ISIMA

Java EE : Présenter une application Web II

Ajax

Présenter une application Web

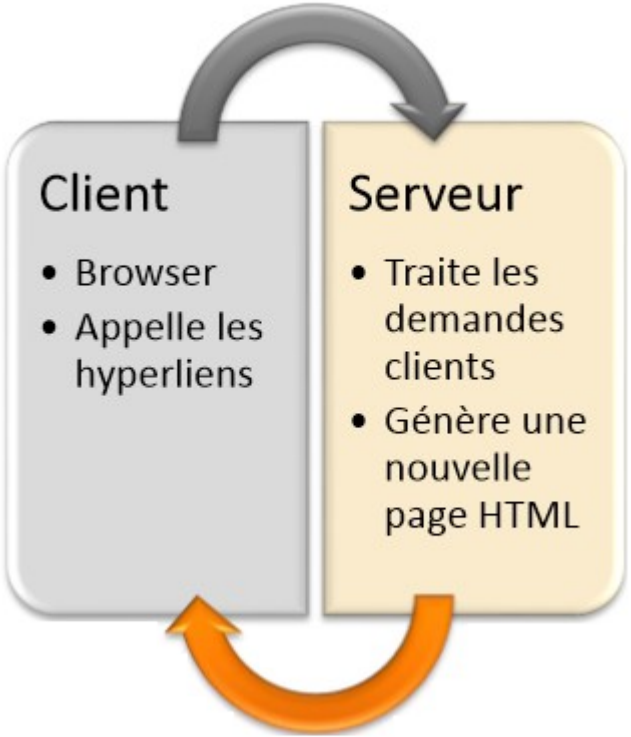
- Previously in « cours de JEE » :
  - Pages statiques
  - Page dynamiques
    - Rôle JSP/Servlet
  - Navigation
  - JSTL/EL

ISIMA

Java™

ORACLE

Un peu d'histoire du Web



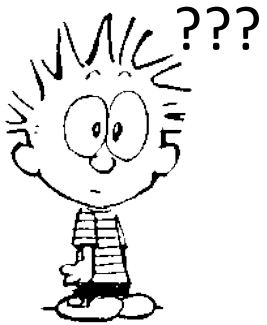
- Cette structure c'est le Web 1.0
  - Les interfaces sont en HTML
  - Les hyperliens constitue le « ciment de l'application »
    - Ils permettent de naviguer d'une page à une autre
  - Peu fluide
    - Les pages sont rechargées intégralement

ISIMA

Java™ ORACLE

Ajax

- Qu'apporte cette solution ?
- Que signifie cet acronyme barbare ?
  - Asynchronous Javascript And Xml



- Change la gestion de la page
  - Les requêtes sont toujours à l'initiative du client
  - Mais il est possible d'utiliser Javascript pour interroger le serveur sans soumettre un formulaire ou passer par un hyper lien

Ajax avec heureCourante

- On prends les même et on recommence ;)
- Un exemple valant beaucoup d'explications…
- Nous allons ajouter un rafraichissement Ajax à heureCourante
  - Toutes les secondes
  - Avec Jquery
    - Instancie les objets nécessaire à l'exécution de requêtes Ajax
      - XMLHttpRequest pour les intimes
  - Et nous supprimerons le bouton « rafraichir »

Rappel de l'exemple précédent

- Nous avons une servlet AfficherHeureCourante
  - redirige vers une JSP
  - génère l'heure courante dans un format lisible

