

Web Technologies

Computer Security

Lecture 9

Tom Chothia

This Lecture

Some basic building blocks of the web:

- HTTP: HyperText Transfer Protocol
- HTML: HyperText Markup Language
- JavaScript
- JSP: Java Server Pages
- SQL: Structured Query Language

Uniform Resource Locators

Protocol

Host

FilePath



http://www.cs.bham.ac.uk/index.html?

field1=valuea&field2=value2



Query String

HTTP

- HyperText Transfer Protocol
- Used to request and deliver webpages.
- Includes:
 - Set of basic commands
 - Header fields
 - Status codes

GET and POST

Key HTTP commands:

- GET: requests a resource, e.g. a webpage in HTTP
- POST: submits data to the server. e.g. from a form on a webpage.

Example

```
laptop:~ laptop$ telnet www.cs.bham.ac.uk 80
```

```
Trying 147.188.192.42...
```

```
Connected to www.cs.bham.ac.uk.
```

```
Escape character is '^]'.  
^C
```

```
GET /index.php
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
  Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/  
  xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml"  
  lang="en">
```

```
<head>
```

The burp Proxy

The burp Proxy lets you monitor and pause an Internet connection.

<http://portswigger.net/burp/proxy.html>

Use it to look at your clients HTTP messages:

1. Run proxy (it opens port 8080),
2. Tell web client to use the proxy
3. See the messages

Headers fields

These provide extra information, e.g.

Host: <host name> is compulsory for
HTTP 1.1,

```
GET /index.php HTTP/1.1
```

```
Host: www.cs.bham.ac.uk
```

Other Headers Fields

`Cookie`: gives a “cookie”

`Accept`: data types client can handle,
e.g. `Accept: text/plain`

`Content-Length`: length of message in bytes.

Full list at: http://en.wikipedia.org/wiki/List_of_HTTP_headers

Cookies

- Cookies let you store a string on the client.
- This can be used to
 - Identify the user,
 - (cookie given out after login)
 - Store user name, preferences etc.
 - Track the user: time of last visit, etc.

How many cookies are in your browser?

Headers and Bodies

HTTP responses are made up of a header and a body:

- The header includes a reply code to tell the client what has happened.
- The body is the resource, e.g. the webpage

Status Codes

- 2--: Success
 - 200 OK: the request worked
 - 201 Created: request worked and server has created a new resource
 - 204 No Content: request worked but there is nothing to return.
 - ...

Status Codes

- 3--: Redirection
 - 301 Moved Permanently: Website has moved
- 4--: Client Error
 - 400 Bad Request: syntax error
 - 401 Unauthorized: needs a cookie?
 - 403 Forbidden: No access allowed

Status Codes

- 5--: Server Error
 - 500 Internal Server Error: general error message
 - 501 Not Implemented: command not supported

Full list of status codes at:

[http://en.wikipedia.org/wiki/
List_of_HTTP_status_codes](http://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

Example

```
laptop:~ laptop$ telnet www.google.co.uk 80
```

```
Trying 173.194.37.104...
```

```
Connected to www.l.google.com.
```

```
Escape character is '^]'.  
^C
```

```
GET / HTTP/1.1
```

```
Host:www.google.co.uk
```

```
HTTP/1.1 200 OK
```

```
Date: Wed, 03 Nov 2010 13:07:18 GMT
```

```
Expires: -1
```

```
Cache-Control: private, max-age=0
```

```
Content-Type: text/html; charset=ISO-8859-1
```

```
Set-Cookie:
```

```
  PREF=ID=e5e59d5c56d4f722:FF=0:TM=1288789638:LM=1288789638:S  
  =qb0MGMEGVmqC7eu0; expires=Fri, 02-Nov-2012 13:07:18 GMT;  
  path=/; domain=.google.co.uk
```

OTHER COMMANDS

HEAD	like GET but only gets header
PUT	uploads resource to server
DELETE	deletes resource from server
TRACE	echoes back message
OPTION	returns supported HTTP
CONNECT	used for tunnelling TCP
PATCH	partially modify resource

