Web application development (part 2)

Netzprogrammierung (Algorithmen und Programmierung V)
Our topics today

Server Sided Approaches
  - PHP and JSPs

Client Sided Approaches
  • JavaScript …
    • Some basics
    • The Document Object Model (DOM)
    • JavaScript Frameworks (e.g., jQuery, Processing.js)
  • Asynchronous Java and XML (AJAX) …
    • Synchronous vs. asynchronous interaction pattern
    • Web application request flow vs. AJAX request flow
    • Core AJAX concepts (esp. XMLHttpRequest)
    • Discussion of a simple AJAX example
  • Applets
  • Web Sockets
Model-View-Controller (Design Pattern)
Server-Side Scripting

Additional **HTML generated sources** embedded in HTML pages

Typical representatives:
- **JSP** - Java Server Pages
- **PHP** - Personal Home Page
- **ASP** - Active Server Pages (Microsoft IIS)
- …
**Two Variants**

Develop Application

- `<HTML>`
- `<xsp:...>`
- `</HTML>`

Runtime

Interpreter

HTML-Document

Develop Application

- `<HTML>`
- `<xsp:...>`
- `</HTML>`

Compile Time (only once)

Compiler

Executable Program

Runtime

HTML-Document
Web application development

Java Server Pages (JSP)
Variant 2, JSP Sequence

HTTP-Server

JSP-page

Servlet generation from JSP (HTML + Scripting)

Client

Servlet

State

Start connection

SQL queries

Results

Database

Request

Response
Java Server Pages (JSPs)

- JSPs have a different syntax than Servlets:
  - Servlets produce HTML code within the Java code
  - JSPs embed Java code in HTML code

- JSPs are translated internally into Servlets by the Web container

- Advantage of JSPs:
  - “Separation of responsibilities” – programming of application logic can be done in Java by programmer; Design of pages can be done by Web designer
  - Component based

- Structure is normal HTML with additional basic tags:
  - Scriptlets, Expressions, Declarations, Directives
Scriptlets

- **Scriptlet Tag**
  - `<% code %>`
  - Embed Java code in JSP document
  - The code will be executed when the JSP is called

```html
<html>
<body>
  <% for (int i = 0; i < 2; i++) { %>
    <p>Hello World!</p>
  <% } %>
</body>
</html>
```
Expressions

Expression Tag

- `<%= expression %>`

- **Embed a Java expression** which will be executed every time when the JSP is loaded
  - Note: no “;” after the Java expression!
  - Otherwise syntax failure

```
<html>
  <body>
    <p>` <%= Integer.toString( 5 * 5 ) %>`</p>
  </body>
</html>
```

```
<html>
  <body>
    <p>25</p>
  </body>
</html>
```
Declaration

Declaration Tag

- `<%! declaration %>`

- Embed Java declaration in JSP
  - e.g. Method declaration:

```html
<html>
<body>
  `<%! public boolean isPositive(int x) {
    if (x < 0) {
      return false;
    }
    else {
      return true;
    }
  } %>`
</body>
</html>
```
URL-encoded Parameter

JSP provides an **object request** which stores attributes and transfers them to the request on the JSP page.

Parameter can set the URL-encoding:

**Example:**

http://localhost/example.jsp?param1=hello&param2=world

```jsp
example.jsp:
<html>
 <body>
  <p><%= request.getParameter("param1") %></p>
  <p><%= request.getParameter("param2") %></p>
 </body>
</html>
```

```html
<html>
 <body>
  <p>hello</p>
  <p>world</p>
 </body>
</html>
```
JSP Action Tags

- `<jsp:property />`
- `property` defined in Tag Library Definition files
- JSP Tag reference:

```html
<html>
<body>
  <jsp:useBean id="user" class="mypackage.User"/>
  <jsp:setProperty name="user" property="userName" value="APaschke" />
  <p>User Name: &nbsp;<jsp:getProperty name="user" property="userName" /></p>
</body>
</html>
```
JSP and Sessions

**Problem:** HTTP is stateless

**Solution** for JSPs: support for Sessions.
- Sessions store attributes.
- Session attribute are persistent by using Cookies or request parameter are transferred

