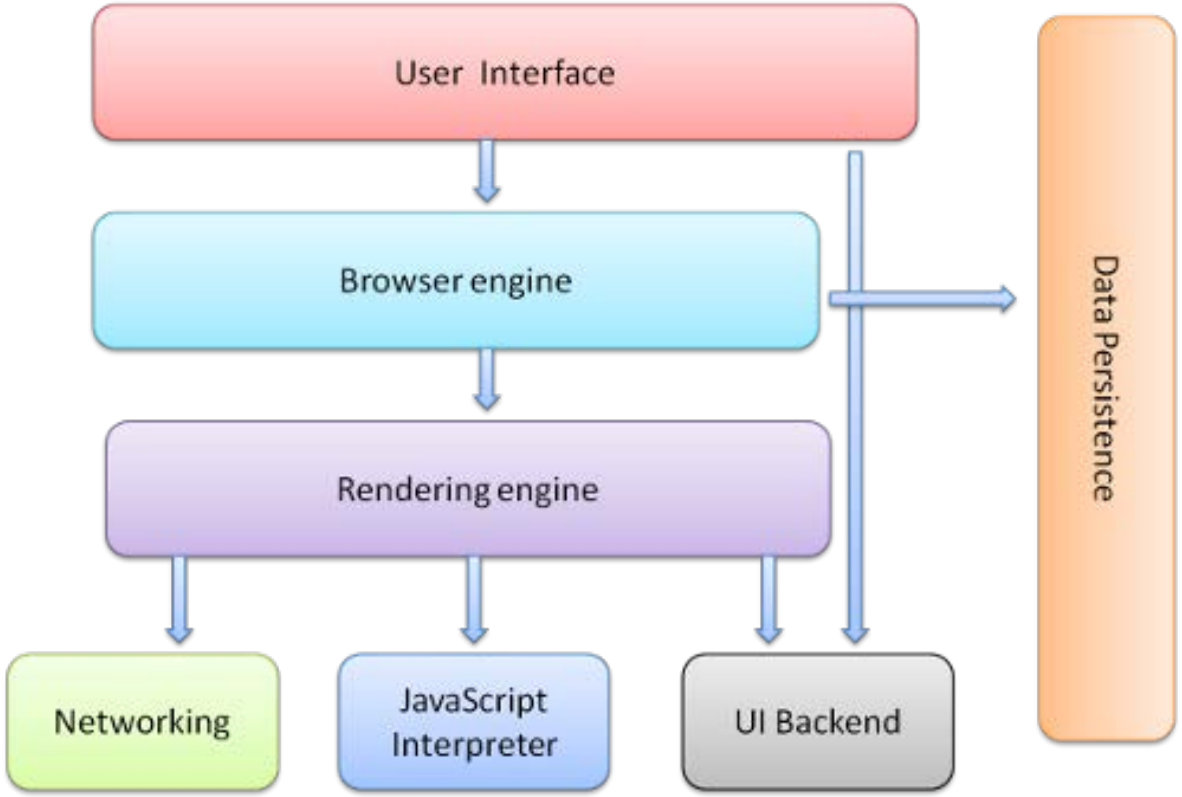
Server Programming
Week 5 Lecture

Outline

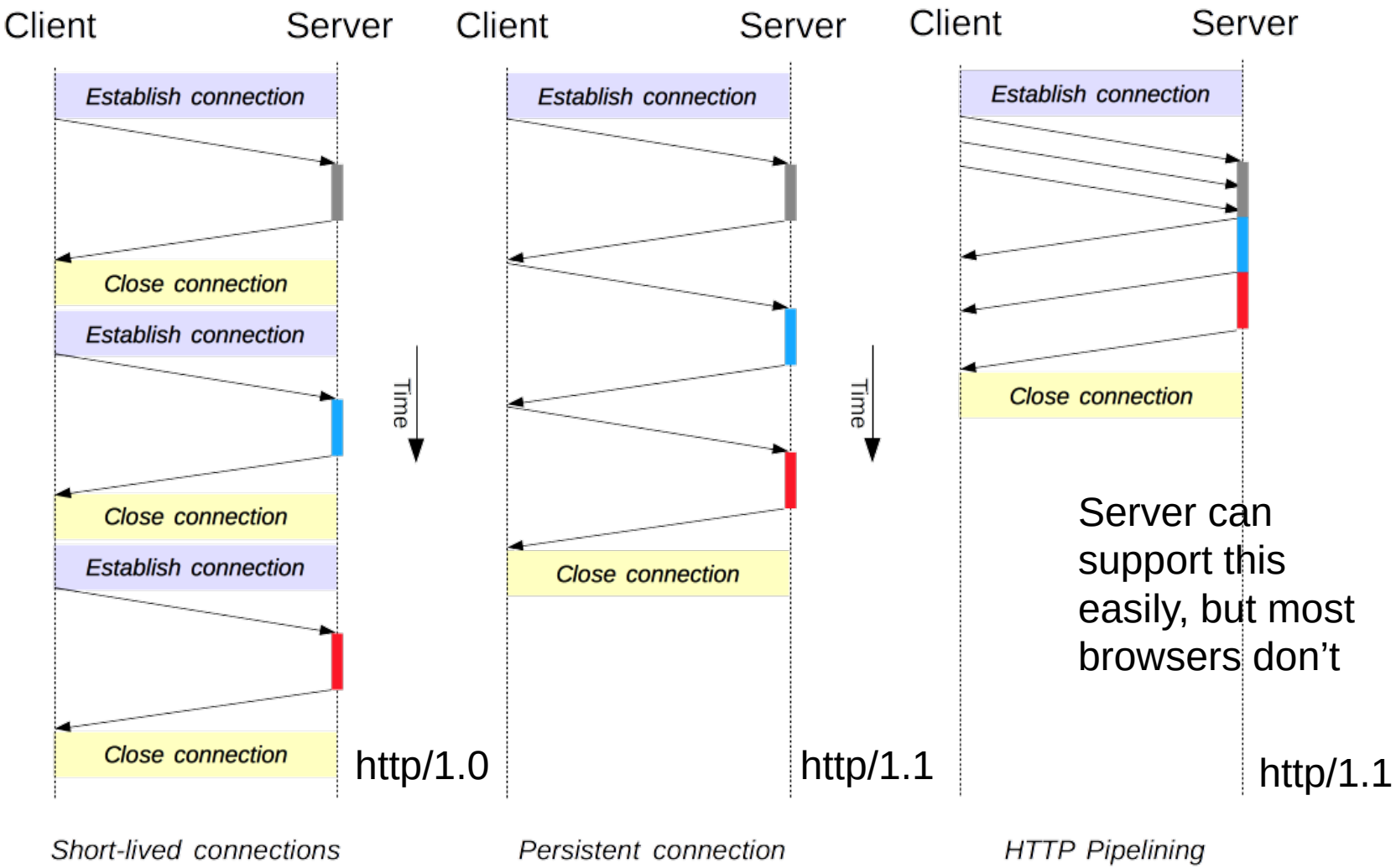
- **Review of Browser**
  - **How browser works**
  - **HTTP Connection Management**
- **How web server works**
- **Java based server application**
  - **Basic Servlet Structure**
  - **Pass data with GET and POST method**
  - **JSP**

How Browsers Work



http://taligarsiel.com/Projects/howbrowserswork1.htm

HTTP Connection Management



https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x

Parallel Connections

- With pipelining, all HTTP/1.x connection is serializing requests
  - Etiher tcp, req1, res1, tcp, req2,res2, … or
  - Tcp, req1,res1,req2,res2
- To improve performance, browser open several connections to each domain, sending parallel requests
  - The maximum number of parallel connection is quite small to avoid giving server the impression of DoS attack
  - Maximum concurrent connections:
    - Chrome: 6
    - IE: 2~8
    - Safari

