Documentation du projet

MiniCyberCoach

Lorenzo Clementi, Gionatan Laveglia

Janvier 2007

dans le cadre du cours

Advanced Software Engineering
SH 2006/07

Software Engineering Group
Département d'Informatique
Université de Fribourg

1 Table des matières

2 Introduction

Le projet MiniCyberCoach a été réalisé pendant le semestre d'hiver 2006/07 dans le cadre du cours de Master *Advanced Software Engineering* à l'Université de Fribourg (Suisse) sous la supervision du prof. Jacques Pasquier-Rocha et du Dr. Patrick Fuhrer.

Le cours introduit les concepts fondamentaux des architectures multi-tiers. Ces architectures prévoient une séparation (éventuellement seulement conceptuelle) du système en trois couches:

- Couche de présentation (*Client/Presentation tier*)
- Couche logique (*Middle/Business tier*)
- Base de données (*Database tier*)

La couche de présentation a comme fonction principale la gestion de l'interaction avec les utilisateurs et l'affichage des résultats produits par la couche logique. Cette deuxième, par contre, s'occupe d'effectuer la logique du travail, en produisant les résultats à partir des données qui sont mises à disposition par la troisième couche du système, c'est-à-dire la couche des bases de données.

Les trois couches sont indépendantes et elles communiquent à travers d'interfaces partagées (Figure 1).

Figure 1 : Les trois composantes d'une architecture 3 tièrs

L'un des principaux avantages de ce type d'architecture est sa modularité qui permet aux différents éléments du système d'évoluer de façon (presque) indépendante. De plus, dans le processus de développement d'un tel système, la séparation du travail résulte plus naturelle et permet aux développeurs de pouvoir se concentrer sur les parties du système qu'ils maîtrisent le mieux.

Le cours *Advanced Software Engineering* a été composé d'une première partie théorique dans laquelle les fondements théoriques des architectures multi-tiers ont été abordés.

Par la suite, en parallèle avec l'avancement du cours, le projet MiniCyberCoach a démarré permettant aux étudiants de se confronter dans la pratique avec les notions apprises dans le cadre du cours.

Dans le prochain chapitre nous allons décrire de façon plus détaillée le travail accompli par notre groupe pour la réalisation du projet MiniCyberCoach, dans le dernier chapitre nous présenterons nos conclusions accompagnés d'une évaluation du projet dans sa totalité.

3 Organisation et réalisation du projet

Le projet MiniCyberCoach a pour but la réalisation d'un outil pour permettre aux athlètes d'enregistrer, visualiser et analyser les données relatives aux entrainements effectués (par exemple la distance parcourue lors d'une séance d'entrainement, ou le taux de battements cardiaques moyen) afin de pouvoir mieux gérer ses propres prestations physiques.

La réalisation du système doit se baser sur une architecture trois tiers dans laquelle la couche logique (*business tier*) doit respecter un paquet de 13 tests unitaires.

En ce qui concerne la couche de présentation des données, par contre, une plus grande liberté est laissée aux groupes d'étudiants: il est possible de réaliser un client web, un *rich client* ou même d'autre types de clients (par exemple, en utilisant les fonctionnalités du *business tier* via des services web). Dans le cadre de ce projet, la couche base de données a une moindre importance : le sujet central est constitué plutôt par la réalisation des couches business + web et de leur coordination.

3.1 Technologies utilisées

MiniCyberCoach a été réalisé avec Java EE (Enterprise Edition); l'IDE utilisé est NetBeans 5.5 en combinaison avec le serveur d'application de Sun Microsystems[1]. Nous avons aussi installé une base de données MySQL.

La version 5 du Java Software Development Kit (et donc de Java EE) a été publiée très récemment (au cours de l'an 2006). Cette nouvelle version a apporté plusieurs modifications au langage: en ce qui concerne la version entreprise, la modification la plus importante a été l'introduction d'une nouvelle spécification pour les Enterprise JavaBeans (EJB) qui constitue une évolution majeure par rapport à l'ancienne version 2.1 (http://java.sun.com/products/ejb/).

