|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module No:** | 6 | **IU No:** | 5 | **Exercise No.** | 5 |

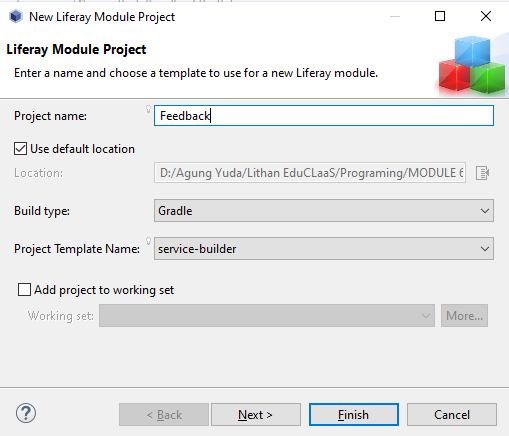
**Name : Agung Yuda Pratama  
Code : BDSE-0922-076/STTB**

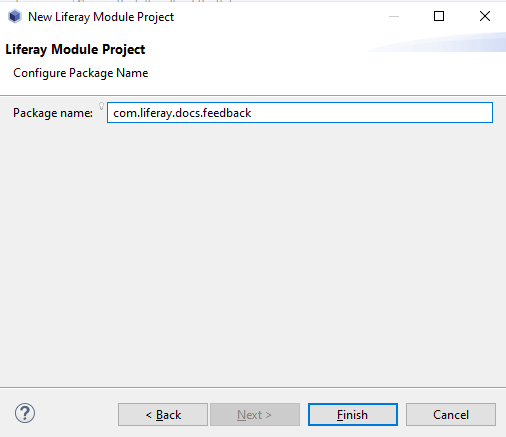
|  |  |
| --- | --- |
| **Lab Assessment Statement** | **Assignment 5 – Develop an Application**  **Referring to Assignment-1's Project Scenario,**  The scope of this assignment is, to implement the previous Feedback portlet to a database-driven application for saving and retrieving the user data. Users can be left a message of his/her opinion to the "Online News Portal".  A screenshot of a cell phone  Description automatically generatedThis is a sample of what the final result should look like.  A screenshot of a cell phone  Description automatically generated  These are the steps provided to build an application.   1. To implement a database-driven application,   right-click on previous Feedback MVC project -> create a new Liferay service builder with package path "com.liferay.docs.feedback".  Use the namspace as FB.   1. Create an entity as a local service in service.xml with entity name Feedback. 2. Define the columns as the following.  * <column name="feedbackId" type="long" primary="true"></column> * <column name="feedbackDate" type="Date"></column> * <column name="feedbackText" type="String"></column> * <column name="feedbackSubject" type="String"></column> * <column name="companyId" type="long"></column> * <column name="groupId" type="long"></column> * <column name="userId" type="long"></column>  1. In service.xml, define the order of feedback entity by ascending feedbackId. 2. In service.xml, define Service Entity Finder Methods for GroupId, CompanyId and feedbackText cloumns. 3. Build your service. If build success, check the service Builder generates classes and interfaces belonging to the persistence layer, service layer, and model layer. 4. Expose the Finder in the Service Layer \*LocalServiceImpl. Implement a service layer for Feedback entity and create a method "addFeeback" that adds Feedbacks to the database. 5. Update controller (your portlet class) action method to handle the Feedback form. 6. Update view.jsp page to display all the feedback. 7. Build your service application, deploy your portlet, and try adding some Feedback for your portal.   **Provide caption and screenshots for each step.**  **Provide screen capture of final result pages.** |
| **Technical Environment** | - |
| **Guidelines** | - |
| **Duration** | 120 mins |

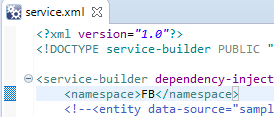
These are the steps provided to build an application.

