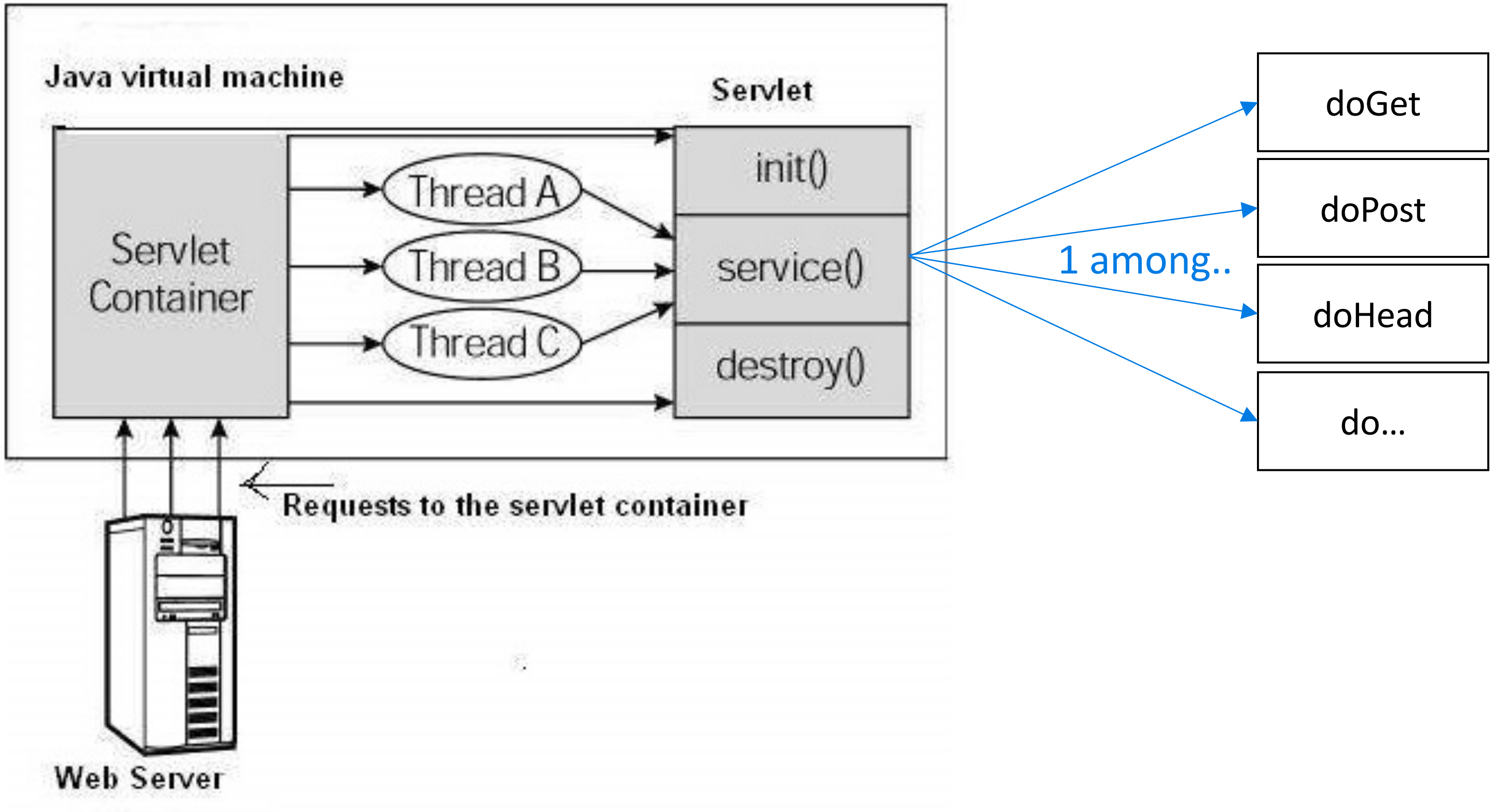
Q

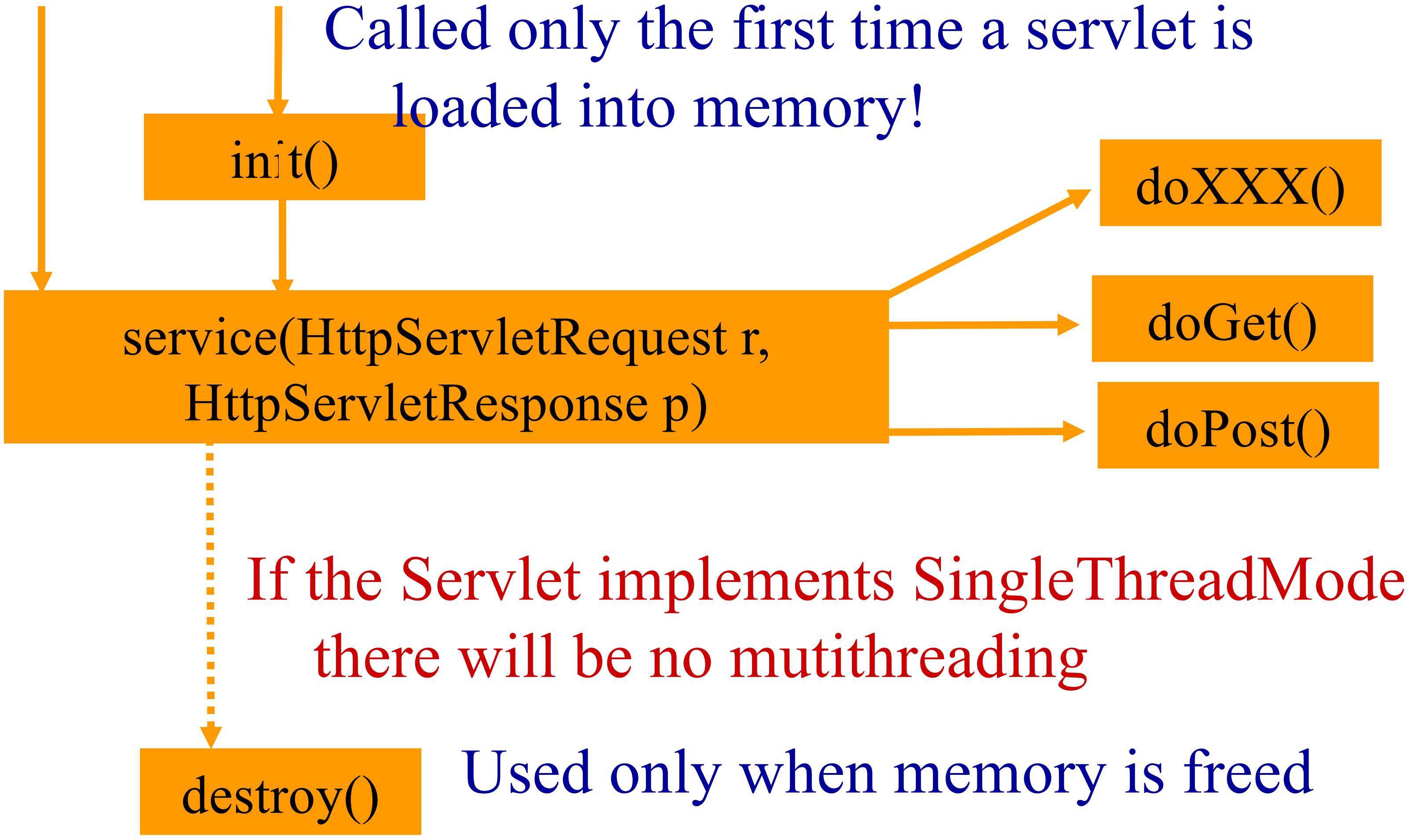
Why are servlet so efficient?

Servlet lifecycle



See https://www.tutorialspoint.com/servlets/servlets-life-cycle.htm

Servlet Lifecycle



Called only the first time a servlet is loaded into memory!

init()

doXXX()

service(HttpServletRequest r, HttpServletResponse p)

doGet()

doPost()

If the Servlet implements SingleThreadModel there will be no mutithreading

destroy()

