Welcome to Web

How things works on WEB?

- Lets discuss something
  - Client
    - Web Browser (IE, Firefox)
  - Server
    - Web Server (Tomcat, IIS)
  - Protocol
    - HTTP
  - Language
    - HTML

Client

As we are talking about web related stuff so, for us client is our very own browsers. Whichever you preferably use.

- What does a Web Client do?
  - Lets user request something
  - Shows the result of the request

Server

You may consider Server as a Hardware Machine or a Software Application, hardly matters. We use both terms for it. Here we are talking about Web Servers.

- What does a Web server do?
  - It takes client's request
  - Gives something back to the client

Protocol

TCP/IP, sounds familiar? These are the agents who takes the burden of delivering. Guess what FedEx do! Here we'll talk about HTTP.

- What does HTTP do?
  - It takes client request to server
  - It brings server response to client

Language

There are so many languages in the World, I understand English but, a web browser understand HTML.

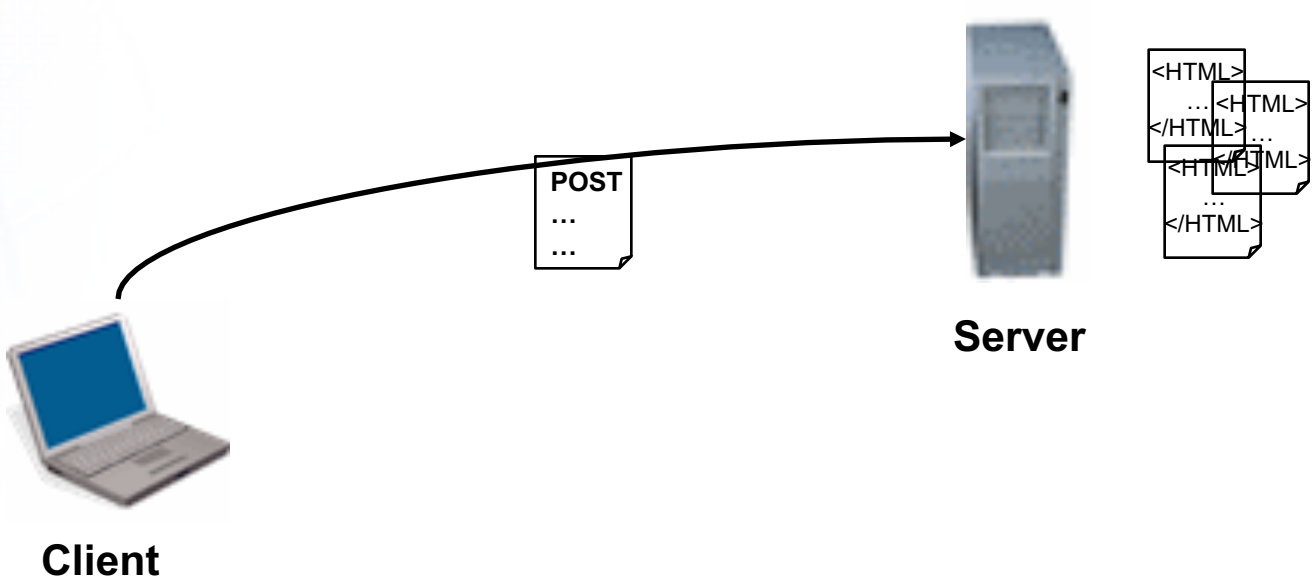
- What does HTML do?
  - Gives instructions to the browser
  - Makes our world colourful.

Lets put them together

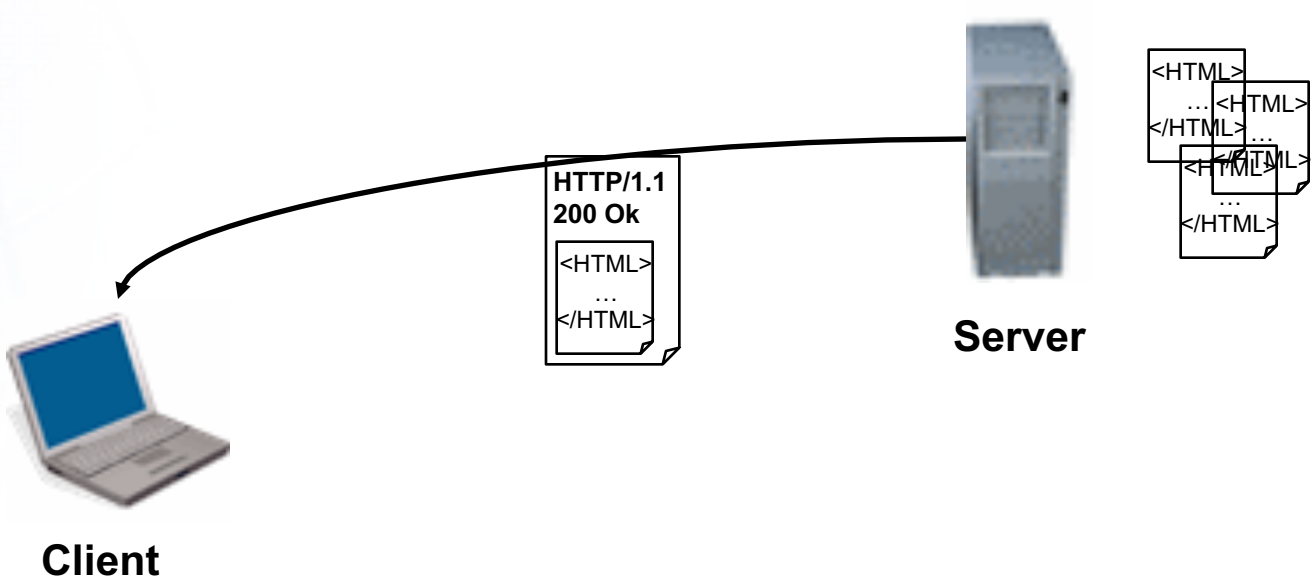
- Browser is client side proxy who sends request.
- Request is some kind of data which is taken by HTTP to the server.
- Web server is an Application who receives the request.
- Server sends the response as an HTML.

How this works?

Client sending the request



Returning response back



What is the HTTP Protocol?