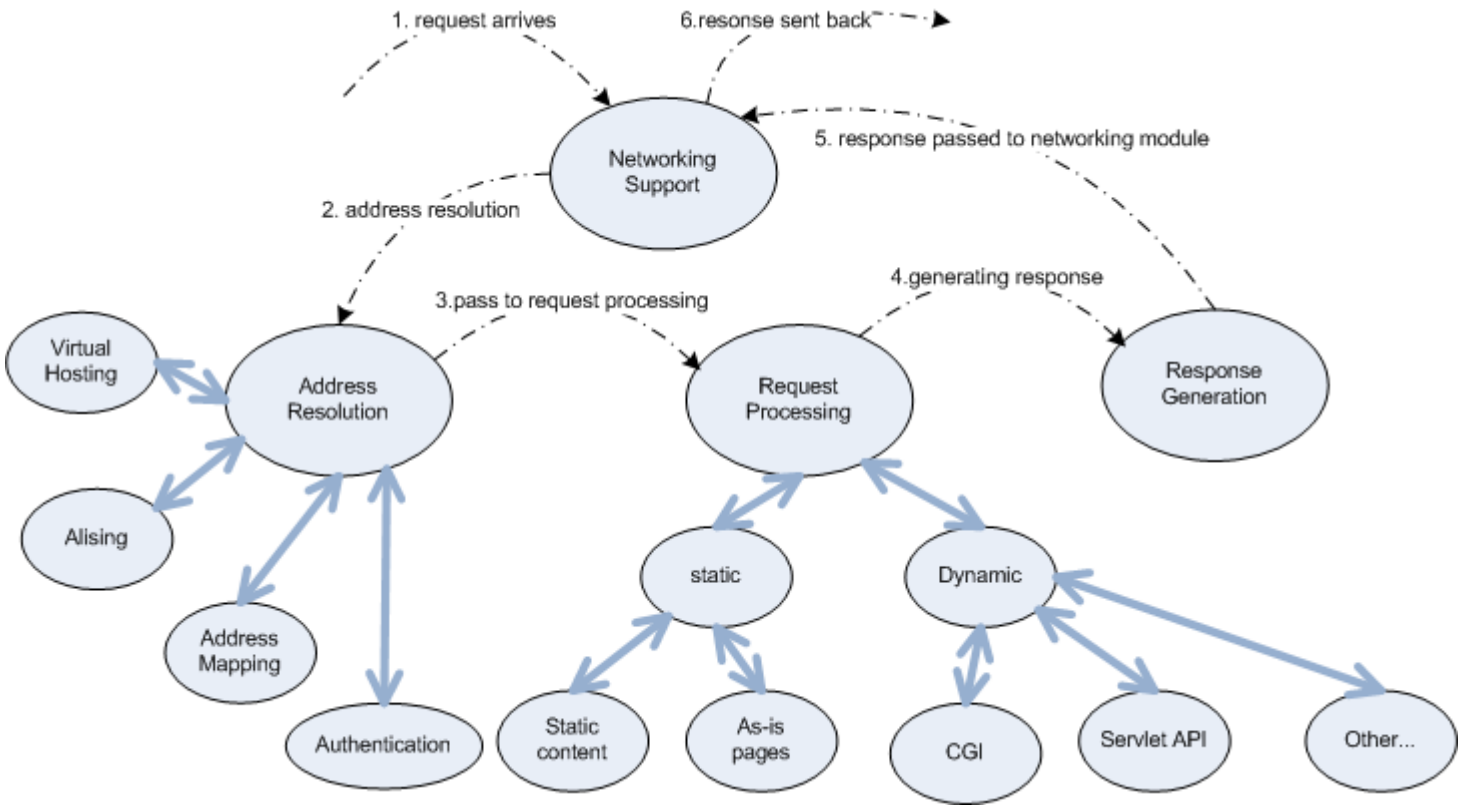
http://sgdev-blog.blogspot.com.au/2014/01/maximum-concurrent-connection-to-same.html

Outline

- Review of Browser
  - How browser works
  - HTTP Connection Management
- How web server works
- Java based server application
  - Basic Servlet Structure
  - Pass data with GET and POST method
  - JSP

How server works

- Most basic web server acts like a file system while file requests are sent using HTTP protocol
- Modern web server manages other resources in addition to static files.



HTTP request processing

- Networking support
  - Persistent connection
  - Multiple requests arrive in one connection
  - Server needs to ensure that responses are sent back in the order of request arrival
- Address Resolution
  - Simple example
    - A file stored under the absolute path

    /usr/mit/*yourlogin*/lib/html/week2.html

    - Is accessible through this URL:

    http://www.it.usyd.edu.au/~*yourlogin*/week2.html
  - Apache administrator has set the mapping between URL path info and local file system path info
    - /~*yourlogin*/week2.htm is mapped to /usr/mit/*yourlogin*/lib/html/week2.html in local file system

Static and Dynamic content

- Static contents are stored in local file system
  - *Static content page*: HTML page, plain text, image files, etc
    - Use the predefined mapping to locate the file
    - Construct HTTP response with header information
  - *As-is page*: static file containing complete HTTP responses (including headers)
    - No response construction is required
    - Server needs a way to tell if a file is as-is. Normally use file extension to indicate.
- Dynamic contents request explicit server side programmatic action to generate response
  - PHP, Perl scripts
  - Application server

Dynamic Content Delivery

- How does a server differentiate between static, dynamic contents and find the correct handlers to generate the dynamic content?
- Address resolution rely on URL path and file extension
  - E.g.
  - ***/~login/xxx.html*** => static content file located under `/usr/mit/login/lib/html/xxx.html`
  - ***/~login/cgi-bin/xxx.cgi*** => cgi file requests language specific interpreter.
  - A url path beginning with **/servlet/** may indicate that the target is a Java Servlet
  - Application server provides configuration files for the mapping

```
– <servlet-mapping>
–     <servlet-name>Greeting</servlet-name>
–     <url-pattern>/Greeting</url-pattern>
–   </servlet-mapping>
```

