David Kunzmann & Miguel Santamaria - 19.06.2016

Introduction

Ce document fait office de rapport pour le projet qui a été réalisé durant le cours de Scala de 3ème année de l'HEIG-VD en filière TIC.
Nous avons décidé de réaliser un projet de gestion de jobs de vacances, à l'aide d'une application Web implémentant un back-end (Play), un front-end (Scala.js) et une base de données (Slick). Dans cette application, il est possible de rechercher et de postuler pour des jobs, ou même d'en proposer un ! Le site contient 3 pages : la page d'accueil / de recherche, la page de détails d'un job et la page d'ajout d'un job.
Le travail a été séparé en deux : David s'occupe du back-end et de la base de données, tandis que Miguel s'occupe du front-end tout en travaillant avec certaines actions du back-end.
Pour utiliser le programme, il faut tout d'abord installer Play Framework et Activator (https://www.playframework.com/documentation/2.5.x/Installing), puis ensuite se rendre à la racine du dossier HolyJobs, et taper `activator run`.

Scala.js

Introduction

Scala.js (https://www.scala-js.org/) est, comme son nom pourrait le présager, une bibliothèque permettant d'écrire du Scala à la place du JavaScript dans la partie front-end d'un site, permettant ainsi d'avoir un

langage typé offrant donc plus de sécurité et plus de contrôle, ainsi que d'avoir une approche plus "fonctionnelle" du code. Cette technologie est encore en version Alpha.

Motivations

A la base, nous souhaitions utiliser Scala.js car cette technologie a été suggérée par les responsables du cours, et aussi par curiosité personnelle.
Nous nous sommes malheureusement rapidement heurté à de nombreux obstacles qui nous on fait perdre un précieux temps. Après discussion avec les responsables, il a été décidé que Scala.js serait retiré du projet et remplacé par du JavaScript, pour des raisons qui seront énoncées plus bas.

Mise en place

Afin de nous faciliter la vie, nous nous sommes basés sur l'exemple donné dans le cours (https://github.com/vmunier/play-with-scalajs-example), que nous avons adapté pour générer notre projet. Les premiers problèmes sont apparus lorsqu'il a fallu installer et configurer Slick dans ce projet (les explications sont disponibles dans le chapitre réservé à Slick).

Analyse de la librairie

Après avoir suivi le tutoriel fourni sur le site de la librairie sans trop de mal, nous avons tenté de l'utiliser dans l'application.
Comme dit plus haut, nous nous sommes très rapidement heurté à un

mur : nous souhaitions en effet utiliser la version JQuery de Scala.js pour réaliser quelques animations (scrolling de la page d'accueil, apparition des détails liés à un job lors du survol de la souris sur l'élément, etc.), mais la documentation fournie (https://www.scala-js.org/api/scalajs-jquery/0.8/#org.scalajs.jquery.JQuery) ne contient que les signatures des fonctions, sans autre explication. C'est après quelques heures de combat que nous avons décidé d'abandonner son utilisation au profil de celle de JS.
La librairie étant en effet actuellement en version Alpha, elle est encore malheureusement trop "nouvelle" et cela se fait ressentir... Elle est tout d'abord très peu documentée (outre la "documentation" de JQuery et de JavaScript, il existe quelques tutoriels et 2-3 exemples basiques par-ci, par-là) et possède une communauté actuellement restreinte. En résumé, elle est donc actuellement utilisable pour des choses dites "basiques" (événements sur des composants du DOM, changements simplistes du DOM, etc.), mais devient très barbante dès que nous souhaitons réaliser des choses plus complexes.

Slick

L'accès ainsi que les requêtes à la base de donnée du projet HolyJobs se font à l'aide de Slick. Nous avons pu tester Slick avec deux différents type de base de donnée: une base de donnée mémoire et une base de donnée persistante.

Mise en place

Beaucoup de problèmes sont survenu lors de la mise en place de Slick. Ceci en partie dû au template de projet pour Scala.js que nous utilisions.

En effet ce dernier offrant beaucoup de possibilités rendais le debbuging plus difficile. Une grande quantité d'options offerte par le template ne nous était d'aucune utilité. Typiquement un projet shared permettant l'échange de donnée entre le backend et le frontend s'est révélé inutile. De plus le template lui-même n'était plus utile lorsque nous avons retiré Scala.js du projet. Nous sommes donc reparti sur une base propre et un projet vide.

Base de données

Lors de la mise en place du projet nous avons choisi de mettre en place Slick avec une base de donnée H2 ceci nous a permis de vérifier que notre projet était foncionnel, mais nous avons églament pu nous familiariser avec Slick et son mode de fonctionnement.

Par la suite nous avons utilisé une base de donnée MySQL persistante. Ce changement implique très peu de modification de notre code. Mis à part l'import du driver MySQL ainsi que la modification de configuration de la base de donnée dans le fichier application.conf, nous avons du modifier des détails tel que le nom des colonnes de la base de donnée(qui doivent être en minuscule). En effet H2 est peu regardant sur la nomenclature mais ceci n'est pas le cas pour MySQL.

Utilisation

L'utilisation de Slick est très intuitive. Mais il faut pour débuter correctement mettre en place les différentes classe qui représenteront les tables de notre base de donnée.

Chaque classe de l'application (Job, Region et Type) doivent être des case class. Ensuite la représentation de cette classe pour la base de

donnée se fait à l'aide d'une classe qui hérite de la classe Table. Cette dernière prend en paramètre un objet Tag qui est le nom de la table dans le base de donnée.

Une fois Slick mis en place les query se font facilement avec leNomDeLaTable.result retournant toutes les lignes de la table sous la forme d'une Sequence. Et pour appliquer un WHERE MySQL il suffit de filtrer le résultat. Une fonctionnalité très agréable de Slick, que nous avons utiliser lors de l'insertion d'un nouveau Job dans la base de donnée, est le **returning**. Ceci permet de retourner une valeur pour nous l'id du Job inséré. Ceci est très pratique car l'id étant généré par le base de donnée nous ne pouvons la connaître avant l'insertion.

Conclusions

Etat des lieux

A l'heure actuelle, le projet n'est pas terminé à 100% (il reste en effet à réaliser les champs de recherche de jobs, ainsi que la pagination), mais nous restons positifs quand à son bon déroulement et quand au suivi de ce qui avait été prévu. Nous arriverons en effet - sauf en cas de tremblement de terre ou autre inconvénients de ce genre - à terminer le projet dans les temps et seront donc prêts pour la présentation.

Conclusion

Ce projet nous a permi de nous exercer dans l'utilisation des technologies Web utilisant Scala, comme le Framework Play, Scala.js et Slick. Nous sommes très enthousiastes quand à Play qui nous a paru être un bon

choix de back-end, ainsi que pour Slick qui offre une nouvelle approche de communication avec la DB, mais sommes au contraires beaucoup moins convaincus par Scala.js...
Il nous a été difficile de mener à bien ce projet, tant nous étions surchargés en fin de semestres, mais sommes heureux d'y être tout de même arrivés.
Nous jugeons notre travail comme étant bon, bien que pas parfait.