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getId()</td>
<td>Returns the session ID.</td>
</tr>
<tr>
<td>getCreationTime()</td>
<td>Returns the time at which the session was created.</td>
</tr>
<tr>
<td>isNew()</td>
<td>Returns true if user’s browser has not yet confirmed the session id.</td>
</tr>
<tr>
<td>invalidate()</td>
<td>Discards the session, releasing any objects stored as attributes.</td>
</tr>
<tr>
<td>getAttribute(String key)</td>
<td>Returns an attribute that is stored in the session.</td>
</tr>
<tr>
<td>setAttribute(String key, String value)</td>
<td>Stores an attribute in the sessions.</td>
</tr>
</tbody>
</table>
Web application development

**Hypertext Preprocessor (PHP)**
Example: Printing a dynamic date

Using CGI
• HTML is embedded within the script-code
• Problem is that it becomes unreadable if there is more HTML than code

```html
print("<HTML>");
print("<H1>Sample</H1>");
var now = new Date();
print("It is now <strong>" + now + "</strong>" ); print("</HTML>");
```

Using PHP
• Script-code is **embedded** in HTML instead of other way around!
• *Inside-out programming*

```html
<HTML>
<H1>Sample</H1>
It is now <strong> <?php echo date ?> </strong>
</HTML>
```
PHP basics – Interpreted Approach

In 1995 developed by Rasmus Lerdorf (member of the Apache Group)

• Originally designed as a tool for tracking visitors at Lerdorf's Web site
• Within 2 years, widely used in conjunction with the Apache server
• Developed into full-featured, scripting language for server-side programming
• Free, open-source,

New to PHP? Then go to this [http://de.php.net/](http://de.php.net/).
Three basic application areas

**Server-side scripting**
- Most traditional and main target field
- Ingredients: PHP parser (e.g., CGI), a web server and a web browser

**Command line scripting**
- Use a PHP script to run it without any server or browser
- Ingredients: PHP parser

**Writing desktop applications**
- Well, is not the very best language to create a desktop application with a graphical user interface
- Ingredients: you know PHP very well, and if you need some advanced PHP features in your client-side applications use PHP-GTK
Basic PHP syntax

A PHP scripting block always starts with `<?php` and ends with `?>`. A PHP scripting block can be placed (almost) anywhere in an HTML document.

```html
<html>
<!-- hello.php -->
<head><title>Hello World</title></head>
<body>
<p>This is going to be ignored by the PHP interpreter.</p>
  <p>While this is going to be parsed.</p>
<p>This will also be ignored by the PHP preprocessor.</p>
  <p>Hello and welcome to <i>my</i> page!</p>
</body>
</html>
```

- `print` and `echo` for output
- a semicolon (`;`) at the end of each statement
Variables and arrays

All variables in PHP start with a $ sign symbol. A variable's type is determined by the context in which that variable is used (i.e. there is no strong-typing in PHP).

```php
$dutch_dessert = Vanillevla ; // string
$x = 28; // integer
$pi = 3.1415; // double
$whatever = TRUE; // boolean
```

Variables are case-sensitive, beware! $alp5 != $ALP5

An array in PHP is actually an ordered map. A map is a type that maps values to keys.

```php
<?php
$arr = array("foo" => "bar", 12 => true);
 echo $arr["foo"]; // bar
 echo $arr[12]; // 1
?>
```

array() = creates arrays

key = either an integer or a string.
value = any PHP type.
Functions in PHP

The PHP function reference is available on http://php.net/quickref.php. It shows the impressive list of functions within PHP.

For example:

**date()** is a built-in PHP function that can be called with many different parameters to return the date (and/or local time) in various formats.

But you also can create your own functions. If you recreate one of the built-in functions, your own function will have no effect.
File Open

The `fopen("file_name","mode")` function is used to open files in PHP.

<table>
<thead>
<tr>
<th></th>
<th>Read only.</th>
<th></th>
<th>Read/Write.</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>Read only.</td>
<td>r+</td>
<td>Read/Write.</td>
</tr>
<tr>
<td>w</td>
<td>Write only.</td>
<td>w+</td>
<td>Read/Write.</td>
</tr>
<tr>
<td>a</td>
<td>Append.</td>
<td>a+</td>
<td>Read/Append.</td>
</tr>
<tr>
<td>x</td>
<td>Create and open for write only.</td>
<td>x+</td>
<td>Create and open for read/write.</td>
</tr>
</tbody>
</table>

For `w`, and `a`, if no file exists, it tries to create it (use with caution, i.e. check that this is the case, otherwise you will overwrite an existing file).

```php
<?php
$fh=fopen("welcome.txt","r");
?>
```

