Vers des frameworks JavaScript

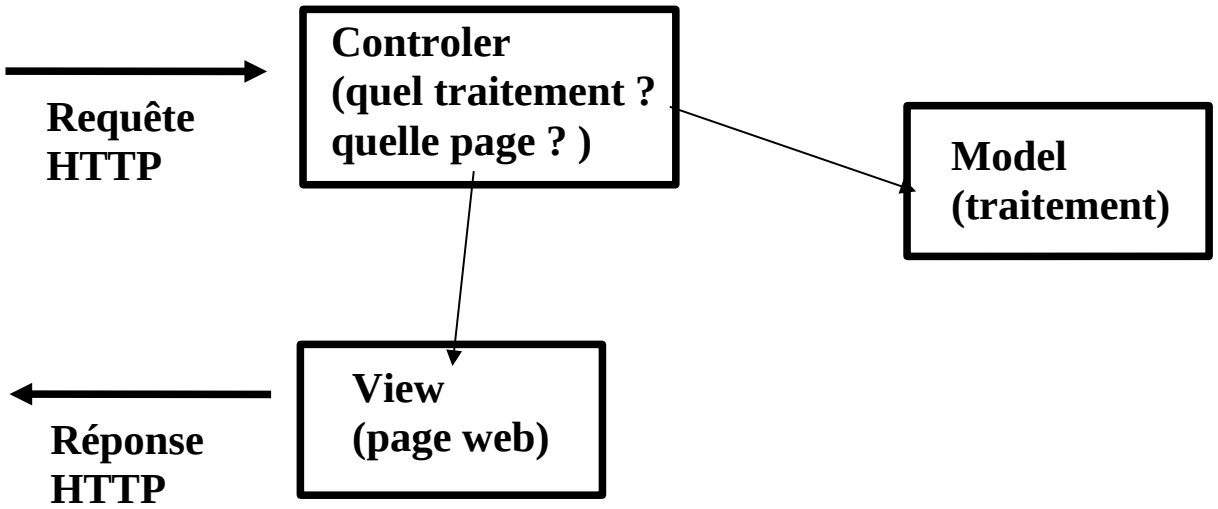
Daniel Hagimont

IRIT/ENSEEIHT
2 rue Charles Camichel - BP 7122
31071 TOULOUSE CEDEX 7

Daniel.Hagimont@enseeiht.fr
http://hagimont.perso.enseeiht.fr

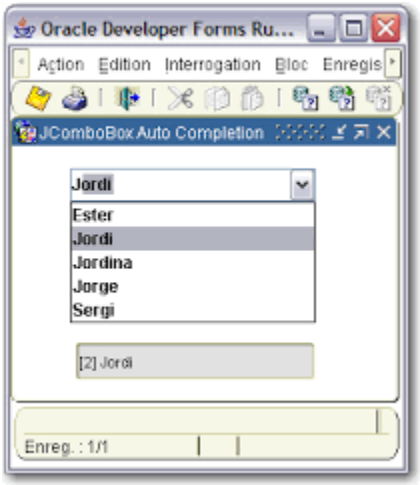
Modèle MVC (rappel)

- Model View Controler
- Séparation entre
  - Le contrôleur : servlet qui aiguille les requêtes
  - Le Modèle : les classes (beans) qui traitent les données
  - La vue : pages JSP pour l'affichage à l'écran



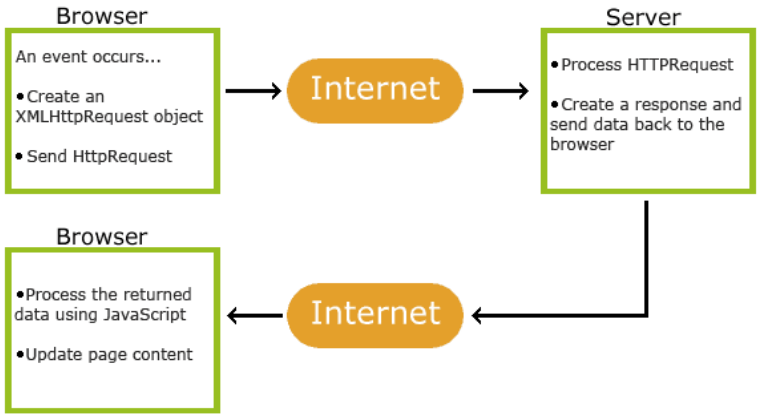
Le problème

- Raffraîchissement de la page dans le navigateur à chaque « submit »
- Effet flash
  - Rechargement de la page entière
  - Même lorsqu'on a modifié un fragment de la page
- Exemple : autocomplétion
  - TextField avec proposition de sélection
  - Recherche dans la base de données



