# Database-enabled web technology
# Web Programming

Instructor: Çağrı Çöltekin

c.coltekin@rug.nl

Information science/Informatiekunde

Fall 2011/12

# About initial project reports

Your initial report should describe what you are going to implement well. It should include,

- the requirements
- your initial database design

Things to think about (while sketching the requirements):

- who are your user(s)?
- do you need multiple user interfaces (e.g., one for end users, one for administrators of the system)
- do you need to store any sensitive information? (credit card numbers, identities, etc.)
- how do you authenticate your users?
- are there possible performance issues? Which columns in your database will be used most in typical queries?

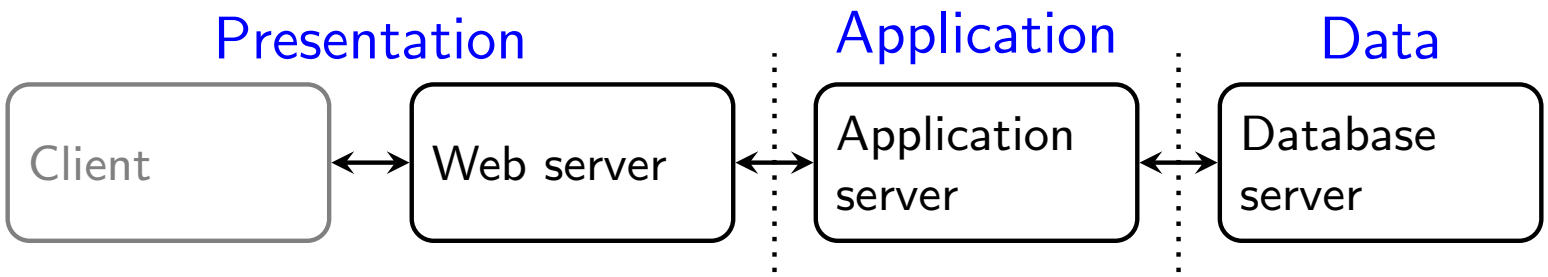
# So far...

- A quick introduction to PHP basics.
- Another introduction on `git` version management system.
- Review of database design and SQL.
- An introduction to using SQL from PHP.

# A first PHP/MySQL example

```
 1  <?php
 2      $host="hostname";
 3      $user="username";
 4      $pass="password";
 5      $db="dbname";
 6      mysql_connect($host,$user,$pass);
 7      mysql_select_db($db);
 8      $q = 'select * from book';
 9
10      $res = mysql_query($q);
11      echo "<table border=\"1\">";
12      echo "<tr><th>ISBN</th><th>title</th></tr>";
13      while ($row = mysql_fetch_assoc($res)) {
14          echo "<tr><td>${row['ISBN']}</td>";
15          echo "<td>${row['title']}</td></tr>";
16      }
17      echo "</table>";
18      mysql_close();
19  ?>
```

# The multi-tier (or 3-tier) architecture

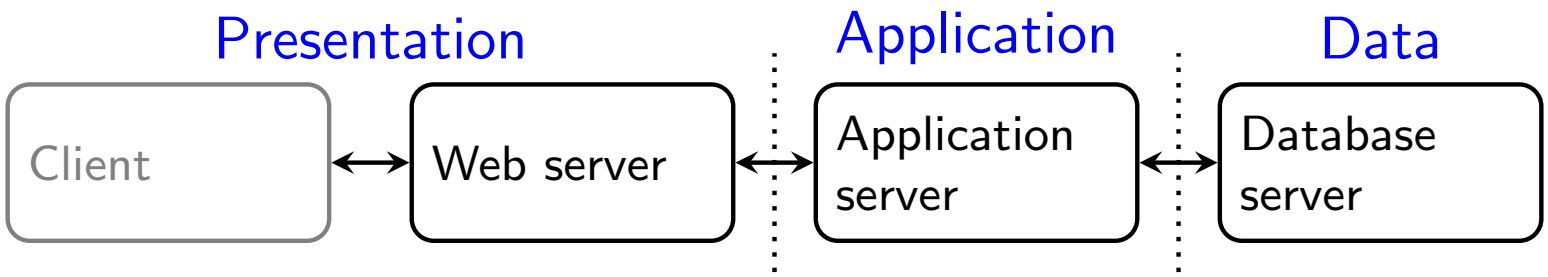



**Presentation** tier interacts with the user (e.g., ask the seat preference in an airline online check-in system).

**Application** tier implements the 'business logic' (e.g., check and reserve a seat, possibly using multiple queries and updates).