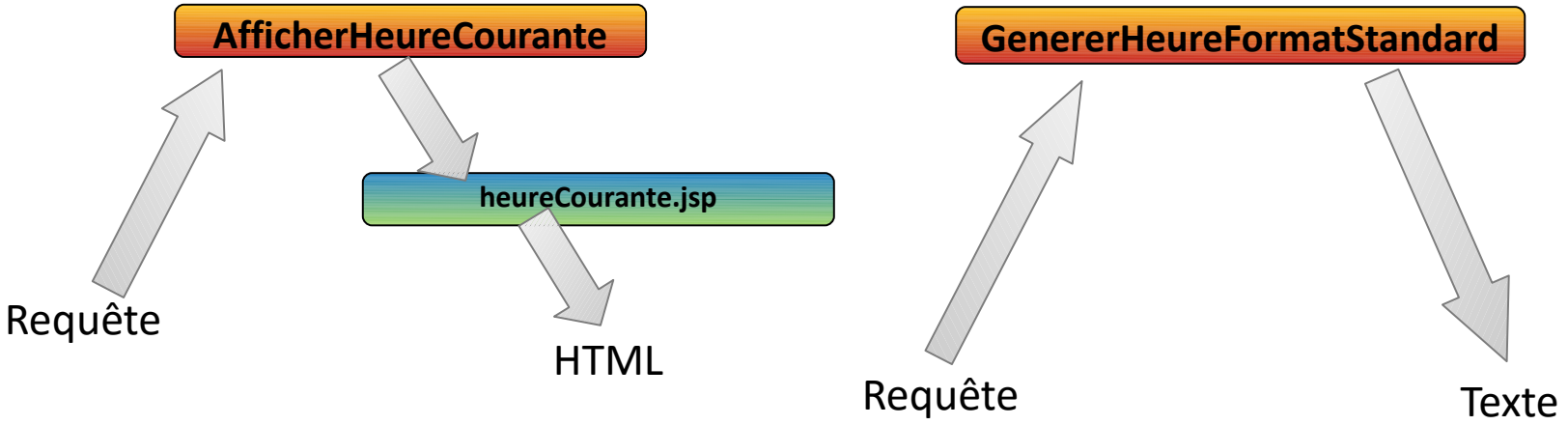
```
@WebServlet("/AfficherHeureCourante")
public class AfficherHeureCouranteServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        final DateTimeFormatter formatteur = DateTimeFormatter.ofPattern("HH:mm:ss 'le' dd/MM/yyyy");
        final String currentDateAsString = formatteur.format(LocalDateTime.now());
        request.setAttribute("currentDateAsString", currentDateAsString);
        request.getRequestDispatcher("/heureCourante.jsp").forward(request, response);
    }
}
```


ISIMA Java ORACLE

Qu'allons nous modifier ?

- Qu'avons-nous besoin en plus ?
  - La requête Ajax permet de ne consommer que des données
    - Pas besoin de HTML pour mettre à jour la page
    - Remplacer l'heure dans la page par l'heure à jour depuis le serveur
  - Créons une servlet qui rend seulement l'heure formatée
    - Sous forme de texte
    - Le résultat sera exploité en JavaScript !

Schéma explicatif



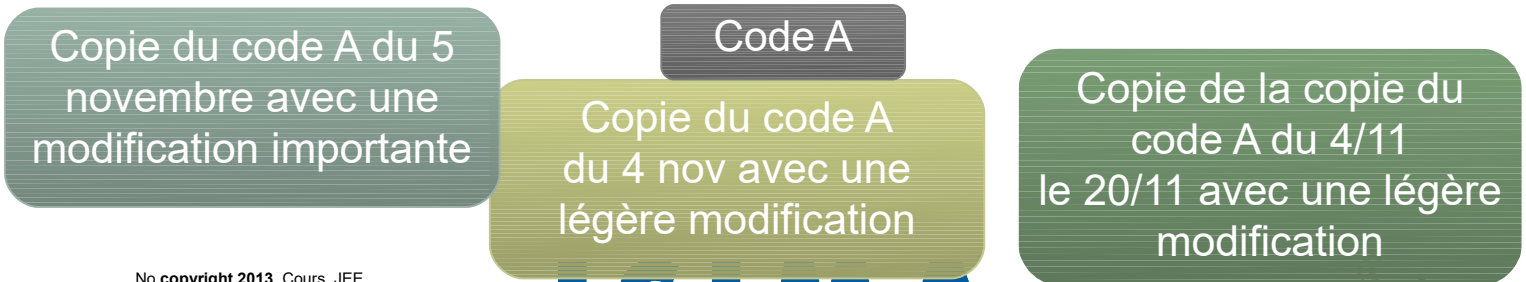
- Les deux Servlets font la même chose à l'exception du format de sortie
- Mutualisons ces deux codes !