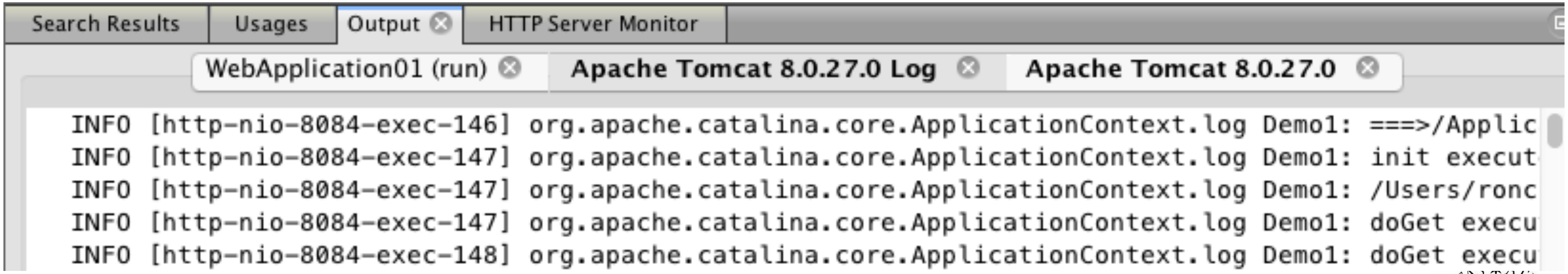
Used only when memory is freed

This code is part of the class HttpServlet

```
protected void service(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
      {
        String method = req.getMethod ();
        if (method.equals ("GET")) {
            ...
            if (ifModifiedSince == -1 || lastModified == -1) doGet (req, resp);
      } else if (method.equals ("HEAD")) { ... ; doHead (req, resp);
      } else if (method.equals ("POST")) { doPost (req, resp);
      } else if (method.equals ("PUT")) { doPut(req, resp);
      } else if (method.equals ("DELETE")) {doDelete(req, resp);
      } else if (method.equals ("OPTIONS")) {doOptions(req,resp);
      } else if (method.equals ("TRACE")) { doTrace(req,resp);
      } else {
         resp.sendError (HttpServletResponse.SC_NOT_IMPLEMENTED,
            "Method '" + method + "' is not defined in RFC 2068");
     }
 }
```

```
@Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
}
@Override
    public void destroy() {
        log("destroy executed");
}
@Override
    public void init() {
        log("init executed");
}
```

Logging



```
INFO [http-nio-8084-exec-146] org.apache.catalina.core.ApplicationContext.log Demo1: ===>/Applic
INFO [http-nio-8084-exec-147] org.apache.catalina.core.ApplicationContext.log Demo1: init execut
INFO [http-nio-8084-exec-147] org.apache.catalina.core.ApplicationContext.log Demo1: /Users/ronc
INFO [http-nio-8084-exec-147] org.apache.catalina.core.ApplicationContext.log Demo1: doGet execu
INFO [http-nio-8084-exec-148] org.apache.catalina.core.ApplicationContext.log Demo1: doGet execu
```

Q

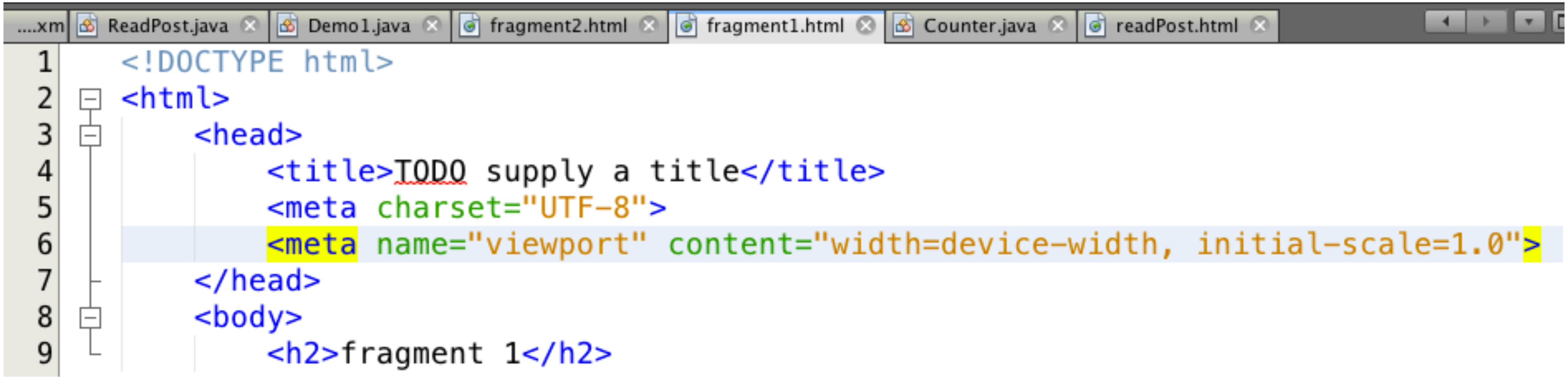
How can we factor out standard pieces of HTML (e.g. header, footer)?

Factoring out some HTML

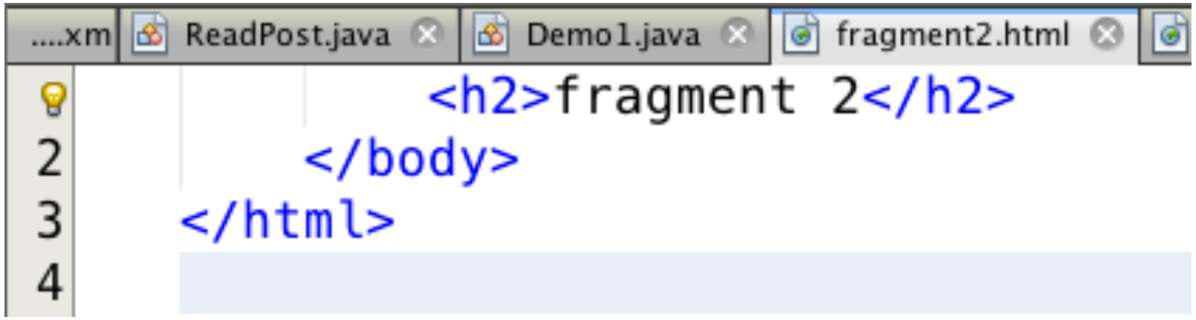
- How can we avoid this horrible stuff?

```
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet ReadPost</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1> fname="+name+"</h1>");
out.println("</body>");
out.println("</html>");
```

