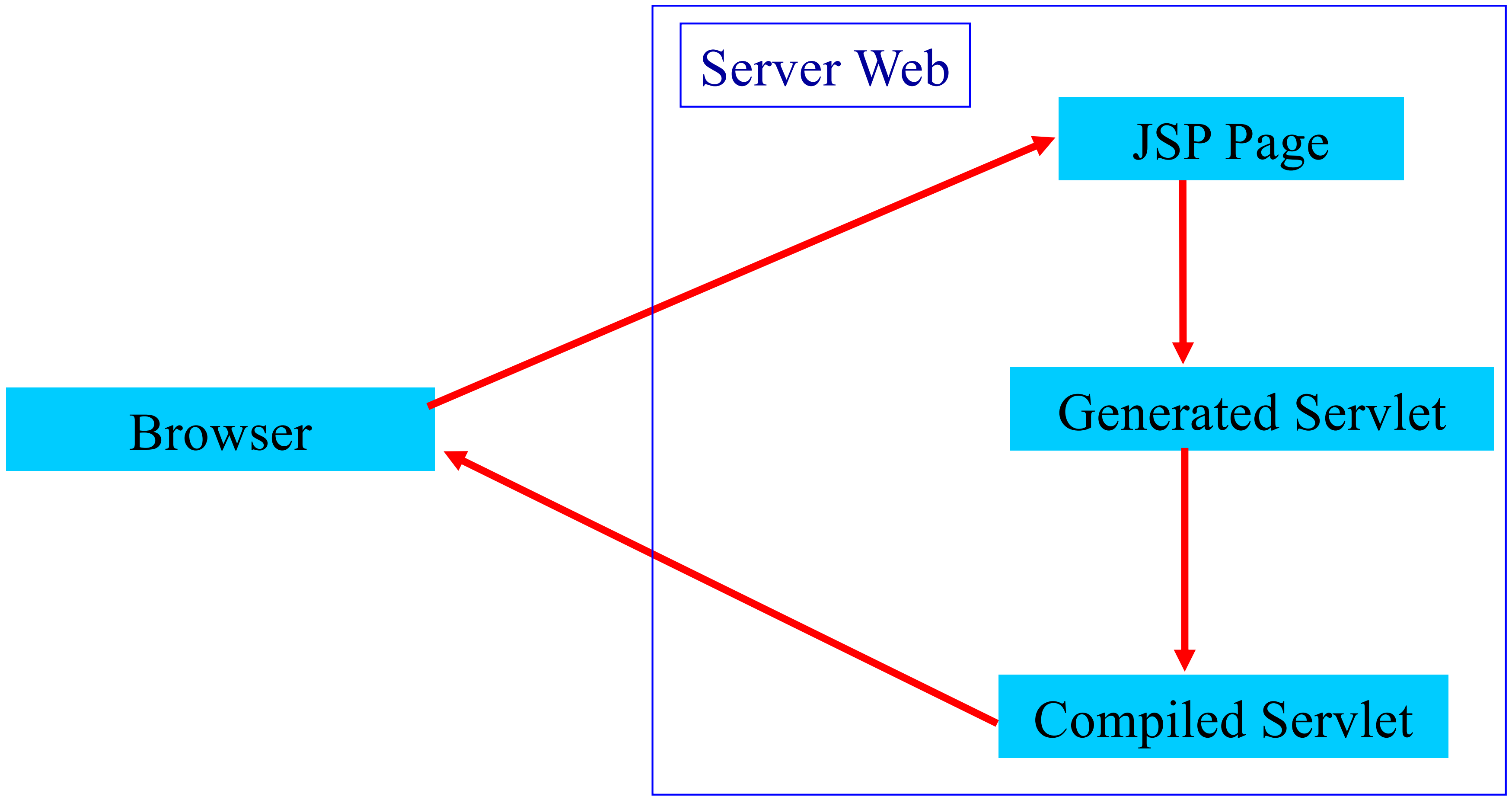
```
<jsp:include page="URL" />
```

For including STATIC or DYNAMIC resources at request time

```
<jsp:forward page="URL" />
```

```
<jsp:useBean id= "instanceName"
  scope= "page | request | session | application"
  class= "packageName.className" type= "packageName.className"
  beanName="packageName.className | <%= expression >" >
</jsp:useBean>
```

JSP Lifecycle



Where is the generated code?

Some IDE's allow you to see the Servlet generated from the JSP, e.g. Netbeans

/Users/ronchet/Library/Application Support/NetBeans/8.2/apache-tomcat-8.0.27.0_base/work/Catalina/localhost/WebAppJSPwithSession/org/apache/jspDemoSession.class

Unfortunately, IntelliJ does not.

https://intellij-support.jetbrains.com/hc/en-us/community/posts/360003783520-Debugging-JSP-in-Intellij-2019-Ultimate

Q

How should I configure Tomcat to use JSPs?