**Data** tier stores the data (e.g., retrieve and/or update the relevant data records).

# The multi-tier (or 3-tier) architecture



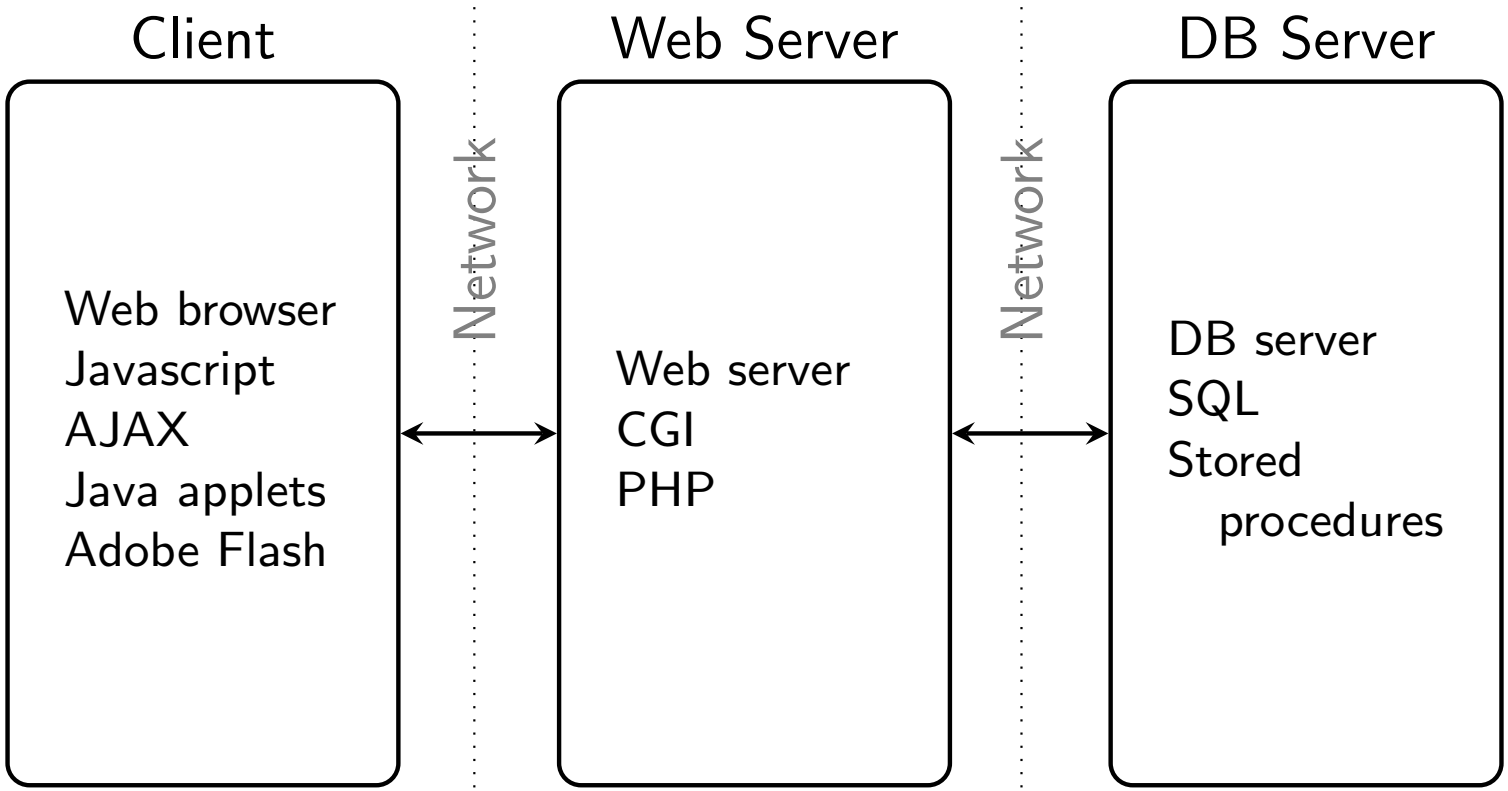

**Presentation** tier interacts with the user (e.g., ask the seat preference in an airline online check-in system).

**Application** tier implements the 'business logic' (e.g., check and reserve a seat, possibly using multiple queries and updates).

