License 3 informatique

Compte Rendu

Projet

Technologie orientée objet

Et

Conception des applications internet

Réalisé par :

Cazaux Axel et Dymko Frederic

Sommaire :

I - Introduction

Ce projet résulte de la conception d'une interface homme-machine pouvant renseigner à l'utilisateurs les différents DNS records d'un site donné par lui-même.

Pour cela, nous avons donc suivi la consigne d'utiliser une communication entre java (utilisation de java 8 durant la conception) et javascript grâce aux technologies JNDI et Websockets.

II - Programmation Java

1 – Le javax.websocket.Session

Nous allons utiliser ici un serveur glassfish (donné dans le cours et sous version 1.15) et donc l'utilisation d'endpoint pour la communication.

Nou aurons donc une session avec les méthodes : onClose(), onOpen(), onError() et onMessage().

La méthode onClose() ne fermera que la session et affichera l'id de cette dite session.

La méthode onError() fera les gestions des erreurs en écrivant un fichier JSON avec les erreurs faites.

La méthode onOpen() va envoyer le tableau des suffixes générer par le JNDI_DNS à la partie javascript pour qu'il puisse être traité directement en javascript.

La méthode onMessage() est un évènement qui récupère les informations depuis une url et qui va renvoyer les données nécessaires grâce au JNDI_DNS sous forme Json.

2 – Class JNDI

Pour le JNDI, on utilise le DNS de google pour avoir plus de précision. Notre JNDI va enregistrer dans un tableau tous les suffixes des url « existants ».

Pour cela, on a décidé d'utiliser la bibliothèque org.json.jsonobject qui permet par rapport à celle de google de ne pas créer de builder. En effet, ici l'on peut créer un objet Json et l'utiliser en même temps.

III – Programmation Javascript

1 – Choix de la structure

Pour coder notre application en javascript, nous nous sommes inspirés du model SAM que nous avions étudier.

Nous utilisons donc par connaissance cette architecture de code et nous l'avons un peu modifié pour simplifier le code et aussi car nous n'avons pas besoin ici d'action.js.

2 – Gestion homme-machine

Avec l'utilisation d'endpoint, il en faut du même type en javascript pour pouvoir communiquer entre serveur et client. Les clients et le serveur communique à l'aide de texte au format JSON (JavaScript Object Notation). La transition de données de javascript à java se fait sous ce format :

```
{
    "type": String,
    "data": {
        "url": String,
        "dns": String
    }
}
```

Les messages envoyés de java à javascript sont de ce format :

```
{
    "type": String,
    "succeed": Boolean,
    "url": String,
    "data": {
        //...
    }
}
```

Après la saisie d'une url de l'utilisateur, afin de savoir s'il faut l'envoyer au serveur java, nous effectuons un post-traitement grâce à une regex sur la chaine de caractère saisie par l'utilisateur et vérifions aussi que le suffixe entré existe bel et bien en parcourant la liste reçue par le serveur à la connexion.

3 – Affichage de l'application

Nous avons utilisé l'outil bootstrap pour faire l'affichage de l'application pour avoir un rendu plus épuré et plus malléable.

L'utilisateur n'aura qu'à taper une url existante (une erreur ne sera pas traitée) et toutes les informations traitées par le JNDI (tous les records enregistrés) seront affichées à l'écran.

IV - Conclusion

Notre application WhoIs permet donc de voir toutes les informations non protégées d'une url donné par l'utilisateur et elle se sert donc de la technologie JNDI pour faire la recherche des records et la traité grâce au serveur glassfish qui permet la communication entre java et javascript.

Voici donc qui conclut ce compte rendu, nous vous remercions de l'avoir lu.