- HTTP is just another network protocol
- Runs on top of TCP/IP
- It has Web specific features
- Conversation structure of HTTP is simple Request, Response sequence
- Takes request from browser and get server response

Request

- Request is browser's interaction with server.
- Key elements of request stream
  - HTTP method
    - Get
    - Post
  - The page URL
  - Form parameters

Response

- Response is servers reply to browser
- Key elements of response stream
  - Status code
    - 200 Ok
    - 404 Not Found
  - Content type
    - Image
    - HTML
  - Content

HTTP Request

- Every request has a header
- Header tells the status of client
- Form data send in different ways known as methods
- Data is sent via several methods

Request Methods

Method	Description
GET	Asks to get the thing(resource/file) at the requested URL.
POST	Asks the server to accept the body info attached. It like a GET with extra info sent with the request
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body
TRACE	Asks for the loopback of the request message, for testing or troubleshooting
PUT	Says to put the enclosed info (the body) at the requested URL
DELETE	Says to delete the thing (resource/file) at the requested URL
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond

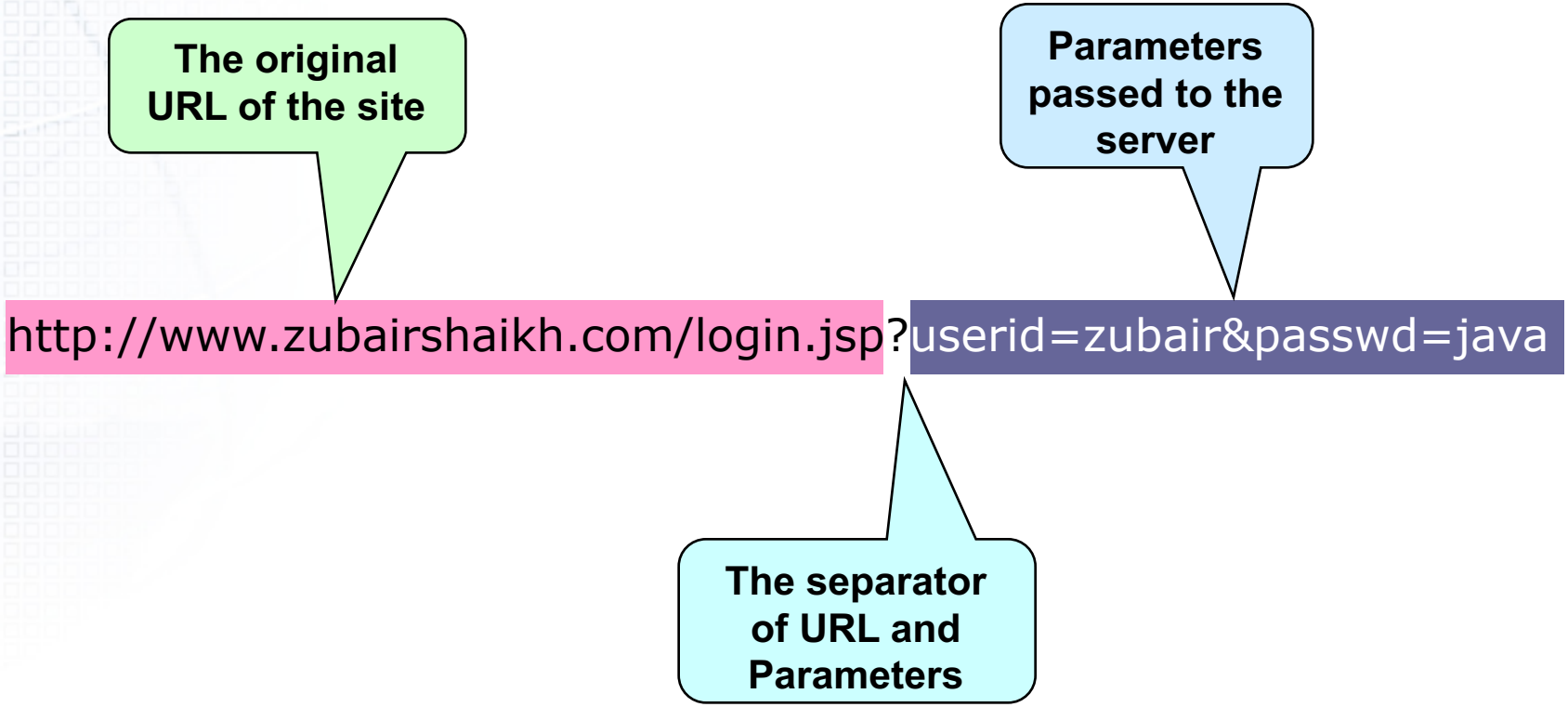
Do we really use all this methods?

Not as such.