1. To implement a database-driven application,

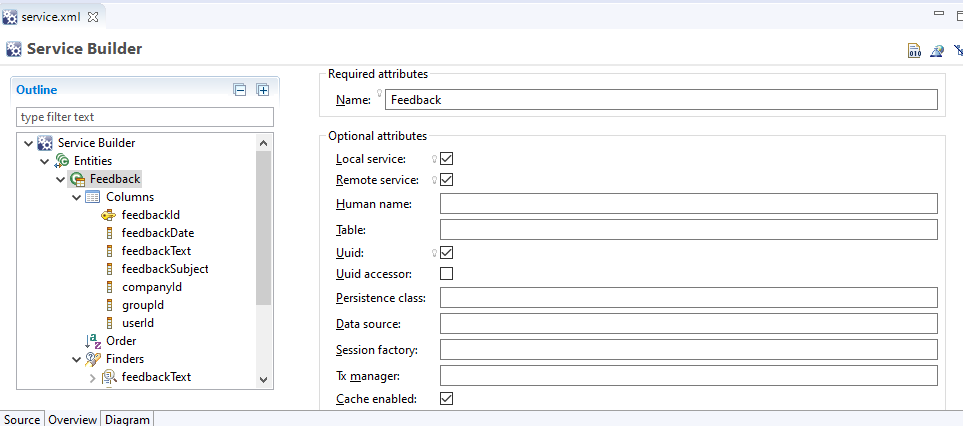
right-click on previous Feedback MVC project -> create a new Liferay service builder with package path "com.liferay.docs.feedback". Use the namespace as FB.

Create a module project and set as below  






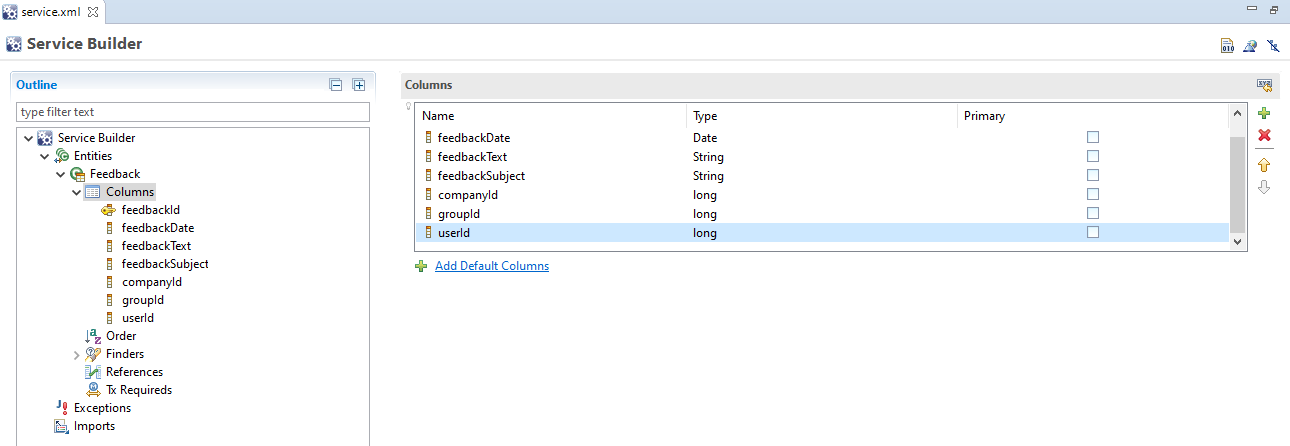
1. Create an entity as a local service in service.xml with entity name Feedback.

Go to service.xml double click on that and click overview  


1. Define the columns as the following.

In the Columns section, you can change the data to this.

* <column name="feedbackId" type="long" primary="true"></column>
* <column name="feedbackDate" type="Date"></column>
* <column name="feedbackText" type="String"></column>
* <column name="feedbackSubject" type="String"></column>
* <column name="companyId" type="long"></column>
* <column name="groupId" type="long"></column>
* <column name="userId" type="long"></column>|

