BucKonnect Part 4 - Design description

Priyanka Sadavartia (sadavartia.1@osu.edu)
Jagannath Narasimhan(narasimhan.23@osu.edu)
Nandakumar Khobare (khobare.1@osu.edu)

We created different session beans and entity beans for the three main tables of BucKonnect which are Users,Groups and Events. The reason for doing this is to make sure that the operations which would be implemented on the three tables in our database are compartmentalized .

The **Entity beans** which are a part of BucKonnectEJB have the @Entity tags , also the tablename(E.g. @Table(name="Events")) tags are present in the entity beans . Along with this the various column name attributes are set (E.g. @Column(name="Event_Id")) along with a getter-setter function combination for each entry.

The **Session beans** which again are a part of BucKonnectEJB define and describe the query for each operation we want to implement which are all the operations that we implemented in part 2 of the project which are the CRUD operations. The session beans define the query for create , update and search for a user, event or group through three separate session beans.

Now these are all the definitions of the entity and session beans . We have backing beans defined in BucKonnectPortal which links these entity and session beans with the front UI end and makes sure the values entered pass from the UI end to the DB end.

Initially to demonstrate the functionality of Ajax we have created welcomeBean.java and welcome.xhtml which without reloading the page displays the entered name by the user.

We have defined separate beans for helping in registering of a user which allows registration into BucKonnect to happen only when :

Validations:

- The username entered is of the format 'xyz.[0-9]@osu.edu'
- The password is a required entry here and the restriction of length is applied on it from 8 - 20.

- Here we have implemented **AJAX** use which checks if the entered username is valid or not, the way this is done is that it checks the already entered values and allows a new user to register only if the entered OSU Email Id is unique.

After a user has registered we have separate xhtml definitions for login and a corresponding backing bean for it. In this we have option for if they are a new user they could signup and the outcome would be the register query.

Then a user would be asked to update his/her profile with details like department,major,interests , about me etc. These changed values reflect back onto the H2-Database in the user table created.

From here we have an option for Home view for the user where there are options for Update profile where a user can update any details that he/she wants to change, Search BucKonnect Portal where there would options to search for users, groups or events which are present on the BucKonnect DB, also view the groups and events that the user is a part of. Each of these above functionalities have their individual backing beans file which links to User,Group or Event session and entity beans corresponding to the table it is dealing with and a separate xhtml page for the UI display to the user.

Thus when a user wants to join a group or event that is where the user and group or user and event databases are involved then they are joined by the foreign key which is the osu_email_id of the user.

Lifetime of each object:

- **User:** An object of user class is created as soon as a new user wants to register, following which a user could also login into BucKonnect . This object is destroyed only when a user logs out from the system. During the time the user is active , the user object could create or search groups and events.
- **Group:** A group object is created whenever a create group request is made and the object is destroyed as soon as the group is created.
- **Event:** An event object is created whenever a create event request is made and the object is destroyed as soon as the event is created.

Thus the **object lifecycle of a user** would go like this.

An object of the **user** type registers and enters into the user database , then after log-in the user can update his/her profile thus filling the other entries of the user table. From here user can search the Buckonnect Portal which would give him three options :

- Search for a different user: this would query the user database and retreive results
- Search for a new group which would be done by groupname: this would link the user and the groups table
- Search for a new event which would be done by eventname: this would link the user and the events table

Now after search ,a user could like to view a user profile or join a group or event . This would be an **entry** point to the user-user , user-group or user-event table depending on the type of request of the user. When a search query returns a valid result then that would again be redirected to users , group or events found xhtml pages showing that the query was successful.

After the query is executed the **exit** from the corresponding database happens.

A new user could even desire to create a group or an event which would again have a backing beans java file which would link itself with the session and entity beans of the corresponding events or group table. Separate UI xhtml pages would be defined for them which would be linked with the backing beans created.

The main linkage would happen at faces-config.xml where the definition of the managed beans would be present and the h2 database link would be completed through the datasource entry at standalone.xml of JBoss 7.1.

Database Schema:

User_Group: Many to many relation between User and Groups

User_Event: Many to many relation between User and Events

User : This corresponds to User ORM.

Groups: This corresponds to Group ORM.

Events: This corresponds to Events ORM.

Non functional requirement:

Security: The password that the user enters on registering into BucKonnect is encrypted and stored into the BucKonnect DB, which makes sure any other user who gets hold of the view of the database,the password of the other users doesn't get compromised. Also when the user wants to login the password entered is matched with decrypted password from the DB.

We have attached out code for this stage of the project.