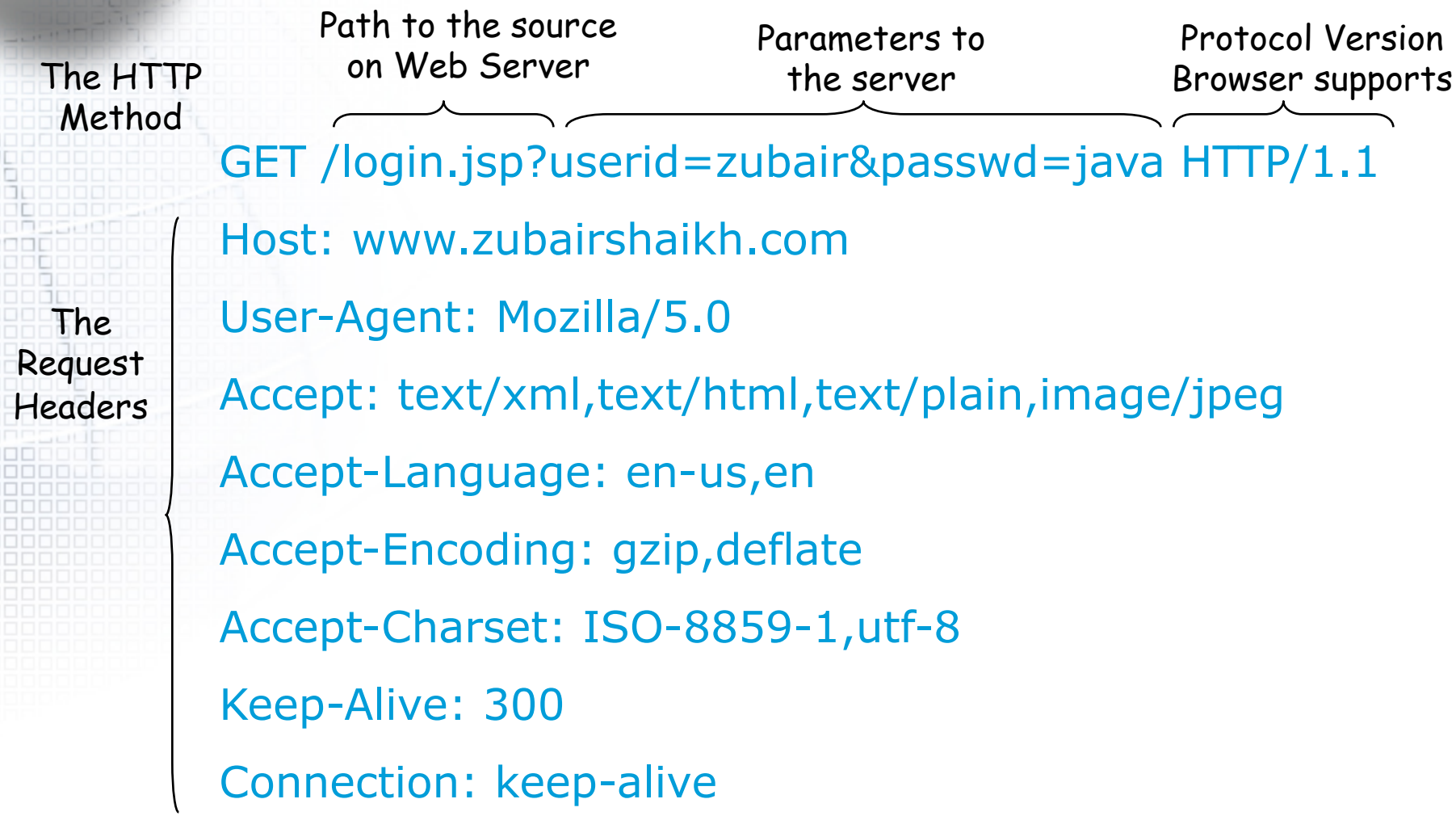
What method we'll be using?

- We mainly use
  - GET
    - Every request is GET
    - Clicking a link
    - Typing URL in browser
  - POST
    - Send user data to the server

A GET Request



Anatomy of GET request



Anatomy of POST request

The HTTP Method

Path to the source on Web Server

Protocol Version Browser supports

```
POST /login.jsp HTTP/1.1
Host: www.zubairshaikh.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive: 300
Connection: keep-alive

userid=zubair&passwd=java
```

The Request Headers

Message body sometimes called "Payload"

HTTP Response

- HTML is a part of HTTP response
- It has response headers with the response content
- Content types are known as MIME
  - text/html
  - text/plain
  - Image/jpeg
  - text/xml

Anatomy of response

Protocol Version that the Web Server supports

HTTP Status code

Text version of Status code

```
HTTP/1.1 200 Ok
Set-Cookie: JSESSIONID=0AAB6C9ASJF84; Path=/login
Content-Type: text/html
Content-Length: 397
Date: Mon, 19 Dec 2005 11:15 GMT
Server: Apache-Coyote/1.1
Connection: close
```

The MIME type of response

HTTP Response Headers

```
<html>
...
</html>
```

Body holds the HTML, or other contents

Why Servlet or JSP?

Static content dilemma

- Static contents sits at the web server end
- Server finds it and hands it to client when requested
- Every client sees the same thing
- Is that sufficient?

But what!!!

- If I want to store users data on the server
- If I want to remember the client
- If I want to show the links based on privileges

Web Server Issues

- Web servers love serving static content
- But sometimes we need more than just static content
- Two things web server alone won't do
  - Dynamic content
  - Saving data on the server

HTTP Protocol issues

- HTTP love to take request to server and bring response to client
- But sometimes we need more than just a data carrier
- HTTP do not remember you

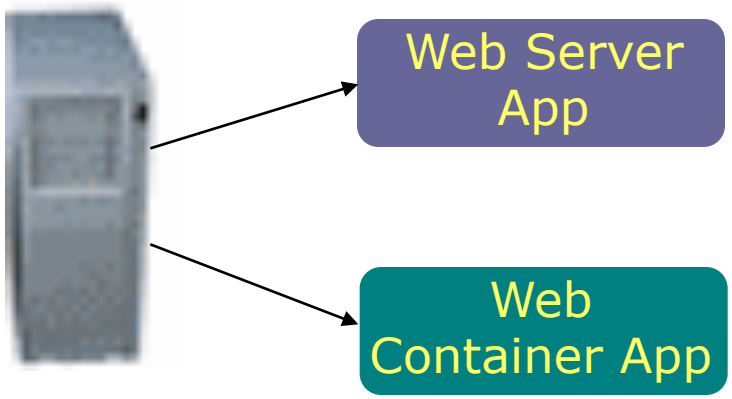
So, we cant help when

- Web server alone just cant help you for achieving dynamic content
- HTTP protocol will never remember you no matter how much you love it