JSP pages

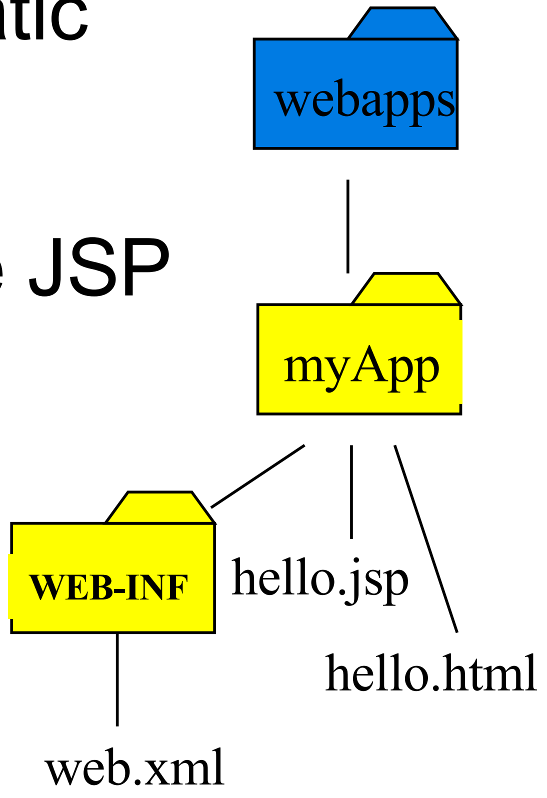
To let Tomcat serve JSP pages, we follow the same procedure that we use for static pages.

In the myApp folder we can deposit the JSP files.

On our Tomcat server, the URL for the hello.jsp file becomes:

http://*machine/port*/myApp/hello.jsp

The WEB-INF directory can be empty.



Q

How can I access request and response in JSPs?

How can I use sessions with JSPs?

Predefined Objects

out	Writer
request	HttpServletRequest
response	HttpServletResponse
session	HttpSession
page	this in the Servlet
application	servlet.getServletContext area shared among all servlets within the same webapp
config	ServletConfig
exception	only in a errorPage
pageContext	

request

```
<%@ page errorPage="errorpage.jsp" %>
<html>
  <head>
    <title>UseRequest</title>
  </head>
  <body>
    <%
       // Get the User's Name from the request
       out.println("<b>Hello: " + request.getParameter("user") + "</b>");
    %>
  </body>
</html>
```

JSP in action – Example part 1

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.util.Date"%>
<%@page language="java"  session="true" %>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type"
             content="text/html; charset=UTF-8">
        <title>Session Test JSP</title>
    </head>
    <body>
      <%!  Integer accessCount; %>
      <%
        accessCount=(Integer)session.getAttribute("accessCount");
         if (accessCount == null) {
            accessCount = 0;   // autobox int to Integer
         } else {
            accessCount = new Integer(accessCount + 1);
         }
         session.setAttribute("accessCount", accessCount);
      %>
```

JSP in action – Example part 2

```
Session is new? <% out.println(session.isNew()); %>
      <h2>You accessed this site " <%= accessCount %>
            times in this session.</h2>
      <ul><li>Your session ID is " <%= session.getId() %></li>
          <li>Session creation time is
               <%= new Date(session.getCreationTime()) %> </li>
          <li>Session last access time is  <%=
               new Date(session.getLastAccessedTime()) %> </li>
          <li>Session max inactive interval  is <%=
               session.getMaxInactiveInterval() %> seconds</li>
       </ul>