3.2 Structuration du projet et répartition du travail

À partir du diagramme de classes (Figure 2) nous avons implémenté les entités en délégant la génération des tables au serveur d'application. La réalisation de la base de données a donc été incluse dans la réalisation de la couche logique du système.

1 Pour plus de détails concernant la plateforme de réalisation, consulter l'annexe.

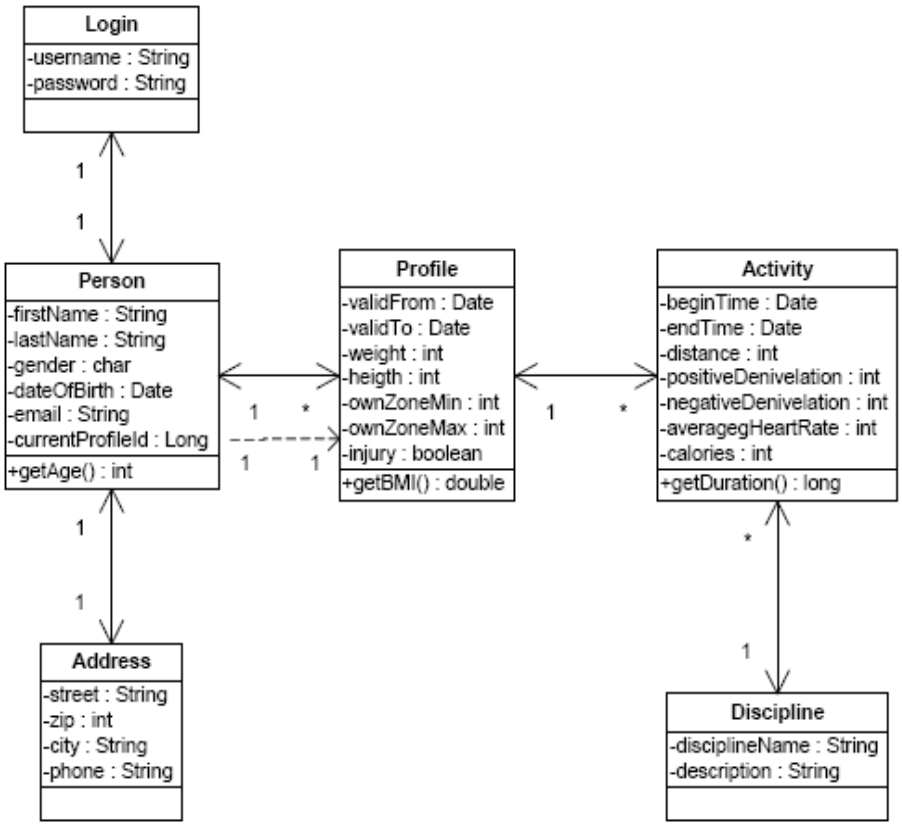


Figure 2 : Diagramme de classes pour les entités

Nous avons séparé la couche présentation et la couché logique en deux différents projets de NetBeans regroupés dans un seul projet *Enterprise application*:

- un projet *EJB Module* pour la couche business
- un projet *Web application* pour la partie web

Ceci permet de développer, déployer et tester ces deux projets de façon indépendante. En effet la réalisation d'un système basé sur une architecture trois tiers permet de séparer les différentes parties du travail, ce qui permet d'avancer en parallèle dans le développement des deux parties tant que les interfaces communes sont respectées.

Néanmoins, dans le cadre de ce projet, il nous a semblé plus intéressant de travailler ensemble dans le développement des différentes parties du projet afin de pouvoir nous familiariser avec toutes les technologies impliquées (qui étaient pour nous des nouveautés) et pouvoir suivre l'avancement du projet dans toutes ses phases.

