Dr Greg Wadley

INFO90002 Database Systems & Information Modelling

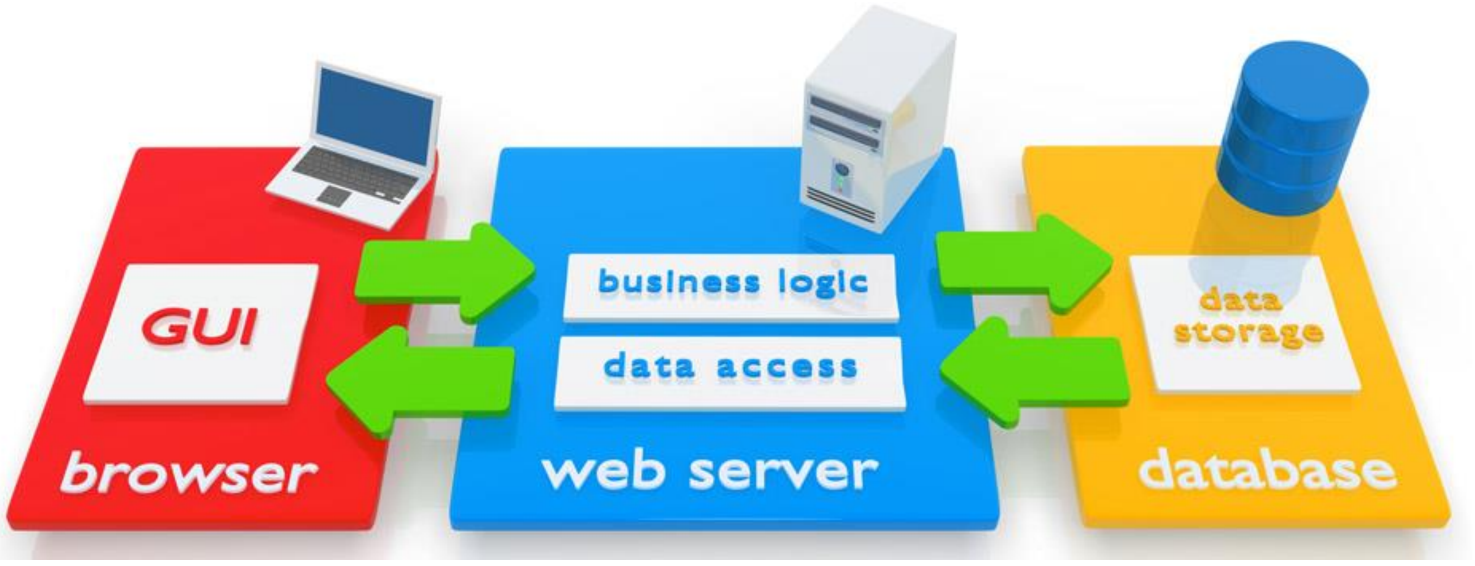
Week 07

Web Apps

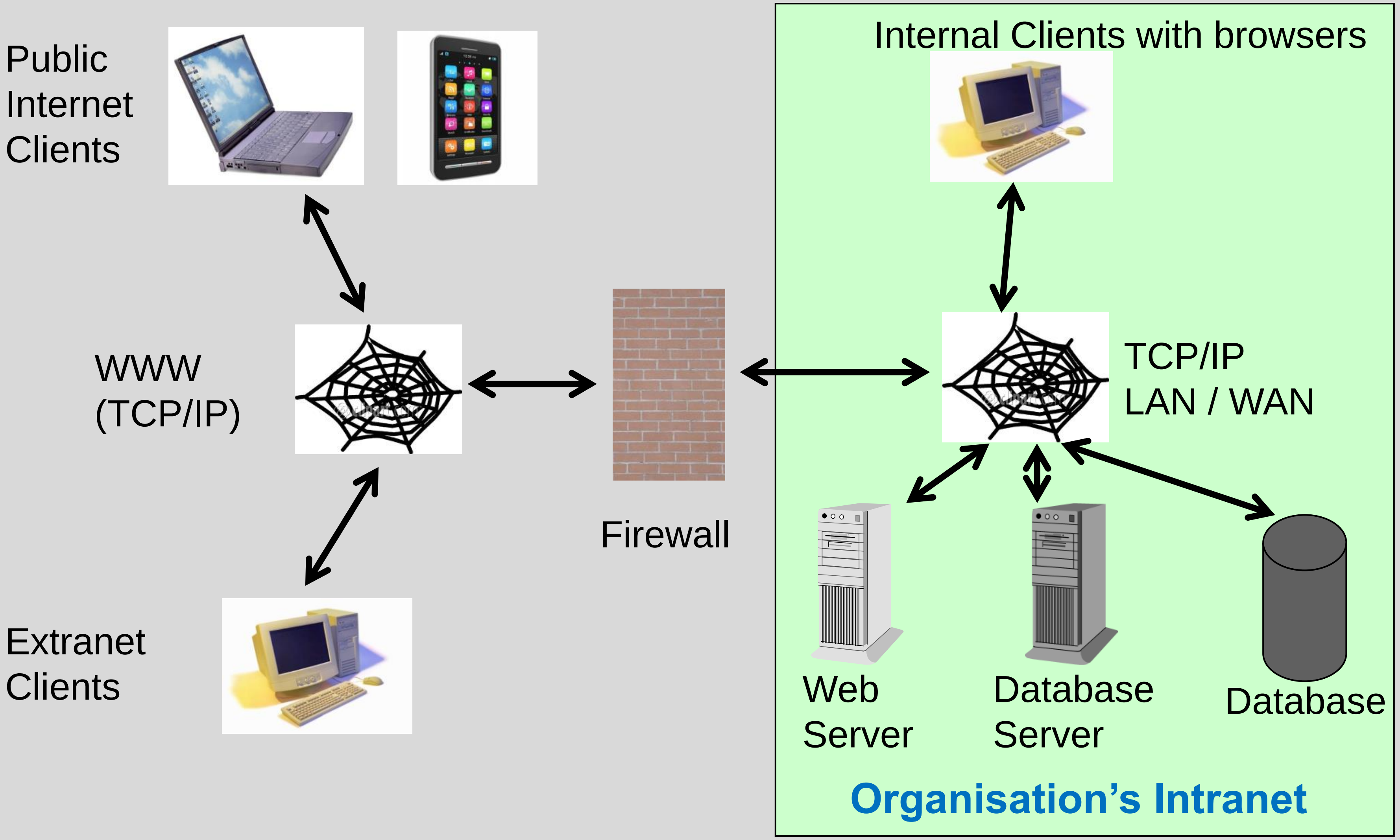
THE UNIVERSITY OF MELBOURNE

Today's Session…

- Why web apps?
- How web apps work
- Making an HTML document
- Connecting to the DB
- Demo web app
- Web services

