Les Framework Java

DWR

Claude Duvallet

Université du Havre
UFR Sciences et Techniques
25 rue Philippe Lebon - BP 540
76058 LE HAVRE CEDEX
Claude.Duvallet@gmail.com
http://litis.univ-lehavre.fr/~duvallet/

DWR

DWR : Direct Web Remoting (1/4)

- Il s'agit d'une librairie Java Open source destinée à aider les développeurs de site WEB pour inclure du Ajax.
- DWR permet d'utiliser des fonctions Java disponible sur un serveur WEB comme si elles faisaient parties du navigateur.
- DWR consiste en 2 partie principales :
  - une partie de code JavaScript pour récupérer des données depuis un serveur web basé sur des servlt en utilisant les principes d'AJAX.
  - une librairie JavaScript qui facilite le développeur de site WEB pour la mise à jour dynamique des pages WEB récupérant des données.
- DWR utilise une nouvelle approche basée sur Ajax en générant du code JavaScript basé sur des classes Java.
- Alors le développeur peut utiliser du code Java depuis du JavaScript comme si ce code était local au navigateur.

DWR : Direct Web Remoting (2/4)

- En réalité le code Java s'exécute sur un serveur WEB et possède donc un accès complet aux ressources du serveur.
- Pour des raisons de sécurité, le développeur WEB doit configurer précisément quelles sont les classes qui peuvent être exportés en tout sécurité (ce qui est souvent fait dans le fichier web.xml ou dwr.xml).
- Cette méthode d'appel des fonction distantes depuis Java vers JavaScript offre aux utilisateurs de DWR un mécanisme proche des classiques RPC tels que RMI ou SOAP tout en ayant le bénéfice que cela fonctionne sur le WEB sans nécessiter l'ajout de plugin au sein du navigateur.
- DWR ne considère pas que le protocole entre le navigateur et le serveurs WEB comme important.
- Il préfère offrir une interface de développement beaucoup plus naturelle.

DWR : Direct Web Remoting (3/4)

- Le plus grand défi est de marrier la nature asynchrone d'AJAX avec la nature synchrone des appels de méthodes Java.
- Dans le modèle asynchrone, les données sont uniquement disponibles quelques temps après que l'appel initial ait été fait.
- DWR résoud ce problème en permettant aux développeurs WEB de spécifier une fonction qui sera appelé lorsque les données retournées utilise un paramètre de méthode externe.
- Cette méthode externe est appelée CallBack Method.
- Voici un exemple de méthode Callback :

```
MJavaClassOnJs.getListProducts(selectedCategory,{
 callback:function(returnedList){
   dwr.util.addOptions(myComboId,returnedList,"productId","productName")
 }
})
```

- La méthode Callback est une fonction à l'intérieur d'un objet Json passé comme paramètre aditionnel de la méthode disante.

DWR : Direct Web Remoting (4/4)

- Dans la version 2.0 de DWR, le support de Reverse Ajax a été ajouté et permet à du code Java s'éxécutant sur le serveur d'envoyer du JavaScript au navigateur.
- Le projet DWR a été créée par Joe Walker en 2004.

DWR : En résumé

- DWR est un framework AJAX pour Java qui permet de "publier" des méthodes Java en Javascript.
- Ces méthodes seront ensuite manipulables dans la page HTML.
- L'utilisation de DWR autorise ainsi la création d'applications Web 2.0 complètes tout en minimisant le code Javascript nécessaire et en apportant la puissance, la souplesse et la robustesse de Java.
- L'ensemble des échanges avec le serveur par XMLHttpRequest sont gérés de manière interne à DWR ce qui dispense le développeur de la résolution de ces problématiques.
- En outre, DWR repose sur le framework Javascript Prototype et offre certaines fonctionnalités et API Javascript permettant la résolution de la majorité des problématiques de compatibilités entre navigateurs.
- DWR fonctionne sur n'importe quel projet Web Java.