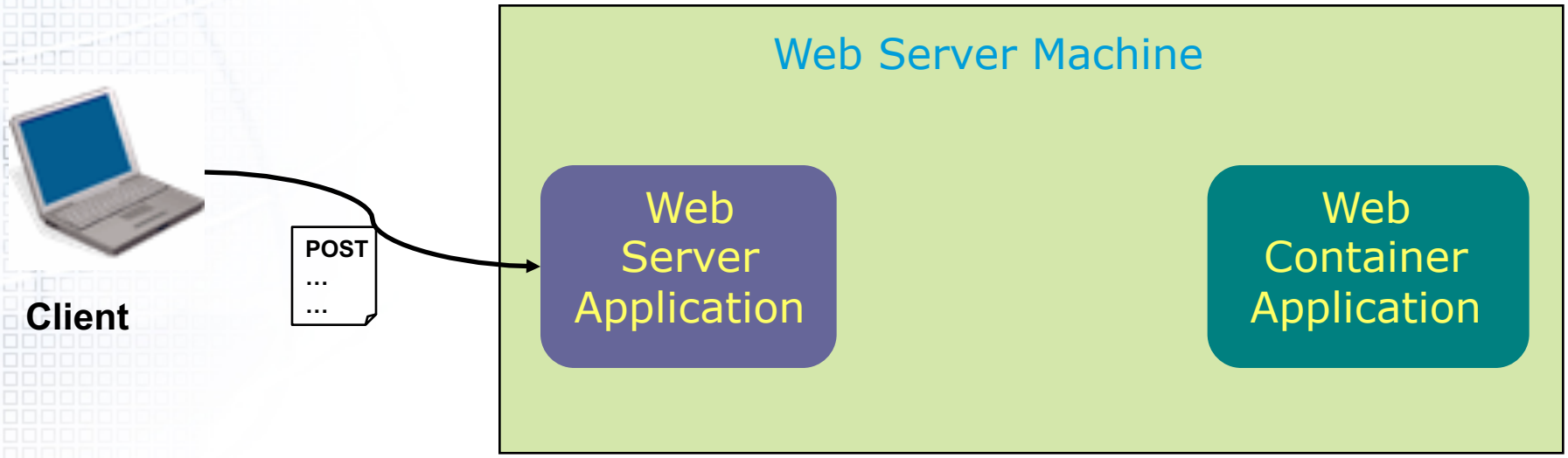
WE need MORE

- Someone who can work in association with web server
- Help you handle dynamic content
- Make you respond HTML from a program