       <p><a  href='<%= request.getRequestURI() %>'>Refresh</a>
       <p><a  href='
            <%= response.encodeURL(request.getRequestURI())%>'>
               Refresh with  URL rewriting</a>
       <form method="GET" action="endSession.jsp">
           <input type="submit" value="End Session">
       </form>
    </body>
</html>
```

Q

How can we make good use of JSPs?

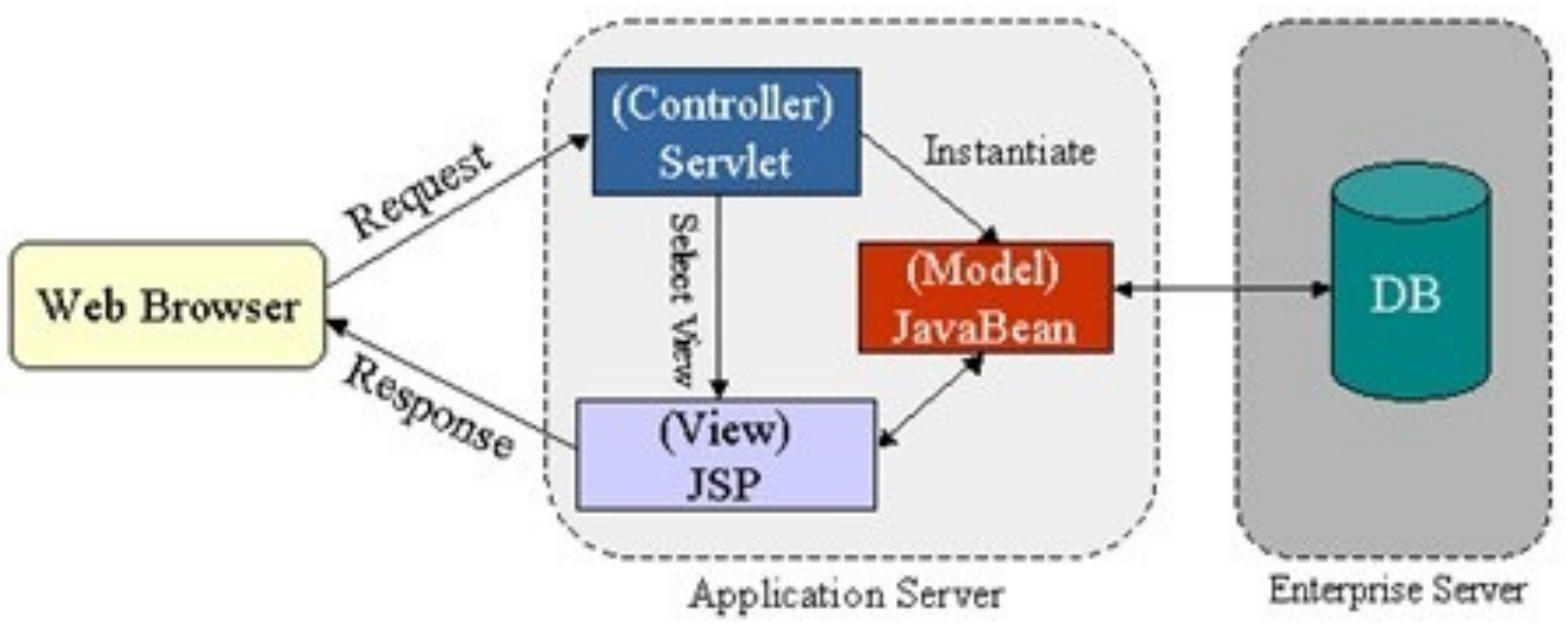
Best practices

- **Don't overuse Java code in HTML pages**
- **Choose the right include mechanism:**
  - *Static data* such as headers, footers, and navigation bar content is best kept in separate files and not regenerated dynamically.
  - Once such content is in separate files, they can be included in all pages using one of the following include mechanisms:
  - Include directive: <%@ include file="filename" %>

    Include action: <jsp:include page="page.jsp" />
- **Don't mix business logic with presentation**
  - JSP code should be limited to front-end presentation.
- **Use filters if necessary** (See next lecture)
- **Use a database for persistent information** (See next lectures)
  - Use connection pooling

JSP usage: MVC pattern

MVC


Web Browser
Request
Response
(Controller) Servlet
Select View
(View) JSP
Instantiate
(Model) JavaBean
DB
Application Server
Enterprise Server

What is a Javabean?

A bean is a Java class that:

- Provides a public no-argument constructor
- Implements java.io.Serializable
- Follows JavaBeans design patterns
  - Has Set/get methods for properties
  - (Has Add/remove methods for events)
- Is thread safe/security conscious
  - Can run in an applet, application, servlet, ...

Example:

```
public class SimpleBean implements Serializable {
        private int counter;
        SimpleBean() {counter=0;}
        int getCounter() {return counter;}
        void setCounter(int c) {counter=c;}
}
```

Standard actions involving beans

<jsp:useBean id=“name” class=“fully_qualified_pathname” scope=“{request|session|application}” />

<jsp:setProperty name=“nome” property=“value” />

<jsp:getProperty name=“nome” property=“value” />

See: https://www.tutorialspoint.com/jsp/jsp_java_beans.htm

Example - BeanOne.java

define the bean

```
package beans;
import java.io.Serializable;
public class BeanOne implements Serializable {
    String name;
    String surname;
    public BeanOne() {}
    public String getName() { return name; }
    public String getSurname() {return  surname;}
    public void setName(String name) { this.name =  name;}
    public void setSurname(String surname) {this.surname = surname;}
    @Override
    public String toString() {
        return "BeanOne{" + "name=" + name + ", surname=" + surname +  '}';
    }
}
```

Example - index.html

main page

```
<!DOCTYPE html>
<html>
<head>
   <title>JSP - Hello World</title>
</head>
<body>
<br/>
<a href="jspOne.jsp">jspOne</a><br>
<a href="jspTwo.jsp">jspTwo</a><br>
<a href="addZ">Add Z</a><br>
<a href="InvalidateServlet">invalidate session</a>
</body>
</html>
```

Example - jspOne.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <title>JSP Page</title>
  </head>
  <body>
    <h1>Setting the property...</h1>
     <jsp:useBean id="myBean1" class="beans.BeanOne" scope="session"/>
     <jsp:setProperty name="myBean1" property="surname" value="de pippis"/>
     <jsp:setProperty name="myBean1" property="name" value="pippo"/>
      <p><%=myBean1.toString()%></p>
      <hr>
      <jsp:include page="index.html"></jsp:include>
</html>
```

set value in the bean

What happens If we change the scope?