Hypertext Markup Language

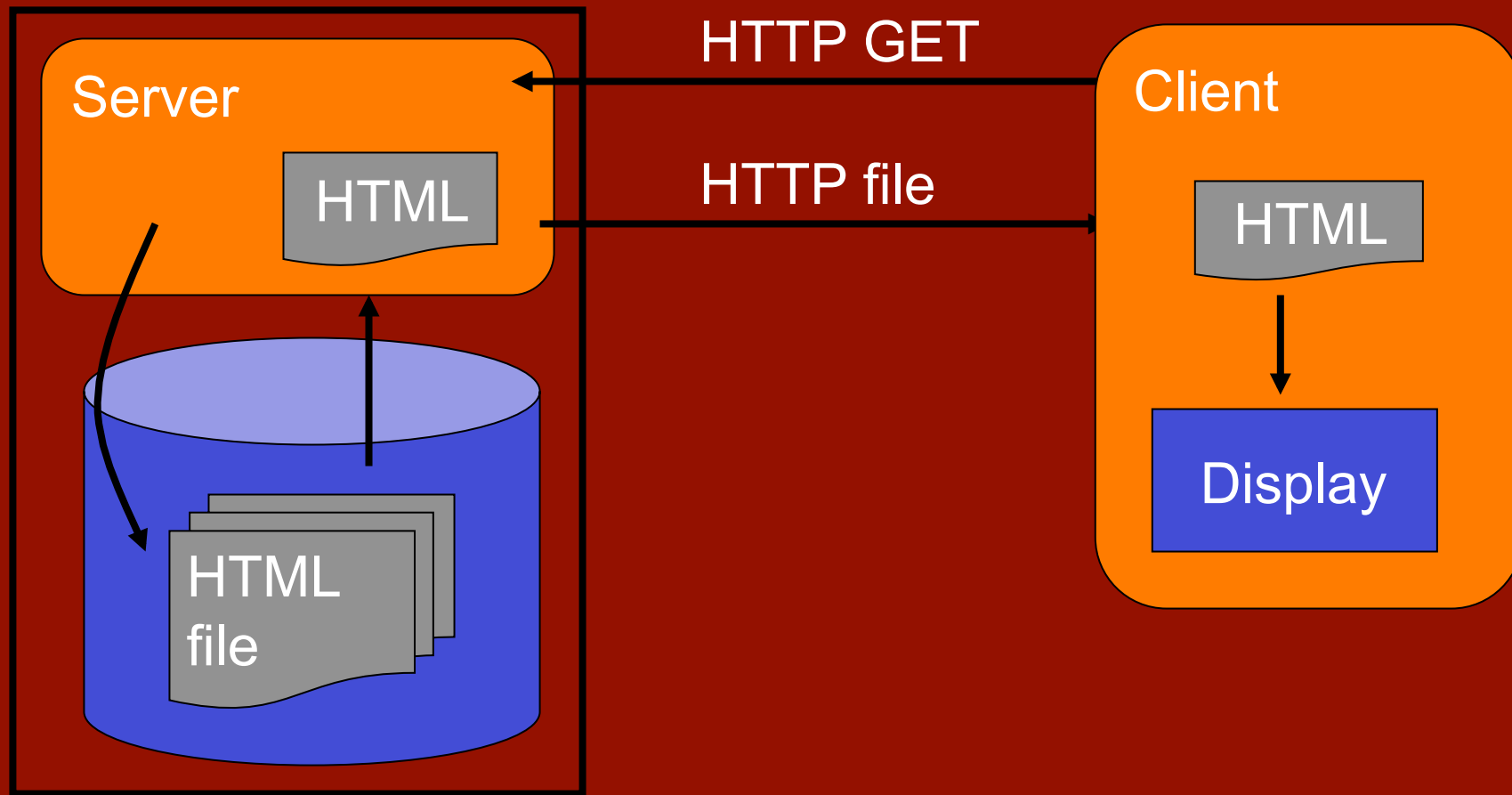
HTML

Tags tell browser how to display text:

```
<b>hello</b> bob = hello bob
```

Links: `link`

Web Pages



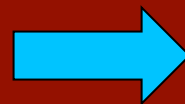
HTML Forms

```
<form action="http://site.com/index.jsp" method="GET">  
  Email: <input type="text" name="email">  
  <input type="submit" value="Submit">  
</form>
```



Email:

Email:



`http://site.com/index.jsp?email=x@y.com`

JavaScript

- JavaScript is a language for web pages, that will run on the client.
- It can be added to any HTML file.
- When the client loads the HTML it executes the JavaScript.
- It's *not* Java, but is kind of like it.

Why Use JavaScript?

- Shift computation onto the client.
- Personalise web pages to the reader.
- Form validation
- Keeping track of users: cookies.
- Pop-up, alerts, new windows

Hello World in JavaScript:

- Put the JavaScript in a HTML web page.