  - "Requested resources xxx is not available" problem is normally caused by not properly configured mapping

Server Side Program Forms

- As a “script”
  - CGI scripts, or Java Servlets, or JavaScript
  - Just like a regular program
  - Parse request using regular expression
  - Write (HTML) response using stream operations provided by the language
- As a Server Page(template processing)
  - PHP, ASP, JSP
  - Program is structured around the returning text page
  - Script code are inserted into the HTML code to execute at certain points
- Combination based on MVC

Server Side Program

- All server side program follows the basic processing steps
  - Parse HTTP request to obtain various information carried in the request
    - Additional parameter, client restriction, cookie and so on
  - Do some processing based on the HTTP request information
  - Generate response
- These steps can be easily seen in “script” style program regardless of the language chosen
- They are hidden in “server page” style program
- There are usually supporting/external services for
  - networking: e.g. sending and receiving request/response
  - Common processing: e.g. extracting header information from both request and response messages
  - Others
- Such supporting services are either provided by <u>server</u> (software such as application server) or <u>language framework</u>

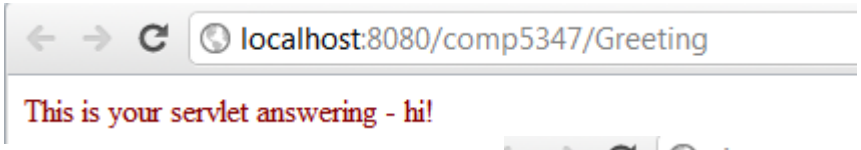
Outline

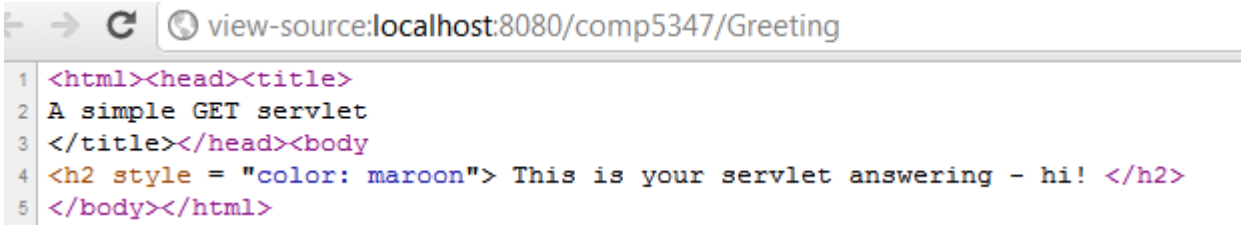
- Review of Browser
  - How browser works
  - HTTP Connection Management
- How web server works
- Java based server application
  - Basic Servlet Structure
  - Pass data with GET and POST method
  - JSP