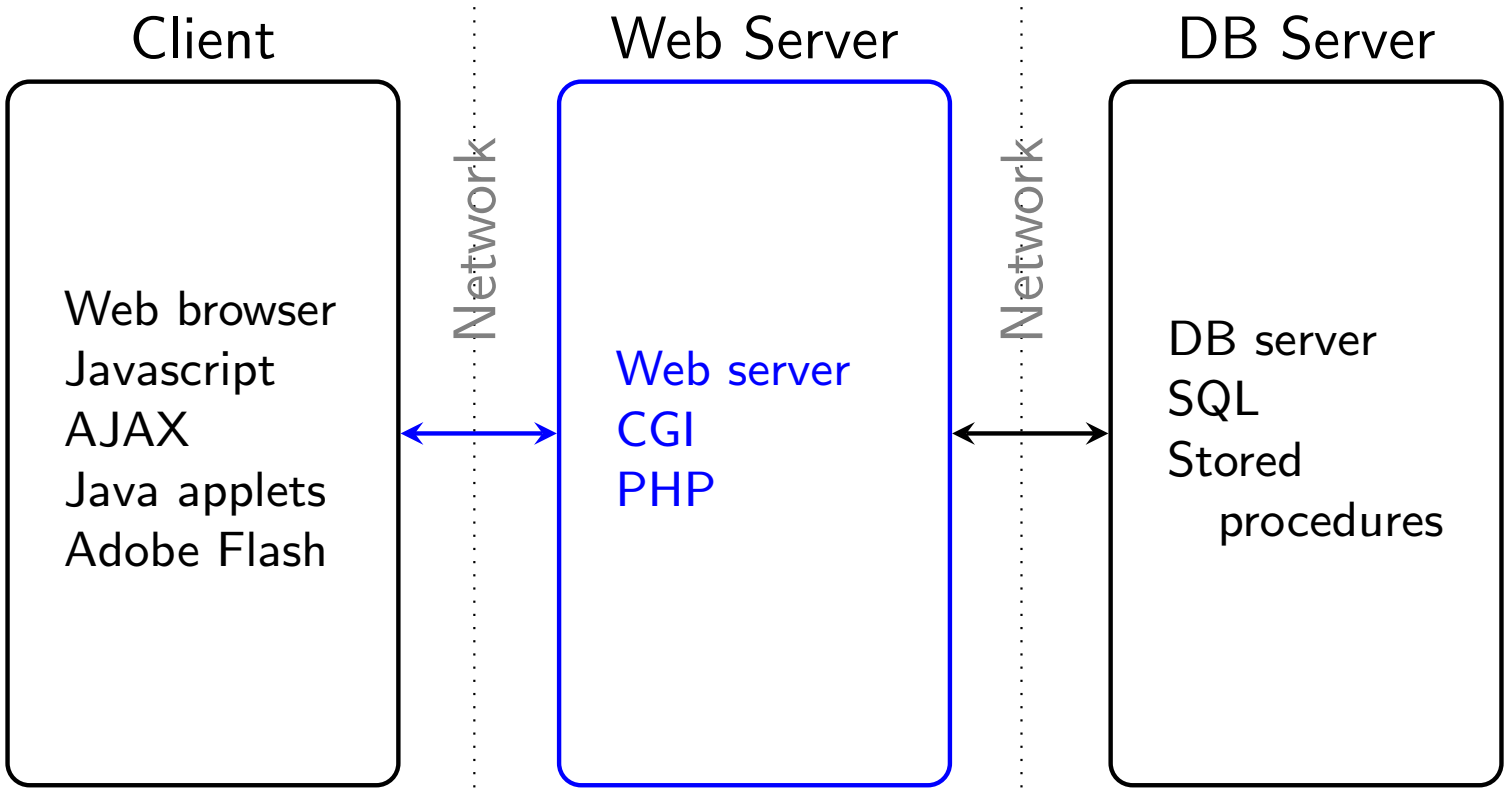
**Data** tier stores the data (e.g., retrieve and/or update the relevant data records).

In practice, division may not match the figure above. However separating presentation from application is always a good idea.

