Ajax

Petite réflexion sur l'exo 1 de la semaine passée

- Nous avons implémenté une Servlet qui réponds à un GET
  - La réponse est de l'HTML fournie directement à partir de l'implémentation en Java.
  - Le bonus montre que c'est peu pratique de travailler de la sorte : on finit par écrire de l'HTML long et compliqué dans des Strings en Java.

JSP

- Il existe une approche classique pour gérer cette manière de travailler :
  - Utilisation de fichiers templates, classiquement JSP en Java.
  - Les données à injecter dans le templates sont retenues dans un modèle Java (un objet).
  - Des tags JSP permettent de faire le lien entre le modèle et l'HTML à générer.
  - La servlet s'occupe de faire tout fonctionner ensemble.

Exemple JSP

```
<HTML>
<HEAD>
<TITLE>Test</TITLE>
</HEAD>
<BODY>
<%!
int minimum(int val1, int val2) {
   if (val1 < val2) return val1;
   else return val2;
}
%>
<% int petit = minimum(5,3);%>
<p>Le plus petit de 5 et 3 est <%= petit %></p>
</BODY>
</HTML>
```

Critique du JSP

- Mélange de Java dans de l'HTML : difficile à maintenir, requiert beaucoup de discipline.
- Navigation "à l'ancienne" : l'unité de communication est la page entière, correspond au Web classique, pas aux applications Webs modernes.
- En général : très verbeux et pas amusant à utiliser.
  - On va donc utiliser une approche différente !

Ce qu’on est capable de faire jusqu’ici

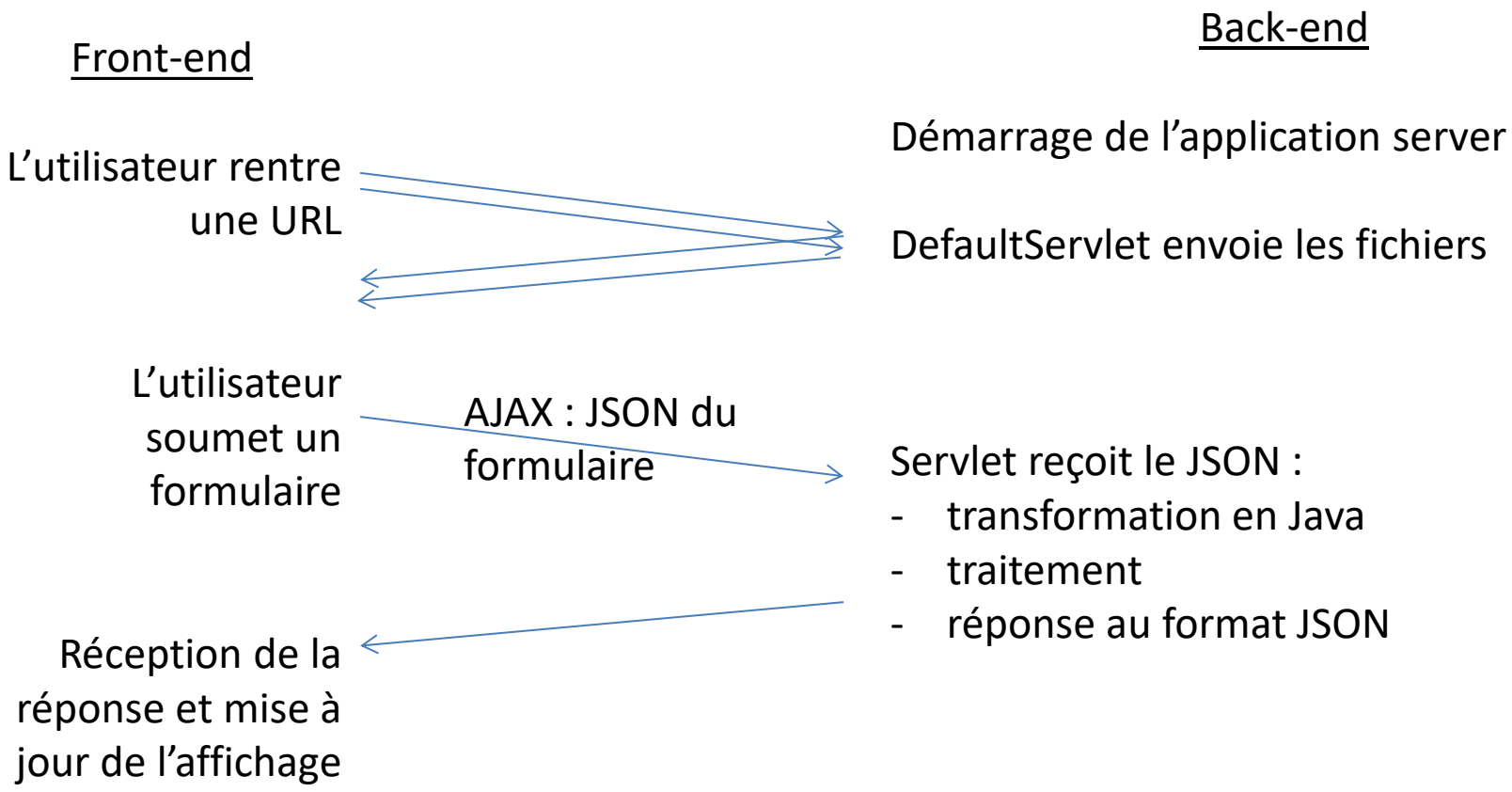
- Ecrire une page HTML avec du CSS et du Javascript.
- Utiliser jQuery pour manipuler dynamiquement cette page.
- Prendre un formulaire et le transformer en JSON.
- Servir des fichiers à partir d’un application server (à l’aide d’une DefaultServlet)

Ce qu'on est capable de faire jusqu'ici

- D'écrire une servlet pour traiter une requête HTTP POST ou GET et envoyer une réponse
- De transformer du JSON en Java.
- Il nous manque un dernière chose :
  - Envoyer une requête à partir du front-end et recevoir la réponse sans quitter la page courante.

=>Ceci s'appelle une requête Ajax

Avec Ajax, on aura donc le design suivant :



Appel Ajax

- Un appel Ajax est donc une requête HTTP, mais qui ne navigue pas vers une nouvelle page.
- Nous utiliserons l'incarnation fournie par jQuery :
  - http://api.jquery.com/jquery.ajax/

Exemple

```
$.ajax({
  url: 'getInfo',
  type: 'POST',
  data: 'id=5',
  success: function(reponse) {
       // traitement de la reponse
       ...
  },
  error: function(e) {
       // en cas d'erreur
       console.log(e.message);
  }
});
```

Dernière précision

- Une requête HTML transporte du texte, pas du JSON.
- JSON.parse : string -> JSON
- JSON.stringify : JSON -> string

```
$.ajax({
  ...
  data: JSON.stringify(json),
  success: function(reponseTxt) {
    var reponse=JSON.parse(responseTxt);
    ...
  }
});
```

GET ou POST

- Une requête Ajax peut tout aussi bien faire du GET que du POST
  - GET : on veut pouvoir bookmarker l'URL de la requête.
  - POST : on ne veut pas pouvoir bookmarker cette URL.
- Comme nos requêtes Ajax communiqueront en JSON, que ce dernier n'est pas à destination du l'utilisateur => requêtes POST