For `x` if a file exists, this function fails (and returns 0).

```php
<?php
if ( (!$fh=fopen("welcome.txt","r")) ) exit("Unable to open file!");
?>
```

If the `fopen()` function is unable to open the specified file, it returns 0 (false).
Form handling

Any form element is automatically available via one of the built-in PHP variables (provided that element has a “name” defined with it).

```html
<html>
  <!-- form.html -->
  <body>
    <form action="welcome.php" method="POST">
      Enter your name: <input type="text" name="name" /> <br/>
      Enter your age: <input type="text" name="age" /> <br/>
      <input type="submit" />
      <input type="reset" />
    </form>
  </body>
</html>
```

```html
<html>
  <!–- welcome.php -->
  <body>
    Welcome <?php echo $_POST["name"]."."; ?><br/>
    You are <?php echo $_POST["age"]; ?> years old!
  </body>
</html>
```

$_POST contains all POST data.

$_GET contains all GET data.
Web application development (part 2)

Client-Sided Approaches
Client Sided WWW Applications

Many tasks can be done by the clients (better load balancing)
- Check user input
- Create graphics
- Manipulation of WWW-Documents
- Additional possibilities for the user interface

Client sided approaches
- Browser Plug-Ins (e.g. Macromedia Flash, …)
- Script languages (JavaScript, …)
- Mobile Code (Java Applets)
- Microsoft ActiveX

Security problem!!!
Web application development

JavaScript
JavaScript (≠ Java)

Scripting language, e.g., for
- Authentification
- Automated (partial) reloads of sites

Embedded in HTML
- `<script type="text/javascript"> … </script>`
- or external file
  - `<script type="text/javascript" src="a.js"></script>`
What can JavaScript do?

**JavaScript gives HTML designers a programming tool** - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages.

**JavaScript can react to events** - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.

**JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element.

**JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing.

**JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and depending on the browser, load another page specifically designed for that browser.

**JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer.
JavaScript object hierarchy

Browser Object Model (BOM)

- Navigator
  - Mime-Type
  - Window
  - Plugin
  - Frame
  - History
  - Location

Document

- Anchor
- Applet
- Area
- Form
- Image
- Link
- Layer

Language

- Array
- Boolean
- Date
- Math
- Number
- String

http://www.comptedoc.org/independent/web/cgi/javamanual/javaobjheir.html
JavaScript basics

Declaration
• Explicit: `var i = 12;`
• Implicit: `i = 12;`

Variable scope
• Global: Declared outside functions and any variable that is implicitly defined
• Local: *Explicit* declarations inside functions

Control and looping
• ‘if’, ‘switch’ statement
• `while`, and `do-while` loops (*break* and *continue* keywords)

Not object-oriented but object-based

```javascript
function Student(studentName, studentAge) {
    this.name = studentName;
    this.age = studentAge;
}

var someStudent = new Student("Michael", 21);
```

Tutorial at
[http://www.w3schools.com/js/](http://www.w3schools.com/js/)
JavaScript output

The document objects allow printing directly into the browser page (amongst other things) 
The window object is implied.

Example: Writing in text or HTML with script or without line-break

document.write("I am <B>BOLD</B>");
document.writeln("I am <U>underlined</U>");

What is the window object?

• It is part of the Browser Object Model (BOM)
• BOM is a a collection of objects that represent the browser and the computer screen
• Objects are accessible through the global objects window and window.screen
• The window object is global to the extent that it represents the very host environment of all other JavaScript objects
Browser Window Object

window object is a JavaScript representation of a browser window

- **closed** - A boolean value that indicates whether the window is closed
- **defaultStatus** - This is the default message that is loaded into the status bar when the window loads
- **Example:** `window.defaultStatus = "This is the status bar";

Selected *build in functions*

- **alert("message")** - string passed to the alert function is displayed in an alert dialog box.
- **window.close()** - function will close the current window or the named window.
- **confirm("message")** - string passed to the confirm function is displayed in the confirm dialog box.
- **focus()** - function will give the focus to the window.
- **open("URLname","Windowname",["options"])** - new window is opened with the name specified by the second parameter.
Form object

The Form object is a property of the document object. This corresponds to an HTML input form constructed with the FORM tag. A form can be submitted by calling the JavaScript submit method or clicking the form submit button.