JavaScript et Ajax

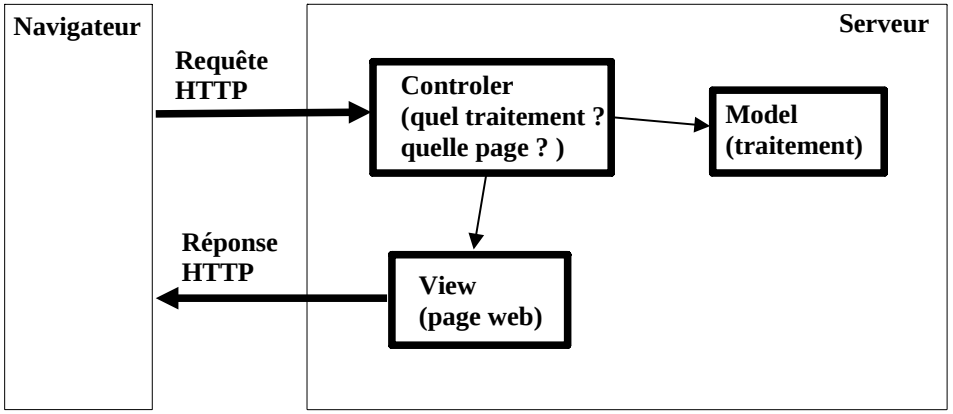
- AJAX = Asynchronous JavaScript And XML
  - Un objet XMLHttpRequest pour charger des données depuis un serveur Web
  - JavaScript and HTML DOM (pour affichage dans le navigateur)

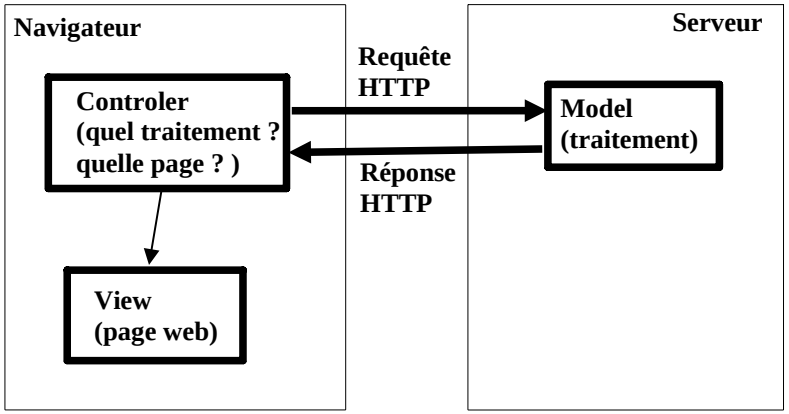



```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>
</div>

</body>
</html>
```

```
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

Généralisation





Le serveur

- Facade : un serveur de Web Services REST
  - Méthodes
  - Paramètres HTTP
  - Échanges de JSON
- Données : JPA
  - Entity beans

Le serveur : exemple du TP

```
@Singleton
@Path("/")
public class Facade {

        @PersistenceContext
        EntityManager em;

        @POST
        @Path("/addperson")
        @Consumes({ "application/json" })
        public void addPerson(Person p) {
                System.out.println("coucou");
                em.persist(p);
        }

        @GET
        @Path("/listpersons")
        @Produces({ "application/json" })
        public Collection<Person> listPersons() {
                return em.createQuery("from Person", Person.class).getResultList();
        }
        ...
```

Frameworks JavaScript

- Il y en a plein
  - Exemples avec jQuery, AngularJS, React
- Single page ou multiple pages

jQuery

- Manipulation du DOM
- Fonctions associées à des événements

Control.js

