Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

6) Overview: web applications

Emmanuel Benoist

Spring Term 2017

Computer Science Division

Table of Contents

Presentation

What is a web application?

- **For the user: it is displayed on a Web browser**
  - Firefox (35%), Chrome (30%), Internet Explorer (19%), Opera (4%), Safari (4%), Android browser, Mozilla, ...[1]
- **Hosted on an HTTP server**
  - HTTP deamon from Apache
  - Java Servers : Apache tomcat, GlassFish, Jetty, JBoss, WebSphere ...
  - IIS (Microsoft Internet Information Server)
- **Uses HTTP**
- HyperText Transfer Protocol
- and a secure version https (using a TLS tuneling)

[1] Source: statistics of the web site `http://www.benoist.ch` January 2013

Static pages

Static pages

- **Written in HTML**
  - HyperText Markup Language
- **Markup Language:**
  - Uses tags `<h1>Title</h1>` `<br>` or `<img>`
- **Tags mark the semantic of the document**
  - Headings: `<h1></h1>` `<h2></h2>` ...
  - Paragraphs `<p>...</p>`
  - Blocks (without intrinsec semantic):`<div>...</div>`

Page HTML

- **A very simple page**

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html> <head> <title>Example 1</title> </head>
<body>
<h1>Example 1</h1>
Hello World
<ul>
  <li>This is a first item</li>
  <li>This is a second item</li>
  <li>This is a third item</li>
</ul>
<a href="index.php">Back</a>
</body> </html>
```

Programming server side

Programming Server Side

- **HTML is rarely static**
  - Static HTML is too complex to manage
  - Text is stored in a DB
  - HTML is created on the fly
- **Different technologies**
  - Script languages: PHP, python,
  - Java : Servlet, JSP, JSF, Struts
  - Microsoft .NET: C#, ASP, ...

PHP

PHP

- **PHP was developed for including programming inside an HTML page**
  - Read inputs given by users
  - Insert loops, conditional branching,
- **It is now much more than that**
  - An Object oriented programming language
  - Lots of libraries for dealing with images, pdf, http, flash, etc.
  - Template engines for programming properly (separation of HTML from programming)
  - Frameworks for programming more efficiently (PEAR, Symphony, Zend, ...).
- **Lots of programs available in PHP**
  - Content management systems (Typo3 for instance)

PHP example

- **A very small program**

```
<html>
<head><title>Example Hello World in PHP</title></head>
<body>
<?php
echo "Hello World <br>\n";
echo "It is:" .time()." (not so clear isn't it?)<br>\n";
echo "more clear:" .date("D, d M Y H:i:s",time());
?>
</body>
</html>
```

PHP presents data from a DataBase

- **Writing HTML by hand is not possible**
  - HTML is tricky for authors
  - Too complex for a normal person
  - It is now too complex to anybody
- **Access to a DB**
  - Select data, update or modify data

Content Management Systems

Content Management Systems

- **Managing a web site manually is not possible**
  - Redactors are not computer science specialists
  - Content must be changed regularly
  - Content is often very dynamic
- **Content Management Systems**
  - Redactors write input in forms,
  - Input is stored inside the DataBase
  - Content of the database is then displayed in HTML
  - `https://staff.ti.bfh.ch/typo3/`

Programming Server Side

- **Other programming languages**
- **Java**
  - Servlet, Java Server Pages, Java Server Faces, Struts, Spring
- **.NET**
  - ASP, C#,

Java Servlets

Java server side

- **Java Servlet**
  - Java program generates the HTML file

```
...
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse ↘
    →response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello␣World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello␣I3ABC␣and␣I4Q!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Servlet using forms I

```
public class HelloBoss extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse ↘
    →response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");

        String username = request.getParameter("username");
        if(username==null){
            printForm(out);
        }
```

Servlet using forms II

```
        else{
            out.println("Hello " +username+"<br>\n");
        }
        out.println("</body>");
        out.println("</html>");
    }
    void printForm(PrintWriter out){
        out.println("<form>");
        out.println("What is your name? ");
        out.println("<input type=\"text\" name=\"username\">");
        out.println("</form>");
        out.println("");
    }
}
```