Now Web Server Machine

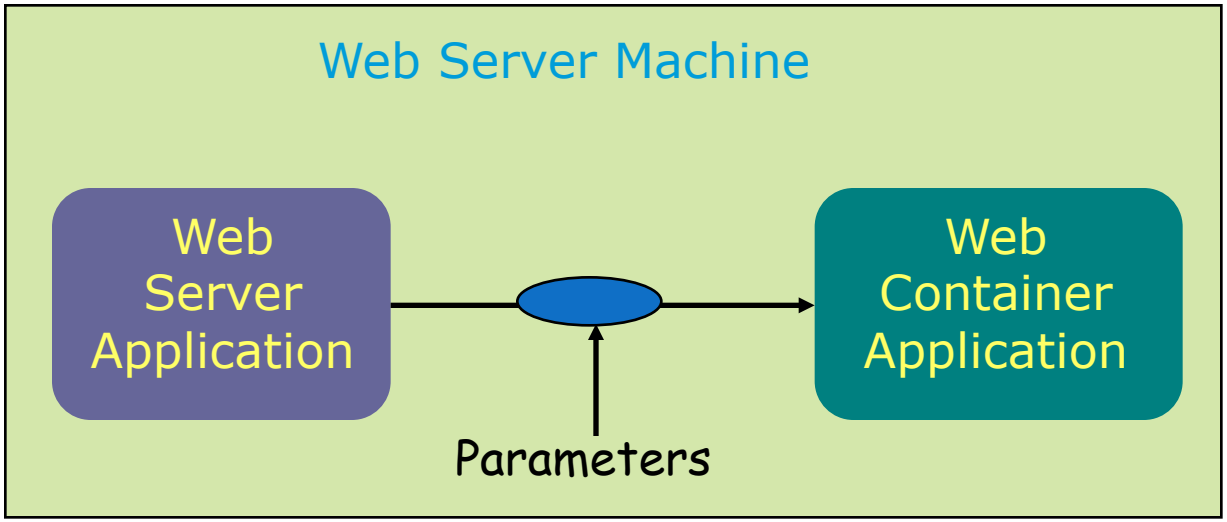


Dynamic contents processing



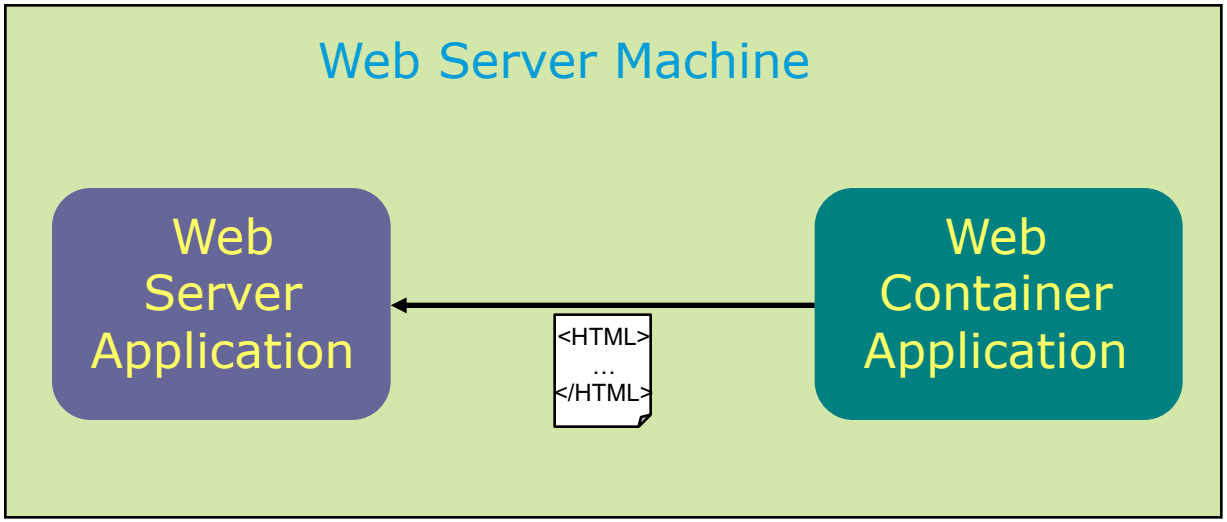
Dynamic contents processing

Client

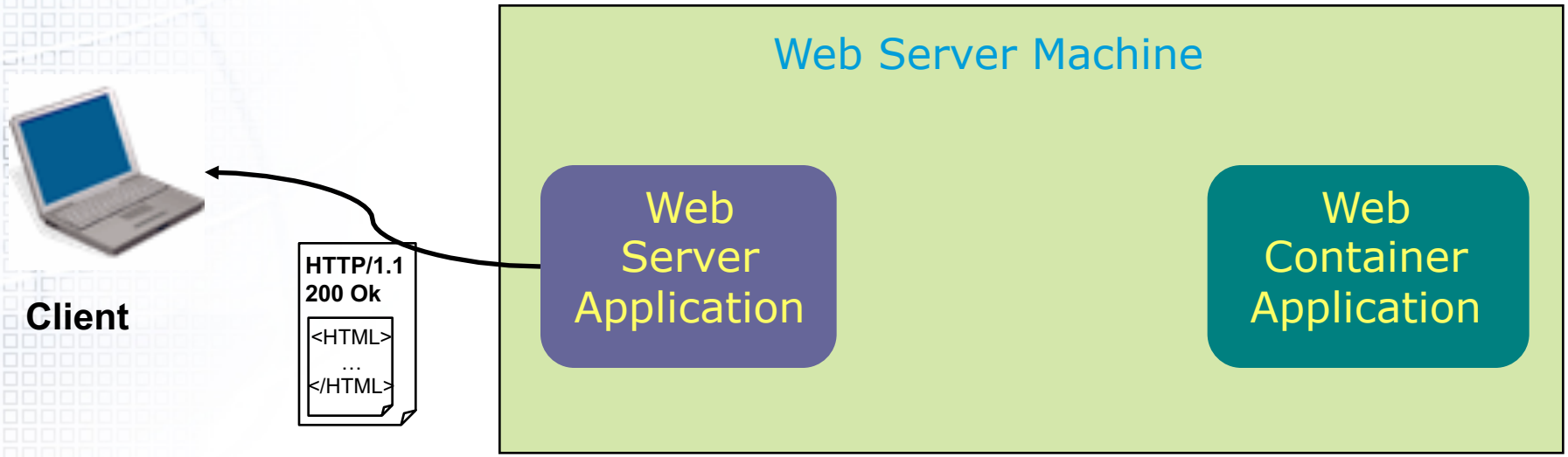


Dynamic contents processing

Client



Dynamic contents processing



Web Container

- Is a Java application design to follow J2EE specification
- It's an application who understands compile and make servlet run

J2EE

The Specification

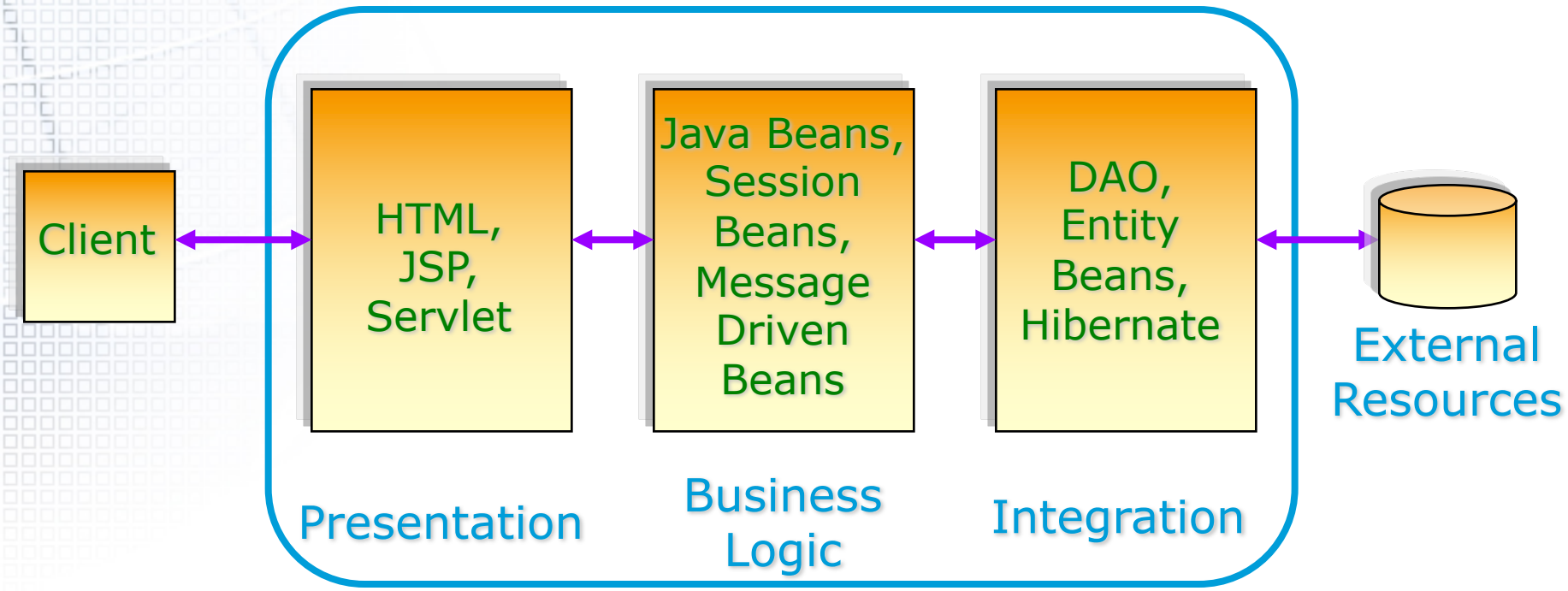
Introduction

- J2EE is Java's specification for enterprise application development
- It cover two major components
  - Web Components
    - For designing presentation layer
  - Distributed Components
    - For developing business components

Cont'd

- Sun has provided the specification for developing components
- The structure or flow of the application is all up to the developer
- In J2EE scenario the application is segregated into three layers
  - Presentation
  - Business Logic
  - Integration

The J2EE Scenario



Advantages

- J2EE Scenario gives extensibility benefits
- The application is loosely coupled
- Presentation, Business Logic and Integration is independent of each other
- Changes if needed in either of this will not affect other parts of the application