L'architecture logicielle : la couche DWR

- La séparation en couches d'une application favorise, entre autres, sa robustesse et son évolutivité.
- Une nouvelle couche applicative a été créée : elle est située dans la couche front, au dessus de la couche service.
- Cette couche a pour tâche de faire l'interface entre le client, la page html et la couche service.
- Cette couche - appelée DWR - accède, contrairement à la couche service à la requête http.
- Cela permet de récupérer des informations telles que la locale de l'utilisateur, son identité, de mettre en forme les formulaires postés, etc.
- Cette couche est réservée à une utilisation dans une application web alors que la couche service peut être publiée à travers des webServices, accédée depuis un client lourd, etc.

Principe de fonctionnement de DWR

- La couche DWR, gérée également par Spring, publie un certain nombre de méthodes java en Javascript.
- Toute classe, définie dans la couche DWR et correctement configurée, publiera les méthodes qui la composent.
- Le moteur DWR est chargé de créer l'interface Javascript permettant d'appeler ces méthodes en AJAX, via XMLHttpRequest.
- Ces aspects sont totalement masqués par le framework DWR.
- DWR transforme aussi des objets java en Javascript.
- Les méthodes publiées retournent des objets java que DWR convertit en Javascript si il y est autorisé explicitement via des déclarations de Converters.
- Ces objets pourront être récupérés en Javascript, après les appels de méthodes et affichés dans la page html.

Un premier exemple (1/4)

1. Créer un objet JavaScript reprenant la structure de données de votre formulaire, le formulaire ici est une page de login :

```
<script type="text/javascript">
var loginForm = {
    login:null,
    password:null
};
</script>
```

2. Ecrire l'équivalent côté Java :

```
public class LoginForm {
    private String login;
    private String password;

    // getters and setters
}
```


Un premier exemple (2/4)

3. Ecrire la classe DWR qui va permettre de valider l'identification de l'utilisateur :

```
import java.util.ArrayList;
import java.util.List;

public class LoginDWR {
    /**
    * Méthode asynchrone permettant de valider l'authentification
    */
    public List validateLogin(LoginForm loginForm, HttpServletRequest request) {
        List errorList = new ArrayList();

        // validation du login
        if (loginForm.getLogin() == null) {
            errorList.add('Your login is required');
        } else {
            if (loginForm.getLogin().length() < 4) {
                errorList.add('Your login must be at least 4 characters');
            } else if (loginForm.getLogin().length() > 10) {
                errorList.add('Your login can not exceed 10 characters');
            }
        }
        ...
```

La suite est à télécharger sur ma page WEB

Un premier exemple (3/4)

4. Exemple de page JSP qui contiendrait ce formulaire et mise en place de la validation asynchrone : télécharger l'exemple sur ma page WEB.

- Explications
  - Le bouton du formulaire appel la méthode javascript `validLogin()`.
  - La méthode `dwr.util.setValue()` est utilisée pour réinitialiser les messages d'erreurs précédents.
  - L'appel à la méthode `dwr.util.getValues()` permet de stocker les informations saisies du formulaire dans l'objet javascript `loginForm`, cette méthode se base sur les identifiants des champs du formulaire pour les mapper dans l'objet donné en paramètre, ici `loginForm`.
  - L'objet DWR est appelé avec la méthode `validateLogin()` en lui donnant en paramètre notre objet `loginForm` pré-rempli, de plus nous spécifions la méthode de retour (callback).

Un premier exemple (4/4)

- Suite des explications
  - La méthode de retour est exécutée automatiquement quand DWR aura terminé l'exécution de sa méthode (return ...).
  - Nous validons ensuite le formulaire récupéré côté serveur dans la méthode DWR pour faire un contrôle de surface tout d'abord puis un contrôle en profondeur en appelant nos services métier.
  - Une liste d'erreurs est finalement retournée au navigateur ou une liste vide si tout est valide.
  - Nous manipulons ensuite légèrement le DOM pour afficher les messages d'erreurs ou de succès avec `dwr.util.setValue()`