Form objects can be accessed by
window.document.myForm OR window.document.forms[0]

Selected properties

action - This specifies the URL and CGI script file name the form is to be submitted to. It allows reading or changing the ACTION attribute of the HTML FORM tag.

target - The name of the frame or window the form submission response is sent to by the server. Corresponds to the FORM TARGET attribute.

length - The number of fields in the elements array, i.e. the length of the elements array.

method - This is a read or write string. It has the value "GET" or "POST".
Form object – objects and methods

Form Objects (one example)
• text - A GUI text field object. Methods are `blur()`, `focus()`, and `select()`
• Attributes
  • `defaultValue` - The text default value of the text field
  • `name` - The name of the text field
  • `type` - Type is "text"
  • `value` - The text that is entered and appears in the text field. It is sent to the server when the form is submitted
• Example “accessing form field values”: `window.document.myForm.firstname.value`

Form object methods
• `reset()` - Used to reset the form elements to their default values
• Example `window.document.myForm.reset();`
• `submit()` - Submits the form as though the submit button were pressed by the user
• Example `window.document.myForm.submit();`
Event handlers

Events are actions that occur usually as a result of something the user does. For example, clicking a button is an event, as is changing a text field or moving the mouse over a hyperlink.

Other events are `click`, `change`, `focus`, `load`, `mouseover`, `mouseout`, `reset`, `submit`, `select` …

You can use event handlers, such as `onChange` and `onClick`, to make your script react to events.

Examples

```html
<input type="button" onClick="javascript:doButton()">
<select onChange="javascript:doChange()"> 
<a onClick="javascript:doSomething()"> </a>
<form onSubmit="javascript:validate()"> 
```
JavaScript principle

Contents

HTML

Behavior

JavaScript
(event handling)

myJS.js

Presentation

CSS

myCSS.css

<head>
<script src="myJS.js">
<link rel="stylesheet"
type="text/css"
href="myCSS.css">
...
From HTML to DOM

HTML is a representation for hypermedia documents
- A representation is required to store and transmit the document
- HTML uses markup for representing the document structure

Browsers must render HTML documents (i.e., apply CSS and execute JavaScript)
1. GET HTML from server and receive as text/html document
2. Parse document and deal with any errors by “fixing them”
3. Interpret document as if it had been error-free
4. GET all additional resources (CSS, images, JavaScript, …)
5. **Build internal model (DOM) based on error-free interpretation**
6. Apply CSS rules to determine styling of document (e.g., margins and font sizes)
7. Render into visual structure
8. **Start executing JavaScript code**
9. **Listen for events (keyboard, mouse, timer) and execute code**
DOM representation of an HTML page

Each HTML document loaded into a browser window becomes a **Document** object

- Provides access to all HTML elements in a page, from within a script.
- Is also part of the **Window** object, and can be accessed through the `window.document` property.

DOM describes how all elements in an HTML page are related to the topmost structure: the document itself.

Example

```
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <p>Lorem ipsum ...</p>
    <p>Lorem ipsum ...</p>
    <p>link <a href="http://html.net">HTML.net</a></p>
  </body>
</html>
```
Accessing elements

By name

document.getElementsByTagName("td")[indexOfColumn]

By ID

document.getElementById("mytable")

Walk the DOM Tree

JavaScript Frameworks

Different needs produce different frameworks

- Dojo (http://dojotoolkit.org/)
- Ext JS (http://www.sencha.com/products/extjs/)
- jQuery (http://jquery.com/)
- Processing.js (http://processingjs.org/)

More comprehensive list is on Wikipedia:

There is no such thing as the “best JavaScript framework”

- For any given project, decide on the support you need
- Evaluate frameworks for the support they provide
- Evaluate for functional requirements (“is there a collapse/expand folder view?”)
- Evaluate for non-functional requirements (“is the framework openly licensed”)
Web application development

Asynchronous Javascript and XML (AJAX)
Conventional Web Applications

Client Browser (State A) -> HTTP REQUEST -> HTTP Server

HTTP RESPONSE
New Page
(state A + ... = state B)

Client Browser (State B)
AJAX basics

Asynchronous Javascript and XML (AJAX) is a technology which allows client-side JavaScript to make requests to a server without causing a reload of the current page in the browser.

Using AJAX and standard DOM methods, client-side JavaScript can request, receive, and visualize information in the context of a single Web page.

The advantage of AJAX over more traditional Web pages is that they better resemble the behavior of desktop applications, providing enhanced features and usability for the user.

The basic idea is, when you work with a web page, the web page itself can be interacting with the server, but without reloading the (whole) page you are on. The result is an uninterrupted user experience, with full server-side capabilities.