Simple Servlet example

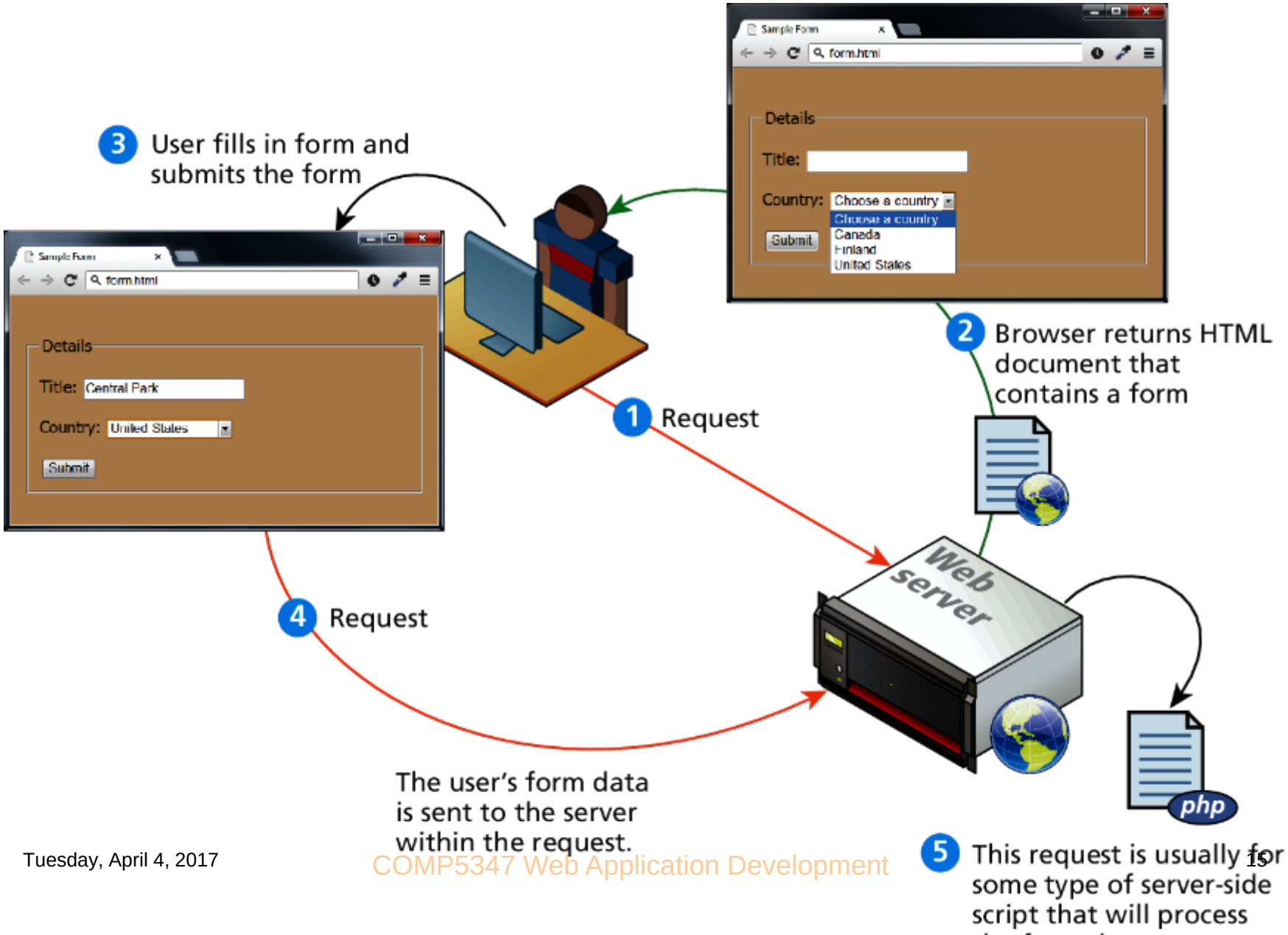
Greeting.java

```
@WebServlet("/Greeting")
public class Greeting extends HttpServlet {
    public Greeting() {
        super();

    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        PrintWriter returnHTML;
        response.setContentType("text/html");
        returnHTML = response.getWriter();
        returnHTML.println("<html><head><title>");
        returnHTML.println("A simple GET servlet");
        returnHTML.println("</title></head><body");
        returnHTML.println("<h2 style = \"color: maroon\"> “This is your servlet answering - hi!
                </h2>");
        returnHTML.println("</body></html>");
        returnHTML.close();
    }
}
```