Example - jspTwo.jsp

show value
in the bean

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Getting the property...</h1>
        <jsp:useBean id="myBean1" class="beans.BeanOne" scope="session"/>
        <jsp:getProperty name="myBean1" property="surname" />
        <jsp:getProperty name="myBean1" property="name" />
        <p><%=myBean1.toString()%></p>
    <hr>
    <jsp:include page="index.html"></jsp:include>
</body>
</html>
```

Example - BeanAccessServlet.java

modify value
in the bean

```
@WebServlet(name = "addZServlet", value = "/addZ")
public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,  HttpServletResponse response)
                throws ServletException, IOException  {
        HttpSession session=request.getSession();
        if (session.getAttribute("myBean1")==null) {
            session.setAttribute("myBean1", new BeanOne());
        }
        BeanOne aBean=(BeanOne)(session.getAttribute("myBean1"));
        aBean.setName(aBean.getName()+"z");
        request.getRequestDispatcher("jspTwo.jsp").forward(request, response);
    }
}
```

InvalidateSessionServlet.java

invalidate session

```
@WebServlet(name = "InvalidateServlet", value = "/InvalidateServlet")
public class InvalidateServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse
           response) throws ServletException,IOException {
    HttpSession session=request.getSession();
    session.invalidate();
    try {
      request.getRequestDispatcher("jspTwo.jsp").forward(request, response);
    } catch (ServletException | IOException e) {
      e.printStackTrace();
    }
  }
}
```

Pay attention to the scope!

JSP

Servlet

```
<jsp:useBean id="myBean1"
class="beans.BeanOne"
scope="session"/>
```

```
Session session=request.getSession();
BeanOne x= (BeanOne )session.getAttribute("myBean1");
```

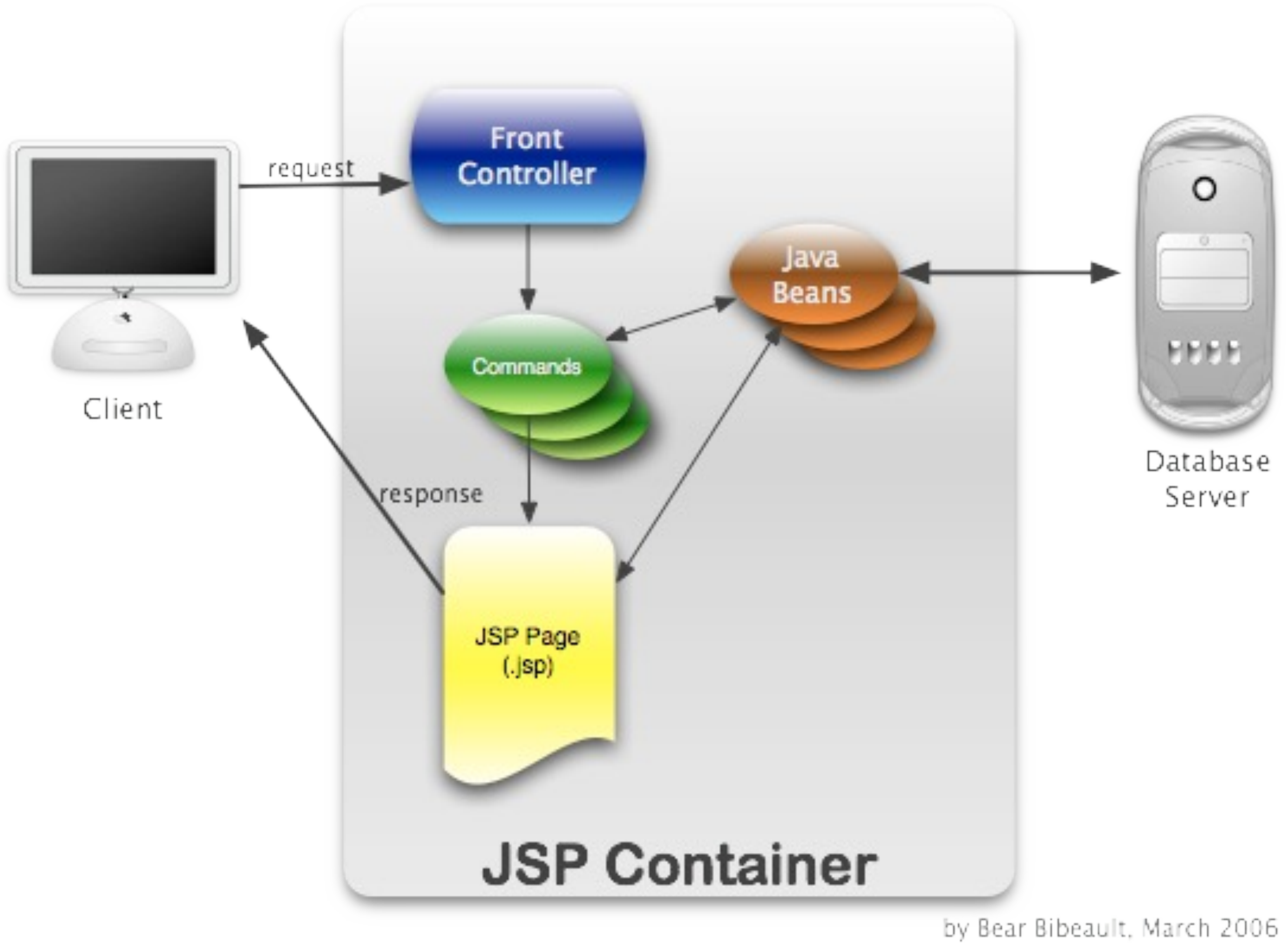
```
<jsp:useBean id="myBean1"
class="beans.BeanOne"
scope="application"/>
```