Factoring out some HTML

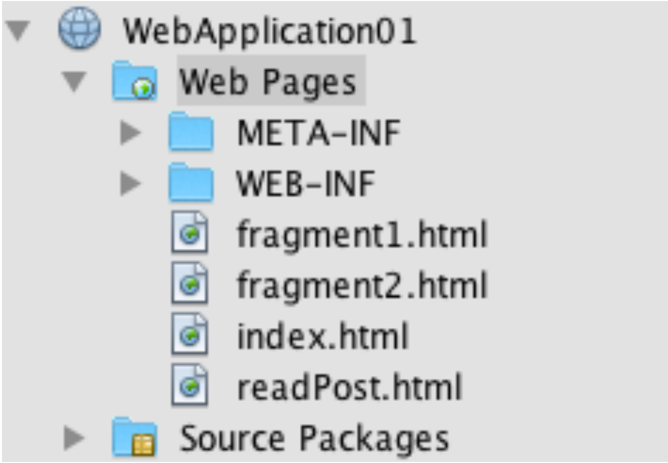


```
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h2>fragment 1</h2>
```



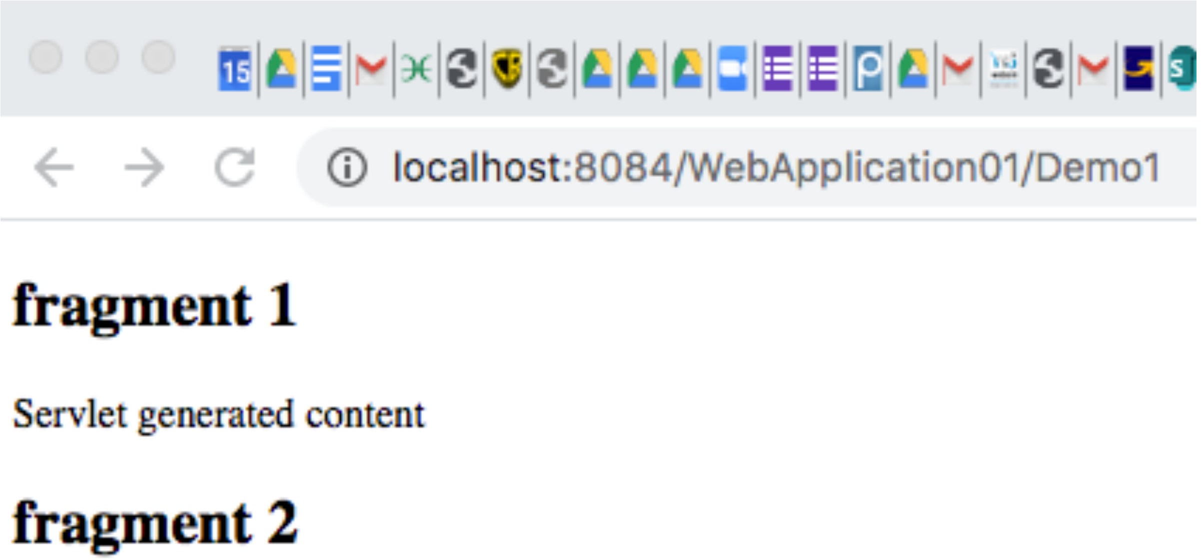
```
        <h2>fragment 2</h2>
    </body>
</html>
```

```
@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);
          out.println("Servlet generated content");
         request.getRequestDispatcher("/fragment2.html")
                .include(request, response);
      }
}
```



```
▼ WebApplication01
   ▼ Web Pages
      ▶ META-INF
      ▶ WEB-INF
        fragment1.html
        fragment2.html
        index.html
        readPost.html
   ▶ Source Packages
```

output



fragment 1

Servlet generated content

fragment 2

Invoking another servlet

```
request.getRequestDispatcher("/myServet")
                .forward(request, response);
```

Q

How can we keep global information in a webApp?