Sending data to server

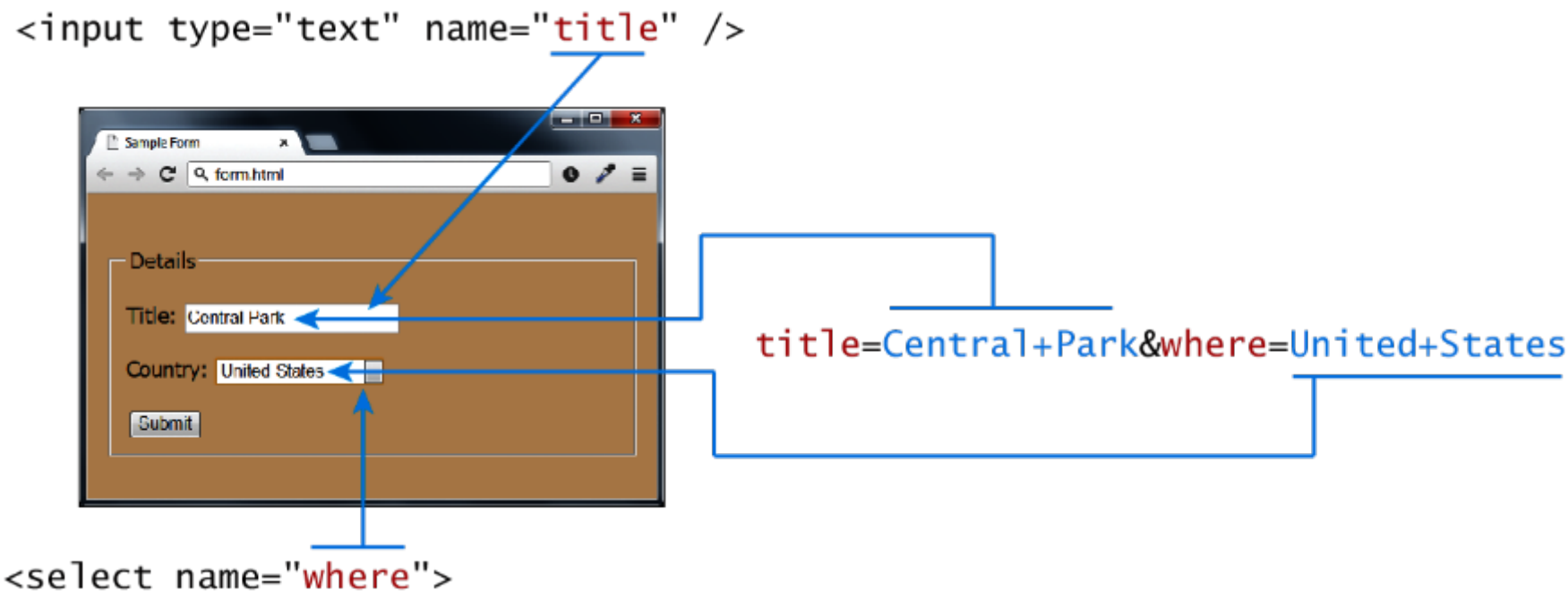


<form> element

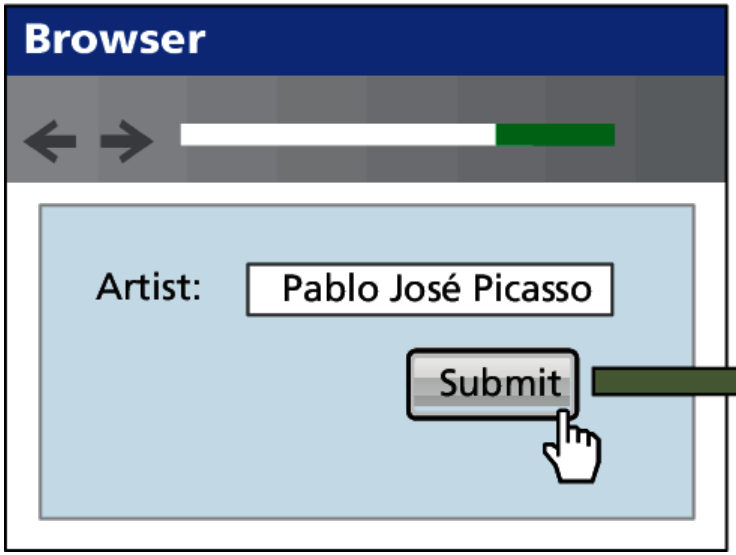
- There are two essential features of any form, namely the **action** and the **method** attributes.
  - The **action** attribute specifies the URL of the server-side resource that will process the form data
  - The **method** attribute specifies HTTP request method
    - GET
    - POST
- GET and POST methods send form data to server in different ways

GET Method

- GET method attached the form data as query string to URL
- General format of query string
  - http://example.com/over/there?title=test&where=here