```
ServletContext context=request.getServletContext();
BeanOne x= (BeanOne )context.getAttribute("myBean1");
```

```
<jsp:useBean id="myBean1"
class="beans.BeanOne"
scope="request"/>
```

```
BeanOne x= (BeanOne )request.getAttribute("myBean1");
```

Front controller pattern



Examples

from:
https://javaranch.com/journal/200603/frontman.html

The Front Controller employed by the Struts package typically uses a servlet mapping of ***.do** where whatever appears as the prefix of the ".do" is used to lookup the actual class path of the Command (called "actions" in Struts) in an internal configuration map.

Another example, the pattern that I usually use, is to employ
a servlet mapping such as **/command/***
where the prefix "command" triggers the front controller,
and the rest of the path info is used to lookup the Command class I
n an internal map.
It would be typical to see URLs along the lines of:

http://some.server.com/webapp/command/deleteItem
http://some.server.com/webapp/command/insertItem
http://some.server.com/webapp/command/doSomethingWonderful

Cookies: legal aspects

Italian regulations

Individuazione delle modalità semplificate per l’informativa e l’acquisizione del consenso per l’uso dei cookie - 8 maggio 2014

- http://garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/3118884
- https://www.garanteprivacy.it/home/doveri
- https://it.wikipedia.org/wiki/Regolamento_generale_sulla_protezione_dei_dati

Italian regulations

- a. Cookie tecnici.

- I cookie tecnici sono quelli utilizzati al solo fine di "effettuare la trasmissione di una comunicazione su una rete di comunicazione elettronica, o nella misura strettamente necessaria al fornitore di un servizio della società dell'informazione esplicitamente richiesto dall'abbonato o dall'utente a erogare tale servizio" (cfr. art. 122, comma 1, del Codice).

Italian regulations

- Essi non vengono utilizzati per scopi ulteriori e sono normalmente installati direttamente dal titolare o gestore del sito web. Possono essere suddivisi in
- cookie di navigazione o di sessione, che garantiscono la normale navigazione e fruizione del sito;
- cookie analytics, assimilati ai cookie tecnici laddove utilizzati direttamente dal gestore del sito per raccogliere informazioni, in forma aggregata, sul numero degli utenti e su come questi visitano il sito stesso;

Italian regulations

- cookie di funzionalità, che permettono all'utente la navigazione in funzione di una serie di criteri selezionati (ad esempio, la lingua, i prodotti selezionati per l'acquisto) al fine di migliorare il servizio reso allo stesso.

- Per l'installazione di tali cookie non è richiesto il preventivo consenso degli utenti, mentre resta fermo l'obbligo di dare l'informativa ai sensi dell'art. 13 del Codice, che il gestore del sito, qualora utilizzi soltanto tali dispositivi, potrà fornire con le modalità che ritiene più idonee.

Italian regulations

- b. Cookie di profilazione.
- I cookie di profilazione sono volti a creare profili relativi all'utente e vengono utilizzati al fine di inviare messaggi pubblicitari in linea con le preferenze manifestate dallo stesso nell'ambito della navigazione in rete. In ragione della particolare invasività che tali dispositivi possono avere nell'ambito della sfera privata degli utenti, la normativa europea e italiana prevede che l'utente debba essere adeguatamente informato sull'uso degli stessi ed esprimere così il proprio valido consenso.

Italian regulations

- 2. Soggetti coinvolti: editori e "terze parti”.
- Un ulteriore elemento da considerare, ai fini della corretta definizione della materia in esame, è quello soggettivo. Occorre, cioè, tenere conto del differente soggetto che installa i cookie sul terminale dell'utente, a seconda che si tratti dello stesso gestore del sito che l'utente sta visitando (che può essere sinteticamente indicato come "editore") o di un sito diverso che installa cookie per il tramite del primo (c.d. "terze parti").

Italian regulations

- Nel momento in cui l'utente accede a un sito web, deve essergli presentata una prima informativa "breve", contenuta in un banner a comparsa immediata sulla home page (o altra pagina tramite la quale l'utente può accedere al sito), integrata da un'informativa "estesa", alla quale si accede attraverso un link cliccabile dall'utente.

Getting deeper with HTML Forms

Forms

Give to the user the possibility to send information to the Web server

The FORM tag defines a form and has the following attributes:

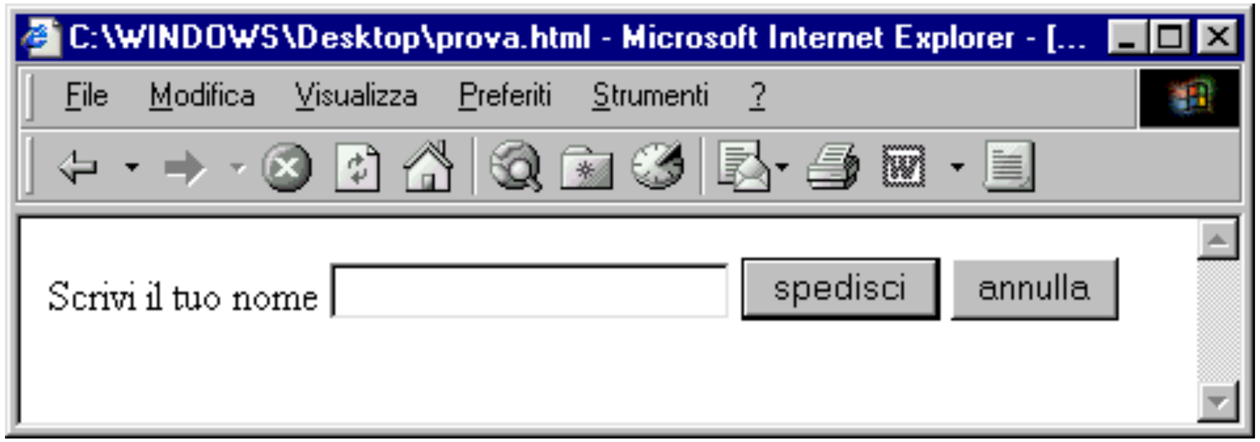
- ACTION identifies the processing engine
- ENCTYPE specificies the MIME type used to pass data to the server (Es. Text/html)

FORM contains the sub-tag (inner tags):

- several tags for collecting data
- An INPUT tag **must be** of type SUBMIT for sending the data
- An INPUT can be of type RESET to cancel all the gathered data

Form - input