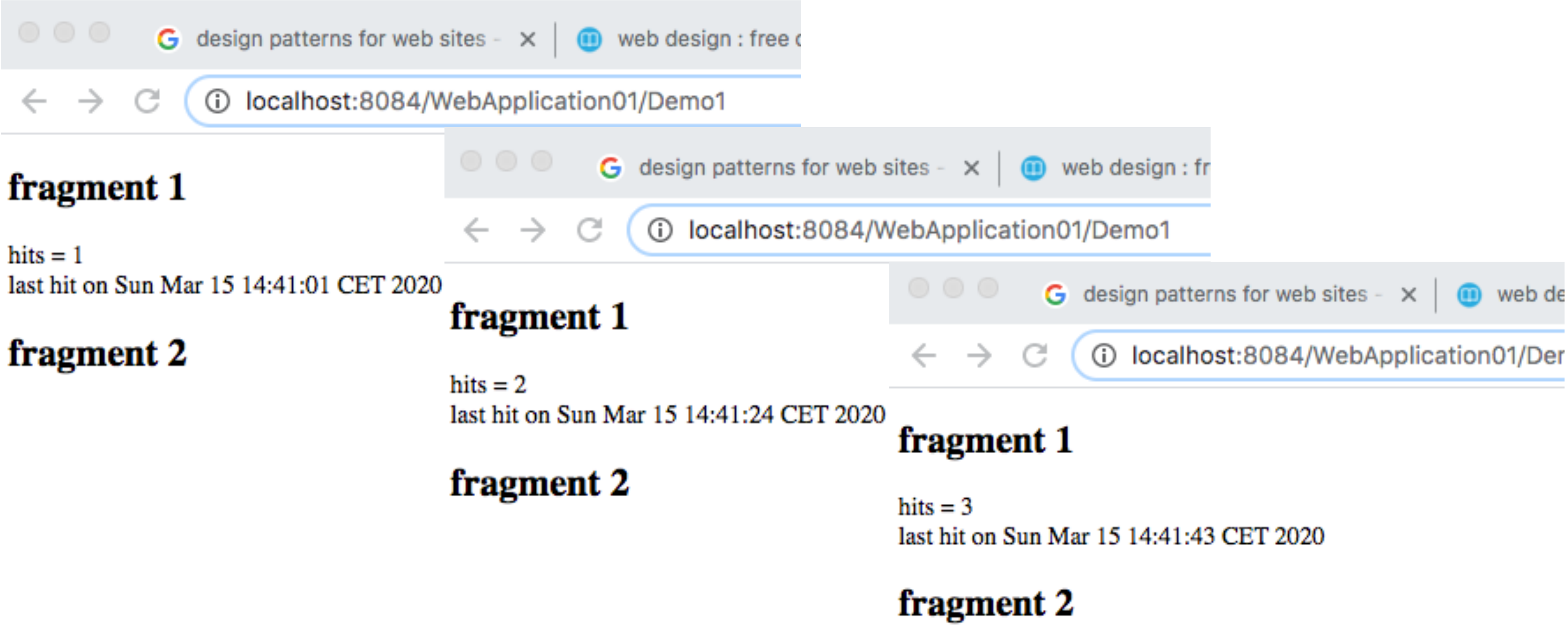
Let us build a hit counter - 1

```
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 c
localhost:8084/WebApplication01/Demo1
design patterns for web sites -
web design : fr
localhost:8084/WebApplication01/Demo1
design patterns for web sites -
web de
localhost:8084/WebApplication01/Der

```
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
```

But if we restart the server, counter restarts from 1!

Q

How can we persist (global) information in a webApp?

How can we persist the counter?

In a file (named counterData)!

1) In init, let us check if file exists. If yes, let us resume the counter, else, let us create a new one.
2) In destroy, let us save counter in conterData.

Java serialization

class A implements Serializable {…}

```
A a1=new A();
A a2;
 …
File myFile = new File(filePath);
…
ObjectOutputStream oi = new ObjectOutputStream(new
          FileOutputStream(myFile));
oi.writeObject(a1);
…
ObjectInputStream oi = new ObjectInputStream(new
          FileInputStream(myFile));
a2 = (A) oi.readObject();
```

(throws various exceptions…

(throws various exceptions…

Let us build a hit counter - 1

```
public class Counter implements Serializable {
    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();
    }
}
```

!

Wait – our counter solution is not thread-safe!

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

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)