# Web-programimg model: what runs where



# Web-programimg model: what runs where




In this lecture we will study the client–server interaction and server-side programming

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`

C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`

C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde

C Resolve the host name: 129.125.2.51

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`

C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde

C Resolve the host name: 129.125.2.51

C Find the default port number for HTTP: 80

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`

C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde

C Resolve the host name: 129.125.2.51

C Find the default port number for HTTP: 80

C Open a TCP connection to the IP:port

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`
C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde
C Resolve the host name: 129.125.2.51
C Find the default port number for HTTP: 80
C Open a TCP connection to the IP:port
S Accept the connection

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`
C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde
C Resolve the host name: 129.125.2.51
C Find the default port number for HTTP: 80
C Open a TCP connection to the IP:port
S Accept the connection
C Send the HTTP request: GET /let/informatiekunde HTTP/1.1...

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`
C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde
C Resolve the host name: 129.125.2.51
C Find the default port number for HTTP: 80
C Open a TCP connection to the IP:port
S Accept the connection
C Send the HTTP request: GET /let/informatiekunde HTTP/1.1...
S Read the request, process it

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`
C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde
C Resolve the host name: 129.125.2.51
C Find the default port number for HTTP: 80
C Open a TCP connection to the IP:port
S Accept the connection
C Send the HTTP request: GET /let/informatiekunde HTTP/1.1...
S Read the request, process it
S Form a response and send it

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`
C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde
C Resolve the host name: 129.125.2.51
C Find the default port number for HTTP: 80
C Open a TCP connection to the IP:port
S Accept the connection
C Send the HTTP request: GET /let/informatiekunde HTTP/1.1...
S Read the request, process it
S Form a response and send it
C Read the response, process it

C: Client, S: Server

# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`
C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde
C Resolve the host name: 129.125.2.51
C Find the default port number for HTTP: 80
C Open a TCP connection to the IP:port
S Accept the connection
C Send the HTTP request: GET /let/informatiekunde HTTP/1.1...
S Read the request, process it
S Form a response and send it
C Read the response, process it
C Close the connection

C: Client, S: Server

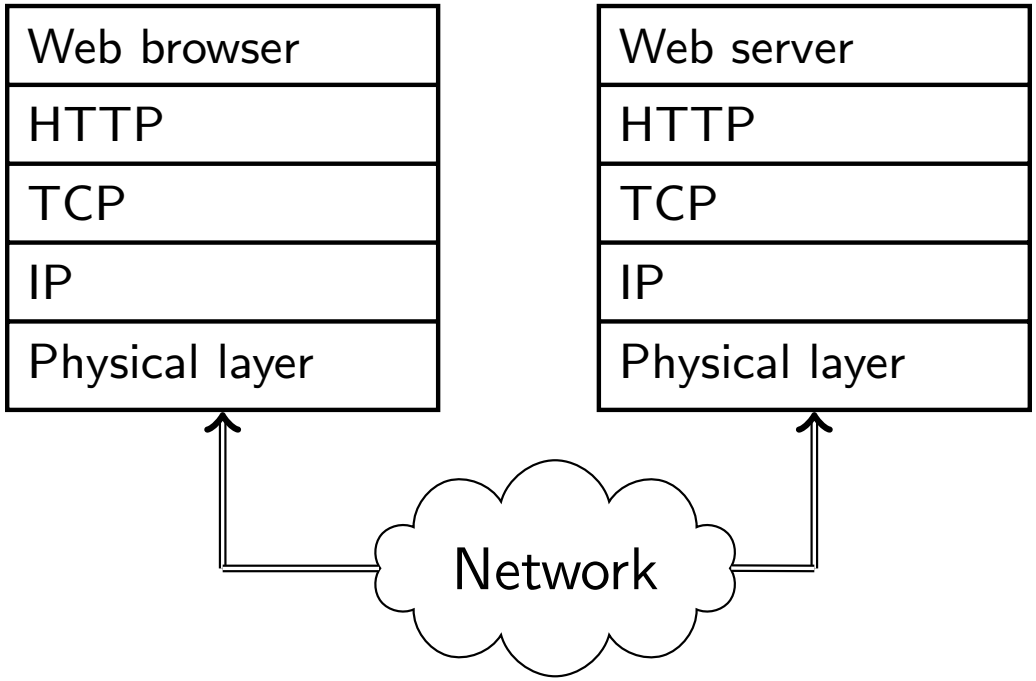
# What happens when you click on a link?

