d'JET
CONSEIL
FORMATION
PROJETS

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# « Belles pages »
# (extraites du cours Symfony)

## D'Jet Conseil – Formation - Projet

d'JET
CONSEIL
FORMATION
PROJETS

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Héritage et inclusion de template

d'JET
CONSEIL
FORMATION
PROJETS

# Héritage de template

père

```
<html>
  <head>
  ...
  </head>
  <body>
  ...
  {% block nom %}

 ... « trou »

  {% endblock %}
  ...

  </body>
</html>
```

héritage

fils

```
{% extends ... %}
{% block title %} ...
{% endblocIk %}
{% block nom %}
Corps du bloc
...
{% endblock%}
```

fils ayant hérité

```
<html>
  <head>
  ...
  </head>
  <body>
  ...
Corps du bloc fils
...
...
...
...
...
...

  </body>
</html>
```

héritage

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

d'JET CONSEIL FORMATION PROJETS

# Exercice : définir un layout pour le projet Simple Stock

- Dans `layout.html.twig` on affiche
  - En haut de page
    - ✓ Le nom de l'application, le nom du stock définis en global dans …/app/parameters
    - ✓ Le nom de l'utilisateur courant et son statut définis en global dans …/app/parameters
  - En milieu de page un « trou » `{% block BLOCA %} … [% endblock %}`
  - En bas de page
    - ✓ « Propulsé par Symfony ( ou « Powered by Symfony »)
    - ✓ Auteurs
  - Fichier à placer dans …`/Resources/views`
- Deux héritiers (fichiers à modifier)

  `…/Default/index.html.twig, …/MonPremier/list.html.twig`
- Attention à la directive extends dans les fichiers héritiers

  `{% extends 'SYM16SimpleStockBundle::layout.html.twig' %}`

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Inclusion de template

- Le template que nous avons défini pour lister se voulait le plus générique possible pour pouvoir lister n'importe quoi. Exemple:
  - Le contenu du stock
    - ✓ Vis, écrous, rondelles etc.
  - La liste des statuts possibles dans le système
    - ✓ Administrateur, gestionnaire, super-utilisateur etc
  - Les utilisateurs connectés au système
    - ✓ gdupont, jdurand etc
- Avec cependant la possibilité de spécificités
  - `liststock.html.twig`, `liststatut.html.twig`, `listuser.html.twig` incluront `list.html.twig`
- Il suffit de placer une fonction include là où on le souhaite

```
{{include('SYM16SimpleStockBundle:MonPremier:list.html.twig')}}
```

- Effet : c'est comme si le code du template inclus était recopié à l'endroit où se trouve l'appel à la fonction `include`

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Inclusion de template

list.html.twig

```
<thead>
...
</thead>
<tbody>
...
...
</tbody
```

Exemple de template à inclure

listuser.html.twig

```
<html>
  <head
  ...
  </hea
  <body
  ... {
{{ incl
  ... {

  </bod
</html>
```

liststock.html.twig

```
<html>
  <head>
  ...
  </hea
  <body
  ... {
{{ incl
  ... {

  </bod
</html>
```

liststatut.html.twig

```
<html>
  <head>
  ...
  </head>
  <body>
  ... {# partie spécifique#}
{{ include(...)}}
  ... {# partie spécifique#}

  </body>
</html>
```

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Effet de l'inclusion

listuser.html.twig

```
<html>
  <head>
  ...
  </head>
  <body>
  ... {# partie spécifique
<thead>
...
</thead>
<tbody>
...
...
</tbody
  ... {# partie spécifique


  </body>
</html>
```

liststock.html.twig

```
<html>
  <head>
  ...
  </head>
  <body>
  ... {# partie spécifique#}
<thead>
...
</thead>
<tbody>
...
...
</tbody
  ... {# partie spécifique#}


  </body>
</html>
```