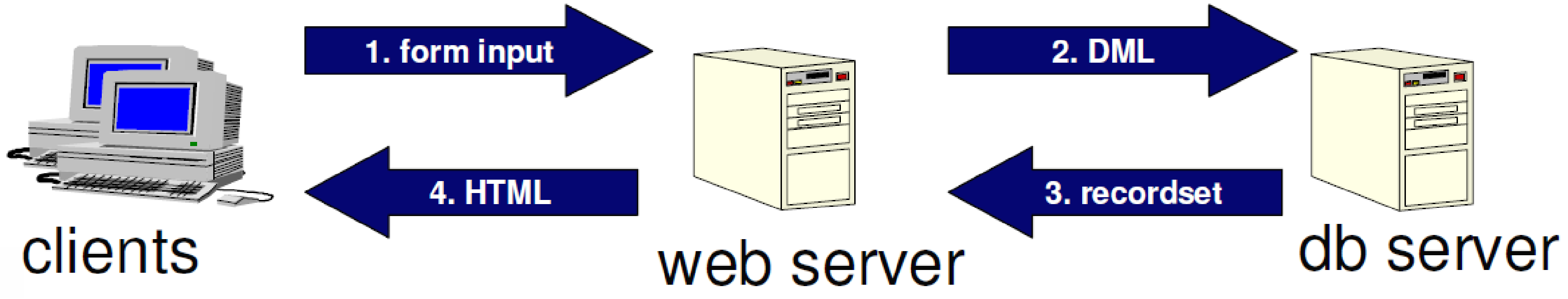


Architecture of a web app



Why create web applications?

- Web browsers are ubiquitous
- No need to install client software for external customers
- Simple communication protocols
- Platform and Operating System independent
- Reduction in development time and cost
- Has enabled eGov, eBusiness, eCommerce, B2B, B2C



Web infrastructure

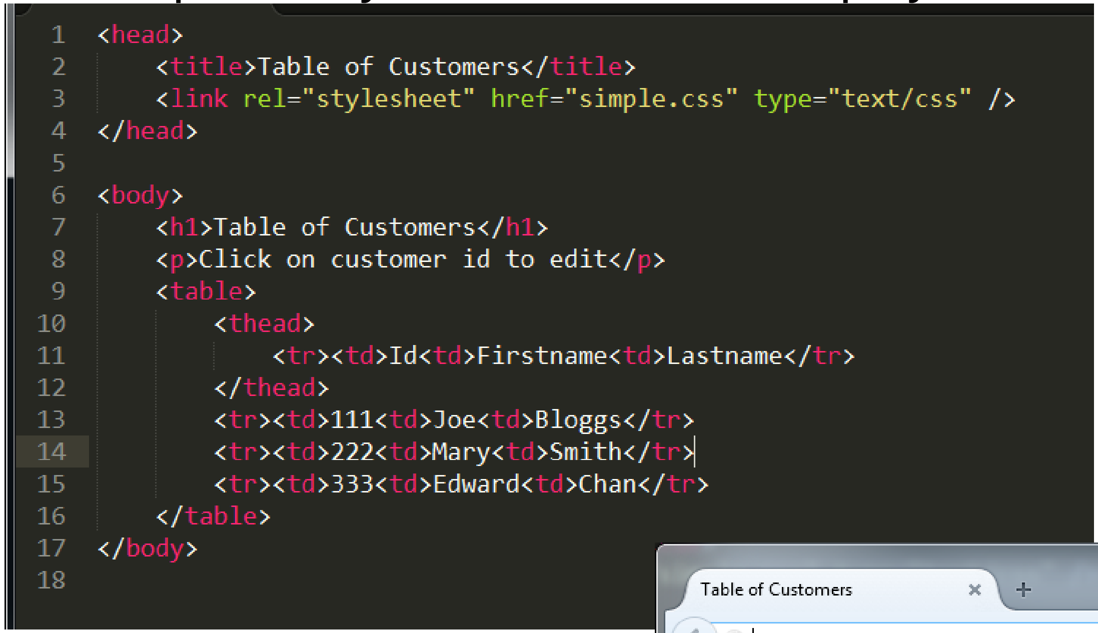
- Browser
  - Software that retrieves and displays HTML documents
- Web Server
  - Software that responds to requests from browsers by transmitting HTML and other documents to browsers
- Web pages (HTML documents)
  - Static web pages
    - content established at development time
  - Dynamic web pages
    - content dynamically generated using data from database
- World Wide Web (WWW)
  - The total set of interlinked hypertext documents residing on Web servers worldwide

Web-related languages

- Hypertext Markup Language (HTML)
  - Markup language used to define a web page
- Cascading Style Sheets (CSS)
  - Control appearance of an HTML document
- JavaScript (JS)
  - Scripting language that enable interactivity in HTML documents
- Extensible Markup Language (XML)
  - Markup language used to transport data between web services

Web page = HTML document

- a structured file of elements defined by HTML tags
- interpreted by web browser for display



```
<head>
    <title>Table of Customers</title>
    <link rel="stylesheet" href="simple.css" type="text/css" />
</head>

<body>
    <h1>Table of Customers</h1>
    <p>Click on customer id to edit</p>
    <table>
        <thead>
            <tr><td>Id<td>Firstname<td>Lastname</tr>
        </thead>
        <tr><td>111<td>Joe<td>Bloggs</tr>
        <tr><td>222<td>Mary<td>Smith</tr>
        <tr><td>333<td>Edward<td>Chan</tr>
    </table>
</body>
```