C Extract the URL: `http://www.rug.nl/let/informatiekunde`
C Parse the URL: Host: www.rug.nl, Protocol: HTTP, Resource: /let/informatiekunde
C Resolve the host name: 129.125.2.51
C Find the default port number for HTTP: 80
C Open a TCP connection to the IP:port
S Accept the connection
C Send the HTTP request: GET /let/informatiekunde HTTP/1.1...
S Read the request, process it
S Form a response and send it
C Read the response, process it
C Close the connection

This is still an overview, a lot more happens under the hood.

C: Client, S: Server

# Layers in communication




- Web server and web browser talks to each other using HTTP.
- The HTTP messages goes through a set of networking layers.
- We are mainly interested in HTTP, some aspects of TCP/IP networking is relevant to web programming.

# Three-slide introduction to TCP/IP (1)

- TCP/IP is the name of the network protocol family used in the Internet.
- It is more than TCP and IP. Just to list a few: `UDP`, `BGP`, `DHCP`, `ICMP`, `DNS`, `...`
- The IP protocol is connectionless, it does it's best to deliver a network packet to it's destination.
- IP does not guarantee the delivery of every packet.
- TCP works on IP, it is connection oriented. With TCP, you do not worry about the lost packets.

# Three-slide introduction to TCP/IP (2)

- The hosts in a TCP/IP network is identified with a unique IP address.
- IP(v4) addresses are 4-byte integers, e.g., 129.125.2.51 (New version of IP, IPv6, uses longer IP addresses).
- DNS maps more human readable domain names, like `www.rug.nl` to IP addresses.

DNS can be used for distributing load:
A particular host name can be assigned multiple IP addresses. For each DNS query, a DNS server will issue one of the IP addresses in a round-robin fashion

# Three-slide introduction to TCP/IP (3)

- Commonly used services have reserved port numbers, for example 80: HTTP, 443: HTTPS, 22: SSH . . .
- A server typically 'listens' on a reserved port for client connections.
- Clients reserve temporary port numbers.
- Each end of a connection is identified by IP address/port number pairs.
- Connection is typically initiated by a client, any of the parties can close the connection.

# Anatomy of a URL

http://www.rug.nl:80/let/informatiekunde
① ② ③ ④

① Scheme indicates the protocol. The rest of the URL may be different depending on the scheme.

② Domain name is the name of the host where the HTTP service runs.

③ Port number can optionally be given in cases where the service do not run on the default port.

④ Path typically identifies the (HTML) files on the server, but can express more than a file name. The interpretation is dependent on the web server.

# HTTP: an overview

- HTTP is a request-response protocol. Clients asks for an operation on a resource, possibly with some content, and server responds, likely with some content.
- The requested operation has to be one of 9 HTTP methods, like `GET`, `HEAD` or `POST`.
- Response message starts with a status message.
- Both request and response can include additional headers, which provide additional information.
- HTTP protocol does not encrypt the communication, nor has it any mechanism to verify the identity of server or the client.
- HTTPS is an extension of HTTP that uses Secure Socket Layer (SSL).

# HTTP requests

```
1  GET / HTTP/1.1
2  Host: www.rug.nl
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:8.0) ...
4  Accept: text/html,application/xhtml+xml, ...
5  Accept-Language: en-us,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Accept-Charset: UTF-8,*
8  Connection: keep-alive
9  Cookie: acceptedLanguages=en; s_nr=1321910886052 ...
10
```

- First line is the actual request, here using method `GET`.
- The rest of the lines are headers that provide additional information.
- The empty line (10) is important. It signals the end of headers.

# HTTP response

```
1  HTTP/1.1 200 OK
2  Date: Wed, 23 Nov 2011 01:11:25 GMT
3  Server: Apache/2.2
4  Last-Modified: Tue, 11 Mar 2008 11:35:02 GMT
5  Content-Length: 260
6  Content-Type: application/html
7
8  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML ...
9  <html>
10 ...
```

- The first line is the status line.
- Again, server gives us a set of header lines followed by an empty line and the content.
- The response can also indicate a permanent or temporary error, or a redirection message.

# HTTP methods

HTTP standard defines 9 methods, but we are only interested in

**GET** Typically used to get a static content (e.g., file). But it can also be used for dynamic content (we will return to this).

**HEAD** It is like GET, but server only responds with headers, no content is transferred.

**POST** is used when client needs to transfer some content. Typically content is the name/value pairs from a HTML form. However, it can be anything that server/client agree on.

Others (for the sake of completeness): PUT, DELETE, OPTIONS, CONNECT, PATCH, TRACE.