AJAX is not a technology per se, but a web development method and its aim is to increase the interactivity of a website.
AJAX - Asynchronous JavaScript And XML

AJAX combines JavaScript, HTML, DHTML, DOM und XML, in order to facilitate programming of interactive websites.

AJAX communicates via JavaScript code and XMLHttpRequest object asynchronously with the server.

After sending request the client can proceed execution.

If the response from the server arrives, the XMLHttpRequest object executes a client-site call back method which e.g. updates the website.

Supported by most modern Web browsers.

Examples which use Ajax: Google Maps, Gmail, Google Docs, Flickr, Facebook, etc. (see e.g. RSS Tec. (Really Simple Syndication, Weblogs, Ticker, …)
Core AJAX concepts

JavaScript includes an XMLHttpRequest object that can fetch files from a web server.

**XMLHttpRequest** can load HTTP requests

- **Synchronously**: The browser will wait for the response. This is a simpler model for the developer, but the resulting blocking is not acceptable.
- **Asynchronously**: The browser will not wait, but the object will call an event handler when the loading status changes. This is what you would use in practice

The contents of the fetched file can be put into current web page using the DOM.
Usage of XMLHttpRequest

Client Browser (State A) → REQUEST → HTTP Server → RESPONSE – New Page

Client Browser (State B) → REQUEST/RESPONSE

Client Browser (State C) → REQUEST/RESPONSE

Client Browser (State D) → REQUEST/RESPONSE
A typical Ajax request

1. User clicks, invoking event handler

2. Handler's JS code creates an XMLHttpRequest object

3. XMLHttpRequest object requests a document from a web server

4. Server retrieves appropriate data, sends it back

5. XMLHttpRequest fires event to say that the data has arrived (this is often called a callback; you can attach a handler to be notified when the data has arrived)

6. Your callback event handler processes the data and displays it
### XMLHttpRequest Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abort()</td>
<td>Stops the current request</td>
</tr>
<tr>
<td>getAllResponseHeaders()</td>
<td>Returns all headers (name and value) as a string</td>
</tr>
<tr>
<td>getResponseHeader(&lt;headerName&gt;)</td>
<td>Returns the value of the specified header</td>
</tr>
<tr>
<td>open(“GET”/“POST”, “URL” [, true/false [, &lt;username&gt;,&lt;password&gt;]])</td>
<td>Opens a connection and retrieves response from the specified URL. Can also specify optional values method (GET/POST), username and password for secured sites</td>
</tr>
<tr>
<td>send(content)</td>
<td>Transmits request (can include postable string or DOM object data)</td>
</tr>
<tr>
<td>setRequestHeader(&lt;name&gt;, &lt;value&gt;)</td>
<td>Assigns values to the specified header</td>
</tr>
</tbody>
</table>
Simple AJAX example

<script type="text/javascript" language="javascript">
    function sndReq() {
        var resObject = new XMLHttpRequest();

        resObject.open('GET', 'laender.php?wo=1', true);
        resObject.onreadystatechange = handleResponse;
        resObject.send(null); }

    function handleResponse() {
        if(resObject.readyState == 4 {
            document.getElementById("hs").innerHTML =
            resObject.responseText; } }
</script>

<span style="cursor: pointer; text-decoration: underline" onclick="sndReq()">Einen Request absetzen.</span>

<span id="hs"></span>
Simple AJAX example (cont.)

```javascript
function sndReq() {
    var resObject = new XMLHttpRequest();

    resObject.open('GET', 'laender.php?wo=1', true);
    resObject.onreadystatechange = handleResponse;
    resObject.send(null);
}

function handleResponse() {
    if(resObject.readyState == 4 {
        document.getElementById('hs').innerHTML=
        resObject.responseText;
    } }
</script>

<span style="cursor: pointer; text-decoration: underline" onclick="sndReq()">
    Einen Request absetzen.
</span>

<span id="hs">
</span>
Simple AJAX example (cont.)

```html
<script type="text/javascript" language="javascript">
    function sndReq() {
        var resObject = new XMLHttpRequest();
        resObject.open('GET', 'laender.php?wo=1', true);
        resObject.onreadystatechange = handleResponse;
        resObject.send(null);
    }

    function handleResponse() {
        if(resObject.readyState == 4 {
            document.getElementById("hs").innerHTML = resObject.responseText;
        }
    }
</script>

<span style="cursor: pointer; text-decoration: underline" onclick="sndReq()">Einen Request absetzen.</span>
</span>
```

```html
<span id="hs"></span>
</span>
```
Simple AJAX example (cont.)