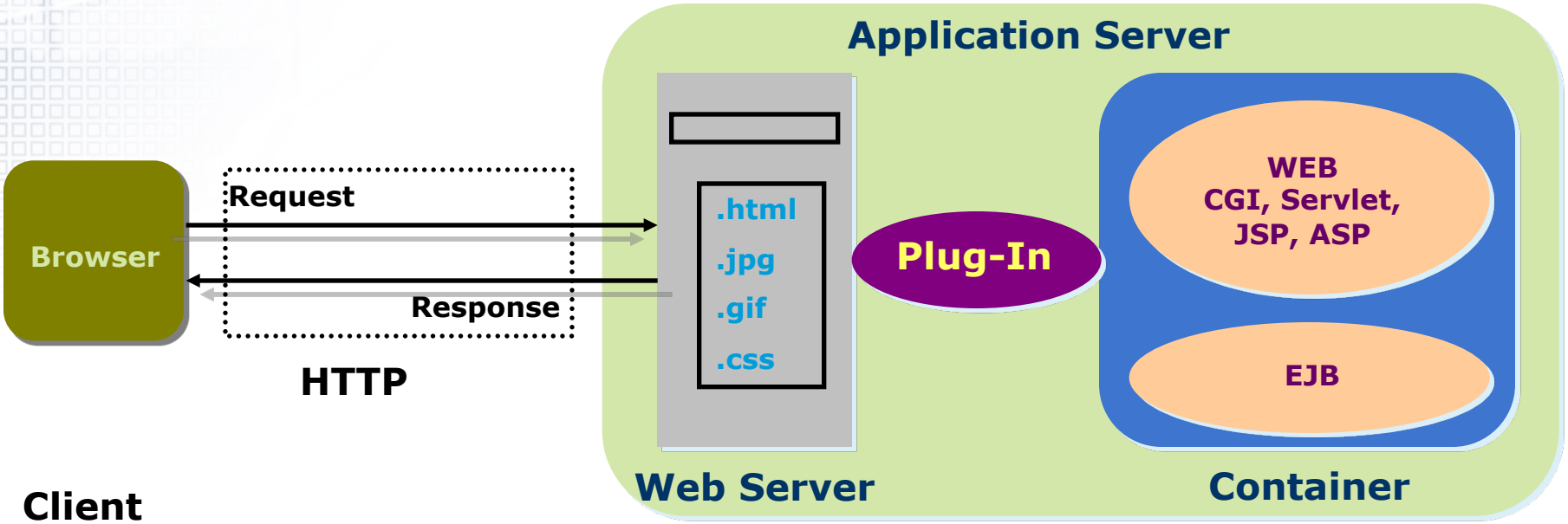
Servlet

Introduction

- Technology for developing dynamic web pages
- Require a web container to run
- A java program to enhance the capability of any server
- Follows internet programming model

Web Programming Model

- HTTP Protocol
- Browser
- Request
- Response



Servlet Programming

Servlet Programming

- Types of Servlet
- Servlet Life-cycle
- doGet() and doPost()
- Servlet API

Types of Servlet

- Generic Servlet
  - Servlet that can work with any protocol and is used for other than web applications
- HTTP Servlet
  - Servlet that can work only with HTTP protocol and commonly used for developing web applications

Servlet Life-Cycle

- init()
- service()
- destroy()

init()

- Code that runs while initialization of the servlet
- Initialization code like creating database connection
- Runs only once in the whole life of a servlet

service()

- Code that executes for every request made by the client
- Identifies the request type and forward to either get or post specific method

destroy()

- Code that runs while destroying a servlet
- Holds cleanup code like closing connection or stream
- Runs only once in a life of a servlet

doGet and doPost

- These methods are HTTP specific service methods
- Called by the service method
- doGet Method
  - Method handles request of get type
- doPost Method
  - Method handles request of post type

Servlet API

The Servlet APIs

- Defines the APIs to the services that the Servlet Container provides to the Servlet
  - Request/Response interfaces
  - Session State services
  - Environment services (configuration, application), etc.
- Defines the Servlet life-cycle and service semantics from the Servlet Container point of view
  - Initialization/Destruction/Service
  - Threading models

The Servlet APIs Packages

- Two packages:
  - javax.servlet - General Servlet related definitions
  - javax.servlet.http - HTTP related extension for the "general" Servlets
- Since we only deal with Web (HTTP) related Servlets we will cover and mix both of them.

The Servlet Interface

- Defines two misc' methods apart from the life cycle methods (getServletConfig(), getServletInfo())
  - Implemented by base classes

ServletConfig

- The Servlet developer uses the ServletConfig object to get Servlet related information.
  - getInitParameter(String name)
    - Returns a string containing the value of the named initialization parameter, or null if the parameter does not exist.
  - getInitParameterNames()
    - Returns the names of the Servlet's initialization parameters as an enumeration of strings.

Cont'd

- getServletContext()
  - Returns the context for the Servlet.
- getServletName()
  - Returns the Servlet name.

ServletContext

- The Servlet developer uses the ServletContext object to get server related (application) information and services.
  - getMimeType(String file)
    - Returns the mime type of the specified file.
  - getRealPath(String path)
    - Applies alias rules to the specified virtual path and returns the corresponding real path.

Cont'd

- log(String msg, Throwable t)/ log(String msg)
  - Write the stacktrace and the given message string to the Servlet log file.

HelloWorld Servlet - Code

```
package com.zubairshaikh.servlet;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

        resp.setContentType("text/html");
        final PrintWriter out = resp.getWriter();
        out.println("<html><body>");
        out.println("<h1>Hello World (With Servlet Accent)</h1>");
        out.println("</body></html>");
    }
}
```

web.xml

```
//initial xml declarations
<web-app>
    <servlet>
        <servlet-name>HelloWorld</servlet-name>
        <servlet-class>com.zubairshaikh.servlet.HelloWorld</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloWorld</servlet-name>
        <url-pattern>/Hello</url-pattern>
    </servlet-mapping>
</web-app>
```

