Web Programming

© Leonidas Fegaras
University of Texas at Arlington
Dynamic Web Pages

- **Static web page**: every time you request this page, you get exactly the same content
  - boring!
- **Dynamic web page**: the page content may change on each request
  - the user interacts with the web site and the web site responds accordingly
- **Common Gateway Interface (CGI)**
  - A simple protocol that can be used to communicate between Web forms and your program
  - A CGI script can be written in any language
    - Need to be able to read input, write to output, and read environment variables
    - PHP, Java, C, C#, Perl, etc
Web Programming

- We need both client-side and server-side programming
  - to improve client-server interaction
  - to reduce bandwidth, server load, response time, etc

- Client-side programming is needed
  - to put dynamic content into an HTML page
  - to react to user events interactively without bothering the server
  - to mimic a GUI using graphics and animation
  - to validate data/queries before they are submitted to the server

- Server-side programming is needed
  - to limit the client interface to a server
    - for security
    - and performance
  - to perform heavy-duty processing, not available at every client
    - database processing
    - file directory service
  - as a broker to web services
Current Situation

- For client-side programming, the choice is clear: **JavaScript**
  - Java applets were a promising idea but failed
- For server-side, there are many choices:
  - For rapid prototyping, most people use **PHP scripts** (some use ASP)
  - For high-performance and portability, people now use **servlets**
- A script is easy to develop and maintain but has a high overhead for the server
  - for each client-server request, a script must be interpreted in a new thread
- **Servlets** are compiled Java programs
  - The Java engine runs continuously and spawns a light-weight thread for each servlet call
- Most web servers use the **apache** web server
  - Microsoft IIS is the second most popular choice, but is not portable
- **Tomcat** is the best choice for a servlet container
HTML Forms

- Forms are the most popular way to make web pages interactive
  - A form on a web page allows the user to enter requested information and submit it for processing

Example:

```html
<form name="input" action="/cgi-bin/login.php" method="GET">
    Username: <input type="text" name="user"><br/>
    Password: <input type="password" name="password"><br/>
    <input type="submit" value="Submit">
</form>
```

- The user types username “Smith” and password “mypass”
- When the user presses Submit, the browser sends the form data to the web server. For GET, it's the same as clicking on the link:

  http://myserver.com/cgi-bin/login.php?username=Smith&password=mypass
When the web server gets this request, it launches the CGI program which was written to process this form.

The CGI program generates a web page in HTML so that the user can see the results of submitting the form.

The CGI program passes the HTML back to the web server.

The web server passes the HTML back to the browser.

What's the difference between the GET and POST methods?

- A web browser downloads most files using GET.
- GET is also used for most form submissions, when the form data are small (when they can fit in the URL).
- When an HTML form is submitted using POST, the form data are attached to the message body, at the end of the POST request.
XHTML

- Need to handle HTML content as data
  - so that we can retrieve arbitrary parts of the HTML document using IDs
  - so that we can selectively change parts of the content of HTML pages
    - required for asynchronous server requests (see Ajax later)
- The obvious choice is to treat HTML as XML
  - Plentiful standards for querying/modifying XML data
- XHTML is HTML in XML form, standardized by W3C
  - It will eventually replace HTML
- What are the differences between HTML and XHTML?
  - XHTML is a stricter and cleaner version of HTML
  - All tag/attribute names must be lowercase
  - All elements must be well-formed. Examples:
    - instead of `<p>`, you write `<p>`...</p>
    - instead of `<br>`, you write `<br/>
- It's very well integrated with Cascading Style Sheets (CSSs)
Enforcing XHTML

- How to enforce it?
  