3.3 Couche base de données

Les "éléments" à la base de MiniCyberCoach sont les six entités suivantes:

- *Person*: données personnelles de l'utilisateur
- *Login*: nom d'utilisateur et mot de passe permettant d'accéder au système
- *Address:* adresse postale et électronique
- *Activity*: données relatives à une séance d'entrainement
- *Discipline*: discipline sportive (par exemple, natation) pour la catégorisation des activités
- *Profil*: données relatives à la condition physique d'une personne

Dans la couche business, ces entités sont modélisées par des ***entity***, des classes Java correspondant à une table de la base de données. Dans le cadre de ce projet on a laisse que le serveur d'application génère automatiquement les tables dans la base de données au moment du déploiement.
Dans le cas du développement d'une application au sein d'une entreprise, il se pourrait que ceci ne soit pas la meilleure solution et que la gestion de la base de données soit faite par un spécialiste qui pourrait ainsi en optimiser le design et les performances.

Les clés primaires de chaque table sont aussi générées automatiquement, sauf dans certains cas particuliers dans lesquels nous les définissons directement. Ceci est nécessaire car, pour pouvoir exécuter les tests unitaires, on doit pourvoir contrôler l'état de la base de données sans devoir passer par une interface web.

Les relations entre les tables (i.e. les clés étrangères au niveau de la base de données) peuvent être réalisées grâce à la méthode des annotations.

3.4 Couche logique

La spécification pour la couche logique du projet MiniCyberCoach a été définie à travers 13 *use cases* et les tests unitaires associés.

Dans le paragraphe précédent, nous avons vu comment les entités (qui *"vivent"* dans le *container* des EJB) génèrent les tables de la base de données. Dans le *business tier*, à chaque entité est associé un ***stateless session bean***, c'est à dire un objet particulier qui contient des méthodes implémentant la logique métier de l'application.

Pour chaque entité définie dans notre projet nous avons implementé un *session bean*, nous avons donc les *beans* suivants:

- AddressFacade.java
- LoginFacade.java
- PersonFacade.java
- ActivityFacade.java
- ProfileFacade.java
- DisciplineFacade.java

Chaque *bean* a accès aux méthodes nécessaires pour créer, parcourir ou détruire des entités. Les méthodes implémentées dans chaque *bean*, exploitent ces fonctionnalités qui sont réalisées automatiquement par le IDE.

Pour que les *beans* puissent avoir des références entre eux mêmes, il faut aussi définir une interface locale pour chacun d'entre eux. Ces interfaces locales permettent en effet à un certain *bean* d'appeler des méthodes implémentées dans un autre *bean* se trouvant dans le même *container*.

La structure de la couche logique est complétée par trois ultérieurs *stateless session beans* permettant de réaliser le *business delegate pattern*; il s'agit des *session beans* suivants:

- PersonManagementSessionBean.java
- ActivityInformationSessionBean.java
- AdminSessionBean.java

À chacun de ces *beans* est associée une interface *remote* offrant les méthodes pouvant être appelées depuis l'extérieur du *container* (c'est-à-dire, par exemple, par la couche présentation du système).

L'utilisation de ces interfaces n'est pas obligatoire et il serait possible d'ajouter une interface remote directement aux session beans. Cependant, leur utilisation est un pattern connu qui permet de mieux structurer la couche *business* de l'application.

3.5 Couche de présentation

Nous avons décidé de réaliser un client web basé sur les technologies *Java Server Faces* (JSF) et *Java Server Pages* (JSP).