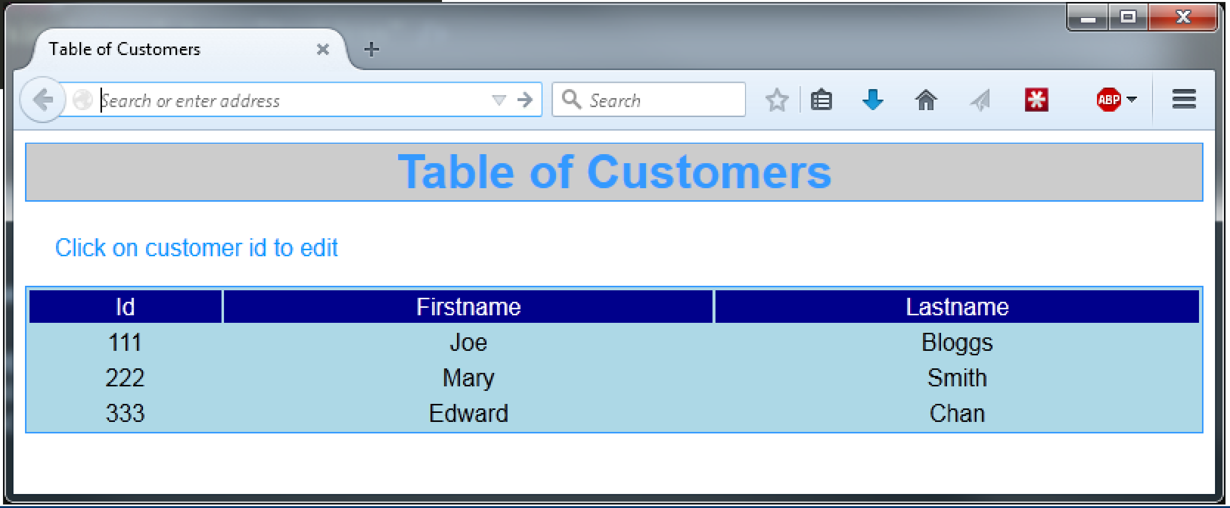


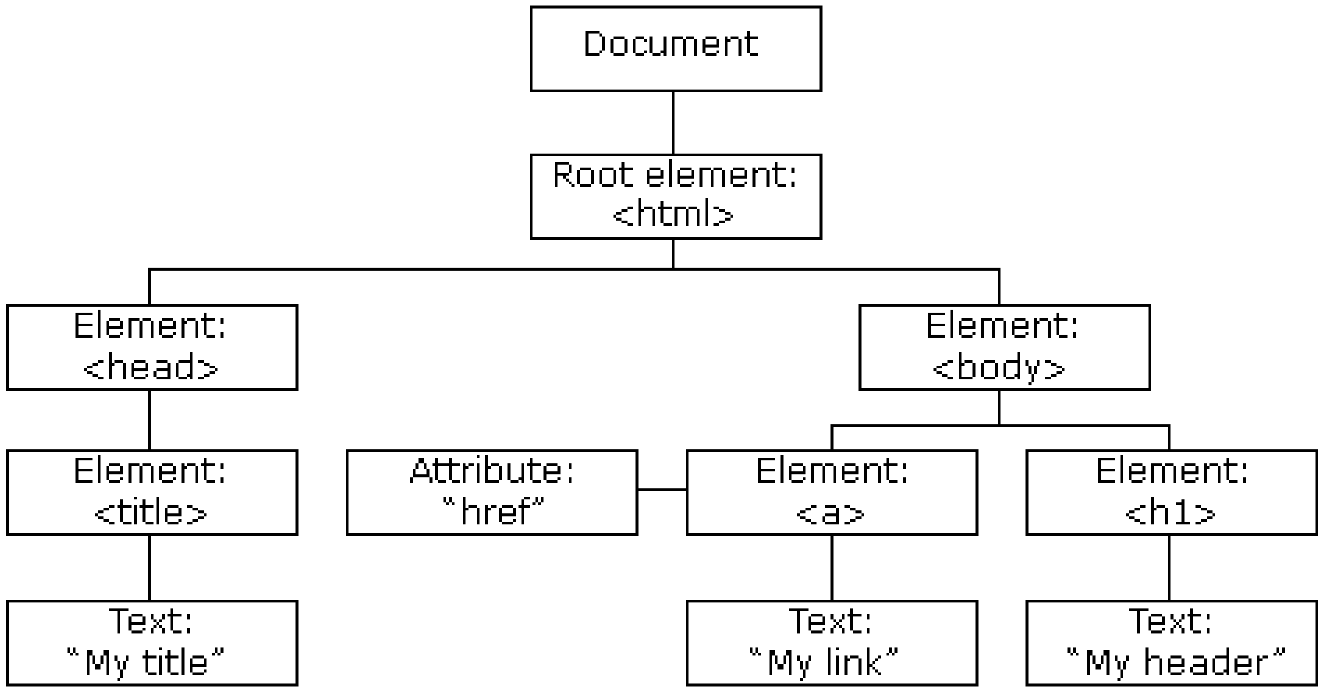
Table of Customers

Click on customer id to edit

Id	Firstname	Lastname
111	Joe	Bloggs
222	Mary	Smith
333	Edward	Chan

Structure of an HTML document

- elements are structured as a tree (one web page = one tree)
- divided into a HEAD and a BODY
- the BODY is what you see displayed in the browser
- BODY is divided into elements such as headings, paragraphs, tables, lists …



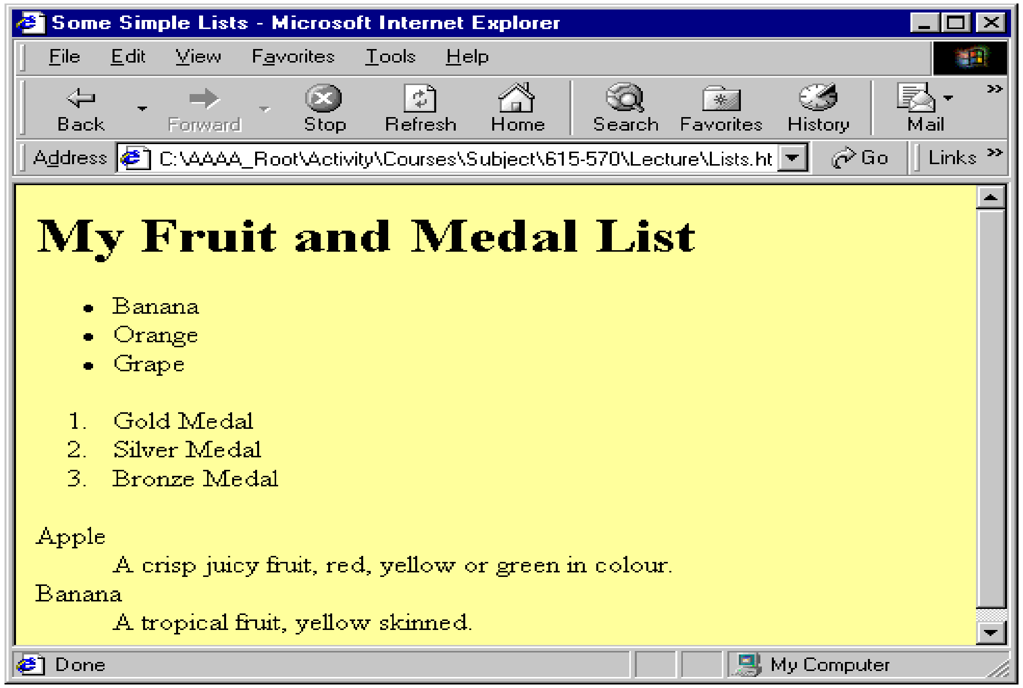
picture source: W3 Schools

Important HTML elements

<HEAD> … </HEAD>	• document header.
<BODY> … </BODY>	• document body
<H1> … </H1>	• Heading type 1
<H6> … </H6>	• … to Heading type 6.
<P> … </P>	• paragraph.
<TABLE>	• table
<TR>	• table row
<TD>	• table data
<UL>	• list
<LI>	• list item

HTML list

```
<HTML> <HEAD> <title>Some Simple Lists</title> </HEAD>
<BODY bgcolor="#FFFF99">
<H1>My Fruit and Medal List </H1>
<UL>
  <LI>Banana</LI>
  <LI>Orange</LI>
  <LI>Grape</LI>
</UL>
<OL>
  <LI>Gold Medal</LI>
  <LI>Silver Medal</LI>
  <LI>Bronze Medal</LI>
</OL>
<DL>
  <DT>Apple
    <DD>A crisp juicy fruit, red, yellow or green in colour.
  <DT>Banana
    <DD>A tropical fruit, yellow skinned.
</DL>
</BODY> <HTML>
```


