Advanced Web Technologies: Targets of this course

- **Know How to developp web sites in Java**
  - What is a Servlet Engine
  - What is a Web Application
- **Knows the specific JSF framework**
  - Principles
  - Knows how to customize the framework
- **Have experienced those technics**
  - In the creation of a “small” web application (homework)

Why do we need something else than PHP

- **Advantages of PHP**
  - Small
  - Easy to use
  - Available on any 5CHF/Month hosting
  - Robust
  - Lot of available libraries
- **Advantages of JSP/JSF**
  - Fully Object Oriented
  - Scallable
  - Reusability of Business Logic
  - Classloader

Possibility to integrate Apache HTTP-Deamon and Tomcat

- **Combine advantages of Apache httpd**
  - Efficiency in file serving (HTTP optimization)
  - Load balancing (can manage more than one Tomcat server)
- **And the ones of Tomcat**
  - Power of Java
  - Servlet/JSP/JSF/...

Developpement of a JSF application

- **Design the architecture of the application**
  - JSP pages containing a component tree
  - Backing Beans
  - Relations between them
- **Design the comportement of the application**
  - Event Handling
  - Action functions
  - Navigation rules

What did we see this semester

- **Servlet/JSP/Tomcat**
  - Presentation of principles of Java Web Applications
  - Servlet class, session and application scope
  - `web.xml`
- **Java Server Faces**
  - Model View Controler design pattern (Swing + Struts = JSF)
  - JSF principles (navigation, backing beans)
  - JSF basic components (in and output)
  - Navigation
  - Event handling
- **Customizing JSF (build reusable objects)**
  - Create new Components
- **Web 2.0**
  - Presentation of the new paradigm (Web 2.0)

Java on a Web Server

- **Needs a server (servlet engine)**
  - Standardized by sun (Servlet, JSP, JSF, taglibs, ...)
  - many implementations Tomcat (Apache fundation), WebSphere (IBM), ...
- **Web Application structure**
  - application contained in a directory (can be ziped in a WAR file)
  - `WEB-INF/` directory
  - File `WEB-INF/web.xml` contains the configuration of the application
  - Directory `WEB-INF/classes/` contains the arborescence of `.class` files
  - Directory `WEB-INF/lib/` contains the `.jar` libraries

Java Concepts contained in Servlets

- **HTTPRequest and HTTPResponse objects**
  - Contain all the information transfered to and from the server
- **Session**
  - Managed automatically by the servlet engine (without programming)
  - Basket used to store the information about a "session" (Session = group of requests that can be attributed to the same user)
  - `session` object contains attribute name-value pairs.
- **Application scope**
  - Contains the configuration of the application (defined in the `web.xml`
  - And object shared by all the users (DB connection or connection pools for instance).

Model View Controler

- **Model**
  - Contains the data
  - knows how to apply the data to the buisiness logic
  - does not know how to be displayed
- **View**
  - Contains the layout part
  - Doesn't have any link with the business logic
  - Doesn't know what it is displaying
- **Controler**
  - Makes the link between model and view
  - Defines what model is displayed using which view element

Java Server Faces = a recursive MVC

- **Model View Controler: first level = application**
  - Model = beans
  - View = JSP document
  - Controler = Faces servlet
- **MVC : second level = component**
  - Model = Value binding
  - View = Renderer
  - Controler = UI Component
  - But here, the choice of the renderer can be left to the faces configuration.

Java Server Faces

- **Standardized by Sun**
  - Many implementations
  - Sun reference implementation
  - Apache: MyFaces
  - Rich Faces (Jboss)
  - ...
- **Architecture**
  - One servlet: faces servlet that handles all the requests (is configured using `faces-config.xml`
  - JSP pages defining "component trees" (using JSF taglibs)
- **Tree of components**
  - Is created at each request
  - Is populated with values sent in the request
  - Transfers the infromation to the backing beans
  - Is responsible for the rendering of the page (each component renders itself)

Main JSF Features

- **Navigation Rules**
  - Action components (links and buttons) can contain a value
  - value can be a string (fixed)
  - value can be the result of an operation (on a managed-bean)
  - the `faces-config` contains the rule for the mapping value/destination.
  - So the page itself does not need to know the possible destinations
- **Event Handling**
  Two sorts of events
  - `ActionEvents`: correspond to an action (links and buttons)
  - `ValueChangedEvents`: correspond to input compenents where something has changed.

Java Server Faces Life-cicle

- The JSF servlet receives the request
- **Restore View** (creates the component tree)
- **Apply request values** (set the values contained in components)
- **Process validation** (attention: the beans do not contain any value at that moment)
- **Update Model values** (transfer the value in the beans)
- **invoke application** (first events and method invocation, then navigation is evaluated)
- **Render Response**
- the http response is returned to the client

JSF is open to new Components

- **You can download new libraries**
  - MyFaces (standard implementation + new features) Tomahawk, Trinidad, Tobago,
  - Rich Faces
  - ...
- **You can create and reuse your own components**
  - Written In java
  - Using Facelet simple functionality

JSF Component Model

- **Much like Swing's component model**
  - It has events and properties
  - also has containers that contain components,
  - and that also are components that can be contained by other containers.
  - In theory, the JSF component model is divorced from HTML and JSP.
  - The standard set of components that ships with JSF has JSP bindings and generates HTML renderings.
- **Component functionality typically centers around two actions: decoding and encoding data.**
  - *Decoding* is the process of converting incoming request parameters to the values of the component.
  - *Encoding* is converting the current values of the component into the corresponding markup, that is, HTML.

Conclusion

- **We have seen a lot**
  - Tomcat, Servlet, JSP, JSF, JSTL, Ajax4jsf,...
- **But it is only an introduction**
  - API's are much larger
  - Functionalities are bigger
  - Only an overview
- **We have focused on only one layer**
  - We have seen only the presentation layer
  - You still have to combine it with a business logic and a persitancy layer
  - Linked with EJB (Entity Java Beans) or JPA (Java Persistancy API - Hibernate)