Mutualisation

- Un soupçon de conception
- Pourquoi mutualiser du code ?
  - Meilleur maintenabilité
    - Un seul point pour modifier le comportement de tous les consommateurs
- La redondance est l'ennemi de la maintenabilité !
- Le copier/coller d'un existant, un ami à très court terme
  - Pas de risque de régression
  - Facilité de mise en place
- Et a long terme ?

Mutualisation

- Un cas en exemple
- J'ai un bug sur le code A
  - Il en existe 4 copies
    - En entreprise nous pouvons trouver des duplications bien plus nombreuses !
  - Je dois faire évoluer A ! Dois-je faire évoluer les copies ? Qui les utilisent ? Est-ce que je suis sur d'avoir bien apréhender tous les cas ?
  - La suppression de la redondance est fastidieuse et dangereuse



Revenons à heure courante

- Mutualisons intelligemment la génération de l'heure
- Comment mutualiser ?
  - L'héritage ?
    - En java l'héritage est limité à une classe
    - Plutôt dans le cadre de mutualisation technique
  - Pattern Utils ?
    - Pratique pour le technique mais moins adapté à un code fonctionnel
  - La délégation ?
    - Methode souple et lisible. Utilisons là (même si Utils pourrait convenir)

Créons une classe GenerateurHeureCourante

```
public class GenerateurHeureCourante {

    private static final String FORMAT_STANDARD_DATE = "HH:mm:ss 'le' dd/MM/yyyy";

    public String nowWithStandardFormat() {
        return
DateTimeFormatter.ofPattern(FORMAT_STANDARD_DATE).format(LocalDateTime.now());
    }
}
```

Que deviens notre servlet existante ?

```
@WebServlet("/AfficherHeureCourante")
public class AfficherHeureCourante extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        final GenerateurHeureCourante generateurHeureCourante = new
GenerateurHeureCourante();
        request.setAttribute("currentDateAsString",
generateurHeureCourante.nowWithStandardFormat());
        request.getRequestDispatcher("/heureCourante.jsp").forward(request, response);
    }
}
```

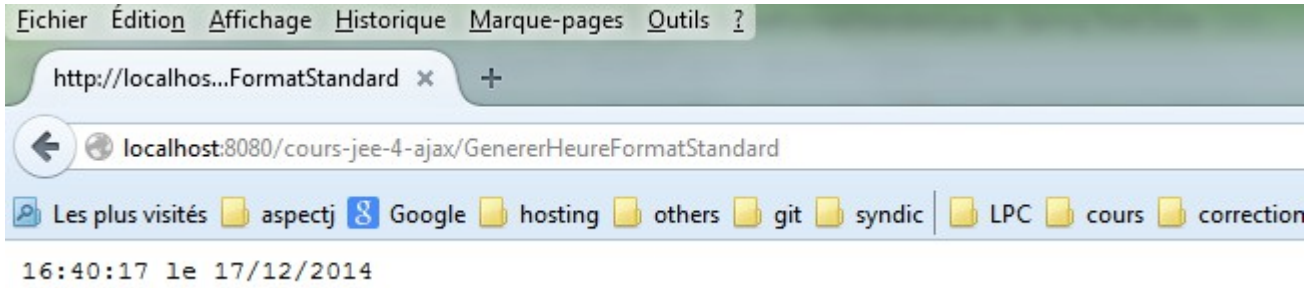
Un petit test

- Toujours vérifier que le code fonctionne toujours

Créons la Servlet GenererHeureFormatStandard

```
@WebServlet("/GenererHeureFormatStandard")
public class GenererHeureFormatStandard extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        final GenerateurHeureCourante generateurHeureCourante = new
GenerateurHeureCourante();
        response.getWriter().write(generateurHeureCourante.nowWithStandardFormat());
    }
}
```

Testons dans le navigateur



Intégrons la solution sur notre page JSP

- En Javascript avec JQuery
- JQuery permet d'interroger le serveur en Ajax
  - Nous allons utiliser la méthode Ajax
    - Les methodes get, post ou load() l'utilisent
  - Pour la documentation allez sur http://jquery.com/
- Nous allons utiliser également la méthode standard JS setTimeout()