URL Encoding



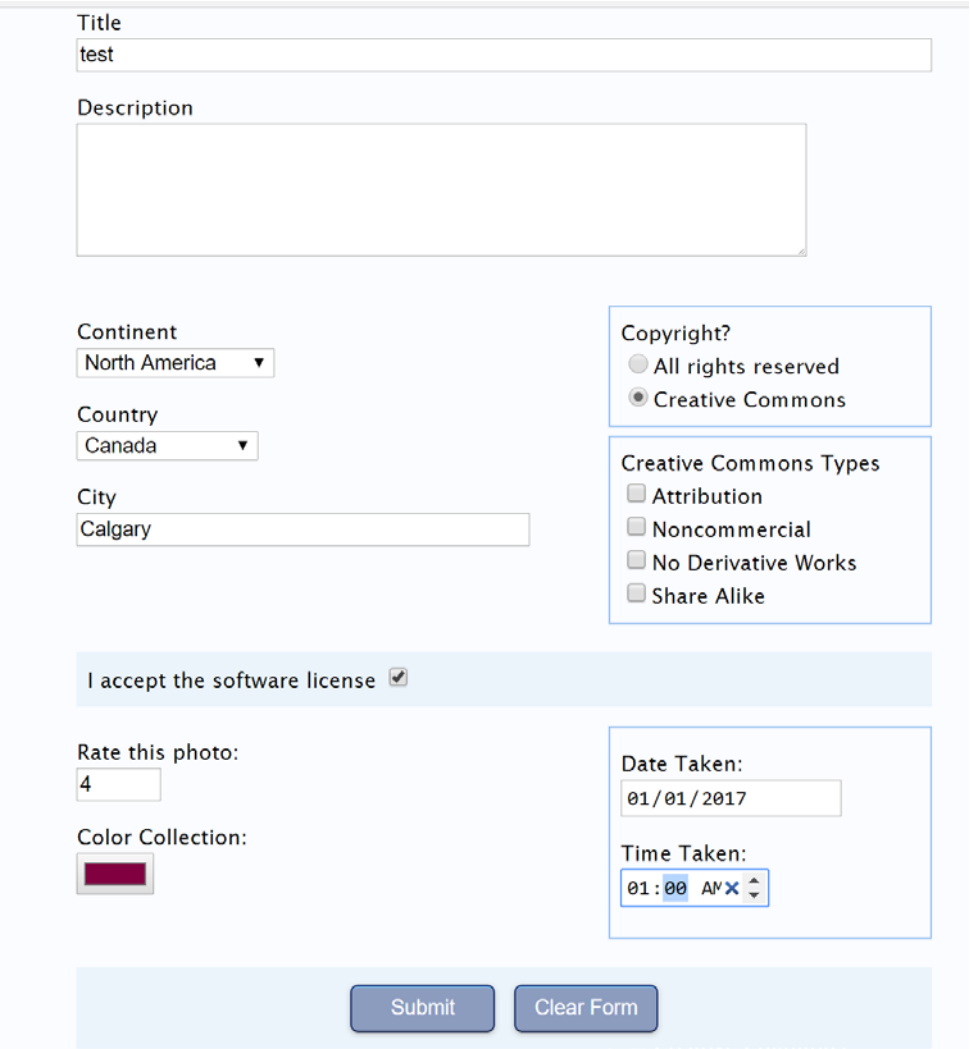
Notice how the spaces and the accented é are URL encoded (in red).

`artist=Pablo+Jos%E9+Picasso`

URL Encoding

POST Method

- POST method sends the form data as part of request body
  - The actual format may look similar to query string



POST request body

Headers Preview Response Timing

General

Request URL: file:///C:/Users/yzho8449/course/comp5347/2017/labs/code/week5-lecture/week3.html

Request Headers

Provisional headers are shown

Content-Type: application/x-www-form-urlencoded
Origin: null
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36

Form Data view source view URL encoded

title: test
description:
continent: North America
country: Canada
city: Calgary
copyright: 2
accept: on

Headers Preview Response Timing

General

Request URL: file:///C:/Users/yzho8449/course/comp5347/2017/labs/code/week5-lecture/week3.html

Request Headers

Provisional headers are shown

Content-Type: application/x-www-form-urlencoded
Origin: null
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36

Form Data view parsed

title=test&description=&continent=North+America&country=Canada&city=Calgary&copyright=2&accept=on&rate=4&color=%23800040&date=2017-01-01&time=01%3A00

GET vs. POST

- GET
  - Data can be clearly seen in the address bar.
  - Data remains in browser history and cache.
  - Data can be bookmarked
  - Limit on the number of characters in the form data returned.
- POST
  - Data can contain binary data.
  - Data is hidden from user.
  - Submitted data is not stored in cache, history, or bookmarks.

GET vs. POST

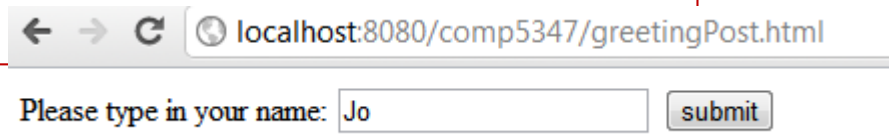
- Implication
  - GET is meant to be used to **query** something from the server **without changing** any server data
  - POST is meant to be used for sending data to be processed and **change something** on the server

Get information from request

- Getting parameters in url or from POST body

greetingPost.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Say Hello</title>
</head>
<body>
<form method="POST" action = "week2/sayHello">
Please type in your name: <input type = "text" name="clientName" ></input>
<input type = "submit" value = "submit"></input>
</form>
</body>
</html>
```

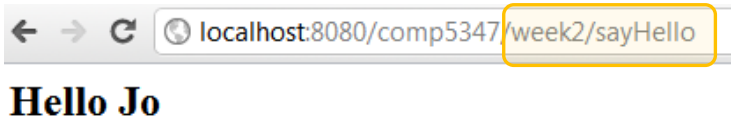


- HTTP POST
  - POST week2/sayHello HTTP/1.1
    clientName= Jo
- HTTP GET
  - GET week2/sayHello?clientName=Jo HTTP/1.1

Get Information from request

GreetingWithName.java