- Put JavaScript between the HTML tags

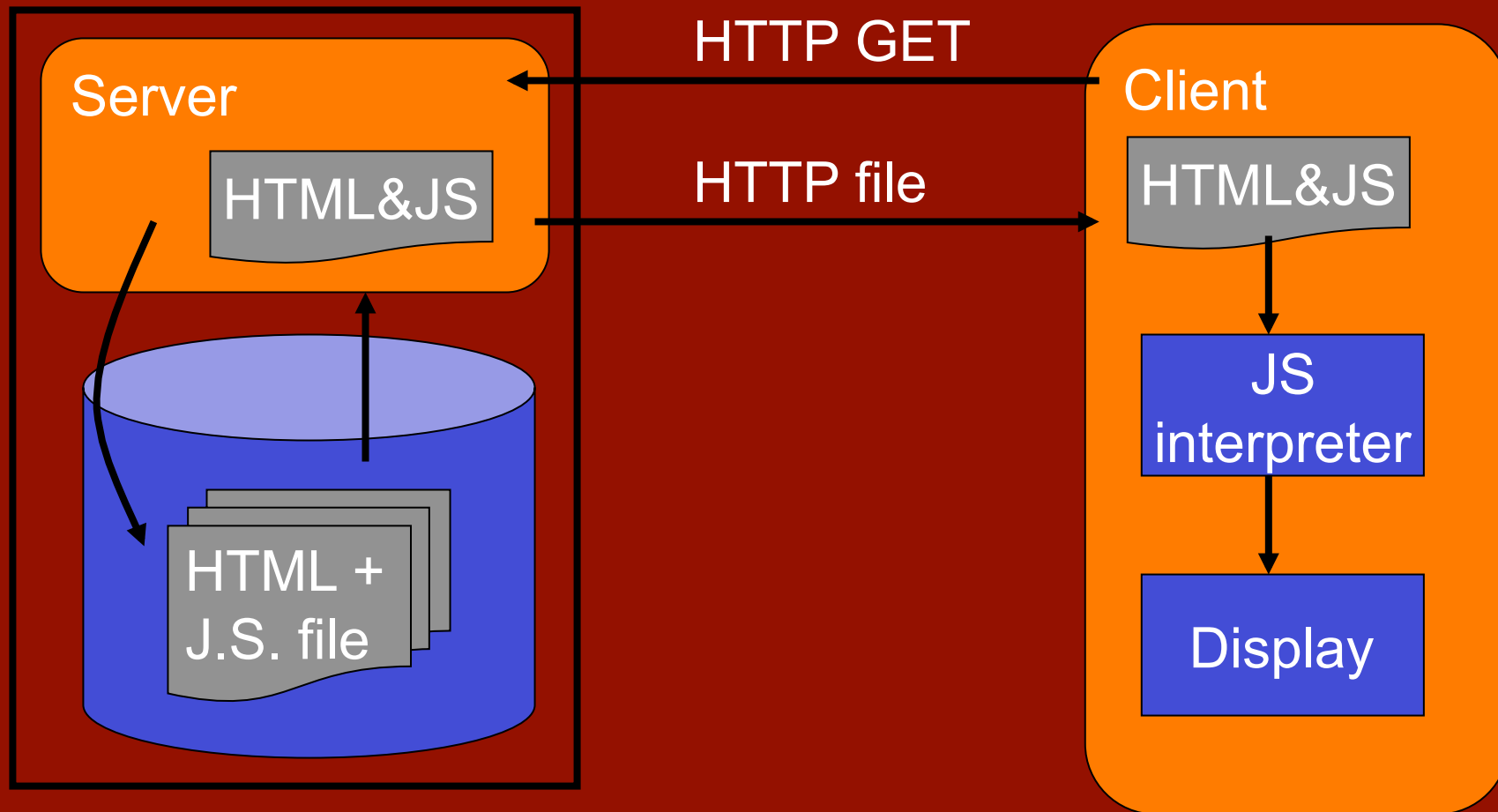
```
<script> ... </script>
```

- The print command in JavaScript is:

```
document.write(<String>);
```

- HTML between the `<noscript> ... </noscript>` will be run if JavaScript is not enabled.

JavaScript



JavaServer Pages (JSP)

JavaServer Pages lets you write dynamic webpages using Java.

You can put the Java in a HTML file.

The Java code will be run on the server every time a page is requested.

To run JSP you need to use a compatible webserver, e.g. Tomcat, Glassfish.

JSP

JSP files end with `.jsp`

Info bar at top with imports etc.