```
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,
            …
    }
}
```

<%@ directives %>

<%! declarations %>

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 Standard actions

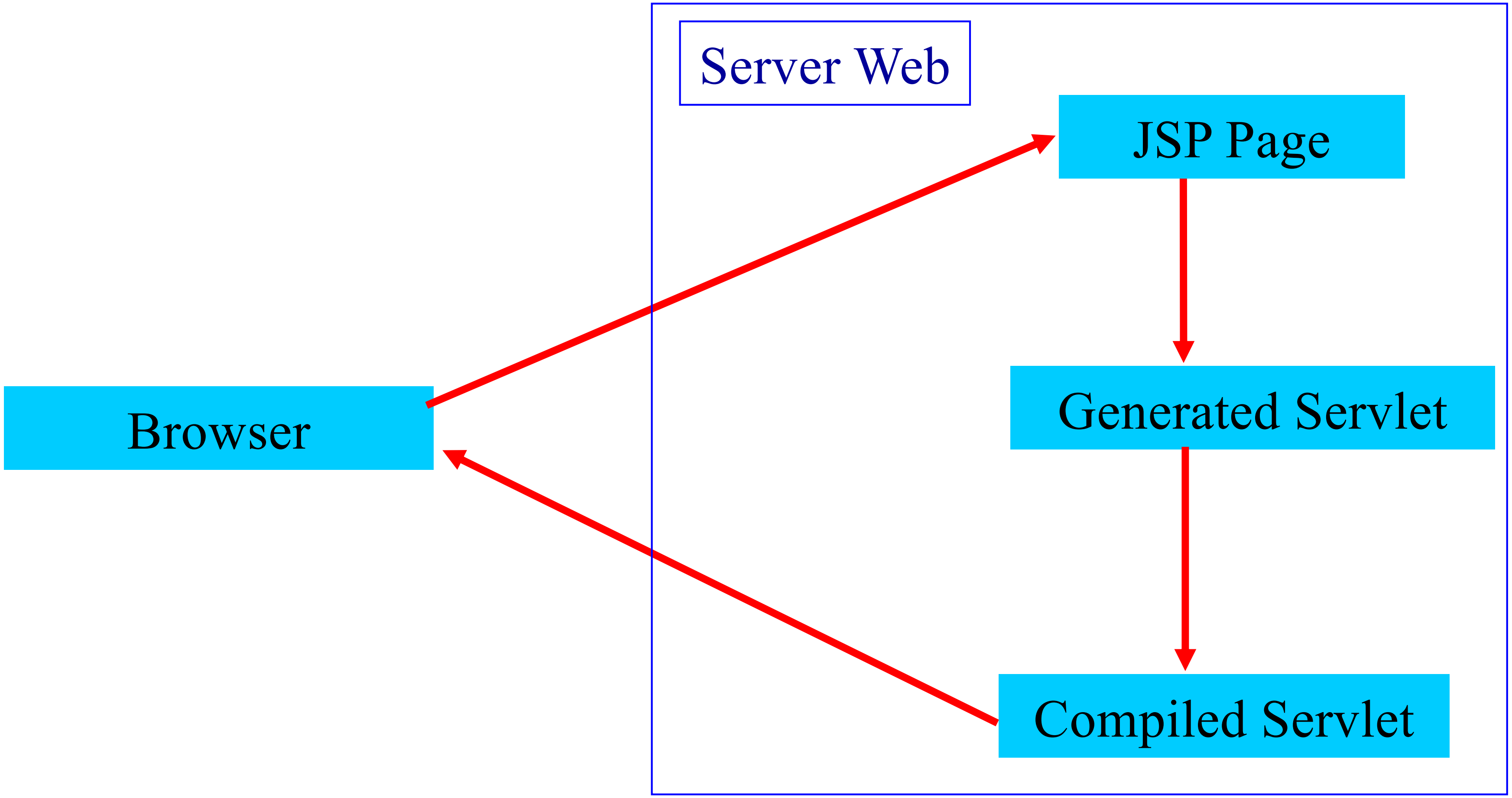
```
<jsp:include page=“URL” />
```

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



Server Web

JSP Page

Generated Servlet

Compiled Servlet

Browser

Where is the generated code?

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

Q

How can I access request and response?

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

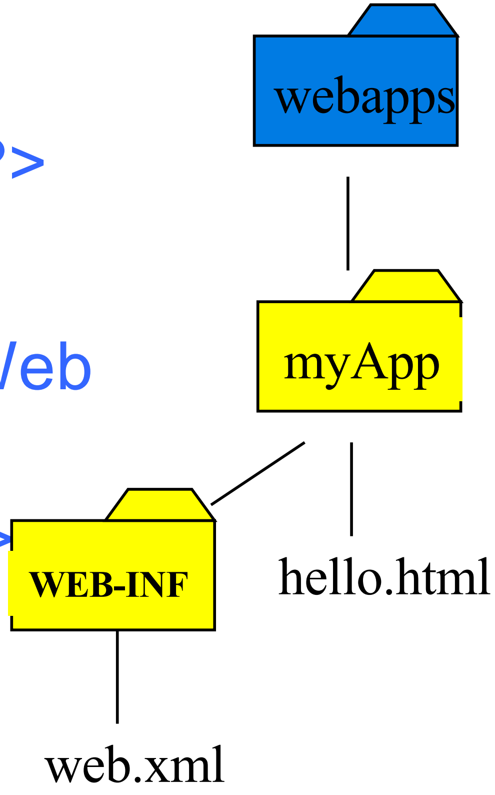
```
<%@ page errorPage="errorpage.jsp" %>
<html>
  <head>
    <title>UseRequest</title>
  </head>
  <body>
    <%
        // Get the User's Name from the request
        out.println("<b>Hello: " + request.getParameter("user") + "</b>");
    %>
  </body>
</html>
```

Q

How should I configure Tomcat to use JSPs?

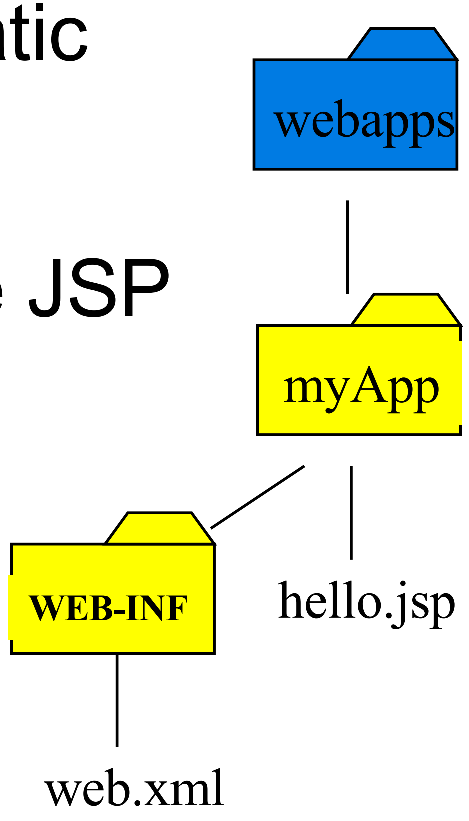
Static pages

- A web.xml file MUST be provided:
- <?xml version="1.0" encoding="ISO-8859-1"?>
- <!DOCTYPE web-app
-     PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN“
-     "http://java.sun.com/dtd/web-app_2_3.dtd">
- <web-app>
- </web-app>



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.

web.xml

Java web applications use a deployment descriptor file named web.xml to determine many things, such as how URLs map to servlets, which URLs require authentication, etc..

web.xml resides in the app's WAR under the WEB-INF/ directory.

See https://cloud.google.com/appengine/docs/standard/java/config/webxml

Why did we not use web.xml so far?

Some of the info expected in the web.xml can be provided via annotation. E.g.