```
<FORM method="POST" action="/cgi-bin/elabora">
   Scrivi il tuo nome
   <Input type="text" size“=25" maxlength="15“ name=“a”>
   <Input type="submit" value="spedisci">
   <Input type="reset" value="annulla">
</FORM>
```

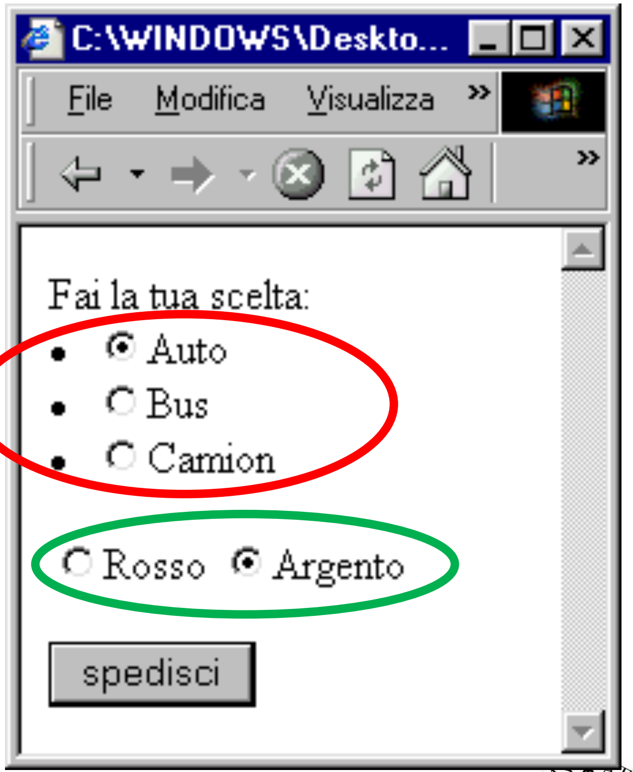

C:\WINDOWS\Desktop\prova.html - Microsoft Internet Explorer - [...

Scrivi il tuo nome

Sends a url of type
http://…/cgi-bin/elabora?a=MarcoRonchetti

Form – input type="radio"

```
<FORM method="POST" action="/cgi-bin/elabora">
  Fai la tua scelta:
  <LI><Input type="radio" name="tipo"
  value="auto" checked>Auto
  <LI><Input type="radio" name="tipo"
  value="bus">Bus
  <LI><Input type="radio" name="tipo"
  value="camion">Camion

  <P><Input type="radio" name="colore"
  value="rosso">Rosso
  <Input type="radio" name="colore"
   value="argento" checked>Argento</P>