liststatut.html.twig

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

d'JET
CONSEIL
FORMATION
PROJETS

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Les formulaires

# Objectif d'un formulaire

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

- Un formulaire à pour objectif d'hydrater un objet.
  - A priori, tout objet peut être hydraté par un formulaire
  - En particulier une entité peut être hydratée par un formulaire
- Rappel : hydrater un objet c'est affecter des valeurs à tout ou partie des attributs de cet objet (en utilisant ses *setters*)
  - Par exemple l'entité Utilisateur peut être hydratée par un formulaire
  - Mais on peut aussi utiliser un formulaire pour hydrater un e-mail que l'on veut envoyer
- Le formulaire est lui-même un objet qui se crée à l'aide de la méthode `createFormBuilder` appliqué à une instance de l'objet que l'on veut hydrater
  - C'est une méthode de la classe `Controller` ce qui veut dire qu'un formulaire est crée dans un `fonction xxxxxAction(…)`
  - Le formulaire proprement dit est généré par la méthode `getForm()`

# Le formulaire pas à pas

```
// créer une instance de l'objet à hydrater
$utilisateur = new Utilisateur
// créer l'objet formulaire pour l'objet à hydrater
$formbuilder = $this->createFormBuilder($utilisateur);
// ajouter les attributs que l'on veut hydrater (pas
   l'id)
$formbuilder
   ->add('nom', text)    // on est pas obligé d'hydrater
   ->add('prenom', text)// tous les attributs de l'objet
   //etc.
// générer le formulaire
$form = formbuilder->getForm()
// afficher le formulaire et le passer à la vues
return $this->render(
   'SYM16SimpleStockBundle:Forms:simpleform.html.twig',
    array('form' => $form->CreateView() ));
```

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

d'JET CONSEIL FORMATION PROJETS

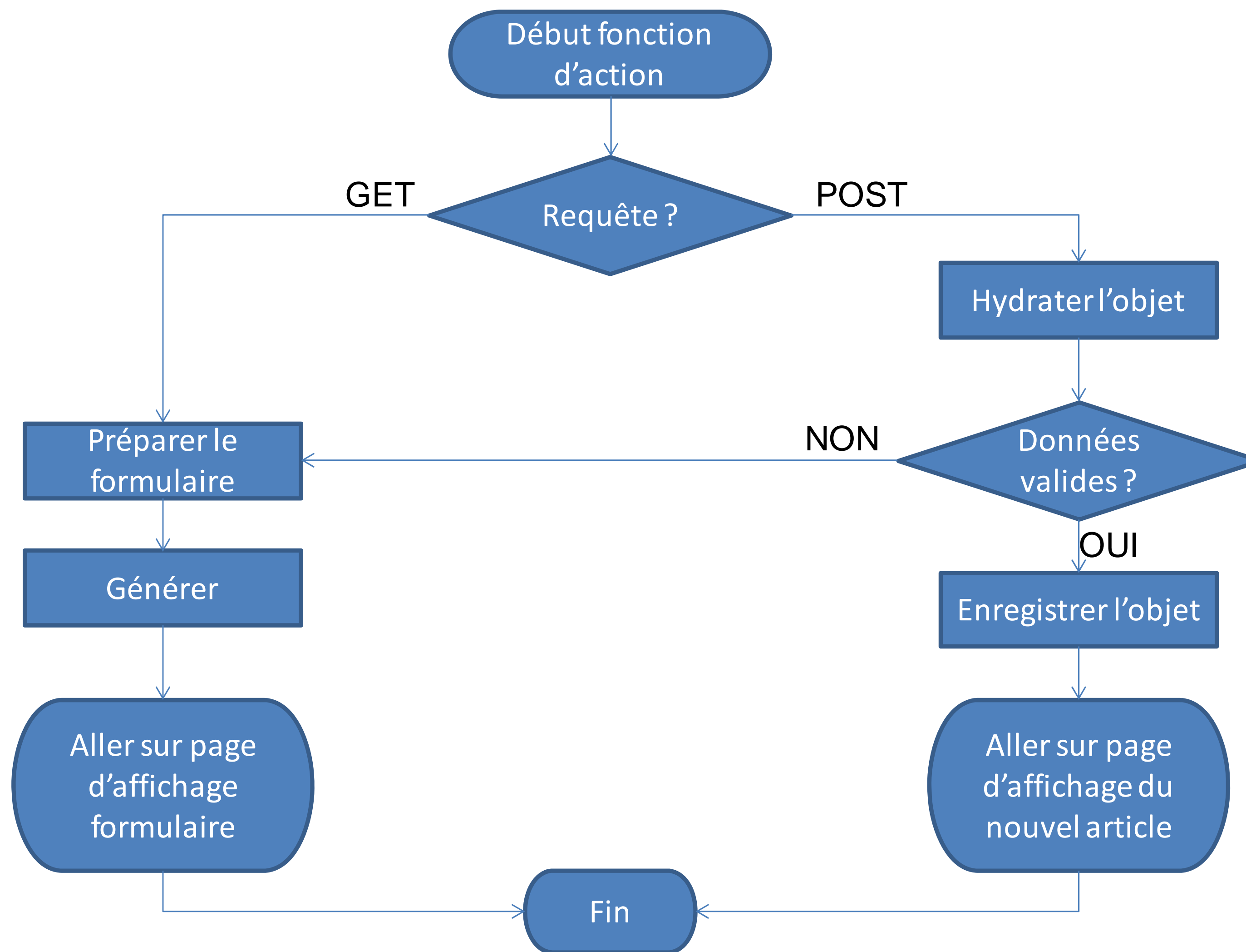
# Pas à pas (suite)