```
package it.unitn.disi.ronchet.myservlets;
@WebServlet(name=”myServlet”,
       urlPatterns = {"/welcome"})
public class Welcome extends HttpServlet
```

Is equivalent to

```
</web-app>
    <servlet>
        <servlet-name>myServlet</servlet-name>
        <servlet-class>it.unitn.disi.ronchet.myservlets.Welcome
        </servlet-class>
    </servlet>
    <servlet-mapping
         <servlet-name>myServlet</servlet-name>
         <url-pattern>/welcome</url-pattern>
    </servlet-mapping>
</web-app>
```

web.xml and annotations together

Whatever is defined in web.xml overwrites annotations.

Try it!

Redefine the URL via web.xml, and see who wins between annotation and configuration.

Q

What does HTML5 add to Forms?

Form – more input types

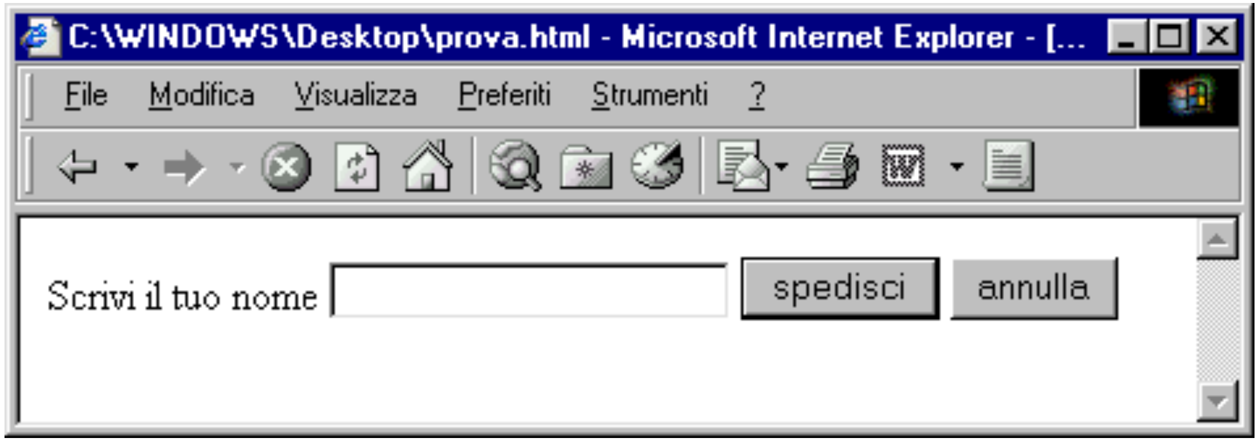
- <input type="button">
- <input type="checkbox">
- <input type="color">
- <input type="date">
- <input type="datetime-local">
- <input type="email">
- <input type="file">
- <input type="hidden">
- <input type="image">
- <input type="month">
- <input type="number">
- <input type="password">
- <input 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">

HTML5: many more types!

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

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>
```