# HTTP headers

- Both requests and responses may be of interest for server-side programming.
- Request headers include users' preferences, such as `Accept-Language` or certain information about users' environment that you may want to know, such as `User-agent`.
- You can set response headers, to communicate with the browsers. For example, `Refresh` will instruct the browser to reload the page after specified time, or `Cache-Control` gives you a way to tell the browser if/how long the content can be cached.

# HTTP Cookies

- A specific HTTP header field `Cookie` (in request) or `Set-Cookie` (in response), is widely used for web programming.
- A cookie is a piece of information a HTTP servers asks the client to retain for until a specific expiry date.
- The server sends a cookie to a client using,
  `Set-Cookie: name=val, expires=datetime, domain=d, path=p`
- The client (if enabled) sends the matching cookie that are not expired with every request.
- Cookies are typically used for session management, (some form of) authentication, or applications like shopping charts.

# HTTP Cookies

- A specific HTTP header field `Cookie` (in request) or `Set-Cookie` (in response), is widely used for web programming.
- A cookie is a piece of information a HTTP servers asks the client to retain for until a specific expiry date.
- The server sends a cookie to a client using,
  `Set-Cookie: name=val, expires=datetime, domain=d, path=p`
- The client (if enabled) sends the matching cookie that are not expired with every request.
- Cookies are typically used for session management, (some form of) authentication, or applications like shopping charts.

# A summary so far

- The WWW, and as a result, the web-programming environment works over HTTP.
- HTTP is a request-response protocol.
- A HTTP request is originated by a client (e.g. browser) and includes a method, and a set of headers.
- A HTTP response includes a status code, additional headers, and the content.
- A server-side web-application (whether it is a CGI program, or a embedded interpreter) has access to raw HTTP data sent by the client, and form the response the way it wants.
- You will not typically deal with the raw HTTP messages, but knowing what lies underneath helps.

# HTML Forms

```
<form action="form.php" method="post">
Name: <input type="text" name="name">
Password: <input type="passowrd" name="pass">
</form>
```

- HTML forms are defined in `<form>` tag.
- Input elements are specified inside the form.
- `<input>` tags specifies most common input types, but some elements have different tags, e.g., `<textarea>`.
- `action` attribute specifies the URL that will handle the form input when submitted.
- `method` is either `GET` or `POST`.
- You can specify a default value, typically using `value` attribute.

# Input types

| | |
|---|---|
| text | A single line text field |
| password | Same as text, but the input is not displayed |
| textarea | A multi-line text field |
| radio | Radio button, among a set of buttons user can select only one |
| checkbox | Like radio buttons, but user can select multiple (or none) |
| option/select | Create drop-down lists. |
| button | A button |
| img | A button with an image |
| file | A file upload button |
| hidden | A static value. |

# Form example (1)

```
 1 <form action="form.php" method="post">
 2 Name: <input type="text" name="name"><br>
 3 Password: <input type="password" name="pass"><br>
 4 Gender:
 5 <input type="radio" name="gender" value="M"> Male /
 6 <input type="radio" name="gender" value="F"> Female
 7
 8 <p>You like:
 9 <input type="checkbox" name="hobby" value="bike"> Biking
10 <input type="checkbox" name="hobby" value="hike"> Hiking
11 <input type="checkbox" name="hobby" value="cook"> Cooking
12
13 <p>How do you feel today?
14 <textarea name="feelings" cols=20 rows=2></textarea><br>
15
16 <p> <input type="reset" value="clear">
17 <input type="submit" value="submit">
18 <input type="image" src="logo.png"><br>
19 </form>
```

# Form example (2)

```
 1 <form action"form2.php">
 2 <p>How do you feel again? <select name="feelings2">
 3 <option>good</option>
 4 <option>fine</option>
 5 <option>bored</option>
 6 <option>sleepy</option>
 7 </select>
 8 <p>and again? <select multiple name="feelings2">
 9 <option>good</option>
10 <option>fine</option>
11 <option>bored</option>
12 <option>sleepy</option>
13 </select>
14 <p>File name: <input type="file" name="file"/><br/>
15 <p><button type="submit" name="submit" value="submit">
16     press here
17 </button>
18 <button type="submit" name="submit" value="submit">
19     <img src="logo.png">
20 </button>
21 </form>
```

# GET or POST ?

- GET method,
  - Encodes the form output in the URL:
    http://my.hostname/form.php?name=myname&age=myage
  - PRO: users can bookmark it.
  - CON: the values are visible on the URL bar (passwords?).
  - CON: there may be size limitation. No set standard, but rule of thumb: no more than about 2K characters.
- POST method,
  - Encodes the form output in the message body.

```
POST /form.php HTTP/1.1
header: value
....

name=myname&age=myage
```

  - PRO: with the help of HTTP, you can pass values securely.
  - PRO: it can handle much more data.
  - CON: you cannot bookmark it output of a form submission.