- A ce stade tout ce qu'on à fait c'est afficher le formulaire
  - Il faut à présent gérer sa soumission
- Ici noter une différence importante par rapport aux habitudes (PHP, HTML)
  - En PHP/HTML on a un fichier php dédié à l'affichage du formulaire et un autre fichier dédié à son traitement (récupération des valeurs etc)
  - Sous Symfony2 c'est la même fonction d'action qui fait les deux. Elle sera appelée deux fois
- Appels de la fonction d'action
  - La première fois elle est appelée par le routeur, suite à la soumission d'une URL (ex : …. `/app_dev/adduser`)
    - ✓ La requête utilisée est GET
  - La seconde fois elle est appelée par le formulaire, suite à sa validation (bouton Submit ou Validez)
    - ✓ La requête utilisée est POST
  - Dans la fonction d'action on va donc tester par quelle méthode de requête elle a été appelée

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Schéma d'exécution de la fonction d'action

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation



# Schéma d'exécution

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

```
// récupération de la requête
$request = $this->get('request');
// test de la méthode
if($request->getMethod() == 'GET'){
   // préparation, génération, affichage formulaire
   ...
else { // on est donc arrivé en ce point par POST
   // hydrater la variable $utilisateur
   $form->bind($request);
   // verifier la validité des valeurs d'entrée
   if($form->isValid()) {
        // enregistrer utilisateur dans la BDD
        ...
        // visualiser les résultats
        ...
   else
        // afficher de nouveau le formulaire
```

d'JET
CONSEIL
FORMATION
PROJETS

Etc...

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

d'JET
CONSEIL
FORMATION
PROJETS

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Listes liées

Où l'art de démêler DOM, JavaScript, AJAX, jQuery, JSON

…

# Définition

- Des listes déroulantes sont liées si
  - Le contenu d'un liste « esclave » dépend du choix fait dans une liste « maitre »
  - Exemple :
    - ✓ Liste maitre : liste des départements de France
    - ✓ Liste esclave : liste des communes du département sélectionné dans la liste maitre
- La liste esclave doit donc forcément être créée dynamiquement puisqu'on ne connait pas à l'avance le choix fait dans la liste maitre.
  - Problème non trivial sous Symfony