HelloWorld Servlet - Output

Form processing

Submitting a Form

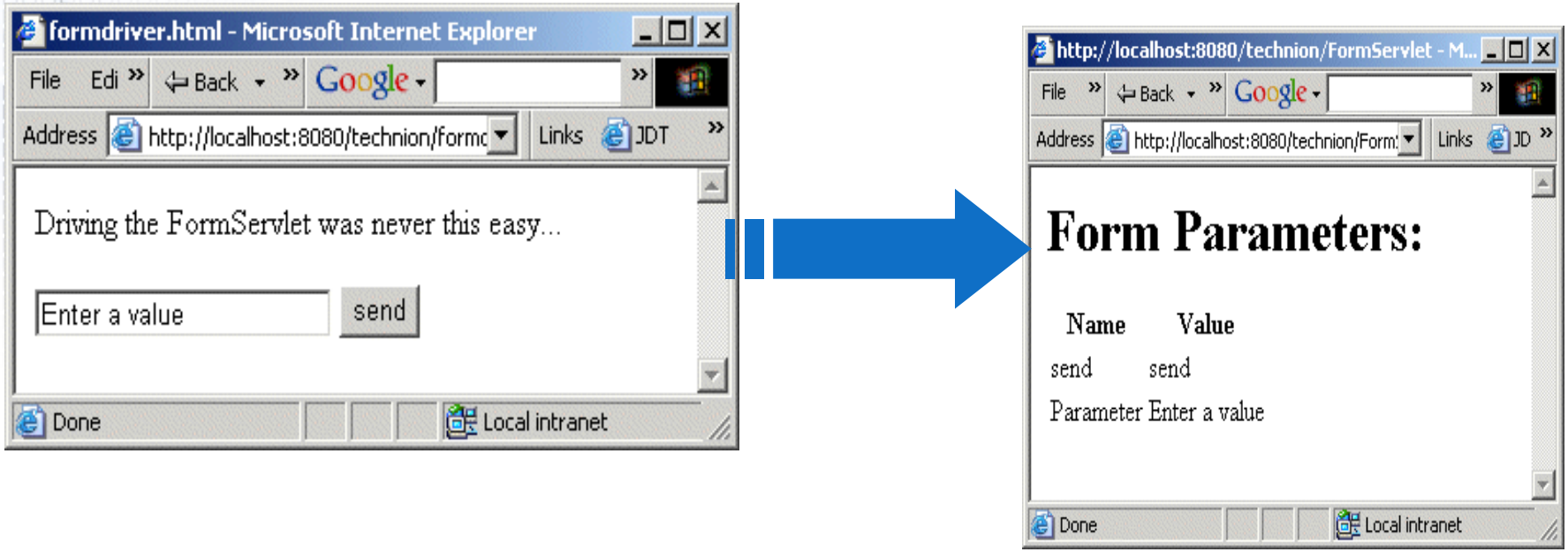
- Form parameters constructs a list of named values
- Form parameters can be obtained using the HttpServletRequest methods:
  - getParameterNames()
    - Returns the parameter names for this request as an enumeration of strings.

Cont'd

- getParameter(String name)
  - Returns a string containing the lone value of the specified parameter, or null if the parameter does not exist.
- getParameterMap()
  - Returns a java.util.Map with the parameters for this request

Form Processing Servlet

- Executed by submitting a Form
- Prints all the submitted Form parameters as a response



FormServlet Servlet Code

```
package com.zubairshaikh.servlet;


public class FormServlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        doGet(req, resp);
    }

    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

        resp.setContentType("text/html");
        final PrintWriter out = resp.getWriter();
        out.println("<html><body>");
        out.println("<h1>Form Parameters:</h1>");
        // Continue on next slide …
    }
}
```

FormServlet Servlet Code

```
        // Started on previous slide
        out.println("<table>");
        out.println("<tr> <th> Name </th> <th> Value </th> </tr>");
        final Enumeration names = req.getParameterNames();
        while(names.hasMoreElements())
        {
            final String name = (String)names.nextElement();
            final String value = req.getParameter(name);
            out.println("<tr><td> " + name + " </td><td> " + value + " </td></tr>");
        }
        out.println("</table>");
        out.println("</body></html>");
    }
}
```

FormServlet Observations

- Reading the posted Form parameters is easy (as it should be)
- We implement both doGet() and doPost()
  - The two HTTP methods used for sending Forms
- The value of the "submit" button is submitted as well

Redirection

Redirection

- Two approaches of redirecting the control
  - sendRedirect()
  - forward()

sendRedirect()

- Always sends a header back to the client/browser
- The browser uses this header to make another fresh request
- sendRedirect has a overhead as to the extra remote trip being incurred
- The advantage is that u can point to any resource (whether on the same domain or some other domain)

forward()

- forward just routes the request to the new resources specified in the forward call
- Route is made by the servlet engine at the server level only
- The request and response objects remain the same both from where the forward call was made

Differences

- In sendRedirect() absolute path of the url needs to be given. The request and response objects will be passed automatically to the new page
- In case of forward() method, request and response objects needs to be send to the new page explicitly

Session state

Keeping track

What is Session State

- We want to save short-term state on behalf of the user:
  - Shopping cart.
  - Authentication information.
- HTTP is stateless:
  - How can we keep state?
- Enter HTTP based user session state
  - Cookies
  - Session State APIs

Cookies

- The first session state solution was cookies, a special set of HTTP headers with special semantics:
  - The server can set a cookie in the client's browser by using the set-cookie header.
  - The client will then send the cookie back to the server on the following requests using the cookie header.
  - The value of a cookie header needs to be an alphanumeric value.

Cont'd

- But Cookies are bad:
  - Not enough abstraction -> hard to use.
    - Need to manually parse a cookie.
  - Not secure enough, data is passing back and forth.
  - Limited in size (~100byte).
  - Uses are afraid of cookies

Session State Servlets Way

- The Servlet container is tracking the user
  - Several tracking tools (not only cookies)
  - Allows the servlets to associate data with the user
- The container exposes APIs to access the session data
  - HttpSession
  - HttpServletRequest.getSession()

HttpSession

- Represents a single user session.
- Exposes the following methods:
  - Object getAttribute(String name)
    - Get a stored value from the session.
  - Enumeration getAttributeNames()
    - Get the named of the stored values.
  - removeAttribute(String name)
    - Remove a values from the session.
  - setAttribute(String name, Object value)
    - Set a value in the session.

HttpSession

- Exposes the following methods:
  - setMaxInactiveInterval(int interval), int getMaxInactiveInterval()
    - Set and get the max inactivity interval that the session can have
  - long getCreationTime(), getLastAccessedTime()
    - Get the time the session was created/last accessed.