# PHP and forms

- Form output from PHP is easy, it is just HTML output.
- If a form uses your PHP script as `action`, you can access the form input using two super globals,
  - `$_POST` if you used `method=post`
  - `$_GET` if you used `method=get`
- PHP also provides a unified associative array `$_REQUEST`, which combines both `$_POST` and `$_GET`.
- All these variables are associative arrays. For example, if you had a form input with `name="username"`, the value could be accessed using `$_REQUEST['username']`.

# A simple form example

```
<!-- This is a simple HTML form
 -->
<form action=form.php method=get>
Username: <input type="text"
                 name="name"/><br>
Password: <input type="password"
                 name="pass"/><br>
          <input type="submit"
                 name="submit"
                 value="submit">
</form>
```

```
<!-- this is a PHP script that
     processes it -->

<?php

if (!isset($_REQUEST['submit'])){
    echo "Why are you here?";
} else {
    echo "Your name is: "
       . "${_REQUEST['name']}<br>";
    echo "And passowrd: "
       . "${_REQUEST['pass']}<br>";
}
?>
```

# PHP and forms: a common trick

```
1  <!-- This is a simple HTML form
2   -->
3  <form action=form.php method=get>
4  Username: <input type="text"
5                   name="name"/><br>
6  Password: <input type="password"
7                   name="pass"/><br>
8            <input type="submit"
9                   name="submit"
10                  value="submit">
11 </form>
```

```
1  <!-- this is a PHP script that
2       processes it -->
3
4  <?php
5
6  if (!isset($_REQUEST['submit'])){
7      echo "Why are you here?";
8  } else {
9      echo "Your name is: "
10         . "${_REQUEST['name']}<br>";
11     echo "And passowrd: "
12         . "${_REQUEST['pass']}<br>";
13 }
14 ?>
```

# PHP and forms: a common trick

```
1
2  <?php
3  if (!isset($_REQUEST['submit'])){
4  ?>
5
6  <form  method="post"
7         action="<?php echo "${_SERVER['PHP_SELF']}";?>">
8  Username: <input type="text" name="name"/><br>
9  Password: <input type="password" name="pass"/><br>
10           <input type="submit" name="submit" value="submit">
11 </form>
12
13 <?php
14
15 } else {
16     echo "Your name is: " . "${_REQUEST['name']}<br>";
17     echo "And passowrd: " . "${_REQUEST['pass']}<br>";
18 }
19 ?>
```

# PHP and cookies

- PHP also gives you an easy way to access and set cookies.
- Cookies are stored in another super global associative array `$_COOKIE`
- Assuming you have a cookie with name `user`, you can access it using `$_COOKIE['user']`.
- Cookies are also present in the combined associative array `$_REQUEST`
- You can set cookies with function `setcookie()`. For example `setcookie($name, $value, $expiry);`
- You have to set the cookies before any content (remember: they are part of the HTTP headers, not the content).
- Technically, you cannot delete a cookie. But if you (re)set it with a expiry date in past, browser will do it for you.

# PHP and cookies

```
 1 <?php
 2     if (!isset($_COOKIE['MyCookie'])) {
 3         setcookie('MyCookie',
 4                   'some value',
 5                   time()+3600*24*7);
 6     }
 7 ?>
 8 <html>
 9 <!--- ... some html stuff -->
10
11 <?php
12     if (!isset($_COOKIE['MyCookie'])) {
13         echo "You do not have the cookie yet.";
14     } else {
15         echo "MyCookie = ${_COOKIE['MyCookie']}";
16     }
17 ?>
```

# Summary & Next week

This week:

- The multi- (or three-)tier architecture.
- The HTTP protocol, and a bit of TCP/IP.
- HTML forms, cookies, and how to use them in PHP.

Next week:

- More on Databases & SQL
- More on using databases through PHP (mainly pear DB).