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Listes liées appliquées au projet Simple Stock)

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

- Quand on crée une référence, le formulaire doit nous proposer deux listes déroulantes
  - Une liste pour choisir un entrepôt parmi les tous entrepôts figurant dans la table `Entrepot`
  - Une liste pour choisir un emplacement de la table `Emplacement` uniquement parmi les emplacements liés à l'entrepôt choisi précédemment
- La liste déroulante `Emplacement` est donc liée à la liste déroulante `Entrepot`
  - Son contenu ne peut être connu qu'une fois le que le choix dans la liste `Entrepot` a été fait.
- Or on a besoin de la liste `Emplacement` avant de soumettre le formulaire au serveur
  - Il faut donc un mécanisme qui va se dérouler en parallèle, déclenché par le fait d'avoir fait un choix d'entrepôt : JS, AJAX, JSON, jQuery

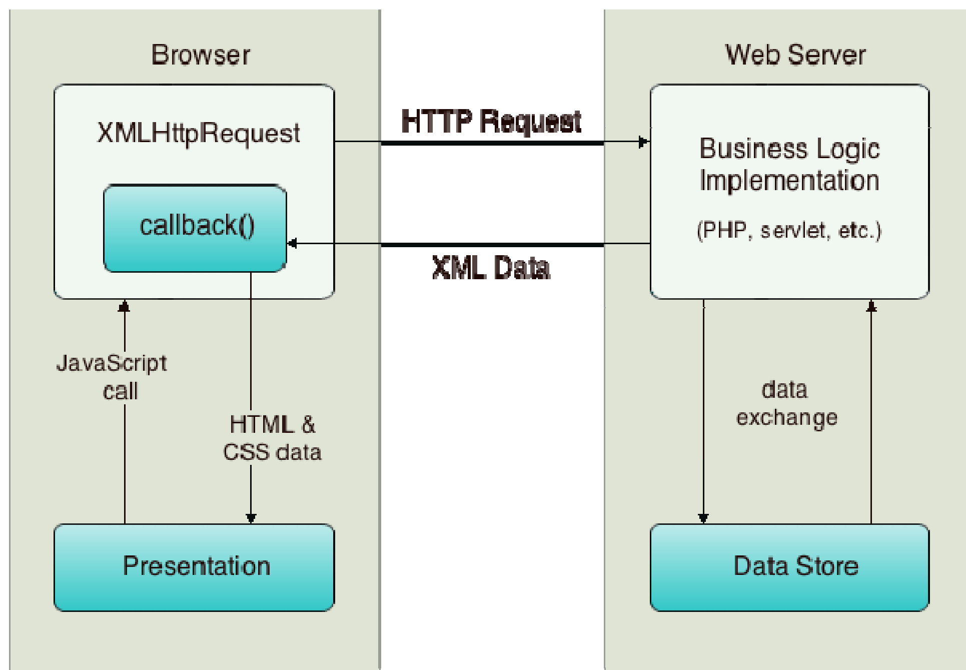
d'JET
CONSEIL
FORMATION
PROJETS

# Principe de résolution

1. On affiche le formulaire avec la liste déroulante `Entrepot`
2. L'internaute fait un choix dans cette liste
3. Cette sélection est envoyée par un script au serveur (tout en restant sur la même page)
4. Le serveur extrait de la base de données les emplacements liés à ce choix et les renvoie au script
5. Le script affiche la liste déroulante `Emplacement` restreinte aux choix possibles compte tenu du choix d'entrepôt
6. L'internaute fait le choix dans cette liste
7. L'internaute complète les autres champs du formulaire si nécessaire
8. L'internaute soumet le formulaire (bouton Envoyer)

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Comment ça marche ?




- Un événement généré par le formulaire appelle un scipt
- Côté serveur, le script construit un objet `XMLHttpRequest` contenant la requête au serveur
- Recevant la requête le serveur élabore une réponse et la renvoie dans un des formats possibles (XML, JSON, …)
- Ceci déclenche un callback() pour informer le script de récupérer les information et de les afficher.