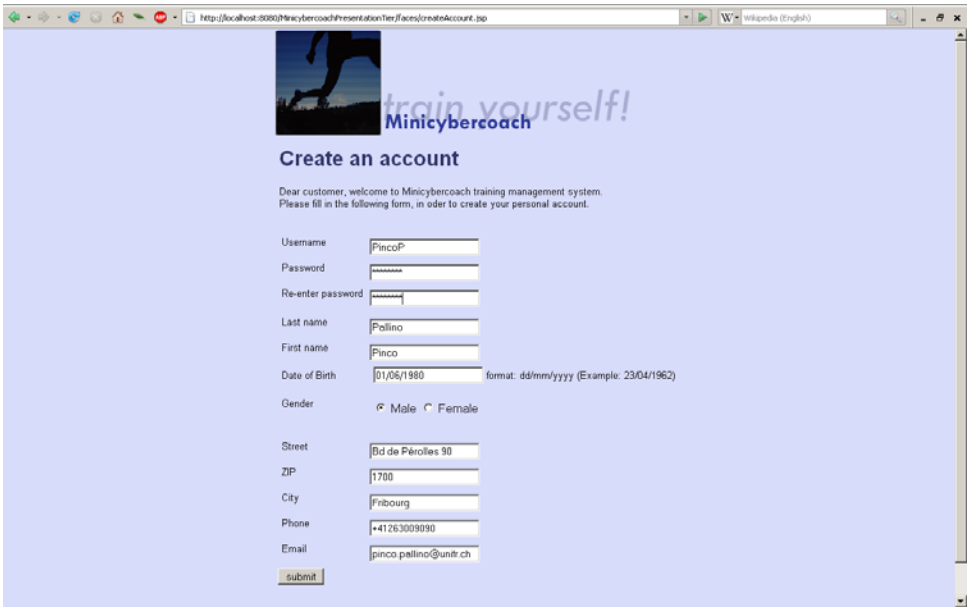
Cette technologie permet d'effectuer une séparation entre affichage et traitement logique. Les fichiers .jsp visualisés dans le navigateur web définissent la structure graphique des pages. À chacune de ces pages .jsp correspond une classe java (appelée ***backing bean***) qui s'occupe de l'accomplissement des tâches logiques nécessaire au traitement des données en provenance du *business tier*.

Nous nous sommes aperçus que la familiarisation avec ces technologies n'était pas si intuitive qu'on pouvait le croire. En effet la réalisation de la couche de présentation nous a requis plus de temps que ce qu'on avait estimé. Nous avons donc décidé de limiter le nombre de fonctionnalités offertes tout en cherchant de couvrir les différents types de traitements que le système doit assurer.

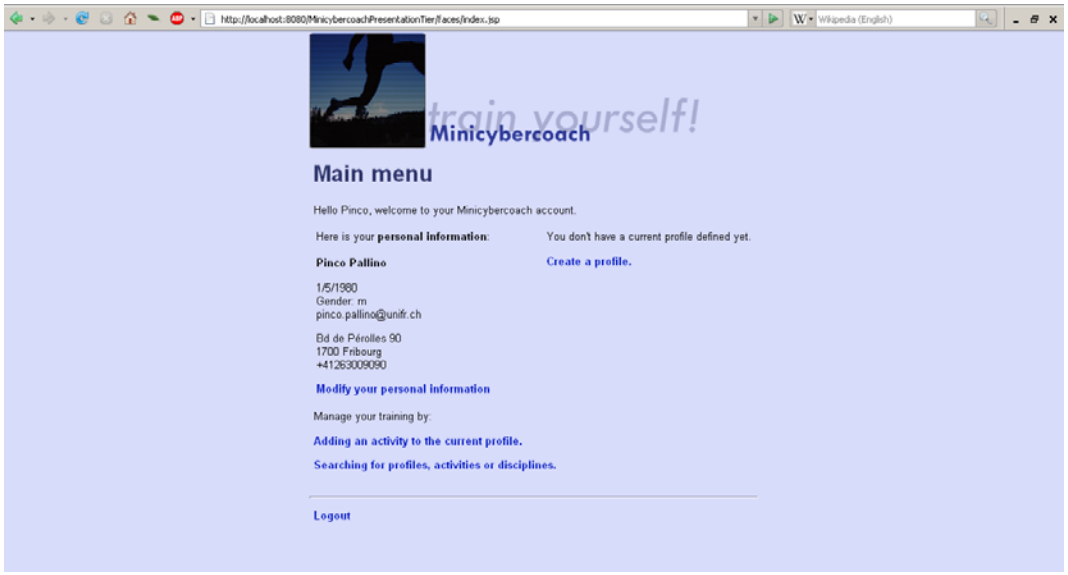
Les fonctionnalités offertes par la couche de présentation sont:

- Créer un nouvel utilisateur
- Procédure de login et logout
- Visualisation et modification des données personnelles
- Création, visualisation et modification des profiles
- Recherche simple de profiles ou activités

Voici quelques screenshots de notre application web:




http://localhost:8080/MinicybercoachPresentationTier/faces/createAccount.jsp
train yourself!
Minicybercoach
Welcome
Your account has been created.
Please log into the system.
Username: PincoP
Password:
Login
or create a new account
University of Fribourg > Software Engineering Group
Gionatan Laveglia, Lorenzo Clementi, 2007


http://localhost:8080/MinicybercoachPresentationTier/faces/index.jsp
train yourself!
Minicybercoach
Main menu
Hello Pinco, welcome to your Minicybercoach account.
Here is your personal information:
Pinco Pallino
1/5/1980
Gender: m
pinco.pallino@unifr.ch
Bd de Pérolles 90
1700 Fribourg
+41263009090
Modify your personal information
You don't have a current profile defined yet.
Create a profile.
Manage your training by:
Adding an activity to the current profile.
Searching for profiles, activities or disciplines.
Logout


http://localhost:8080/MinicybercoachPresentationTier/faces/search.jsp
train yourself!
Minicybercoach
Search page
Search for Profiles
From 2003-12-25 12:00 Format: yyyy-MM-dd hh:mm
To 2007-01-24 00:03 Format: yyyy-MM-dd hh:mm
submit
Logout Main Page

3.6 Structure globale et patterns

On va utiliser un exemple concret afin de mieux illustrer la structure globale du système.