Java Server Pages - JSP

Java Server Pages

- **JSP**
  - Originally: write java inside an HTML page
  - Java Servlet is generated from the JSP code
  - The Servlet is compiled and executed
  - Now: add new tags to HTML
  - Libraries of tags exists that can be used

JSP [2]

```
<%@ taglib prefix="mytag" uri="/WEB-INF/jsp2/jsp2-example-taglib.tld" %>
<html>
  <head>
    <title>JSP 2.0 Examples - Repeat SimpleTag Handler</title>
  </head>
  <body>
    <h1>JSP 2.0 Examples - Repeat SimpleTag Handler</h1>
    <hr>
   <p>...</p>
   <p>...</p>
   <br>
    <b><u>Result:</u></b><br>
    <mytag:repeat num="5">
      Invocation ${count} of 5<br>
    </mytag:repeat>
  </body>
</html>
```

[2] Standard Apache JSP example `http://localhost:8080/examples/jsp/`

Java Server Faces - JSF

Java Server Faces

- **Integrated Framework**
  - Separates the different parts of the server
  - Model View Controler (MVC) design pattern
- **Three components**
  - Controler : the JSF servlet was written, we only use and configure it
  - View : XHTML containing special tags
  - Model : Java Beans making the glue between web tier and java libraries
- **Components and Events**
  - Very similar to Swing
  - Components are defined that are reusable
  - We define a tree of components in the XHTML document
  - Programming is done using Events, the page (i.e. HTTP-Request) is not central anymore.

JSF Hello world I

- **The XHTML file**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//↘
→EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
    <f:view contentType="text/html" />
    <h:head>
        <title>Hello World!</title>
    </h:head>
    <h:body bgcolor="white">
        <h2>My name is Emmanuel. What is yours?</h2>
        <h2>Hi, #{helloBean.greeting}</h2>
        <h:form id="helloForm" >
            <h:graphicImage id=''logo"
␣␣␣␣␣␣␣␣␣␣␣␣url="#{resource['bfh.jpg']}" alt="BFH Logo" />
            <h:inputText id="username" value="#{helloBean.name}" />
```

JSF Hello world II

```
            <h:commandButton id="submit" action="#{helloBean.↘
            →response}" value="Submit" />
        </h:form>
    </h:body>
</html>
```

Programming client side

Programming Client Side

- **Applet**
  - Java program written to be executed inside the browser
  - Limited by a security "sandbox"
  - Not easy to integrate inside the layout of the page
  - Contains directly the Java buttons and UI's.
  - Not used any more (blocked in browsers)
- **Flash**
  - Used for small animations, or games
  - Programming language Object oriented
  - But also some odd concepts (scenario, timeline,...) comming from animation
- **JavaScript**
  - Originally: for testing inputs (the phone number must have this form)
  - Now: The page became an application
  - Ajax, makes the page dynamic,

Javascript

Javascript

- **Reacts to events**
  - Value changed, click, double click, mouse over, key up, key down, etc
- **Manipulate the page**
  - Document Object Model (DOM)
  - Tree representing the HTML
  - Can be manipuled by JavaScript
- **Exchange information with servers**
  - Ajax with the own server
  - JSON with other servers

JavaScript example

```
<html><head><title>Test</title></head>
<body>
<ol id="Liste"> <li>Element</li> </ol>
<script language="JavaScript" type="text/javascript">
<!--
firstchild = document.getElementById("Liste").firstChild;
document.getElementById("Liste").removeChild(firstchild);

for(var i = 0; i < 10; i++) {
  var nouveauLI = document.createElement("li");
  var numeroli = i + 1;
  var nouveautexteli = document.createTextNode("It is the item number " ↘
  → + numeroli);
  document.getElementById("Liste").appendChild(nouveauLI);
  document.getElementsByTagName("li")[i].appendChild(nouveautexteli);
}
//-->
</script> </body></html>
```

Ajax

Ajax (Asynchron JavaScript and XML

- **Principles**
  - The page becomes an application
  - Information is communicated with the server asynchron
  - Page is updated on the fly
- **Difficulty**
  - Program is both on the client and the server
- **Adventage**
  - Application is dynamical
  - Very similar to mobile apps
  - Client is not “thin” anymore

Call an AJAX request

