CVO Antwerpen
centrum voor volwassenenonderwijs

Persoonlijk Dossier: Biodiversity Oman

Project Werk

Naam: Lenny Donnez

Schooljaar 2014-1015

Inleiding

We hebben Java gebruikt voor back-end, waaronder we de JDBC Library hebben gebruikt om de connectie te verzorgen tussen de databank en onze Java code. Deze beslissing werd genomen aangezien de meeste van de groepsleden met deze Library het beste overweg konden en vooralsnog een deadline hadden.
Als databank hebben we MySQL gebruikt, die na enkele hevige maar productieve discussies snel op orde kwam.

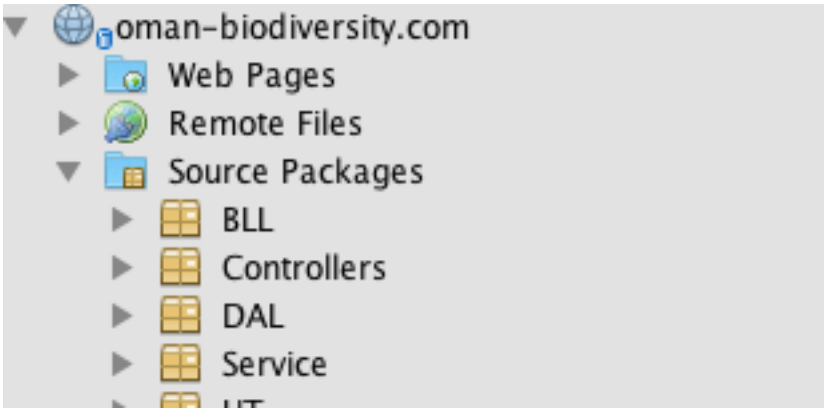
Om onze data naar onze front-end te sturen hebben we een ‘middenlaag’ van Jquery AJAX calls gebruikt. Deze is een eenvoudige maar doeltreffende manier om data dynamisch op onze pagina te laden. Verder is er ook gebruik gemaakt van Twitter Bootstrap 3. Vooral van de handige grid hebben we gretig gebruik van gemaakt. Om onze applicatie op een eenvoudige manier responsive te krijgen te ondersteunen voor oudere browsers, wat met de ingebouwde implementatie van CSS (flex) nog niet het geval is.

.

Mijn Aandeel

Na onze laatste vergadering voor het ontwerp van onze databank zijn we meteen aan onze back-end begonnen en hebben we ons klasse diagram in de praktijk omgezet naar onze BLL.
Hierbij heeft ieder van ons een verantwoordelijkheid gekregen. Ik zelf heb de back-end code geschreven voor onze gebruikers management interface. Omdat we initieel dachten dat de organismen ingevoerd gingen worden door ‘stagiairs’ en professoren, hebben we gekozen voor twee soorten gebruikers namelijk de gewone gebruiker en de admin. Deze wordt tevens in de databank onderscheiden door een eenvoudige boolean.

Aangezien ik één van de eerste was dat ging werken aan de back-end heb ik het project initieel opgezet, en onze mappenstructuur aangebracht. Hiervoor heb ik gekozen om ons strict aan het MVC patroon te houden. Met dit als resultaat:



Onze mappenstructuur is verdeeld onder
- BLL: Business Logic
- DAL: Data Acces met al onze functies die onze connectie met de DB verzorgen
- Service: Die als ons model dient en onze Data klaarmaakt om naar onze DAL te sturen.
- Controllers: dispatcher van Get en Post requests.

Verder heb ik de DAL klasse voor de user interface onder verdeeld in enkele functies. Deze kan je uiteraard vinden in de DAL package onder DaUserAccount.

DaUserAccount Functies

public static void insert(UserAccount user)

Deze functie neemt een User object en slaat deze vervolgens op in de databank

public static UserAccount selectByUsername(String username)

Deze functie neemt een gebruikersnaam en retourneert daar een User object terug met al zijn gegevens behalve het wachtwoord.

public static void update(UserAccount user)

Neemt een bestaand User object en overschrijft het met de bestaande gegevens in de databank.

public static void delete(String username)

Neemt een username en verwijderd al zijn gegevens.

updatePassword(String password, String username)

Update het bestaande wachtwoord met een nieuwe hashed wachtwoord(BCrypt)

public static boolean checkPassword(String username, String password)

Neemt het wachtwoord van een gebruikersnaam om na te gaan of het overeen komt met het hashed wachtwoord in de databank.(Bcrypt)

public static boolean checkUsername(String username)

Kijkt na of deze gebruikersnaam nog niet bestaat

public static void setSuperUser(String username)

zet de Admin Boolean in de databank op 1 om een gebruiker een admin te maken.

public static List<UserAccount> selectAll()

Selecteert alle Users uit de databank en steekt de gegevens van elke gebruiker in een Object waarna dit object in een List geretourneerd wordt.

public static boolean isAdmin(String username)

Kijkt na of een gebruikersnaam Admin is of niet

public static void setNormalUser(String username)

Zet de admin boolean in de databank op 0 om een gebruiker terug een gewone gebruiker te maken.

public static List search(String keyword)

Neemt een kenwoord en zoekt in de UserAccount tabel, of dit keyword overeen komt met voornaam, achternaam, email of voornaam+achternaam. Omdat de voor -en achternaam allebei in een aparte tabel zitten heb ik CONCATENATE in onze query moeten gebruiken om deze 2 waarden samen te kunnen voegen.

public static boolean isLastUser()

Deze functie kijkt na of een een gebruiker de laatste gebruiker is in onze databank, om te vermijden dat deze nooit kan verwijderd worden (Deze functie werd geschreven door Bert).

De objecten dat meegestuurd worden in deze DAL functies worden in onze Service laag klaargemaakt. Deze functies kan je vinden in de Service package onder ServUserAccount. Elke parameter van deze functies wordt meegegeven door zijn aparte controller. Waarbij het Model(Services) Deze in objecten steekt en die vervolgens naar onze DAL stuurt.

Connectie Back -en Front End

Om onze gebruikers de beste User Experience te geven hebben we AJAX gebruikt om onze data te presenteren in onze views. Hiervoor hebben we de ingebouwde AJAX functies van de JQuery Library gebruikt. Deze ronde we in aparte functies stoppen om zo onze code makkelijk uit te kunnen breiden.

Omdat niemand van mijn groep AJAX calls heeft geschreven heb ik op voorhand enkele AJAX calls voor de user management gedocumenteerd.