Le code HTML

- Les modifications de la présentation

```
<div class="container">
      <div class="jumbotron">
            <h1>Heure courante</h1>
            <p class="lead">
                  Cette page a été générée à <span id="heureCourante">${currentDateAsString}</span>
            </p>
      </div>
</div>
```

- Suppression du bouton
- Ajout d'un attribut span autour de l'heure avec un id heureCourante

ISIMA

Java™

ORACLE

Le code Javascript

- Situé avant la fermeture de </body>

```
<script type="text/javascript">
	$(function() {
		rafraichirDansUneSeconde();
	});

	function rafraichirDansUneSeconde() {
		setTimeout(rafraichirHeureCourante, 1000);
	}

	function rafraichirHeureCourante() {
		$
				.ajax({
					url : '${pageContext.request.contextPath}/GenererHeureFormatStandard',
					success : function(data, textStatus, jqXHR) {
						$('#heureCourante').text(data);
					}
				});
		rafraichirDansUneSeconde();
	}
</script>
```

Décodons ce code Javascript

- Instruction par instruction (part 1)

```
$(function() {
    rafraichirDansUneSeconde();
});
```

- Deux points remarquables
  - $(function) est une fonction de Jquery
    - Elle appelle la méthode passée en paramètre une fois seulement que la page est chargée
  - Nous appelons la méthode `rafraichirDansUneSeconde`
    - Nous aurions pu écrire également `$(rafraichirDansUneSeconde)` ce qui aurait eu le même effet

Décodons ce code Javascript

- Instruction par instruction (part 2)

```
function rafraichirDansUneSeconde() {
    setTimeout(rafraichirHeureCourante, 1000);
}
```

- Deux points remarquables
  - La méthode setTimeout(function, timeInMs) appelle la méthode passé en paramètre au bout d'une seconde
  - `rafraichirHeureCourante :` en Javascript le nom de la fonction peut-être utilisé comme une référence passable en parametre !

Décodons ce code Javascript

- Instruction par instruction (part 3)

```
function rafraichirHeureCourante() {
      $.ajax({
            url : '${pageContext.request.contextPath}/GenererHeureFormatStandard',
            success : function(data, textStatus, jqXHR) {
                  $('#heureCourante').text(data);
            }
      });
      rafraichirDansUneSeconde();
}
```

- $.Ajax()
  - La structure de configuration est en JSON
    - C'est une association clé/valeur très puissante
    - url de la servlet (remarquons l'utilisation de l'EL pour le contexte)
      - Il est possible de ne pas le mettre auquel cas le contexte est celui de la page courante

Décodons ce code Javascript

- Instruction par instruction (part 4)

```
function rafraichirHeureCourante() {
      $.ajax({
            url : '${pageContext.request.contextPath}/GenererHeureFormatStandard',
            success : function(data, textStatus, jqXHR) {
                  $('#heureCourante').text(data);
            }
      });
      rafraichirDansUneSeconde();
}
```

- $.Ajax()
  - Success() une référence à une fonction lorsque la requête ajax est terminée
    - Data contient le texte de la page générée
  - $('#heureCourante') : Selector signifiant « je recupere le composant avec l'id heureCourante »

Décodons ce code Javascript

- Instruction par instruction (part 5)

```
function rafraichirHeureCourante() {
      $.ajax({
            url : '${pageContext.request.contextPath}/GenererHeureFormatStandard',
            success : function(data, textStatus, jqXHR) {
                  $('#heureCourante').text(data);
            }
      });
      rafraichirDansUneSeconde();
}
```

- $().text() remplace le texte de la balise par ce qui est passé en paramètre
- `rafraichirDansUneSeconde();`
  - On relance le rafraichissement une seconde apres