```
function showCustomer(str) {
  xmlHttp=GetXmlHttpObject();
  if (xmlHttp==null) {
    alert ("Your browser does not support AJAX!");
    return;
  }
  var url="getcustomer.php";
  url=url+"?q="+str;
  url=url+"&sid="+Math.random();
  xmlHttp.onreadystatechange=stateChanged;
  xmlHttp.open("GET",url,true);
  xmlHttp.send(null);
}
```

Google Web Toolkit - GWT

Google Web Toolkit

- **What is GWT?**
  - A development environment in pure Java for rich web applications
  - Provides Java for programming both client and server sides
- **Advantages of GWT**
  - Homogenous environment
  - Testing of a web application (using JUnit)
- **Not integrated in JSF**
  - Concurent system developed by google

Principle

- **Write Java code**
  - Use Java on Server Side
  - But also on a Client Side
  - Communication is handeled conveniently
- **Tests in a JVM**
  - Testing is done using JUnit
  - Plugin in the browser
  - Tests are conducted inside one JVM (based on Java Code)
- **Compile into Javascript**
  - Creates different versions for different browsers
  - Each browser receives only the “right” version
  - Can be deployed on Java Servers
  - Or any other server (if the server part is not Java)

Hello World Application

- **Download the GWT**
  - From Google Code Web site
  - `http://code.google.com`
- **Create an application**
  - Execute
    `./webAppCreator -out /home/bie1/test/ ch.bfh.awt.Hello`
  - A default application is created
  - Includes `ant` and `Eclipse` project files
- **Test the Application**
  - Go to the directory
  - execute `ant devmode`
  - Install the plugin in your browser
  - Test the application

Directories created

- **Source files:** `/src/`
  - Package for client side application : `/src/ch/bfh/awt/client`
  - Server side classes : `/src/ch/bfh/awt/server`
  - The file `/src/ch/bfh/awt/Hello.gwt.xml` contains the configurations for the GWT application
- **Web Application:** `/war/`
  - Contains html, css, javascript, gifs, and the like
  - Contains the `WEB-INF/` directory (where the server classes are automatically compiled
  - The directory `/war/` will receive the JavaScript files compiled from the client application
  - At the end the content of this directory is copied to the server

Hello World

- **Hello.html: contains a real HTML**
  - Containing layout,
  - References to images, JavaScript, CSS
- **Reference to the script loading the files**

```
<script type="text/javascript" language="javascript" src=↘
→"hello/hello.nocache.js"></script>
```

- **And it contains place-holders that will be manipulated from "Java".**

```
<div id="nameFieldContainer"></div>
<div id="sendButtonContainer"></div>
<div style="color:blue;" id="responseContainer"↘
→></div>
```

HTML FIle

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="Hello.css">
    <title>Web Application Starter Project</title>
    <script type="text/javascript" language="javascript" src="hello/hello.nocache.js"></script>
  </head>
 <body>
...
    <h1>Web Application Starter Project</h1>

        Please enter your name:
         <div id="nameFieldContainer"></div>
        <div id="sendButtonContainer"></div>
       <div style="color:blue;" id="responseContainer"></div>
 </body>
</html>
```

Java File

- **Contains the definition of the user interface**
  - Definition of the Widgets used,
  - Panels,
  - Text fields,
  - buttons
  -
- **Extends the `EntryPoint` class**
  - Defines the `onModuleLoad()` function.
- **Defines the Event Handling**
  - Defines functions to be executed when an Event is fired.

