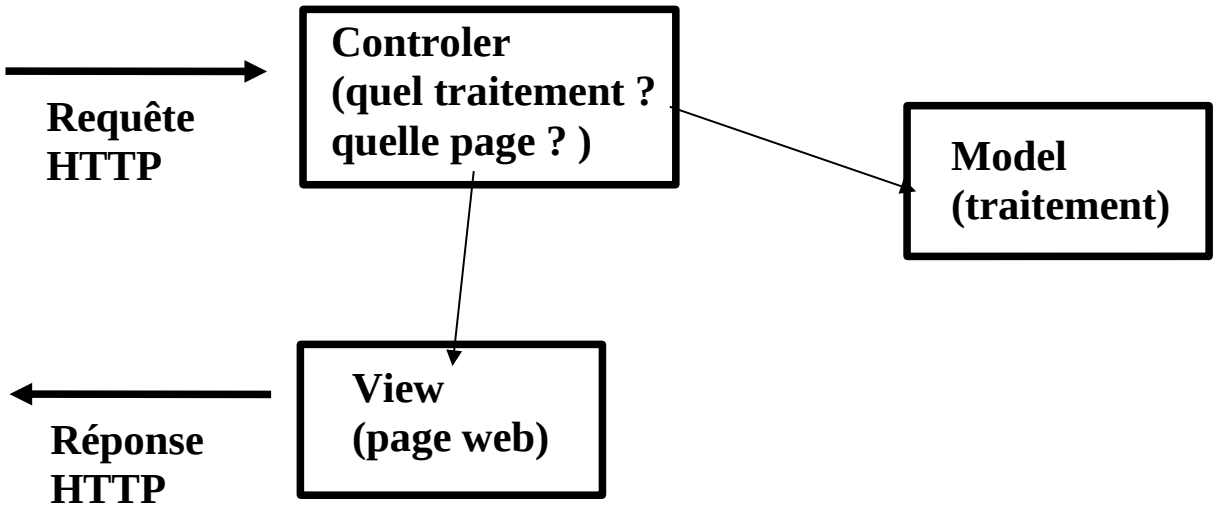
# Vers des frameworks JavaScript

**Daniel Hagimont**

**https://www.google.fr/search?q=daniel+hagimont+home+page**

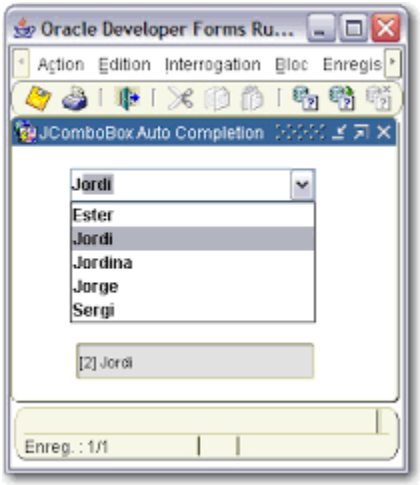
# Modèle MVC (rappel)

- 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



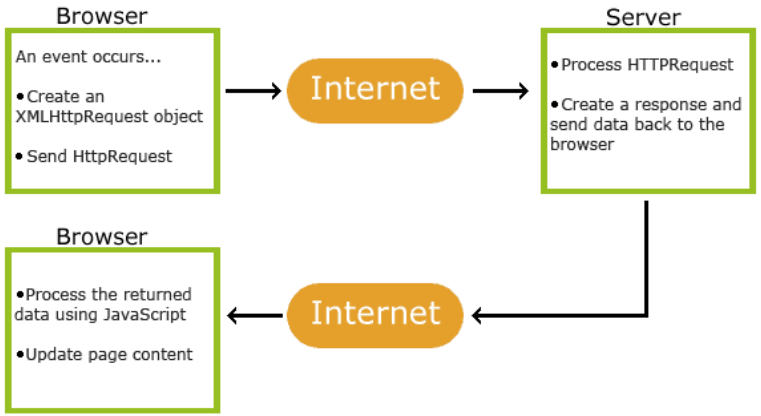
# Le problème

- 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 (supporté par les navigateurs)
  - 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", "Serv", true);
  xhttp.send();
}
```

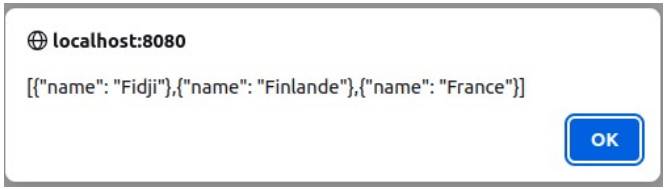
# Autocomplétion - servlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    String start = request.getParameter("start");
    if (start == null) response.getWriter().append("");

    String res = "";
    boolean first = true;
    for (int i=0; i<liste.length; i++) { // liste = la liste des pays
        String country = liste[i];
        if (country.startsWith(start)) {
            if (first) {
                first = false;
                res += "[";
                res += "{\"name\": \""+country+"\"}";
            } else {
                res += ",";
                res += "{\"name\": \""+country+"\"}";
            }
        }
    }
    res += "]";

    response.setContentType("application/json");
    response.getWriter().append(res);
}
```