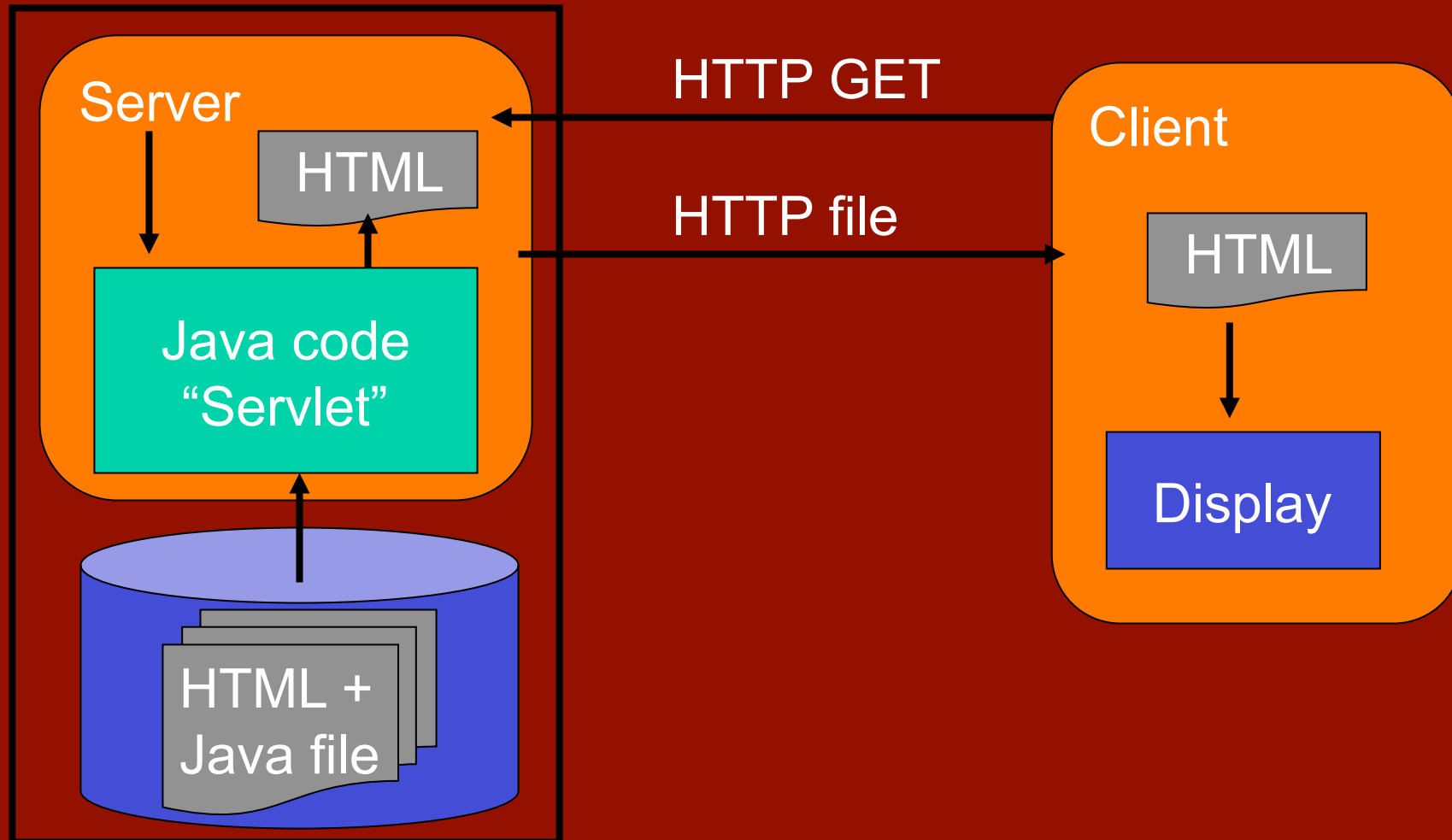
Place Java code inside `<% %>`

All other code is normal HTML

Only runs in a container

- (easy with Netbeans)

JSP



SQL

- To store data about users and content, most website will use a SQL database.
- This is a standard database format, which you should know.
- See e.g.: `http://www.w3schools.com/sql/default.asp`