Some Simple Lists - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address C:\AAAA_Root\Activity\Courses\Subject\615-570\Lecture\Lists.ht
Go Links
Done
My Computer

My Fruit and Medal List

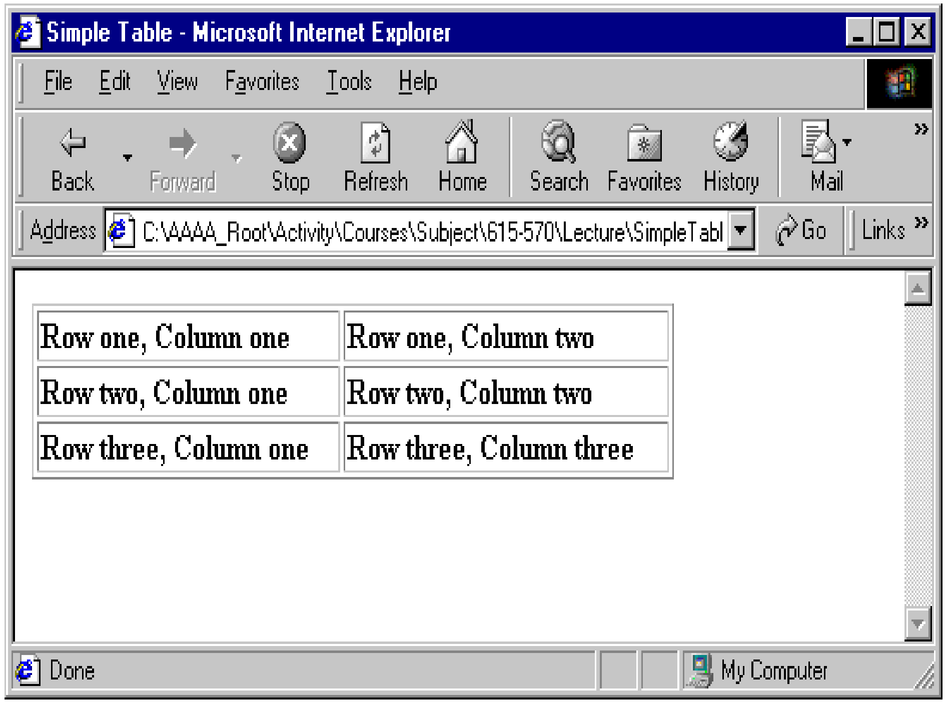
- Banana
- Orange
- Grape

1. Gold Medal
2. Silver Medal
3. Bronze Medal

Apple
    A crisp juicy fruit, red, yellow or green in colour.
Banana
    A tropical fruit, yellow skinned.

HTML table

```
<HTML>  <HEAD> <TITLE>Simple Table</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</HEAD>
<BODY>
<TABLE>
 <TR>
   <TD>Row one, Column one</TD>
   <TD>Row one, Column two</TD>
 </TR>
 <TR>
   <TD>Row two, Column one</TD>
   <TD>Row two, Column two</TD>
 </TR>
 <TR>
   <TD>Row three, Column one</TD>
   <TD>Row three, Column three</TD>
 </TR>
</TABLE>  </BODY> </HTML>
```

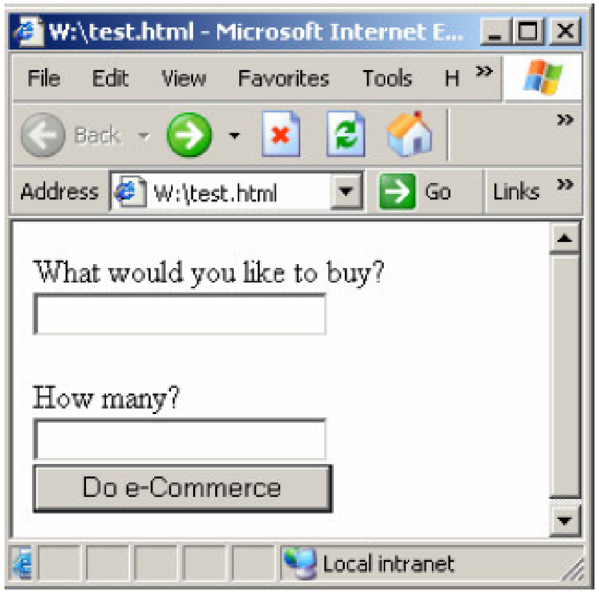

Simple Table - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address C:\AAAA_Root\Activity\Courses\Subject\615-570\Lecture\SimpleTabl
Go Links
Done
My Computer

Row one, Column one	Row one, Column two
Row two, Column one	Row two, Column two
Row three, Column one	Row three, Column three

HTML form

- Forms allow users to input data to a web page
- The web server process the user's input using the file named in the 'action' attribute.

```
<form action = "buy.pl" method="post">

<p> What would you like to buy? <br>
<input type="text" name="product">

<p> How many? <br>
<input type="text" name="quantity"> <br>

<input type="submit" value="Do e-Commerce">
</form>
```


W:\test.html - Microsoft Internet E...
File Edit View Favorites Tools H
Back
Address W:\test.html
Go
Links
What would you like to buy?
How many?
Do e-Commerce
Local intranet