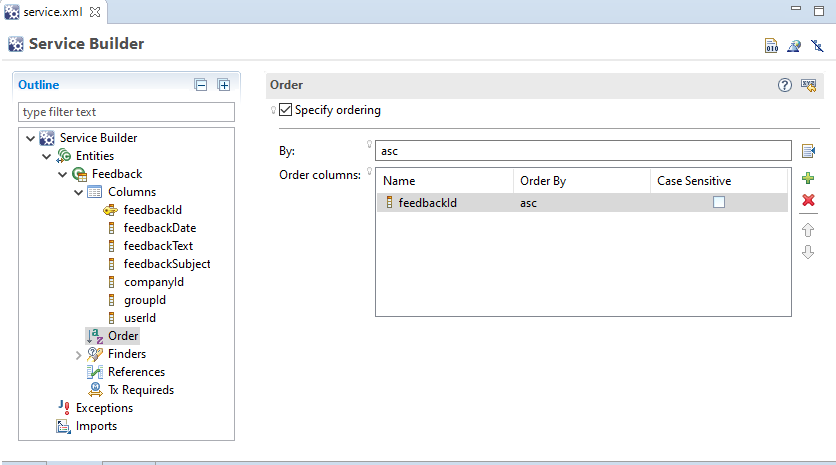
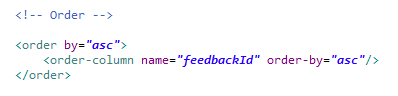


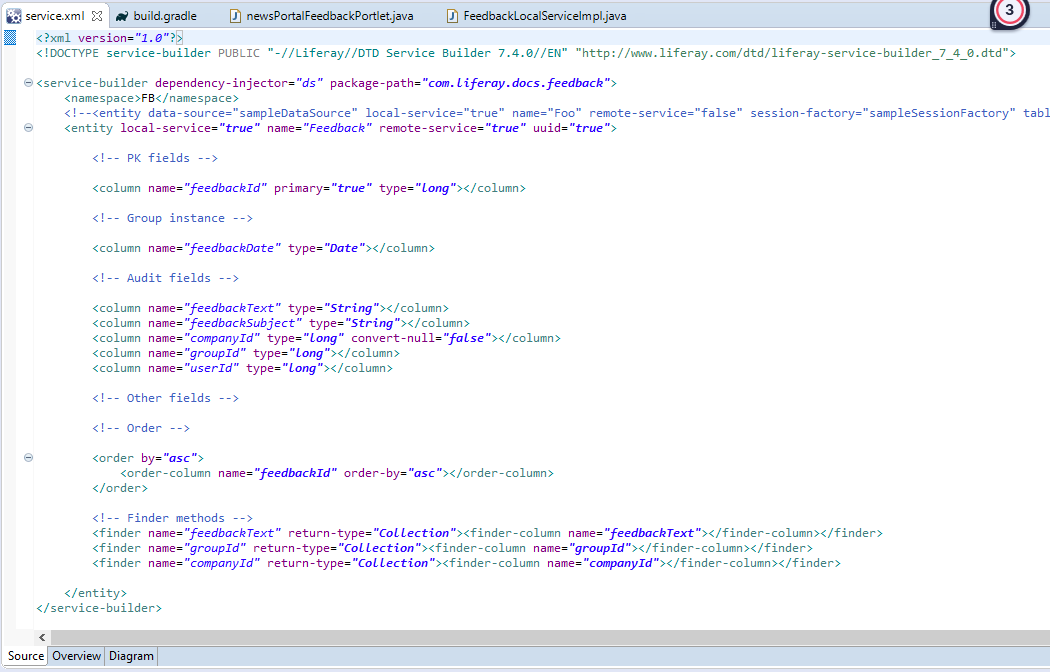
This is the result



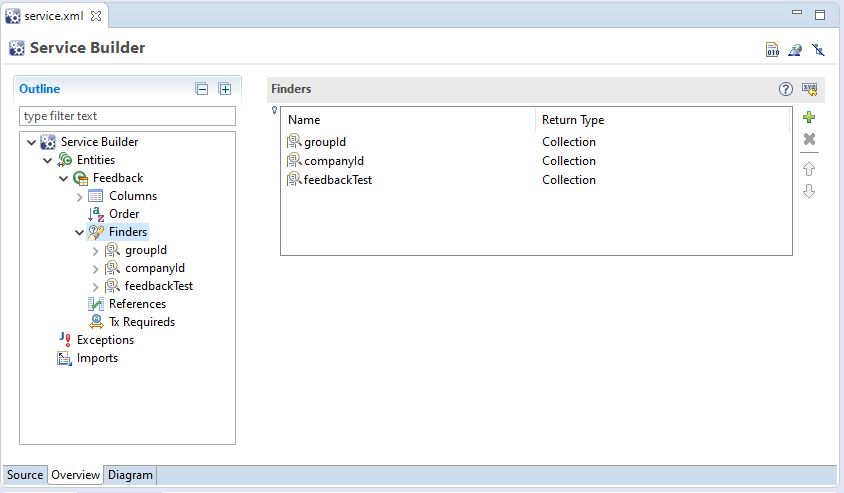
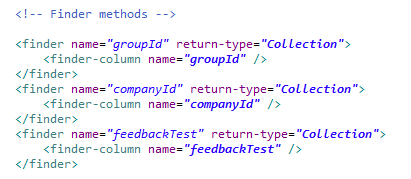
1. In service.xml, define the order of feedback entity by ascending feedbackId.