```
$(document).ready(function() {
    loadMain();
});

function loadMain() {
    $("#Main").load("Main.html", function() {
        $("#BTAddPerson").click(function() {
            ...
        });
        $("#BTAddAddress").click(function() {
            ...
        });
        $("#BTAssociate").click(function() {
            ...
        });
        $("#BTList").click(function() {
            ...
        });
    });
}
```

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script src="Control.js"></script>
</head>
<body>

<div id="Main">
</div>
<br>
<label id="ShowMessage">
</label>

</body>
</html>
```

Main.html

```
<input type="button" id="BTAddPerson" value="Add personne">
<input type="button" id="BTAddAddress" value="Add address">
<input type="button" id="BTAssociate" value="Associate">
<input type="button" id="BTList" value="List">
<br>
```

jQuery

- Manipulation du DOM
  - Accès à une saisie
    - $("#FirstName").val();
  - Modification d'un bloc (div)
    - $("#Main").load("AddPerson.html", function() {...}) ;
    - $("#Main").empty();
    - $("#Main").append(...);
- Appels Ajax

```
jQuery.ajax({
    url: url,
    type: "GET",
    success: function (response) {...}
    error: function (response) {...}
});
```

AngularJS

- Manipulation du DOM
- Fonctions associées à des événements

index.html

```
<script>
function click(button, scope, http) {
    switch (button) {
        case "addPerson" :
            scope.showAddPerson = true;
            break;
        …
    }
}

function OK(action, scope, http) {
    switch (action) {
        case "addPerson" :
            // use scope.person
            break;
        …
    }
}

var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope,$http) {
   $scope.doClick=function(button) {click(button,$scope,$http);}
   $scope.doOK=function(action) {OK(action,$scope,$http);}
});

</script>
```

index.html

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

<button ng-click="doClick('addPerson')">Add person</button>
<button ng-click="doClick('addAddress')">Add address</button>
<button ng-click="doClick('associate')">Associate</button>
<button ng-click="doClick('list')">List</button>
<br>
   <div ng-show="showAddPerson">
      <form novalidate>
          First Name: <input type="text" ng-model="person.firstName"><br>
          Last Name:  <input type="text" ng-model="person.lastName"><br>
          <br>
          <button ng-click="doOK('addPerson')">OK</button>
      </form>
   </div>
   ...
```

AngularJS

- Manipulation du DOM
  - Accès à une saisie
    - Liaison entre éléments de saisie HTML et variables JS (ng-model)
  - Modification d'un bloc (div)
    - Utilisation des variables JS dans HTML
- Appels Ajax

```
http.get("rest/listpersons").then(function(response) {
    if (response.status == 200) {
        scope.listPersons = response.data;
        …
    }
}) ;
```

```
<form>
    Select a person :<br>
    <label ng-repeat="p in listPersons">
        <input type="radio" ng-model="$parent.onePerson"
                value="{{p.id}}">{{p.firstName}} {{p.lastName}}<br>
    </label>
</form>
```

React

- Manipulation du DOM
  - Composant = fonction
    - Peut inclure un état (persistent)
      - const [fname, setFname] = useState("");
    - Retourne le code HTML
  - Composant affiché dans un div
    ReactDOM.render()
  - Composant re-affiché si état modifié

- Fonctions associées à des événements
- Appels Ajax
  - fetch(URL).then((response) => {…});
  - async function invoke() {
    res = await fetch(URL) ;
    if (res.ok) return await res.json() ;}

index.html

```
<!DOCTYPE html>
<html>
<body>
   <div id="root"></div>
</body>
</html>
```

Main.js

```
ReactDOM.render(<Main />, document.getElementById('root'));

function Main() {
 const addPerson = () => {
    ReactDOM.render(<AddPerson />, document.getElementById("Worker"));
 }
 const addAddress = () => {
    ReactDOM.render(<AddAddress />, document.getElementById("Worker"));
 }
 const associate = () => {
    ReactDOM.render(<Associate />, document.getElementById("Worker"));
 }
 const list = () => {
     ReactDOM.render(<List />, document.getElementById("Worker"));
 }
 return (
 <>
  <div id="Main">
        <button onClick={addPerson}>Add personne</button>
        <button onClick={addAddress}>Add address</button>
        <button onClick={associate}>Associate</button>
        <button onClick={list}>List</button>
  </div>
  <br/>
  <div id ="Worker">
  </div>
 </>
 );
}
```