Hello.java

```
package ch.bfh.awt.client;
import .....
/** Entry point classes define <code>onModuleLoad()</code>. */
public class Hello implements EntryPoint {
public void onModuleLoad() {
    final Button sendButton = new Button("Send");
    final TextBox nameField = new TextBox();
    final Label responseLabel = new Label();
    RootPanel.get("nameFieldContainer").add(nameField);
    RootPanel.get("sendButtonContainer").add(sendButton);
    RootPanel.get("responseContainer").add(responseLabel);
  nameField.setFocus(true);
  ... // Event Handling
 }
}
```

Widgets

- **List of default widgets**
  - Buttons: `Button`, `PushButton`, `RadioButton`,`CheckBox`,,,
  - Calendar: `DatePicker`
  - Lists : `ListBox` , `CellList`,
  - Trees: `MenuBar`, `Tree` with `CellTree`,
  - Panels: `PopoupPanel`, `StackPanel`, `HorizontalPanel`, `VerticalPanel`,
  - `http://code.google.com/intl/fr-FR/webtoolkit/doc/latest/RefWidgetGallery.html`
- **Possibility to write your own widgets:**
  - `http://code.google.com/intl/en/webtoolkit/doc/latest/DevGuideUiCustomWidgets.html`
  - Composite components (composition of existing components)
  - or from scratch in Java code

Example: StockWatcher[3]

- **An interface to watch stock values**
  - Presentation (when deployed on localhost) `localhost:8080/stockWatcherGWT`
- **User Interface: One page**
  - One page
  - A list containing the stocks
  - A field to type the stock into
  - A button to add a new stock
- **Back-office**
  - No back-office today
  - Communications with the servers are seen in the next course
  - Communication available:
    - Remote Procedure Call (RPC) in Java
    - Call to JSON data on the same server (PHP for instance)
    - Call to JSON data on another server (against the same origin policy).

[3] Source:`http://code.google.com/intl/fr-FR/webtoolkit/doc/latest/tutorial/`

JavaScript Frameworks

JavaScript Frameworks

- **JQuery**
  - To manipulate the DOM
  - To communicate in JSON with the server
- **AngularJS**
  - To develop using “Model View Controler” design Pattern
  - Communication with the server made transparent

JQuery

JQuery

- **Uses Selectors to select elements in the DOM (same syntax as for CSS)**

  $("p").hide()

  Demonstrates the jQuery hide() method, hiding all <p> elements.

  $("#test").hide()

  Demonstrates the jQuery hide() method, hiding the element with `id="test"`.

  $(".test").hide()

  Demonstrates the jQuery hide() method, hiding all elements with `class="test"`.

  $(this).hide()

  Demonstrates the jQuery hide() method, hiding the current HTML element.

Add an Event handler

- **You can define an event for any selector (all elements of one tag, one id, one class, . . .)**
  Add the function for each <p> element that hides the element when clicked.

```
$("p").click(function(){
    $(this).hide();
});
```

Add an alert on mouse over the element with id p1:

```
  $("p").click(function(){
    $(this).hide();
});
```

Modify the DOM

- **Add elements in the DOM**
  Append a child to a node

  ```
  $("p").append("Some appended to each paragraph.");
  ```

  Add a child as first child

  ```
  $("p").prepend("Some prepended text.");
  ```

- **Add text before and after** `-after()` **or** `before()`

  ```
  var txt1 = "<b>I </b>"; // Create element with ↘
  →HTML
  $("img").after(txt1);
  ```

Other Functions in JQuery

- **Ajax**
  - Connect to the same server

```
$("button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

or

```
$("button").click(function(){
    $.post("demo_test_post.asp",
    {
        name: "Donald Duck",
        city: "Duckburg"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