In the Overview section. Go to the order section, and you can change the Order like this

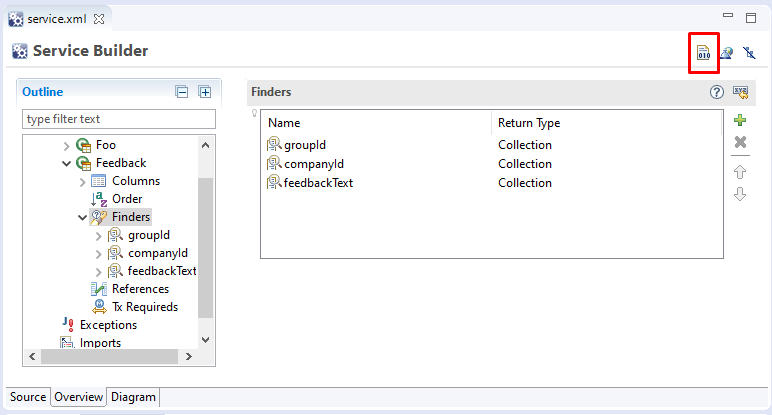
  




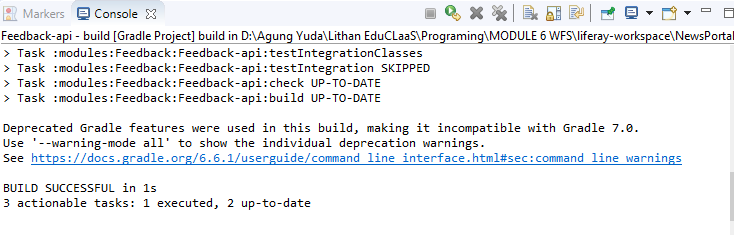
1. In service.xml, define Service Entity Finder Methods for GroupId, CompanyId and feedbackText columns.

Go to Finders section, and change the field same like this.  
  


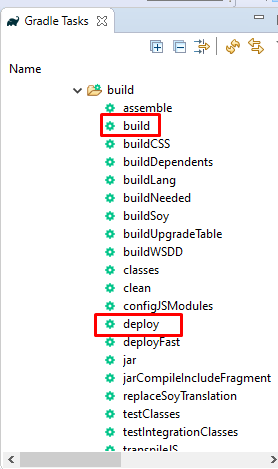
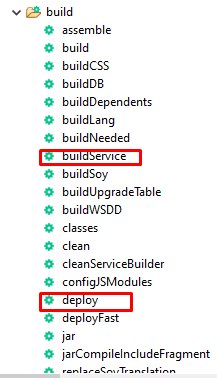
1. Build your service. If build success, check the service Builder generates classes and interfaces belonging to the persistence, service, and model layers.

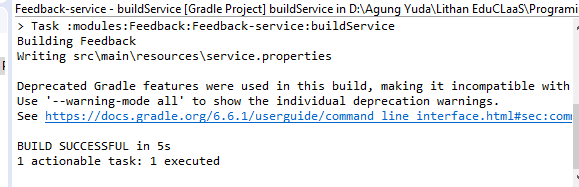
After you completed the task click build this one  