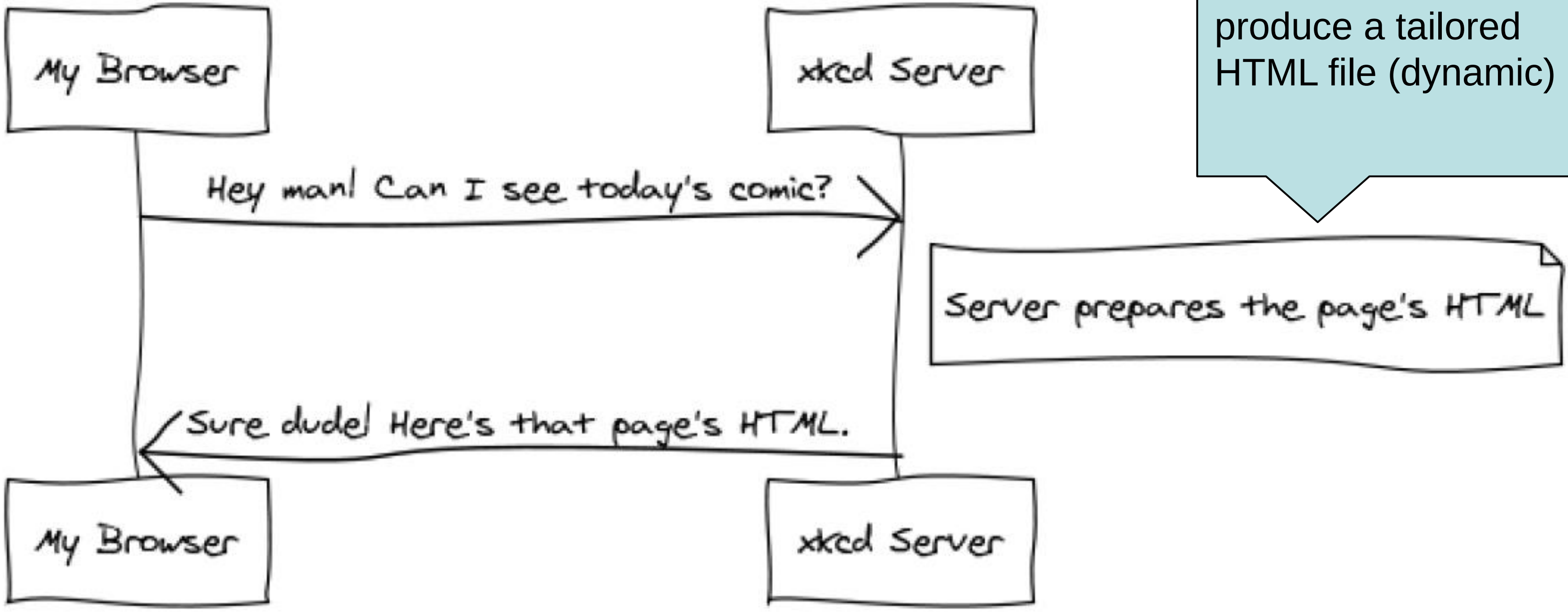
Example HTML form

browser displays form,
sends input data to a script called 'buy.pl'

HTTP: how HTML documents move

- User wants to see a web page
- Types URL into browser
- Browser fetches page from server and displays it

could simply load a pre-prepared HTML file (static), or query the database to produce a tailored HTML file (dynamic)



My Browser
xkcd Server
Hey man! Can I see today's comic?
Server prepares the page's HTML
Sure dude! Here's that page's HTML.
My Browser
xkcd Server

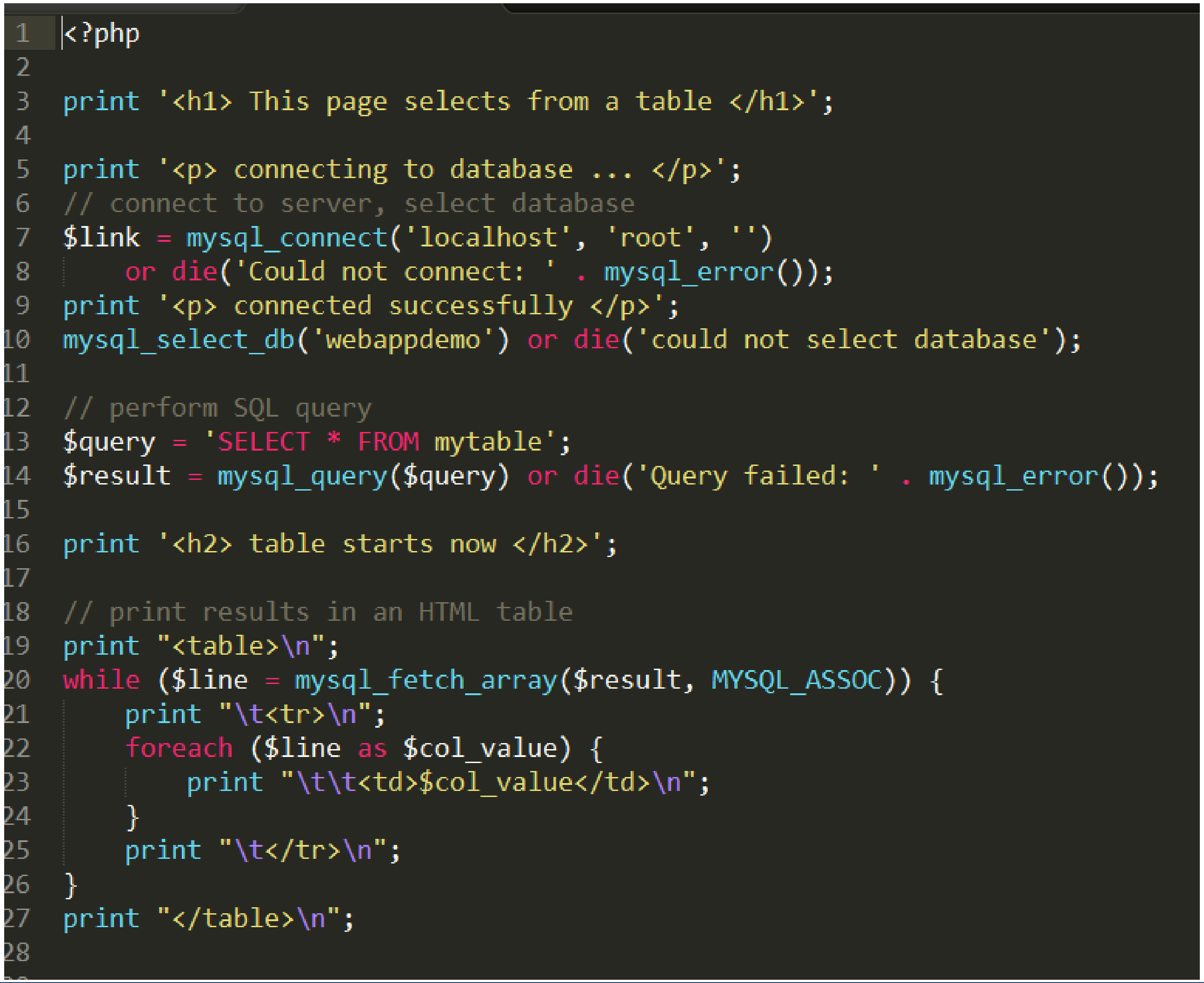
picture source: Symfony Book

Static vs Dynamic web pages

- STATIC web page
  - the URL identifies a file on the server's file system
  - server fetches the file and sends it to the browser
  - the file contains HTML
  - browser interprets the HTML for display on screen
- DYNAMIC web page
  - URL identifies a program to be run
  - web app runs the program
  - program typically retrieves data from database
  - elements such as TABLE, LIST are populated with data
    - web app uses LOOPS to fill the contents of TABLEs and LISTs.
    - e.g. SELECT * FROM Product; (returns a set of product entities)
    - FOR p IN ProductList, print a row in HTML table