AngularJS

Angular JS

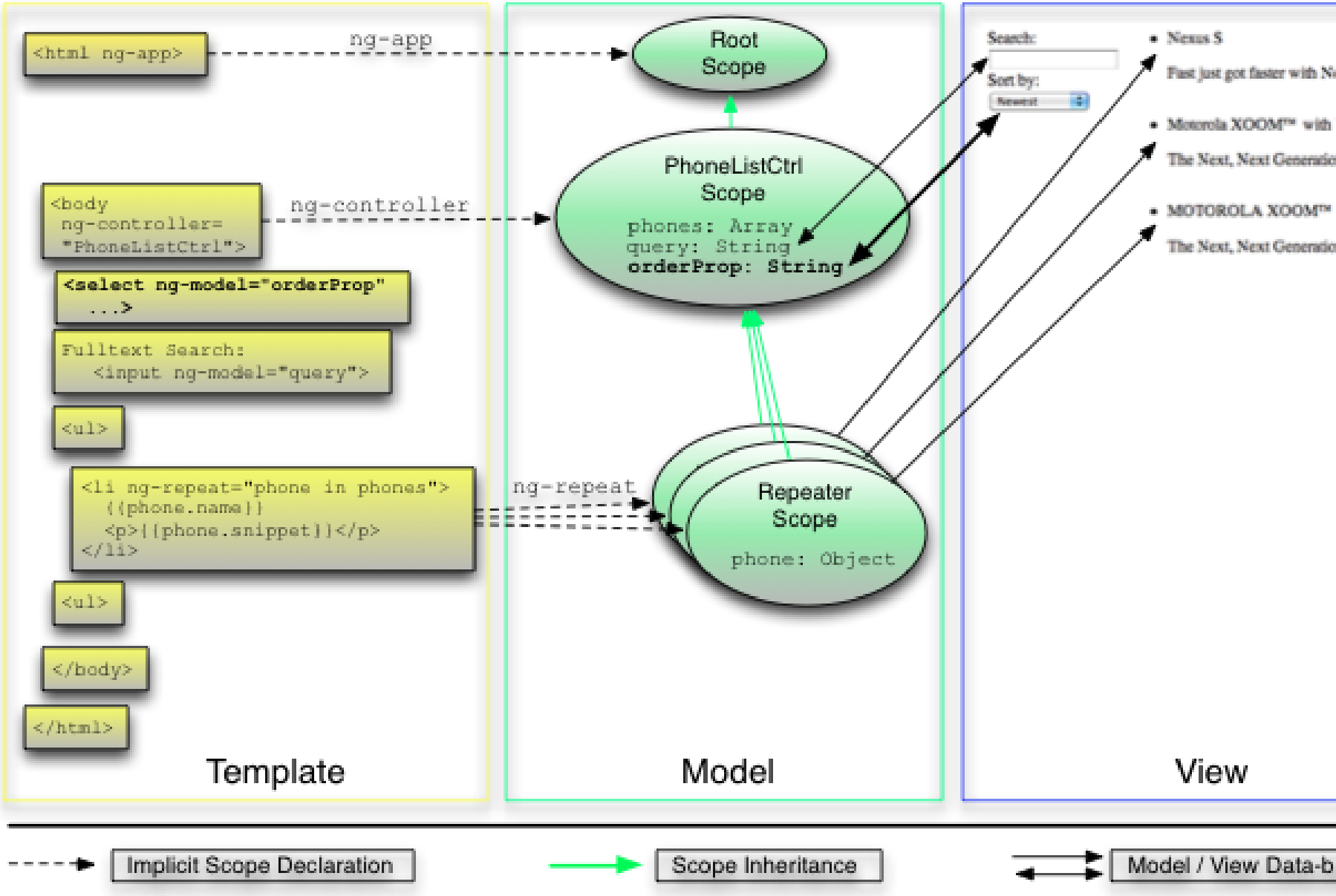
- **Complete framework for developing web application**
  - From Templating (insert value in a HTML page)
  - To Model View Controler design pattern

Template Engine

- **Write a loop in the list of `phones`**

```
<html ng-app="phonecatApp">
<head>
  ...
  <script src="bower_components/angular/angular.js"></script>
  <script src="js/controllers.js"></script>
</head>
<body ng-controller="PhoneListCtrl">
  <ul>
    <li ng-repeat="phone in phones">
      <span>{{phone.name}}</span>
      <p>{{phone.snippet}}</p>
    </li>
  </ul>
</body>
</html>
```

Model View Controler



Conclusion

- **Web Application**
  - Formerly: Pages displayed one after the other
  - Now integrated applications, one page changes
- **Technolgies used client side**
  - HTML to define the Document Object Model
  - CSS to define the layout
  - JavaScript for interacting with the user and the server
- **Languages used server side**
  - Anything you want
  - We will start with PHP to show the principles
  - Principles: Forms, sessions, DOM, ...