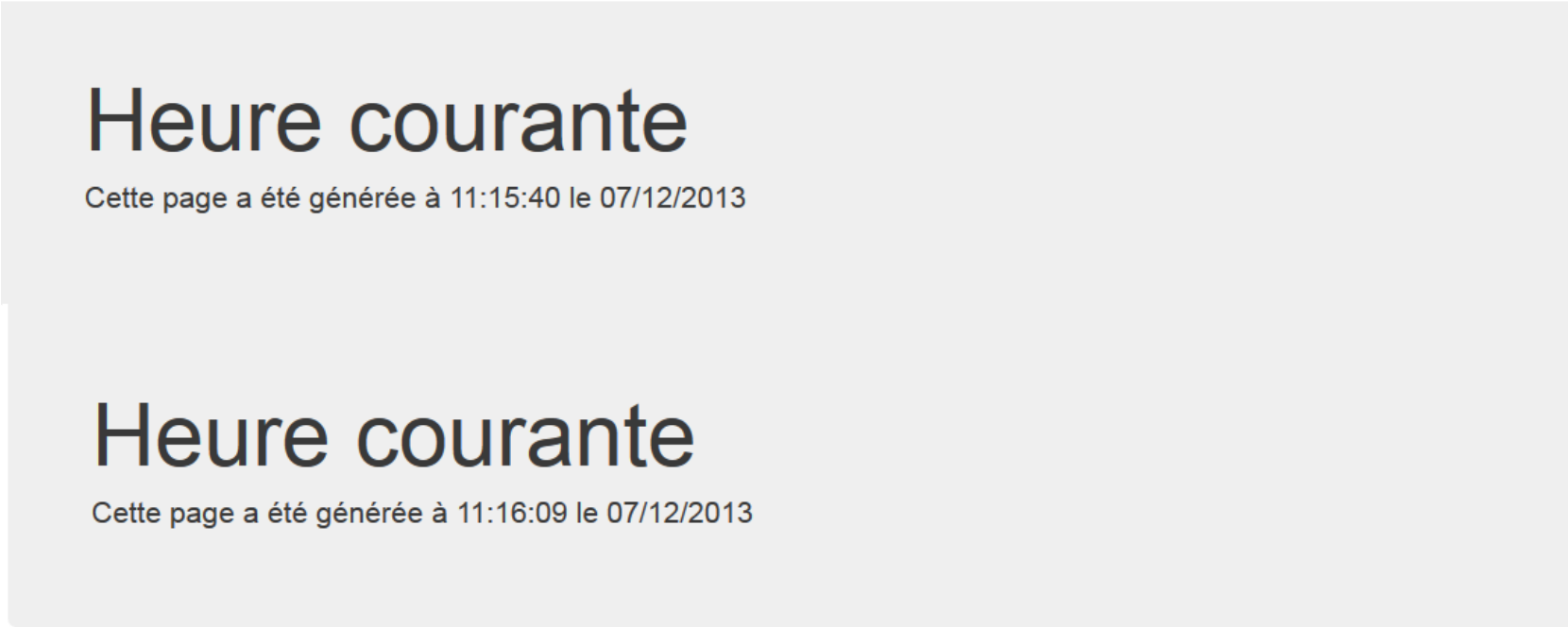
ISIMA

Java™

ORACLE

Et le résultat

- Un rafraichissement automatique



Quelques mot sur JQuery

- Une librairie concise

- JQuery a de grosses qualités :
  - Une API Ajax riche
  - Une grande concision
    - « The Write Less, Do More, JavaScript Library.”
- JQuery est extensible avec un système de plugins
- Vous trouverez dans le cours une page jquery-examples.jsp
  - Elle contient des exemples de fonctions utiles dans Jquery pour manipuler et accéder à des propriétés du DOM

La manipulation des DOM

- Les selectors (avec des exemples par classe et id) :
  - Permet de retrouver des éléments dans la page à partir de leurs noms, leurs attributs ou bien leurs classes CSS
  - La documentation est ici : http://api.jquery.com/category/selectors/
  - Les selectors renvoient toujours des ensembles sur lequel il est possible de faire des opérations
  - Les ensembles peuvent être aussi parcouru avec la fonction each() dont vous trouverez un exemple d'utilisation
- Vous trouverez une utilisation de la fonction val() permettant d'obtenir des valeurs pour des input
  - http://api.jquery.com/val/

La manipulation du DOM

- $(selector).attr() permet de récupérer des informations sur un attribut. Par exemple :

```
<input id="xxx" type="button">
```