```java
package greeting;
public class GreetingWithName extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
            processRequest(request,response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
            processRequest(request,response);
    }
    private void processRequest(HttpServletRequest request, HttpServletResponse response)throws
            ServletException, IOException{
            String clientName = request.getParameter("clientName");
            PrintWriter returnHTML;
            response.setContentType("text/html");
            returnHTML = response.getWriter();
            returnHTML.println("<html><head><title>A simple GET servlet</title></head>");
            returnHTML.println("<body><h2> Hello " + clientName + " </h2></body></html>");
            returnHTML.close();
    }
}
```



web.xml

```xml
<servlet>
    <servlet-name>GreetingWithName</servlet-name>
    <servlet-class>greeting.GreetingWithName</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>GreetingWithName</servlet-name>
    <url-pattern>/week2/sayHello</url-pattern>
  </servlet-mapping>
```

Java Server Page

- JSP takes the “template processing” approach
  - It looks like HTML page (template) but contains other types of text for processing purpose
  - JSP page needs to be processed at server side to be converted into normal HTML page
- JSP technology has evolved over time, there are all sorts of things in various JSP pages.
  - Snippet of Java Code (Java scriptlet) and Expressions
  - Special elements that looks like HTML elements but can only be processed by a web server

The earliest version of JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<%!
int count = 0;
%>
<html>
<head>
<title>Count the number of visits</title>
</head>
<body>
<%
session = request.getSession(true);
// Set the session valid for 5 secs
session.setMaxInactiveInterval(5);
if (session.isNew()) {
count++;
}
%>
You have visit the page <%= count %> times.
</body>
</html>
```

directives

declarations

Java scriplet

Java expressions

JSP with JSTL elements

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2     pageEncoding="ISO-8859-1"%>
3 <%@ taglib prefix="c" uri = "http://java.sun.com/jsp/jstl/core" %>
4 <%@ taglib prefix ="sql" uri = "http://java.sun.com/jsp/jstl/sql" %>
5
6 <sql:setDataSource driver="com.mysql.jdbc.Driver"
7 url="jdbc:mysql://localhost:3306/comp5347shops_development" scope="session"
8 user="root" password="basser" />
9
10 <sql:query var="categories" >
11 SELECT title, description,img_url FROM categories;
12 </sql:query>
13
14 <html>
15 <head>
16 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
17 <title>JSTL example</title>
18 </head>
19 <body>
20 <table border = "1" cellpadding="4">
21 <tr>
22 <td>Title</td><td>Description</td><td>Img_url</td>
23 </tr>
24 <c:forEach var="category" items="${categories.rows}">
25 <tr>
26 <td>${category.title}</td>
27 <td>${category.description}</td>
28 <td>${category.img_url}</td>
29 </tr>
30 </c:forEach>
31 </table>
32 </body>
33 </html>
```

directives

JSP Expression Language (EL)

http://localhost:9080/SimpleApp/pagecountJSTL.jsp

Title	Description	Img_url
lab solution	all lab solutions	/ai.png

MVC structure

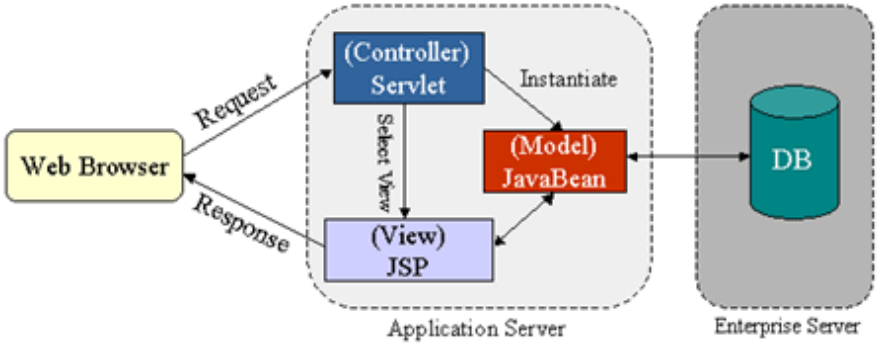


Diagram from http://java.sun.com/developer/technicalArticles/javaserverpages/servlets_jsp/

- General structure
  - User requests are directed to the *controller* servlet
  - The *controller* servlet accesses and modifies the *model*(s), possibly delegating the processing to helper classes
  - The controller servlet selects and passes control to the appropriate JSP page responsible for presenting the view
  - The view accesses models to obtain data for display
  - The view is processed and returned to the requesting user

References

- Head First Servlets & JSP
  - Chapter 2 and 7
- Web application architecture. 2nd edition, Wiley
  - Chapter 6