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

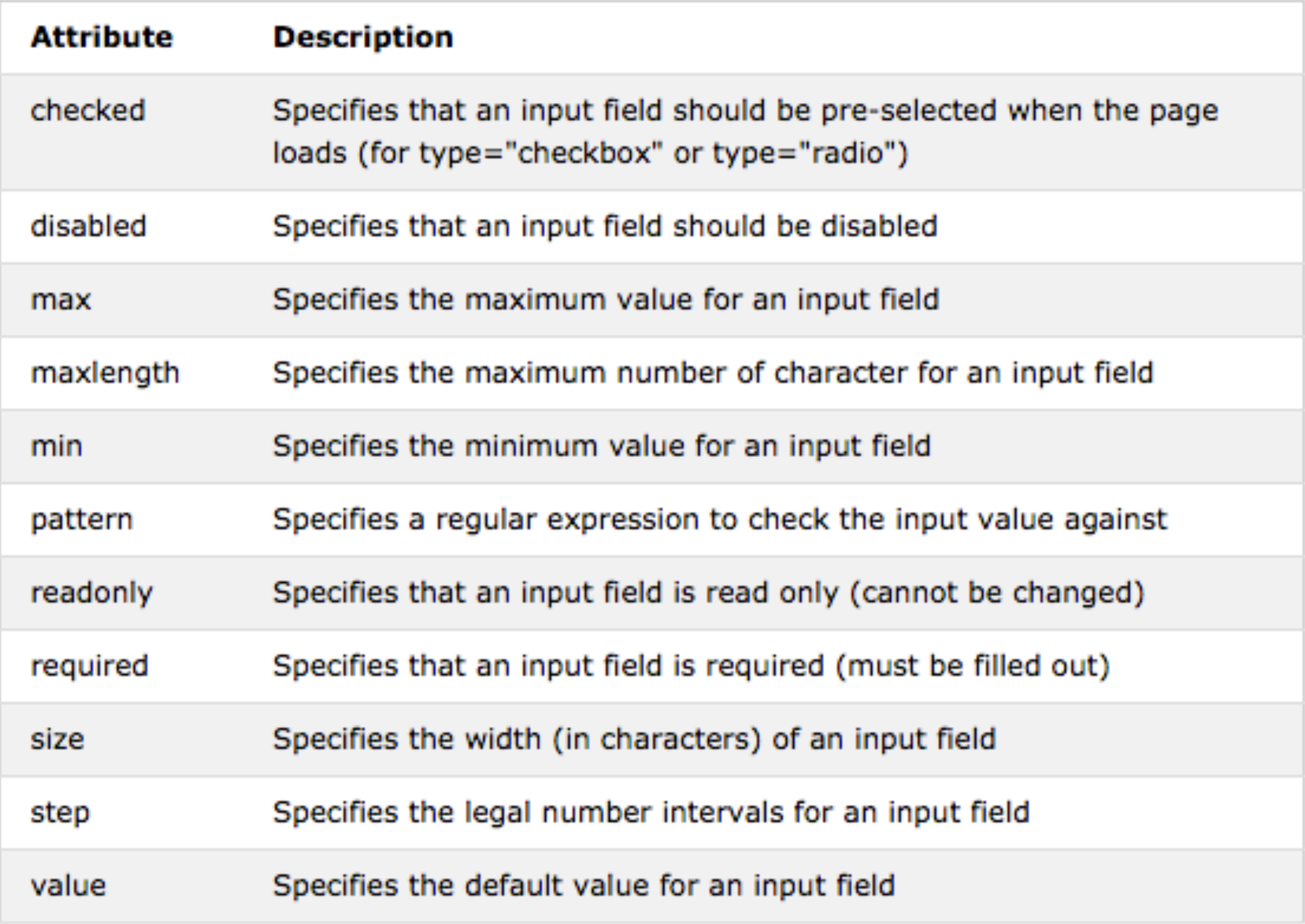
Forms – how many buttons?

- Up to HTML 4:
  - At most 2: “submit” 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>
```

Forms – restrictions



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

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.

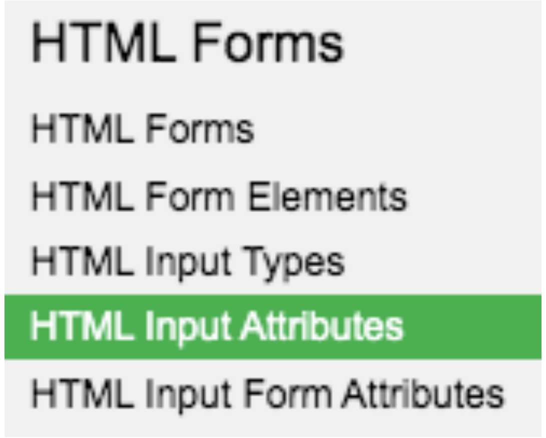
HTML Form: attributes

Various attributes allow customizing Input and forms.

See:

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

W3schools:



HTML Forms

HTML Forms

HTML Form Elements

HTML Input Types

HTML Input Attributes

HTML Input Form Attributes