DEI Departamento Engenharia Informática

IPL escola superior de tecnologia e gestão instituto politécnico de leiria

Morro do Lena, Alto Vieiro · Apart. 4163·2401 – 951 Leiria
Tel.: +351·244 820 300 · Fax.: +351·244 820 310
E-mail: estg@estg.ipleiria.pt · http://www.estg.ipleiria.pt

Desenvolvimento de Aplicações Empresariais – 2016-17-1S

Engenharia Informática – 3.º ano – Ramo SI

Worksheet 2

Topics: (C)RUD operations.

In worksheet 1, you developed the feature that allows the user to create new students and save them in the database. In this worksheet you will develop features to list, view the details, update and remove students.

1. Create the named query "getAllStudents" for the *Student* entity.
2. Write the *List<Student> getAll()* method in the *StudentBean* EJB. You must call the *createNamedQuery(...)* method from the entity manager to create the query and, then, call the *getResultList()* method from the resulting query to get the students' list.
3. Write the *List<Student> getAllStudents()* method in the *AdministratorManager* Managed Bean (MB), which calls the *getAll()* method from the *StudentBean* EJB.
4. Modify the *createStudent()* method in the *AdministratorManager* MB so that it now returns the string "index?faces-redirect=true". This will redirect the user to the index page after she/he creates a new student if the operation succeeds. If not, the application should remain in the same page.
5. In the index.xhtml facelet, add, before the "New student" link, a h:dataTable component that allows all the students to be listed. For now, the table should have three columns: *Username*, *Name* and *Email*. For details on how to use the h:dataTable component, please consult the Order project.
6. Run and test the application.
7. Add the *currentStudent* attribute (*Student* entity type) and the respective getter and setter methods to the *AdministratorManager* MB.
8. Create the admin_students_details.xhtml facelet. It should show the data of the current student defined in the *AdministratorManager* MB, except the password one... Use the h:outputText component to show the data values; use also the h:outputLabel and the h:panelGrid components (how many columns are needed here?). This page should also have a link that allows the user to return to the index page. Note: for now, this page shows the

same information shown in the data table of the index page; maybe this will change in the future…

9. Add a 4th column named "*Actions"* to the data table in the index page. Add to this column a h:commandLink component. Set the *value* attribute to "Details" and the *action* attribute to "admin_students_details?faces-redirect=true". Add to the h:commandLink component the core component

   ```
   <f:setPropertyActionListener target="#{administratorManager.currentStudent}" value="#{student}" />
   ```

   This component will set the *currentStudent* attribute of the *AdministratorManager* MB to the *Student* instance corresponding to the data table line of the "Details" link clicked by the user, so that the admin_students_details page shows the right information.
10. Run and test the application.
11. Add to the StudentBean EJB the *void update(…)* method that receives the attributes of a student, finds it in the database and update its data. Use de *find(…)* method of the entity manager to find the student. Use the *merge(…)* method of the entity manager to update the student.
12. Write the *void updateStudent()* method in the *AdministratorManager* MB, which calls the *udpate()* method from the *StudentBean* EJB, sending it the attributes of the *currentStudent* variable.
13. Create the admin_students_update.xhtml facelet. It should allow the user to modify the data of the *currentStudent* set in the *AdministratorManager* MB. This facelet is similar to the administrator_students_create facelet. What should be different here?
14. Add to the "Actions" column of the data table in the index.xhtml facelet, a command link that allows the update feature to work properly. Proceed similarly as in step 9.
15. Run and test the application.
16. Add to the *StudentBean* EJB the *void remove(…)* method that receives the student's ID (username) as argument. Use de *find(…)* method first… Use the *remove(…)* method of the entity manager to remove the student.
17. Add to the "Actions" column of the data table in the index.xhtml facelet, a command link that allows the user to remove the student from the database. You should set the *actionListener* attribute of the command link to "#{administratorManager.removeStudent}", so that the event listener method *void removeStudent(ActionEvent event)* in the *AdministratorManager* MB (see next step) is called when the link is clicked. You need also to add to the command link component the core component

    ```
    <f:param name="deleteStudentId" id="deleteStudentId" value="#{student.username}" />
    ```

    that defines the parameter *deleteStudentId* and sets its value to the student's username corresponding to the line of the clicked link. See in the next step how you can get the parameter's value from the event passed to the listener in the MB.
18. Write the *void removeStudent(ActionEvent event)* listener method in the *AdministratorManager* MB, which calls the *remove()* method from the *StudentBean* EJB,

sending it the ID of the student to be removed. You must start by getting the parameter value set in the "Remove" command link of the index facelet in the following way:

```
UIParameter param = (UIParameter) event.getComponent().findComponent("deleteStudentId");
String id = param.getValue().toString();
```

19. Run and test the application.

Note: you may/should consult the Order project that comes with the Java EE Tutorial.

Bibliography

Java EE Tutorial 7 (all of it).