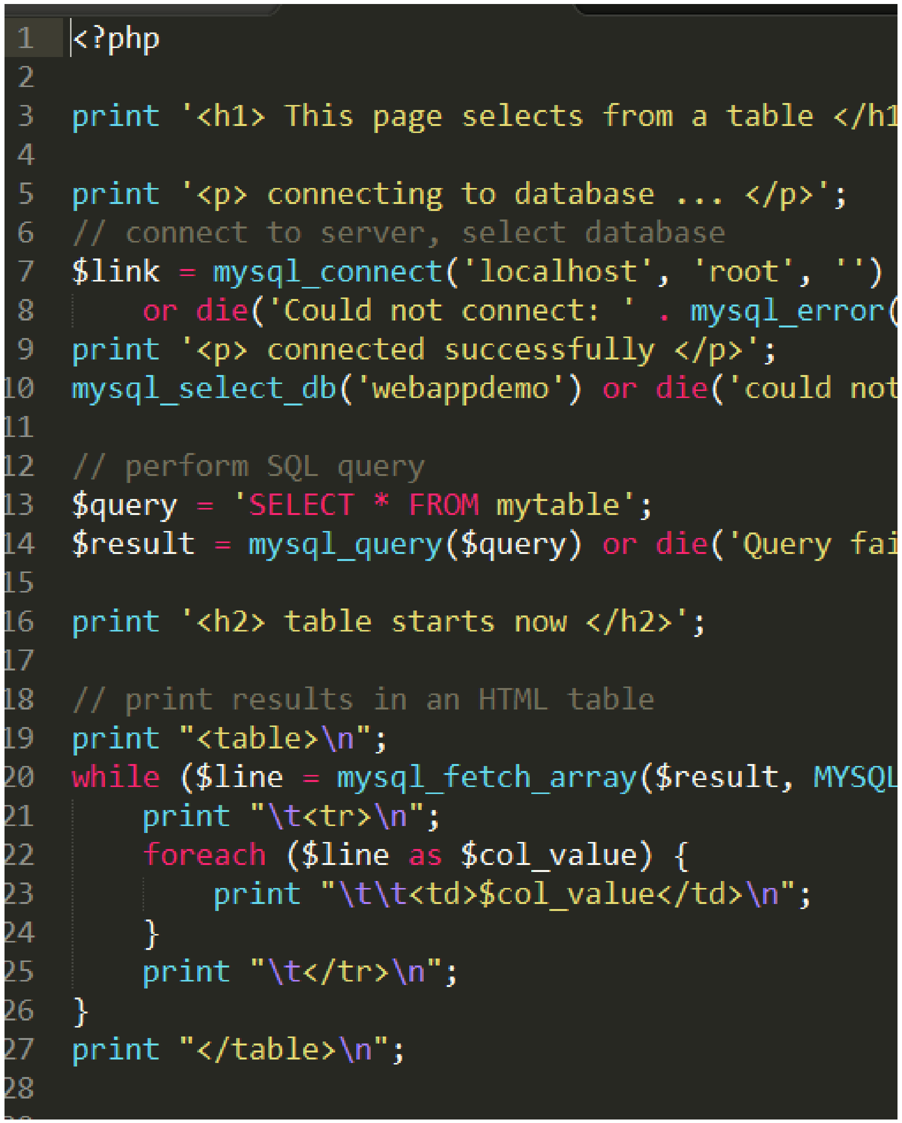
Simple web app using PHP and SQL

- program logs into db
- selects all rows from database table
- displays them inside an HTML table

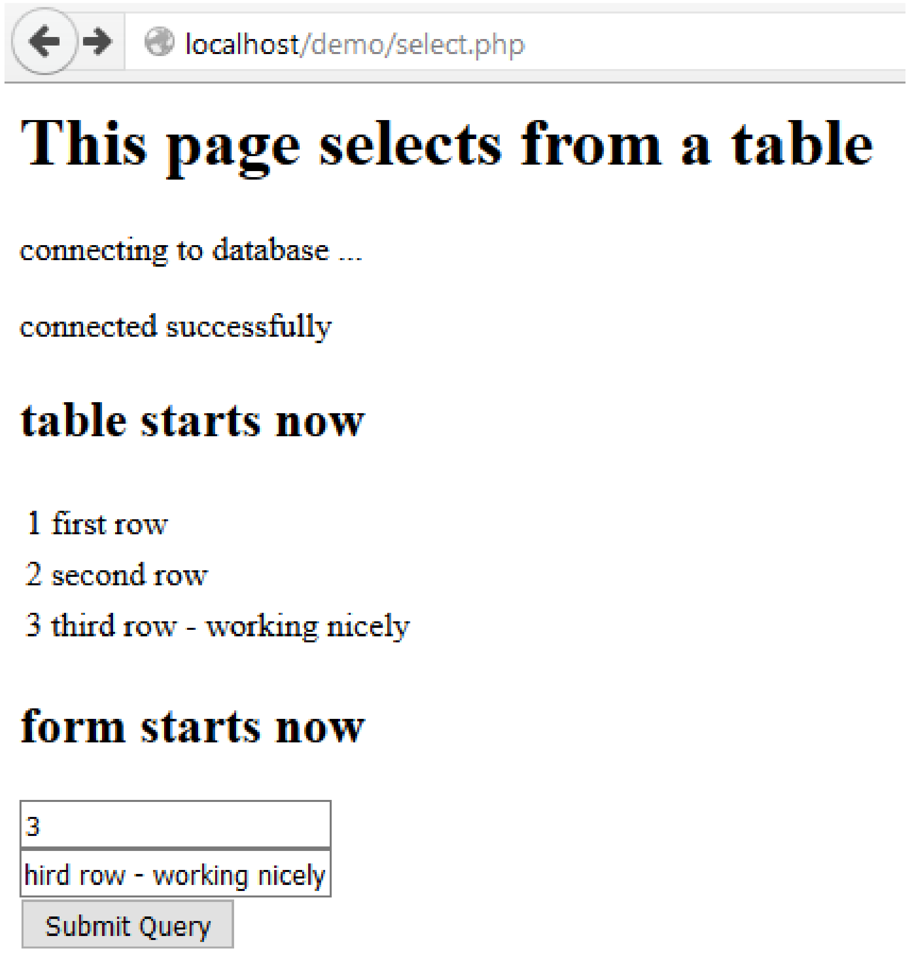


```php
<?php

print '<h1> This page selects from a table </h1>';

print '<p> connecting to database ... </p>';
// connect to server, select database
$link = mysql_connect('localhost', 'root', '')
    or die('Could not connect: ' . mysql_error());
print '<p> connected successfully </p>';
mysql_select_db('webappdemo') or die('could not select database');

// perform SQL query
$query = 'SELECT * FROM mytable';
$result = mysql_query($query) or die('Query failed: ' . mysql_error());

print '<h2> table starts now </h2>';

// print results in an HTML table
print "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    print "\t<tr>\n";
    foreach ($line as $col_value) {
        print "\t\t<td>$col_value</td>\n";
    }
    print "\t</tr>\n";
}
print "</table>\n";