  <Input type="submit" value="spedisci">
</FORM>
```



Fai la tua scelta:

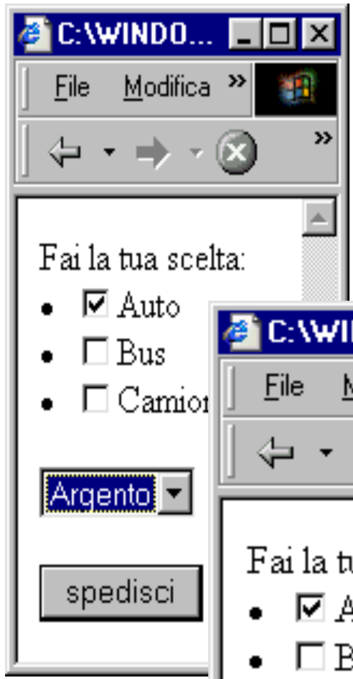
- [✓ Auto]
- Bus
- Camion

Rosso [✓ Argento]

spedisci

Form – input type="checkbox" - select

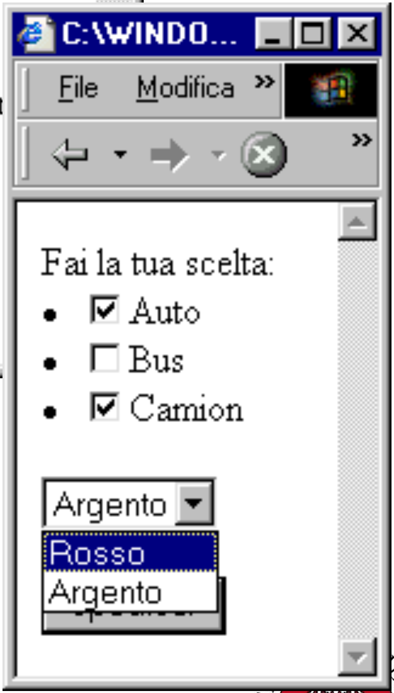
```
<FORM method="POST" action="/cgi-bin/elabora">
  Fai la tua scelta:
 <LI><Input type="checkbox"
 name="tipo" value="auto" checked>Auto
 <LI><Input type="checkbox"
 name="tipo" value="bus">Bus
 <LI><Input type="checkbox"
 name="tipo" value="camion">Camion

 <P><Select name="colore">
 <option>Rosso
 <option selected>Argento
 </select></P>

 <Input type="submit" value="spedisci">
</FORM>
```



Fai la tua scelta:
- ☑ Auto
- ☐ Bus
- ☐ Camion

Argento

spedisci

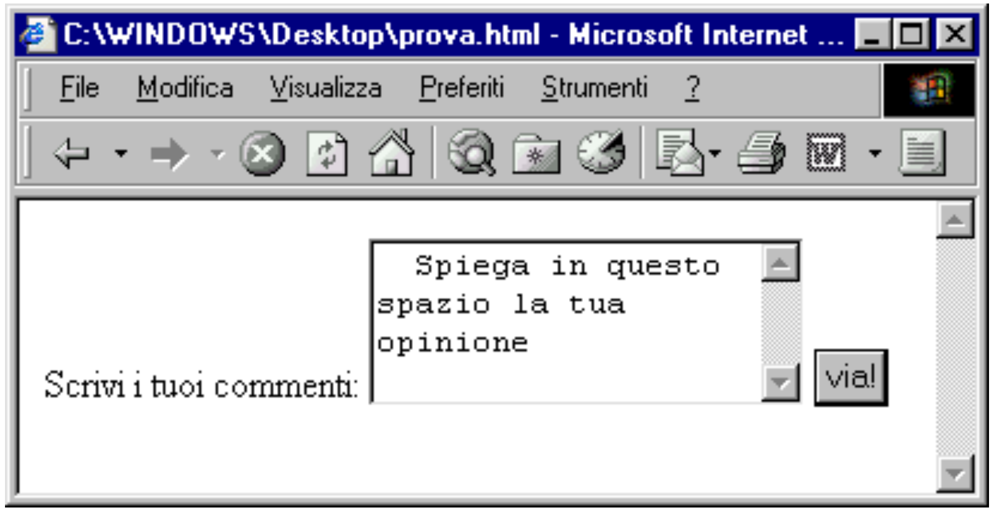


Fai la tua scelta:
- ☑ Auto
- ☐ Bus
- ☑ Camion

Argento
Rosso
Argento

Form – textarea