  - `$('#xxx').attr('type')` permet de récuperer le type (button)
  - `$('#xxx').attr('type', 'text')` permet de changer de l'input button en type texte (champ de saisie libre)
  - http://api.jquery.com/attr/
- $().prop() est l'équivalant de attr en plus intelligent. Il ne fonctionne qu'avec des attributs standards.
  - http://api.jquery.com/prop/

La manipulation du DOM

- Pour pouvoir exécuter des évènements sur les clicks par exemple, vous devez lier des évènements aux fonctions javascript
  - http://api.jquery.com/on/
- Dans le prochain TP nous utiliserons uniquement des évènements click comme vous le verrez en exemple mais il en existe bien sûr beaucoup d'autre.

Encore du JQuery

- JQuery est capable de parser du JSON envoyé depuis un serveur
- Cette fonction est très intéressante car elle permet de :
  - Transporter des informations de façon concise
  - Est supportée nativement par le langage Javascript
- Utilisons le pour passer la date et l'heure depuis notre serveur vers notre client
- Commençons par créer le service JSON coté serveur

ISIMA Java ORACLE

Restituer du JSON coté serveur

- Utilisons l'API standard de JEE
- Depuis JEE7 a été introduit une API de génération de JSON
- Le JSON permet de structurer les informations et d'envoyer sous forme de clés valeurs des informations
- Vous devrez télécharger le jar javax.json-1.0.4.jar car Tomcat ne fournit pas l'implémentation de ce service
  - Copiez le jar dans WEB-INF/lib de votre projet
  - http://central.maven.org/maven2/org/glassfish/javax.json/1.0.4/javax.json-1.0.4.jar

La structure JSON générée

- Pour respecter la séparation des préoccupations nous allons créer une classe se chargeant de la sérialisation
- Nous modifierons ensuite la Servlet GenererHeureFormatStandard afin qu'elle intègre ce mécanisme de sérialisation
- En dernier lieu nous devrons adapter le code Javascript afin de réceptionner et utiliser cette nouvelle valeur sous forme Json

Le classe de serialisation HeureCouranteJSONSerializer

```
public class HeureCouranteJSONSerializer {

    public void serialize(GenerateurHeureCourante generateurHeureCourante, OutputStream
outputStream) {
        final JsonObjectBuilder job = Json.createObjectBuilder();
        job.add("date", generateurHeureCourante.nowWithStandardFormat());
        Json.createWriter(outputStream).writeObject(job.build());
    }

}
```

Détaillons ce code

```
final JsonObjectBuilder job = Json.createObjectBuilder();
```

- La classe Json est une classe utilitaire. C'est le point d'entrée de toutes les opérations Json

```
job.add("date", generateurHeureCourante.nowWithStandardFormat());
```

- Permet d'ajouter un champ du nom date avec la valeur générée par la méthode nowWithStandardFormat

ISIMA

Java™

ORACLE

Détaillons ce code

```
Json.createWriter(outputStream).writeObject(job.build());
```

- Cette opération permet de copier dans l'outputStream passée en paramètre le flux Json désérliasé
- Voyons comment nous allons l'intégrer dans la Servlet ce serializer