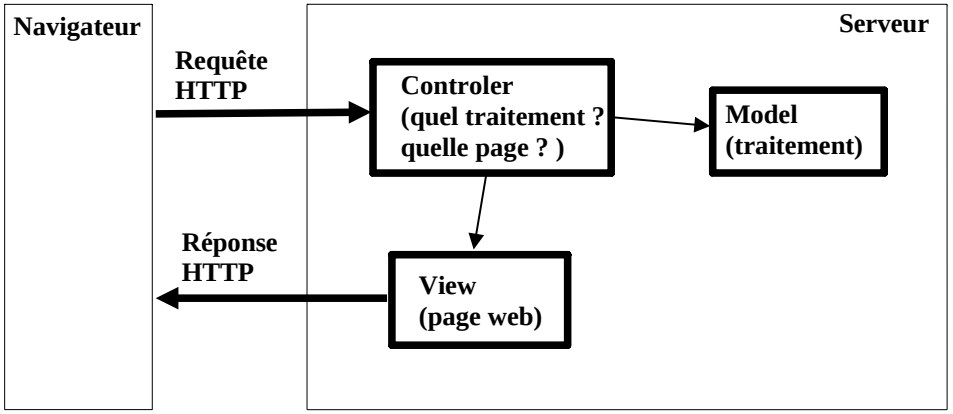


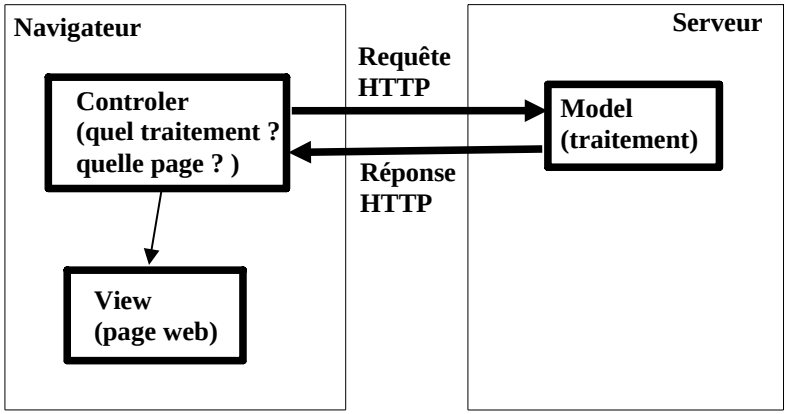
# Autocomplétion – page HTML/Javascript

```
<script>
function loadData() {
   let input = document.getElementById("saisie");
   let start = input.value;
   const xhttp = new XMLHttpRequest();
   xhttp.onload = function() {
      let data = JSON.parse(this.responseText);
      let select = document.getElementById("choice");
      let newoptions = "<option value=\"\">Choose a country</option>";
      data.forEach(function(country) {
         newoptions += "<option value=\"\">"+country.name+"</option>";
      });
      select.innerHTML = newoptions;
   }
   xhttp.open("GET", "Serv?start="+start, null);
   xhttp.send();
}
</script>

<input type="text" name="saisie" id="saisie" oninput="loadData()"> <br>
<select name="choice" id="choice" size="10">
   <option value="">Choose a country</option>
</select>
```

# Généralisation



# Le serveur

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

# Le serveur : exemple du TP

```
@RestController
public class Facade {

   @Autowired
   PersonneRepository pr;
   @Autowired
   AdresseRepository ar;

   @GetMapping("/ajoutpersonne")
   public void ajoutPersonne(@RequestParam("nom") String nom, @RequestParam("prenom") String prenom) {
            Personne p = new Personne();
            p.setNom(nom);
            p.setPrenom(prenom);
            pr.save(p);
    }

   @GetMapping("/ajoutadresse")
   public void ajoutAdresse(@RequestParam("rue") String rue, @RequestParam("ville") String ville) {
            Adresse a = new Adresse();
            a.setRue(rue);
            a.setVille(ville);
            ar.save(a);
   }
```

# 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


**index.html**

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

- Manipulation du DOM
  - Composant = fonction
    - Peut inclure un état (persistent)
      const [fname, setFname] = useState("");
    - Retourne le code HTML
  - Composant peut inclure un composant
    - Etat
  - Composant re-affiché si état modifié
- Fonctions associées à des événements
  - handleSubmit
- Appels Ajax
  - fetch(URL).then((response) => {...});
  - async function invoke() {
    res = await fetch(URL) ;
    if (res.ok) return await res.json() ;}

**Main.js**

```
function Main() {
 const [currentComponent, setCurrentComponent] = useState(null);
 const [message, setMessage] = useState("");

 return (
  <>
   <div id="Main">
    <button onClick={() => setCurrentComponent(
                <AddPerson setMessage={setMessage}
                           setComponent={setCurrentComponent}
                 />)}>Add person</button>
    <button onClick={() => setCurrentComponent(
                <AddAddress setMessage={setMessage}
                            setComponent={setCurrentComponent}
                 />)}>Add address</button>
    <button onClick={() => setCurrentComponent(
                <Associate setMessage={setMessage}
                           setComponent={setCurrentComponent}
                 />)}>Associate</button>
    <button onClick={() => setCurrentComponent(
                <List setMessage={setMessage} />)}>List</button>
   </div>
   <br />
   <div id="Message">{message}</div>
   <br />
   <div id="Worker">{currentComponent}</div>
  </>
 );
}
```