```javascript
function sndReq() {
    var resObject = new XMLHttpRequest();

    resObject.open('GET', 'laender.php?wo=1', true);
    resObject.onreadystatechange = handleResponse;
    resObject.send(null);
}

function handleResponse() {
    if(resObject.readyState == 4 {
        document.getElementById("hs").innerHTML = resObject.responseText;
    } }

</script>

<span style="cursor: pointer; text-decoration: underline" onclick="sndReq()">
Einen Request absetzen.
</span>

<span id="hs"></span>
```
Simple AJAX example (cont.)

```javascript
function sndReq() {
    var resObject = new XMLHttpRequest;
    resObject.open('GET', 'laender.php?wo=1', true);
    resObject.onreadystatechange = handleResponse;
    resObject.send(null);
}

function handleResponse() {
    if (resObject.readyState == 4) {
        if (resObject.status == 200) {
            do something with resObject.responseText;
        } else {
            code to handle the error;
        }
    }
}
```

```html
<span style="cursor: pointer; text-decoration: underline" onclick="sndReq()">Einen Request absetzen.</span>
</html>
AJAX examples

This is a very simple idea, but it can have very nice results, such as:

Google Maps (and A9 Maps)

- An image displaying a map of a specified region is inserted into a web page
- The user clicks the scroll left button
- The JavaScript module loads a new image and inserts it into the current document

Google Suggest (and Amazon Suggest):

- The user types in part of a keyword
- The JavaScript module sends the partial keyword to the server, and gets back a list of matches, which are shown to the user
Using AJAX - Pros and Cons

PROS
[+] Enhanced User Experience
[+] Reduced Bandwidth Usage
[+] Increased User Productivity
[+] Increased Compatibility
[+] Faster & Enhanced Application Development

CONS
[-] Learning Curve For Developers
[-] Effects On End Users
[-] Not Search Engine Friendly
[-] Dependency Of Browser Settings
Web application development

Java Applets (Mobile Code)
Java-Applet

```java
import java.applet.*;
import java.awt.*;

public class FirstApplet extends Applet {

    // The Methode paint implements the Applet
    public void paint(Graphics g) {
        g.drawString("Hello World!", 25, 50);
    }

} /* Class FirstApplet */
```

0efc345f6b7723ba83f2d90ca4 (…)
Example: Client Sided Application

Presentation/Interaction (e.g. Applet)

Server-Prozess

Application Logic

Database

Data
WWW and Java Applets

An appropriate Browser (with Java interpreter) can execute Applets
- Animations, Presentation of Graphics, Audio functionality
- Local Interaction (Mouse, Keyboard)
- TCP/IP-Connections to Server sided applications

Problem: Security
- No pointer
- Validation at compile time
- Verification of byte codes by the Bytecode verifier
- Applets have no access to local file system (except trusted applets!)
- Usual browser policy: Network connections from applets only to the Host from which the applet was loaded!
Web application development (part 2)

Summary
What have we discussed today?

We looked at server sided scripting approaches (JSP and PHP).

We discussed the client side scripting (run in the browser) based on JavaScript (another example is e.g. VB script).

Motivated by the JavaScript principle we talked about the Document Object Model; why it is useful and how can it be used?

We introduced the Asynchronous Java and XML (AJAX) which allows client-side JavaScript to make requests to a server without causing a reload of the current page in the browser. We discussed the interaction patterns and the request flows and compared it to a typical web application. You should know the steps of a typical Ajax request.

We briefly talked about client side application programs that are mobile programs that run as an independent program in the client machine. The typical example are Java Applets.

We also shortly introduced Websockets, an approach for full-duplex single socket connection.
Questions

• How does Server-sided Scripting work and explain it on the example of JSP (or another technology)? What are the benefits?
• What is the difference between interpreted and compiled (server-sided) scripting?
• Name two client-sided approaches for dynamic web application programming.
• What is JavaScript and how does it work? What are the differences to server-sided scripting?
• How does Ajax work and what are the benefits and drawbacks?
References


Additional resources:
http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications
http://www.comptechdoc.org/independent/web/cgi/javamanual
http://www.quirksmode.org/dom/intro.html
http://www.w3schools.com/js/default.asp