  ```html
  <html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Title goes here</title>
  </head>
  <body>
    ... the body goes here
  </body>
  </html>
  ```

- A stricter form (enforces checking against the XHTML DTD):
  
  ```html
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  <head>
    ...
  </head>
  ```
Cascading Style Sheets (CSS)

- Need to separate
  - **Functionality**: done with a combination of server- and client-side programming
    - developed by web application programmers
  - **Content** (layout): specified with HTML/XHTML tags
    - designed by web page designers
  - **Style** (color, fonts, etc): specified with CSS

- **CSS**: *Cascading Style Sheets*
  - allows developers to control the style of multiple elements and Web pages

- The CSS syntax is made up of three parts:
  - selector { property: value }

- Example:
  - `p.center { text-align: center; color: black; font-family: arial }

- Used as:
  - `<p class="center">...</p>`
Cookies

- Small bits of text stored at the client side (on the browser)
- When a user connects to a server for the first time, the server may create a cookie, which will be stored on the client's browser
  - The cookie is associated with the domain of this particular server
  - Cookies can be read only from the domain that created them
- Cookies are part of the HTTP header, so setting a cookie must be put in HTTP before any output is sent to the browser
- After the first visit to a web site, the browser returns a copy of the cookie to the server each time it connects
- Cookies have an expiration date after which they are deleted
  - After they expired, the next connection to the server is like the first time
- Example:
  Name: FPS, Domain: yahoo.com, Expire: 07/02/2008 01:00:00 PM
Session Tracking

- Maintains information about a visitor as she navigates through a server site
  - preserves certain data across subsequent accesses
  - maintains the illusion of a session that spans multiple pages
- Keeps track of visitors by issuing them cookies with randomly generated session IDs
  - The server uses the session ID cookie to remember the visitor
- Session data are serialized and stored at the server after a visitor access finishes
  - They are recreated and loaded on the subsequent access
- If the user browser doesn't accept cookies, it automatically adds the session ID to URLs and to the forms submitted by the user:
  
  For example:
  
  `<a href="http://server.com/index.html" >...<a/>
  
  generates the URL
  
  http://server.com/index.html?ID=64cc3ab764da69864ca3874ab3276
JavaScript

- For client-side programming the choice is clear: JavaScript
  - It is a scripting language (interpreted)
  - It is usually embedded directly into HTML pages
- Looks a little bit like C
  - but it's not that related to Java (although it has OO features now)
  - unlike C, it's not strongly typed
    
    ```javascript
    var x = 1;
    ```
- More information at:
  
  http://www.w3schools.com/js/js_intro.asp
Embedding JavaScript

- To embed JavaScript code into an HTML page:
  <script type="text/javascript">
    document.write("Hello World!");
  </script>

- Using external JavaScript code:
  <script src="myScript.js"></script>

- Best place to embed code is before the HTML body

- The JavaScript code can change the content of the embedding HTML page interactively
  - It sees the embedding HTML document as a node tree (HTML DOM)
  - When JavaScript updates the node tree, the browser automatically redraws only the parts of the web page that correspond to the updated nodes
  - This makes the browser look like a regular GUI
The HTML Document Object Model (DOM) defines a standard way for accessing and manipulating HTML documents:

- It presents an HTML document as a tree-structure (a node tree), with elements, attributes, and text.
- Similar to XML DOM.

**document**: The root of the embedding HTML document.

Methods to locate an element anywhere in the HTML document:

- `getElementById('id')`
- `getElementsByTagName('tag')`

To navigate starting from a node:

- `parentNode()`
- `firstChild()`
- `nextChild()`
- `lastChild()`
- `childNodes()`
Interactive Actions at the Client Side

- This form does not perform any action at the server side:
  
  ```html
  <form>
    Your name: <input id="input">
    <input type=button onclick="copy();" value="proceed">
    Greeting: <input id="display">
  </form>
  <div id="output">&nbsp;</div>
  ```

- The code of `copy()`:
  ```javascript
  function copy () {
    var text = document.getElementById("input").value;
    document.getElementById("display").value = "Hello "+text+"!";
    output = document.getElementById("output");
    output.replaceChild(document.createTextNode("Hello "+text+"!")),
    output.firstChild);
  }
  </script>
Asynchronous Server Requests

When the user pushes the Submit button on a form:

```html
<form action="script.php">
```

the current page is erased and replaced with a new page, which is the output of the `script.php`

- very expensive for the server
  - it has to resend parts of the erased web page again
- very annoying for the client
  - she has to wait for the new web page, looking aimlessly on an empty page

- Cheating: use hidden frames
  - Client gets a vector of frames from the server
  - Only one frame is displayed each time
  - Gives the impression of interaction
  - Assumes that you don't need server data while navigating through frames

- General solution: Dynamic HTML (DHTML)
  - Using asynchronous server requests
  - Introduced by Microsoft (AJAX: Asynchronous JavaScript and XML)
AJAX

- Uses asynchronous data transfer between the browser and the web server, allowing web pages to request data from the server instead of whole pages
  - When the client sends a GET/POST request, it doesn't wait for a response
    - it sets a handler to be evoked when it receives the response from server
    - the server response is typically XML or XHTML data
    - when the client handles the response, it uses this data to update the web page using the XML or the HTML DOM
  - This is accomplished with the XMLHttpRequest object
- You can get more information at:
  - [http://www.w3schools.com/ajax/default.asp](http://www.w3schools.com/ajax/default.asp)
Example

- On click, we want to evoke *serveraction.php* at the server side asynchronously:
  
  ```html
  <form>
    Comment: <input name="comment">
    <input type=button onclick="sendRequest();" value="process XML">
  </form>
  
  <div id="output"></div>
  
  Notice that there is no Submit button

  We want also the result to appear in the `<div id="output">` section instead of redrawing the entire page
The XMLHttpRequest Object

- Need to create an XMLHttpRequest and bind its onreadystatechange method:

```javascript
<script type="text/javascript">
    var request = new XMLHttpRequest();
    function displayResult () {
        if (request.readyState == 4) {
            var xmlDoc = request.responseXML.documentElement;
            var text = xmlDoc.getElementsByTagName("deptname")[0].childNodes[0].nodeValue;
            document.getElementById("output").innerHTML = text;
        }
    }
    function sendRequest () {
        request.onreadystatechange = displayResult;
        request.open("GET","serveraction.php?file=cs.xml",true);
        request.send(null);
    }
</script>
```
Server-side Programming

- This is the heavy-duty web programming
  - Typically uses database connectivity to
    - verify the client
    - store the client's state and current session data
    - store site data (products, transactions, etc)
  - Provides services to clients through complex application programs

- Any programming language can be used
  - C and Perl scripts in CGI used to be the most popular choices
  - PHP scripts are the most popular choice for fast development
    - ASP scripts from .NET are also popular but they are particular to IIS
  - For high-performance, web developers now use Java Servlets
    - or the more convenient JavaServer Pages (JSP), which are translated to servlets
PHP

- Stands for "PHP: Hypertext Preprocessor"
- A widely-used Open Source general-purpose scripting language
- Especially suited for Web development
  - Mostly used at the server side
- Can be embedded into HTML
- Facilitates rapid prototyping
  - Easy to learn by a novice
  - Powerful enough for a professional web application developer
- Not a good choice for high-load web servers
  - Each PHP script must be interpreted and evaluated in a new thread
- More information and manuals at
  http://www.php.net/
  http://www.w3schools.com/php/default.asp
Interacting with a Client

- The HTML code at the client side:
  ```html
  <form action="action.php" method="GET">
    <p>Your name: <input type="text" name="name" /></p>
    <p>Your age: <input type="text" name="age" /></p>
    <p><input type="submit" /></p>
  </form>
  ```

- The PHP file `action.php` at the server:
  ```php
  <html><body>
  Hi <?php echo $_GET['name']; ?>.
  You are <?php echo $_GET['age']; ?> years old.
  </body></html>
  ```
HTTP Authentication with PHP