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Le Document Object Model (DOM)

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

- Le DOM décrit une interface permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents XML (Source : Wikipédia)
- Le DOM est une représentation structuré (arborescente) d'un page

- Soit un élément d'un page (HTML)
- `<p id="paragraphe01" class="texte">Bonjour à tous</p>`
- L'instruction JS suivante permet d'accéder à cet élément
- `document.getElementById("paragraphe01");`
- L'instruction suivante permet d'agir sur lui en changeant sa couleur
- `document.getElementById("paragraphe01").style.color = "#ddcddc";`

# JavaScript (JS), Ajax, jQuery

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

- JS
  - C'est un langage de script exécuté côté client permettant d'accéder au DOM
- Ajax (Asychronous JavaScript and XML) est un regroupement de méthodes
  - HTML et CSS
  - JS pour accéder aux éléments du DOM
  - Une méthode pour échanger des données de façon asynchrone avec le serveur en utilisant l'objet `XMLHttpRequest`
  - Un format d'échange de données avec le navigateur (XML, HTML, JSON)
- jQuery est une bibliothèque de JS qui simplifie la manipulation du DOM, la gestion des événements et l'utilisation d'Ajax

# Json

- JSON (JavaScript Object Notation), est un format de données textuelle générique. Il permet de représenter de l'information structurée comme le permet XML par exemple [Wikipedia].

JSON

```
{ "menu":
  { "id": "file",
    "value": "File",
    "popup":
       { "menuitem": [
           { "value": "New",
   "onclick":      "CreateNewDoc()"
   },
           { "value": "Open",
   "onclick": "OpenDoc()" },
          { "value": "Close",
   "onclick": "CloseDoc()"
         }
       ]
     }
   }
}
```