```
<FORM method="POST" action="/cgi-bin/elabora">
    Scrivi i tuoi commenti:
   <Textarea
   name="commenti" rows="4" columns="14">
  Spiega in questo spazio la tua opinione
   </TEXTAREA>
   <Input type="submit" value="via!">
</FORM>
```


C:\WINDOWS\Desktop\prova.html - Microsoft Internet ...
File Modifica Visualizza Preferiti Strumenti ?
Scrivi i tuoi commenti:
Spiega in questo spazio la tua opinione
via!

HTML5: many more types!

See https://www.w3schools.com/html/html_form_input_types.asp

Form – more input types

- <input type="button">
- <input type="checkbox">
- <input type="color">
- <input type="date">
- <input type="datetime-local">
- <input type="email">
- <input type="file">
- <input type="hidden">
- <input type="image">
- <input type="month">
- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="tel">
- <input type="text">
- <input type="time">
- <input type="url">
- <input type="week">

https://www.w3schools.com/html/html_form_input_types.asp

Forms – how many buttons?

- Up to HTML 4:
  - At most 2: “exec” and ”cancel”

- HTML 5: as many as you want!

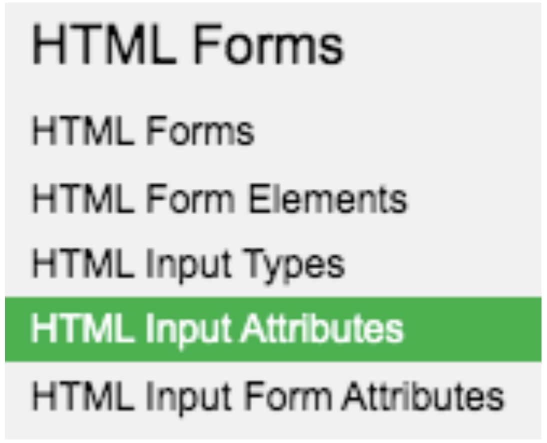
```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formaction="/action_page2.php" value="Submit as Admin">
</form>
```

HTML Form: attributes

Various attributes allow customizing Input and forms, e.g.:

- formaction
- formenctype
- formmethod
- formtarget

W3schools:



HTML Forms

HTML Forms

HTML Form Elements

HTML Input Types

HTML Input Attributes

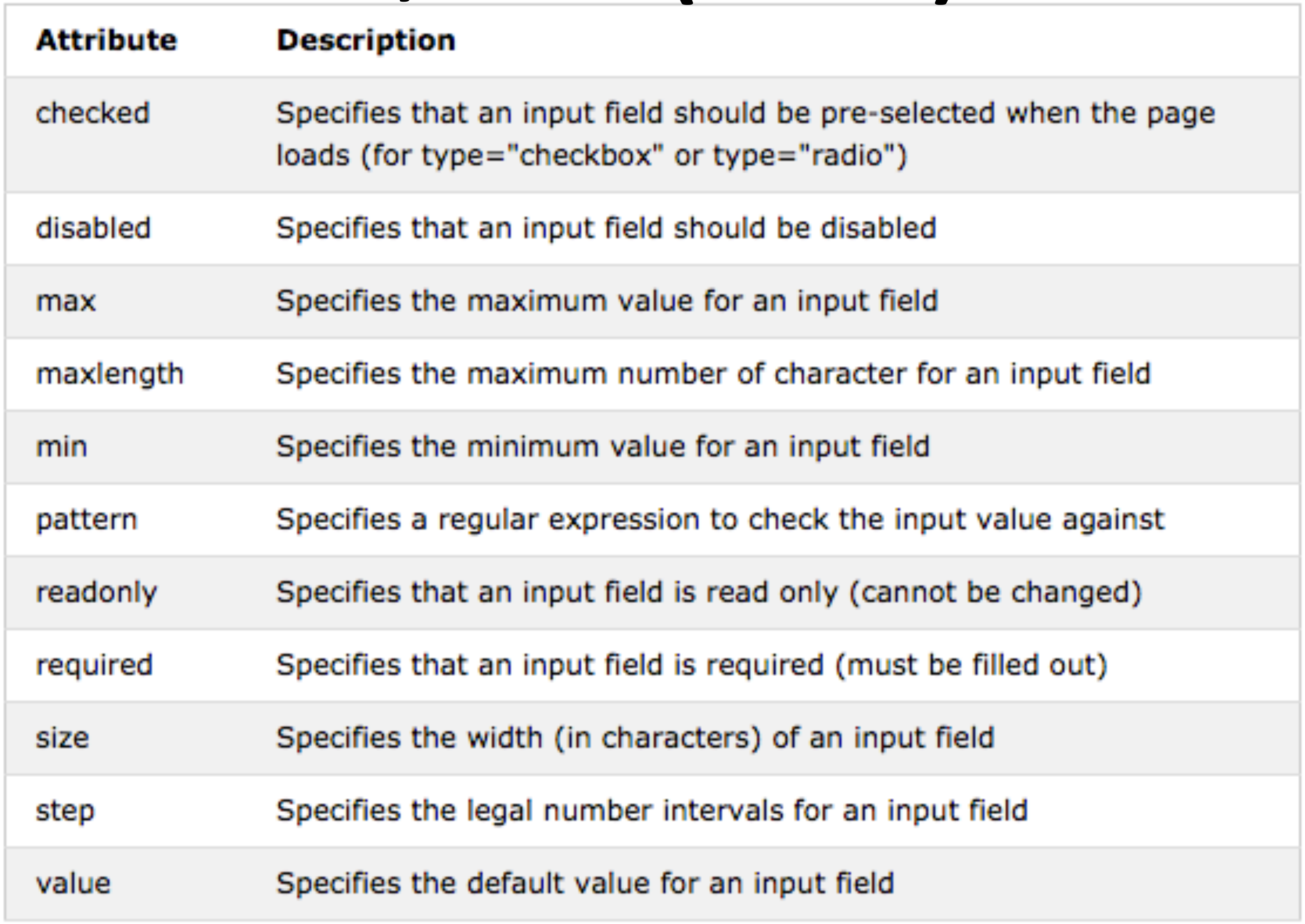
HTML Input Form Attributes

https://www.w3schools.com/html/html_form_attributes_form.asp

HTML Form, with (some) validation

- https://www.w3schools.com/html/html_forms.asp

HTML Form, with (some) validation



Attribute	Description
checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

https://www.w3schools.com/html/html_form_attributes.asp

Forms – restrictions: patterns

```
<form>
  <label for="phone">Enter your phone number:</label>
  <input type="tel" id="phone" name="phone"
          pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

The pattern attribute works with the following input types:
text, date, search, url, tel, email, and password.