```

Simple web app: *select*



```
<?php

print '<h1> This page selects from a table </h1

print '<p> connecting to database ... </p>';
// connect to server, select database
$link = mysql_connect('localhost', 'root', '')
    or die('Could not connect: ' . mysql_error(
print '<p> connected successfully </p>';
mysql_select_db('webappdemo') or die('could not

// perform SQL query
$query = 'SELECT * FROM mytable';
$result = mysql_query($query) or die('Query fai

print '<h2> table starts now </h2>';

// print results in an HTML table
print "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL
    print "\t<tr>\n";
    foreach ($line as $col_value) {
        print "\t\t<td>$col_value</td>\n";
    }
    print "\t</tr>\n";
}
print "</table>\n";

```



localhost/demo/select.php

This page selects from a table

connecting to database ...

connected successfully

table starts now

1	first row
2	second row
3	third row - working nicely

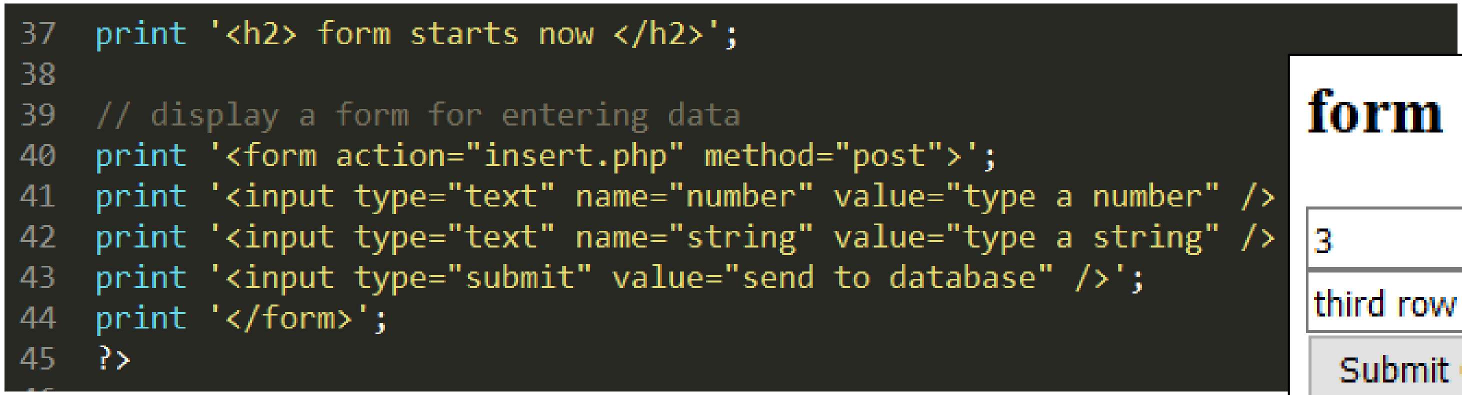
form starts now

3

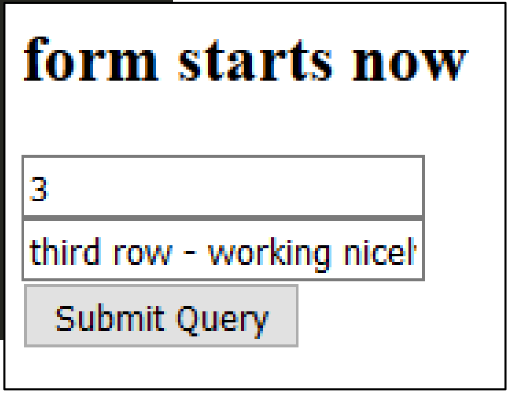
hird row - working nicely

Submit Query

Simple web app: *insert*

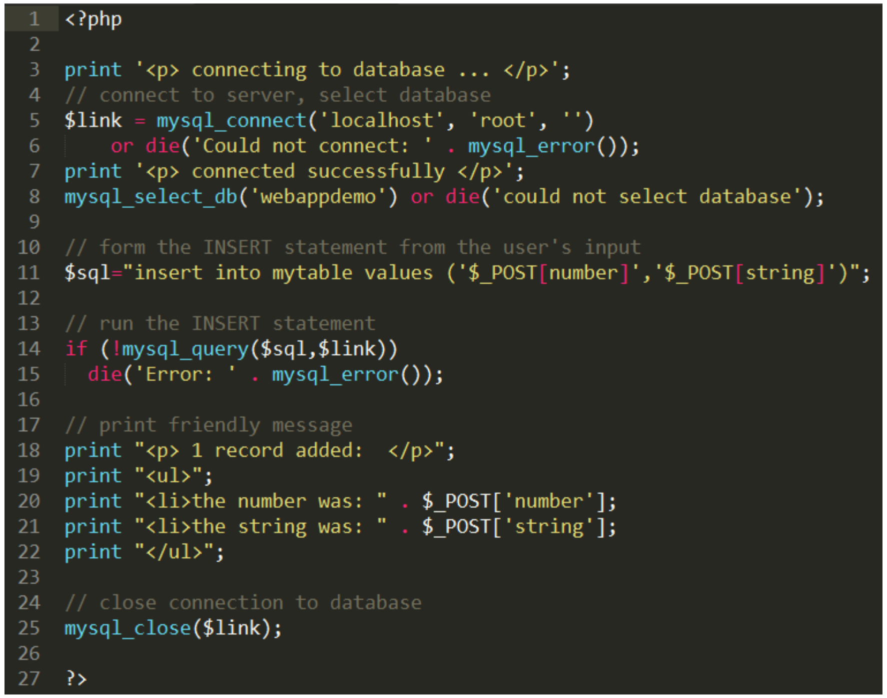


```php
print '<h2> form starts now </h2>';

// display a form for entering data
print '<form action="insert.php" method="post">';
print '<input type="text" name="number" value="type a number" />';
print '<input type="text" name="string" value="type a string" />';
print '<input type="submit" value="send to database" />';
print '</form>';
?>
```

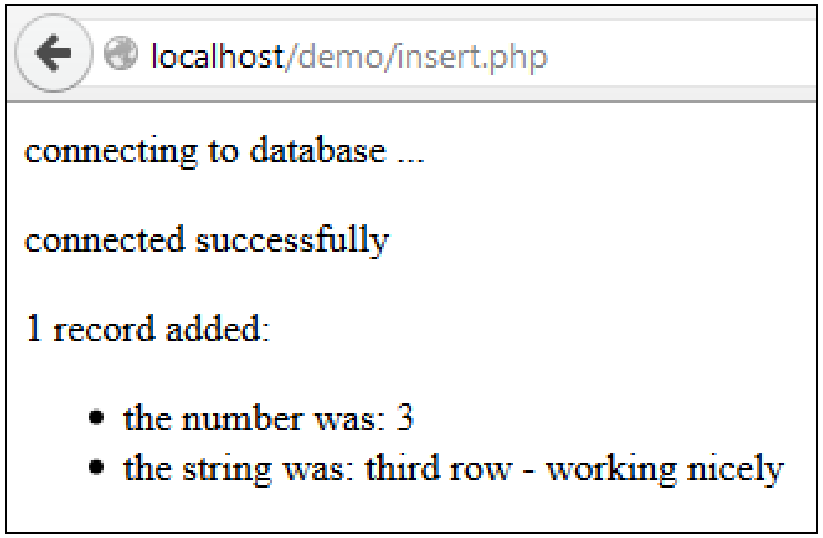


form starts now

```
3
third row - working nicel
```



```php
<?php

print '<p> connecting to database ... </p>';
// connect to server, select database
$link = mysql_connect('localhost', 'root', '')
    or die('Could not connect: ' . mysql_error());
print '<p> connected successfully </p>';
mysql_select_db('webappdemo') or die('could not select database');

// form the INSERT statement from the user's input
$sql="insert into mytable values ('$_POST[number]','$_POST[string]')";

// run the INSERT statement
if (!mysql_query($sql,$link))
  die('Error: ' . mysql_error());

// print friendly message
print "<p> 1 record added:  </p>";
print "<ul>";
print "<li>the number was: " . $_POST['number'];
print "<li>the string was: " . $_POST['string'];
print "</ul>";

// close connection to database
mysql_close($link);

?>
```