- Use the header() function to send an "Authentication Required" message to the client browser, causing it to pop up a username/password input window.
- Once the user has filled in a username and a password, the URL containing the PHP script will be called again with the predefined variables PHP_AUTH_USER and PHP_AUTH_PW set to the user name and password.

```php
<?php
if ( !isset($_SERVER['PHP_AUTH_USER'])
 || !isset($_SERVER['PHP_AUTH_PW']) ) {
    header('WWW-Authenticate: Basic realm="Member Area"');
    header("HTTP/1.0 401 Unauthorized");
    echo "You must enter in a username and password combination!";
    exit;
}
// validate user $_SERVER['PHP_AUTH_USER']
?>
```
Authentication using MySQL

- Need to connect to MySQL database first. File connect.php:
  
  ```php
  <?php
  include('db_login.php');
  $connection = mysql_connect($db_host,$db_username,$db_password);
  if (!$connection)
    die("Could not connect to the database: <br/>".mysql_error());
  $db_select = mysql_select_db($db_database);
  if (!$db_select)
    die("Could not select to the database: <br/>".mysql_error());
  ?>
  
  - The file db_login.php defines the $db_... variables
    
    ```php
    $db_host='localhost';
    $db_database='users';
    $db_username='someMySQLuser';
    $db_password='xxxx';
    ?>
  ```
The User Database

- The table **members** in the database users:

  ```
  create table members ( 
      sid      char(10) primary key not null, 
      name     varchar(30) not null, 
      password varchar(32) not null, 
      email    varchar(40) 
  );
  ```

- The user validation code:

  ```
  $result = mysql_query("SELECT * FROM members 
    WHERE name="."$_SERVER['PHP_AUTH_USER']."" 
    AND password="."MD5($_SERVER['PHP_AUTH_PW'])."""");
  if (!$result || !mysql_fetch_row($result)) {
    header('WWW-Authenticate: Basic realm="Member Area"');
    header("HTTP/1.0 401 Unauthorized");
    echo "Your username and password combination was incorrect!";
    exit;
  }
  ```
Application: Change Password

- HTML at the client:
  
  ```html
  <form action="update.php" method="POST">
    New password: <input type="password" name="password" size=20/><br/>
    <input type="submit" value="Change" />
  </form>
  ```

- PHP at the server, file update.php:
  
  ```php
  <?php
  require_once('login.php');
  $name = $_SERVER['PHP_AUTH_USER'];
  if (isset($_POST['password']) && strlen($_POST['password']) > 4) {
    mysql_query("UPDATE members
      SET password='' . MD5($_POST['password']) . "'"
      WHERE name='' . $name . "''")
    or die('Cannot change the password');
    print 'Success!';
  }
  ?>
  ```
Application: Submitting a File

- HTML at the client:
  ```html
  <form enctype="multipart/form-data" action="handin.php" method="POST">
    <input type="hidden" name="MAX_FILE_SIZE" value="3000000" />
    Submit this file: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
  </form>
  ```

- PHP at the server, file handin.php:
  ```php
  <?php
  $file = $_FILES['userfile']['name'];
  move_uploaded_file($_FILES['userfile']['tmp_name'], "somedir/" . $file);
  ?>
  ```
Cookies and Sessions

- You can set cookies using the `setcookie()` function
  
  ```php
  setcookie("MyCookie", "some info", time()+3600); /* expires in 1 hour */
  ```
- Can be accessed on the next page load using `$_COOKIE`
  
  ```php
  echo $_COOKIE["MyCookie"]; 
  ```
- To initialize a session, use `session_start()`
- To access/change a session variable, use `$_SESSION`
  
  ```php
  <?php
  session_start();
  if (!isset($_SESSION['count']))
    $_SESSION['count'] = 0;
  $_SESSION['count']++;
  print 'you have visited ' . $_SESSION['count'] . ' times';
  ?>
  ```