Cont'd

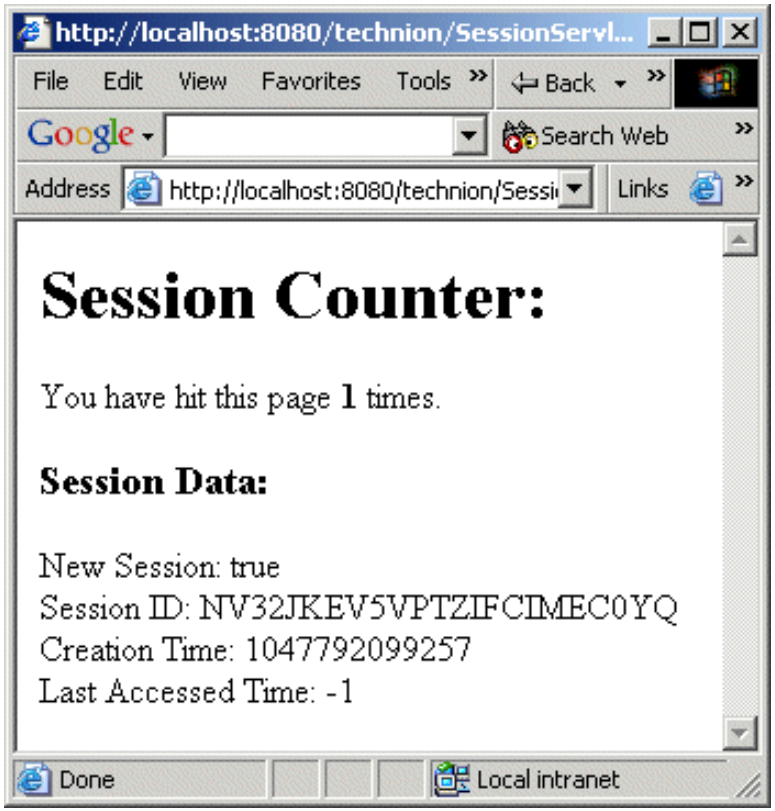
- String getId()
  - Get the id used for this session.
- invalidate()
  - Dispose the session.
- boolean isNew()
  - Was this session created now, and did arrive to the user (yet).

Session State Method

- HttpSession getSession()
  - If a session is associated with the current request, this method will return it.
- HttpSession getSession(boolean create)
  - Return a session that is associated with the current request. If the session does not exists, create new session.

Session Counter Servlet

- Counts the number of per-user hits on the servlet
- Prints some of the session meta data



SessionServlet Code

```
package com.zubairshaikh.servlet;

public class SessionServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

        // Get the session object
        final HttpSession session = req.getSession(true);
        resp.setContentType("text/html");
        final PrintWriter out = resp.getWriter();
        // Continue on next slide …
```

SessionServlet Code

```
// Started on previous slide
out.println("<html><body>");
out.println("<h1>Session Counter:</h1>");

Integer ival = (Integer) session.getAttribute("sessiontest.counter");
if(null == ival) {
    ival = new Integer(1);
} else {
    ival = new Integer(ival.intValue() + 1);
}
session.setAttribute("sessiontest.counter", ival);
out.println("You have hit this page <b>" + ival + "</b> times.<p>");
out.println("<p>");
// Continue on next slide …
```

SessionServlet Code

```
        // Started on previous slide
        out.println("<h3>Session Data:</h3>");
        out.println("New Session: " + session.isNew());
        out.println("<br>Session ID: " + session.getId());
        out.println("<br>Creation Time: " + session.getCreationTime());
        out.println("<br>Last Accessed Time: " +
                    session.getLastAccessedTime());

        out.println("</body></html>");
    }
}
```

SessionServlet Observations

- One must start the session first thing in the service phase
- Initially, values may be null unless initialized using a session listener

Hidden Fields - Code

```
package com.zubairshaikh.servlet;

public class HiddenFieldServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
         RequestDispatcher rd=null;
         response.setContentType("text/html");
         PrintWriter out = response.getWriter();
         out.println("<html>");
         out.println("<head><title>
          HiddenFeildServlet</title></head>");
         out.println("<body>");

//continue on next slide
```

Hidden Fields - Code

```
        //Creating the form with hidden fields
        out.println("<FORM
        ACTION=\"http://localhost:8080/examples/servlet/HiddenFeildServlet\"
        METHOD=POST>");

        out.println("<INPUT TYPE = hidden NAME = user VALUE = James>");
        out.println("<INPUT TYPE = hidden NAME = session VALUE =12892>");

        out.println("<INPUT TYPE = submit value=\"Finished Shopping\">");
        out.println("</FORM>");
        out.println("</body></html>");
        out.close();
    }

    //continue on next slide
```

Hidden Fields - Code

```
//Process the Http Post request
public void doPost (HttpServletRequest request, HttpServletResponse response)
throws ServletException,IOException
{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>HiddenFieldServlet</title></head>");
    out.println("<body>");

    // Get the hidden inputs and echo them
    String user = request.getParameter("user");
    String session = request.getParameter("session");

    out.println("<H2>Hello "+ user +" your session id is "+ session +"</h2>");
    out.println("</form></body></html>");
    out.close();
}
}
```

CookieServlet - Code

```
package com.zubairshaikh.servlet;

public class CookieServlet extends HttpServlet {

    private String getCurrentUser(String value){
        String username=new String("");
            if(value.equals("100")){
            username=new String("Zubair");
         }
        return username;
    }

//continue on next slide
```

CookieServlet - Code

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    Cookie[] cookielist = request.getCookies();
    String user = null;
    String responsestring = null;

    if(cookielist != null){
        for(int x=0;x<cookielist.length;x++){
            String name = cookielist[x].getName();
            if(name.equals("Zubair")){
                user = getCurrentUser(cookielist[x].getValue());
                break;
            }
        }
    }

//continue on next slide
```

CookieServlet - Code

```
        if(user == null){
            Cookie cook = new Cookie("Zubair","100");
            cook.setMaxAge(300);
            response.addCookie(cook);
            responsestring = new String(“<h2>Welcome to our site,  "+
                "we have created a session for you </h2>");
        }
        else{
            responsestring = new String(“<h1>Hello  :"+ user);
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>CookieServlet</title></head>");
        out.println("<body>");
        out.println(responsestring);
        out.close();
    }
}
```

Thank You

Zubair Shaikh
me@zubairshaikh.com