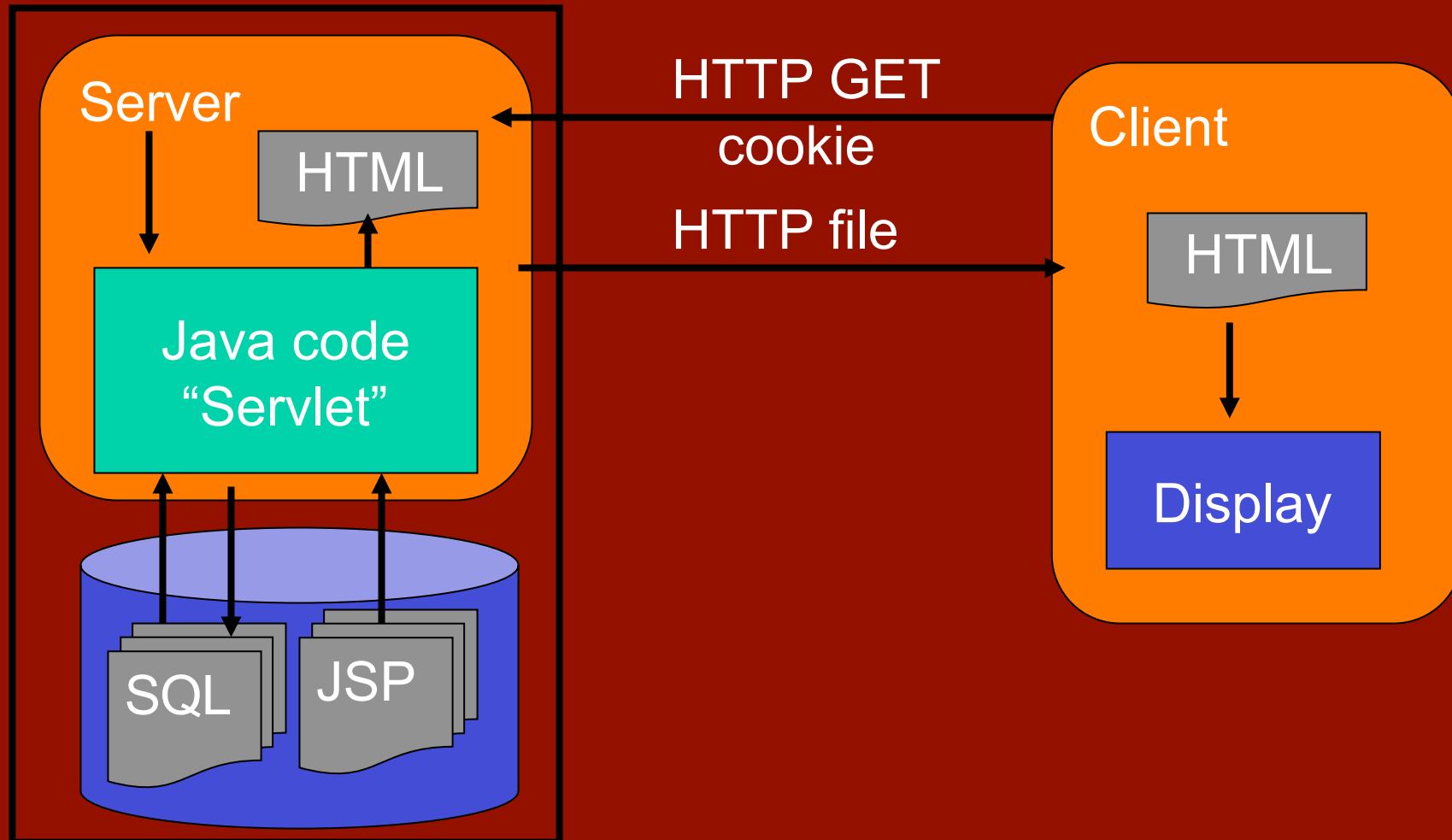
Some Key SQL Commands

```
SELECT LastName FROM namesTable  
        WHERE FirstName='Fred'
```

```
INSERT INTO namesTable VALUES  
        ('John', 'Smith')
```

```
DROP TABLE namesTable
```

A typical web set up



Typical Web Setup

HTTP website:

```
<form action="http://site.com/index.jsp" method="GET">  
  Email: <input type="text" name="email">  
  <input type="submit" value="Submit">  
</form>
```



Email:

Users browser:

Email:



`http://site.com/index.jsp?email=x@y.com`

Typical Web Setup

```
http://site.com/index.jsp?email=x@y.com
```

JSP page reads and processes:

...

```
email=request.getParameter("email");
```

```
stmt.execute("INSERT INTO table
```

```
                VALUE(`"+id+"', `"+email+"`);
```

```
%>
```

```
<b>Your e-mail has been added</b>
```

Other Popular Web Technologies

PHP: plays a similar role to JSP

CGI: like JSP but with Perl instead of Java

ASP: Microsoft's version of JSP

AJAX: Asynchronous Javascript And XML

Next 2 lectures:

How this all goes wrong.

In particular:

- Stealing cookies,
- SQL injection
- Cross site scripting attacks (XSS)
- Cross-site request forgery (CSRF)