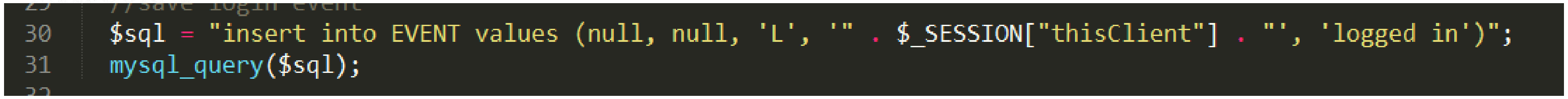
connecting to database ...

connected successfully

1 record added:

- the number was: 3
- the string was: third row - working nicely

Problems with old-style web apps



```
//save login event
$sql = "insert into EVENT values (null, null, 'L', '" . $_SESSION["thisClient"] . "', 'logged in')";
mysql_query($sql);
```

- Placing “raw” SQL inside PHP/HTML files
  - Mixes presentation, business, database logic
  - Hard to maintain when things change
  - Want separation of concerns e.g. MVC
- Lots of reinvention of wheels
  - each dev writes their own solution to common features
  - e.g. login security, presentation templates, database access
- Increasing variety of clients e.g. phones and tablets
  - Manually program for different platforms
- => web application frameworks
  - examples: Ruby on Rails, .Net, Symfony, AngularJS, Django

Web Services

- The WWW allows humans to access remote databases
- Web Services allow *computers* to access remote databases
- 2 major approaches: SOAP and REST
  - Simple Object Access Protocol
  - Representational State Transfer
- structured data usually returned in XML or JSON format
- REST nouns are resources, addressed via URIs
- REST verbs correspond to DML statements
- GET (select), POST (insert), PUT (update), DELETE (delete)
- Try this example web service
  https://www.googleapis.com/books/v1/volumes?q=quilting

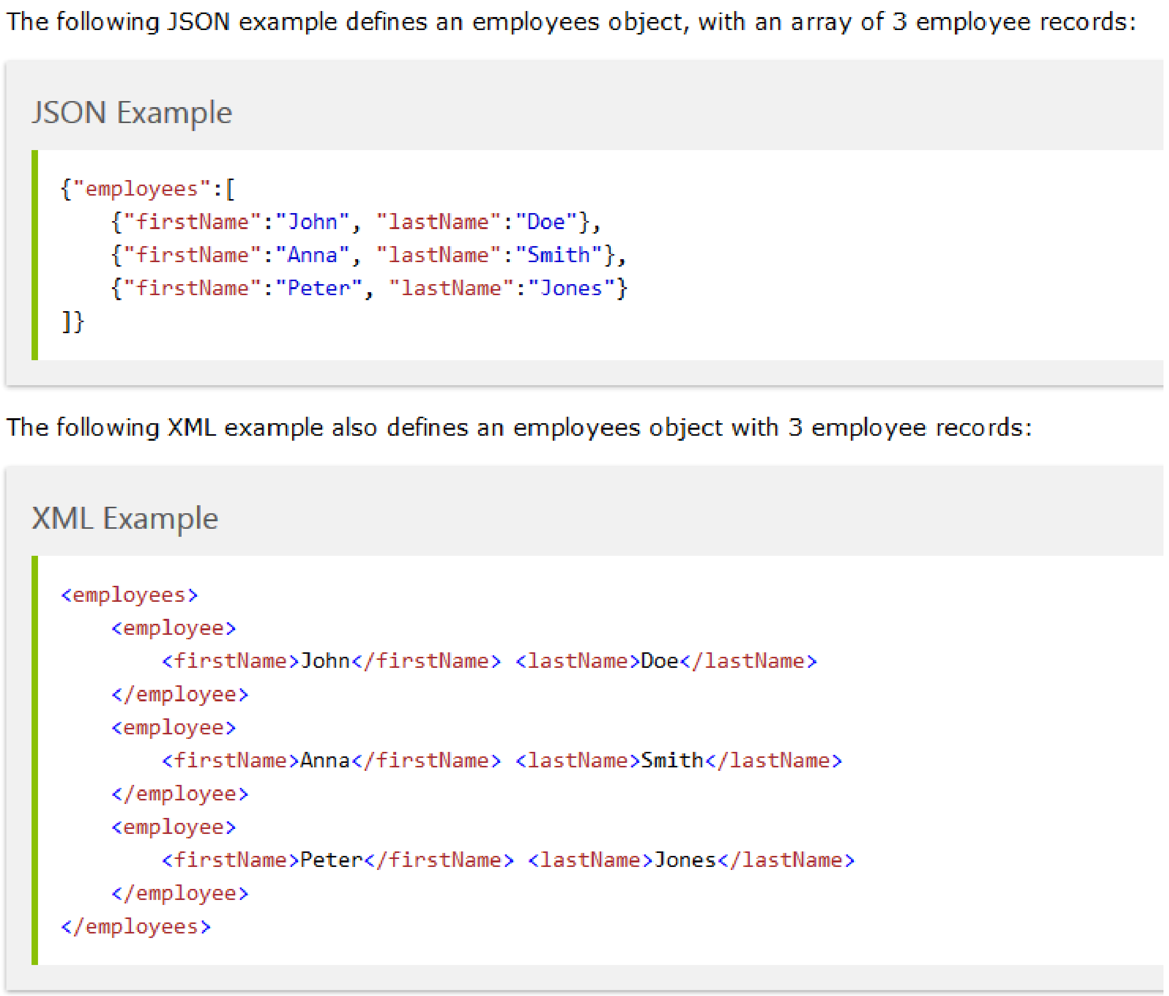
XML and JSON data formats

- used by web services for data exchange

- XML
  eXensible Markup Language

- JSON
  JavaScript Object Notation

(example sourced from W3 schools)



The following JSON example defines an employees object, with an array of 3 employee records:

JSON Example

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

The following XML example also defines an employees object with 3 employee records:

XML Example

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```