ISIMA

Java™

ORACLE

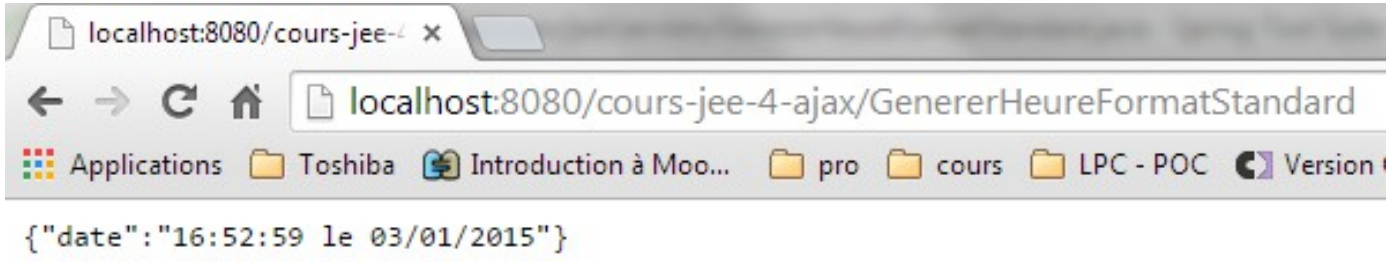
Notre GenererHeureFormatStandard

- Voyons le résultat de l'intégration de notre sérialisation dans la servlet

```
@WebServlet("/GenererHeureFormatStandard")
public class GenererHeureFormatStandard extends HttpServlet {


    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        final GenerateurHeureCourante generateurHeureCourante = new GenerateurHeureCourante();
        final HeureCouranteJSONSerializer serializer = new HeureCouranteJSONSerializer();
        serializer.serialize(generateurHeureCourante, response.getOutputStream());
    }
}
```

Voici le résultat dans Chrome

- Le flux générée est au format json



Les modifications en Javascript

```
function rafraichirHeureCourante() {
      $.ajax({
            url : 'GenererHeureFormatStandard',
            dataType : 'json',
            success : function(data, textStatus, jqXHR) {
                  $('#heureCourante').text(data.date);
            }
      });
      rafraichirDansUneSeconde();
}
```

- Nous avons simplement précisé un dataType afin que JQuery parse le Json
- Ensuite data contient notre structure : data.date permet d'accéder au champs

All right !

Approfondissons le Javascript

- Quelques conseils
- Mettez vos Javascript dans des fichiers js
- Cela vous permettra de structurer vos applications
- Pour rappel, l'import d'un javascript :

```
<script type="text/javascript"
    src="${pageContext.request.contextPath}/js/xxx.js"></script>
```


ISIMA Java ORACLE

Exemple nous allons modifier heureCourante

- Créons le script heure-courante.js

```
$(function() {
	RafraichirDansUneSeconde();
});

function rafraichirDansUneSeconde() {
	setTimeout(rafraichirHeureCourante, 1000);
}

function rafraichirHeureCourante() {
	$.ajax({
		url : '${pageContext.request.contextPath}/GenererHeureFormatStandard',
		dataType : 'json',
		success : function(data, textStatus, jqXHR) {
			$('#heureCourante').text(data.date);
		}
	});
	rafraichirDansUneSeconde();
}
```

Limites

- Les fichiers js ne sont pas parsés par le moteur JSP (et il ne le faut pas)
- L'expression `{pageContext.request.contextPath}/` ne fonctionne pas
- Nous utilisons donc un chemin relatif au domaine de la page
  - url : `'GenererHeureFormatStandard'`
- Il faut être prudent cela peut parfois provoquer des anomalies

Ca fonctionne encore

Terminons par quelques notions de Javascript

- Javascript est un langage objet de type « prototypale »
- Il est faiblement typé
- La portée des variables est fonction du contexte
- Il ne faut jamais oublié la présence de l'objet Window auxquels sont rattachées les variables

Quelques subtilités de Javascript

- Références de fonctions

```
function somme(a,b) { return a + b; }
function produit(a,b) { return a * b; }

function calculSurToutLeMonde(calcul, startValue, chiffres) {
     var result = startValue;
     for(var i = 0; i < chiffres.length; i++){
          result = calcul(chiffres[i], result);
     }
     return result;
};

//premier calcul : resultat 6
alert(calculSurToutLeMonde(somme, 0, new Array(1,2,3)));
//second calcul : resultat 24
alert(calculSurToutLeMonde(produit, 1, new Array(1,2,3,4)));
```

Quelques subtilités de Javascript

- Closures

```
function createGetter(something) {
    return function() {
        return something;
    }
}
var a = 50;

var func50 = createGetter(a);
a = 'a text';

var funcAText = createGetter(a);
alert(func50());
alert(funcAText());
```


ISIMA

Java™

ORACLE

Résumons

- Ajax
- Permet de rafraichir une page partiellement
- Est à l'initiative du client
- Nécessite une adaptation coté serveur
  - Il peut générer du HTML mais aussi du texte ou du JSON
- Nécessite d'implémenter coté client du code Javascript
- Nous l'utiliserons via la libraire JQuery