HTML

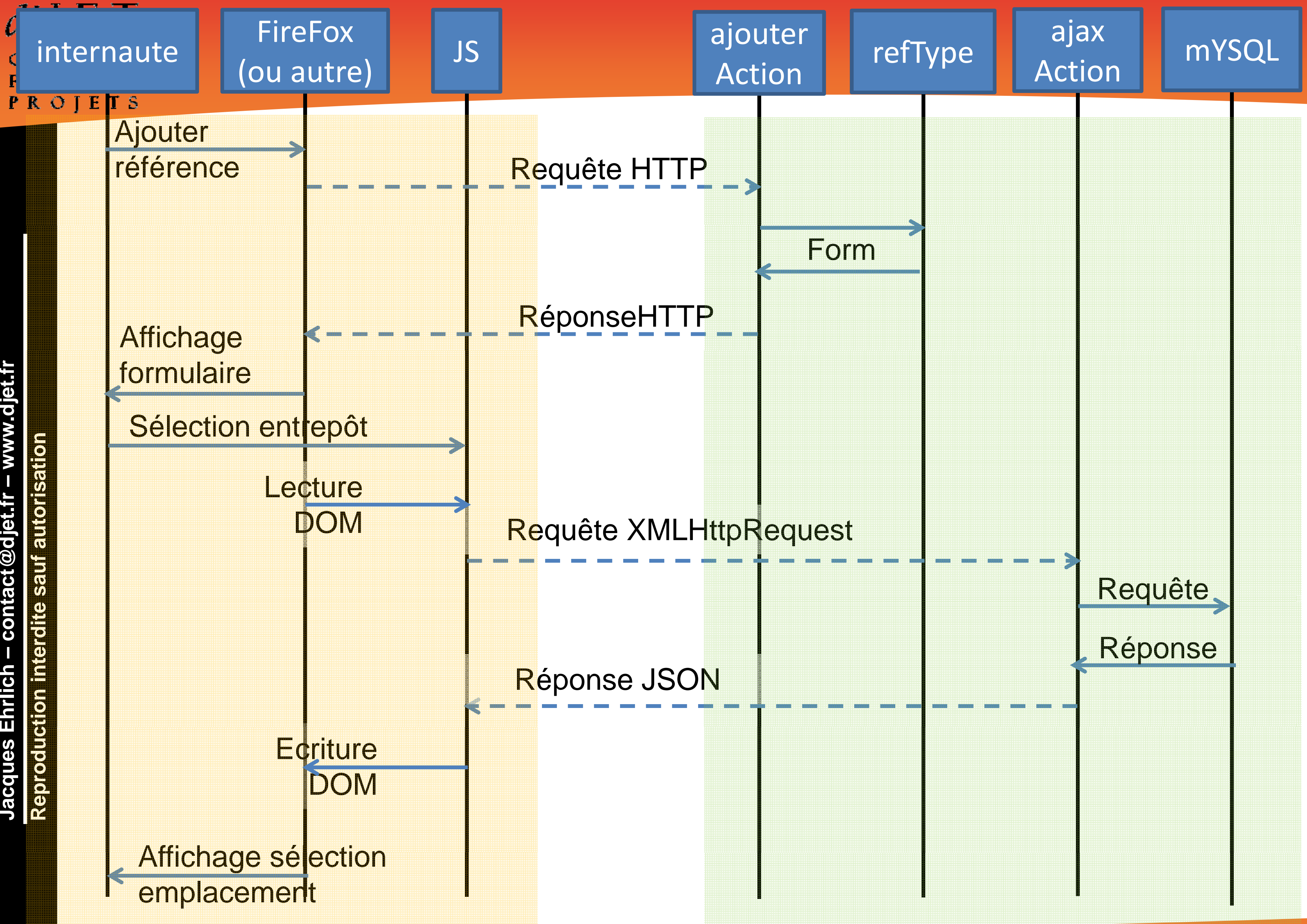
```
<menu id="file" value="File">
   <popup>
     <menuitem value="New"
   onclick="CreateNewDoc()" />
     <menuitem value="Open"
   onclick="OpenDoc()" />
     <menuitem value="Close"
   onclick="CloseDoc()" />
   </popup>
 </menu>
```

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

# Intégration dans Symfony

- L'action `ajouterAction()` de contrôleur `ReferenceController` affiche le formulaire `ReferenceType`
- L'internaute en selectionnant dans ce formulaire l'entrepot, déclenche l'exécution d'un script situé dans la vue :
  - `…/Resources/views/Reference/ajouter.html.twig`
- Le script récupère via le DOM du formulaire, l'info requise (le choix dans la liste Entrepot), construit l'objet `XMLHttpRequest` et l'envoie au serveur.
- Le serveur reçoit l'objet `XMLHttpRequest` qui contient notamment une url permettant d'appeler la fonction PHP qui va traiter la réponse
  - Il s'agit de `referenceAjaxAction(Request $request)`
- ainsi que le paramètre (le numéro d'id de l'entrepôt) passé par GET ou POST
- Le serveur traite la requête et répond au format JSON (ou un autre …)
- La réponse est récupérée pas le script
  - Ce sont les choix d'emplacements possibles
- Le script modifie la liste déroulante Emplacements via le DOM

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation


internaute
FireFox (ou autre)
JS
ajouter Action
refType
ajax Action
mYSQL
Ajouter référence
Requête HTTP
Form
RéponseHTTP
Affichage formulaire
Sélection entrepôt
Lecture DOM
Requête XMLHttpRequest
Requête
Réponse
Réponse JSON
Ecriture DOM
Affichage sélection emplacement
CLIENT
SERVEUR
Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

d'JET
CONSEIL
FORMATION
PROJETS

Jacques Ehrlich – contact@djet.fr – www.djet.fr
Reproduction interdite sauf autorisation

Copyright d'JET Conseil Formation Projet
Reproduction interdite sans autorisation