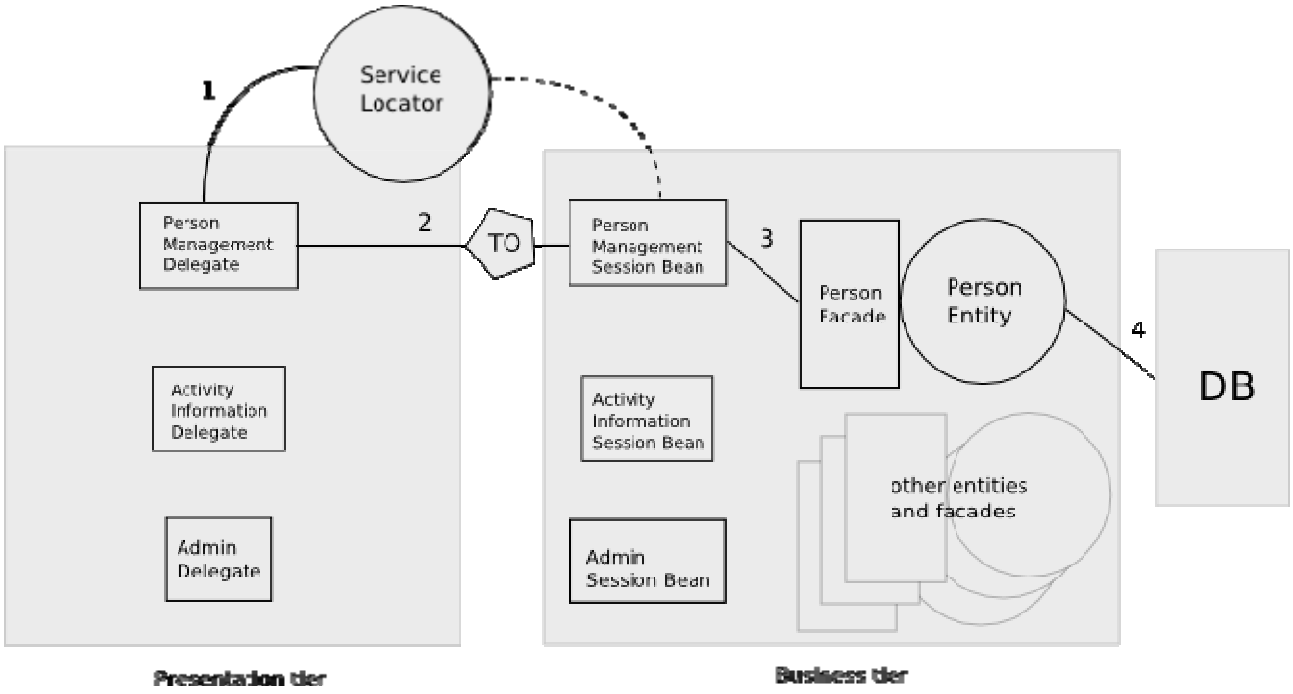


Figure 3 : Interaction entre les différentes couches

Lorsqu'un utilisateur du système fait appel à une fonctionnalité à travers l'interface web, les événements suivants se produisent:

0. Soumission d'un formulaire sur une page .jsp.
1. Le *backing bean* associé à la page .jsp demande au *Service Locator* une instance du *business delegate* côté serveur.
2. Le *backing bean* fait le *marshalling* des données dans un *Transfert Object* (TO), le TO ainsi obtenu est envoyé vers le *business tier* au *business delegate* concerné.
3. Le *business delegate* effectue le *dé-marshalling* et au travers des interfaces locales il opère sur les *entity bean*.
4. Les données modifiées sont persistées dans (ou retirées depuis) la base de données.
5. Si nécessaire le *business delegate* renvoi un nouvel TO au *web tier*.

L'utilisation des *Tranfert Objects*, des *Business Delegates* ainsi que du *Service Locator* est un exemple d'application de *Design Patterns* dans des applications multi-tiers.

4 Conclusion

Nous aimerions tout d'abord souligner le fait que, dans le cadre des études de Master, la majeure partie des cours s'occupent des aspects théoriques de l'informatique. Dans ce contexte, le cours *Advanced Software Engineering* constitue une exception que nous avons beaucoup appréciée. En effet ce cours nous a donné un aperçu de quelques technologies de plus en plus utilisées dans le monde du travail que nous considérons importantes pour la réussite de notre future vie professionnelle.

Nous avons bien apprécié le fait d'être confrontés à une technologie de pointe très jeune telle que *Java Enterprise Edition*, bien que cela ait parfois produit quelques difficultés dans l'encadrement du projet. Dans la phase de développement, nous avons pu constater que la puissance de ces technologies est corrélée avec un niveau de complexité souvent élevé. En effet l'apprentissage et l'utilisation ne sont pas toujours aisée ce qui rend les temps de familiarisation très longs.

L'intérêt suscité en nous par la possibilité d'utiliser ces technologies, nous a poussés à travailler ensemble pour la totalité du projet bien que celle-ci ne soit pas la solution optimale pour ce qui concerne le temps investi. Pour cette raison le temps estimé pour la réalisation de la partie web du projet n'a pas été suffisant pour l'implémentation de toutes les fonctionnalités souhaitées. Dans le souci de respecter les délais imposés, nous avons donc décidé de limiter les fonctionnalités offertes tout en cherchant de couvrir les différents types de *use case* côté web. Avec davantage de temps à disposition, nous aurions implémenté des fonctionnalités permettant de traiter les disciplines, élargi les recherches (en particulier aux activités et aux disciplines) et amélioré le côté "user-friendly" de l'interface web en introduisant la validation des données et en facilitant leur saisie (listes déroulantes, etc).

En conclusion nous pouvons affirmer que, malgré la charge de travail et le grand nombre d'heures de travail investies dans le projet, ce cours a certainement constitué une expérience utile et enrichissante.

Annexe A

Hardware & software utilisés:

- Laptop avec processeur Intel Pentium Mobile 2.0 Ghz, 1GB RAM.
- OS: Windows XP Pro SP2
- NetBeans 5.5 avec Sun Java System Application Server 9.0 U1 Patch 1
- Java EE 5 SDK Update 1 (java version 1.5.0_10)
- MySQL 5.0.27 Community server
- MySQL JConnector/J 5.0.4