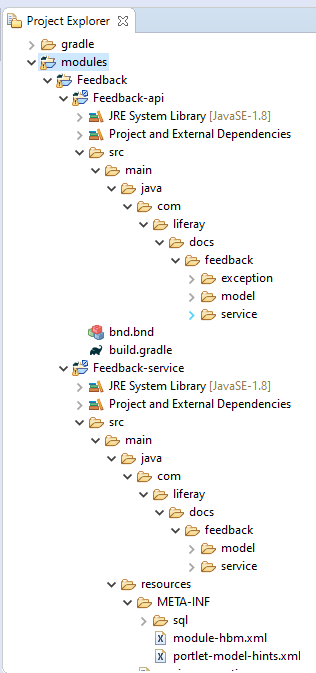
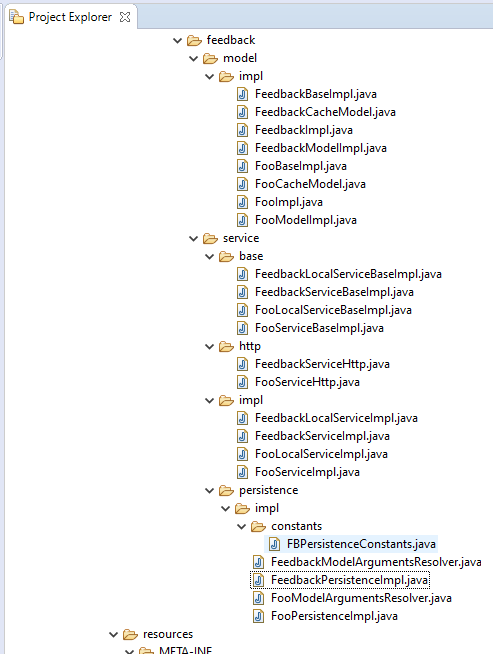
Go to Gradle task, select Feedback service and Feedback API, and click build

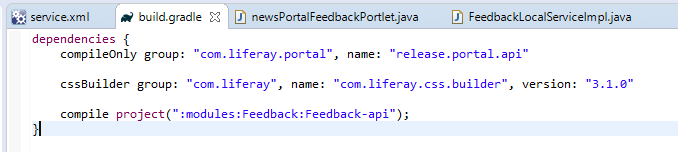


After that build the API first and builld the service until successfull

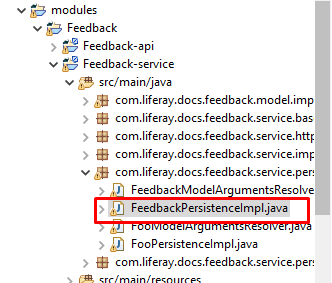




Compile the code so we can see the database  
  
Dont forget to see the database is updated or not



1. Expose the Finder in the Service Layer \*LocalServiceImpl. Implement a service layer for Feedback entity and create a method "addFeeback" that adds Feedbacks to the database.   
   go to this FeedbackPersistenceImpl.java



**Finder Method for groupId**

@Override

**public** List<Feedback> findBygroupId(**long** groupId) {

**return** findBygroupId(

groupId, QueryUtil.***ALL\_POS***, QueryUtil.***ALL\_POS***, **null**);

}

@Override

**public** List<Feedback> findBygroupId(**long** groupId, **int** start, **int** end) {

**return** findBygroupId(groupId, start, end, **null**);

}

@Override

**public** List<Feedback> findBygroupId(

**long** groupId, **int** start, **int** end,

OrderByComparator<Feedback> orderByComparator) {

**return** findBygroupId(groupId, start, end, orderByComparator, **true**);

}

@Override

**public** List<Feedback> findBygroupId(

**long** groupId, **int** start, **int** end,

OrderByComparator<Feedback> orderByComparator, **boolean** useFinderCache) {

FinderPath finderPath = **null**;

Object[] finderArgs = **null**;

**if** ((start == QueryUtil.***ALL\_POS***) && (end == QueryUtil.***ALL\_POS***) &&

(orderByComparator == **null**)) {

**if** (useFinderCache) {

finderPath = \_finderPathWithoutPaginationFindBygroupId;

finderArgs = **new** Object[] {groupId};

}

}

**else** **if** (useFinderCache) {

finderPath = \_finderPathWithPaginationFindBygroupId;

finderArgs = **new** Object[] {groupId, start, end, orderByComparator};

}

List<Feedback> list = **null**;

**if** (useFinderCache) {

list = (List<Feedback>)finderCache.getResult(

finderPath, finderArgs);

**if** ((list != **null**) && !list.isEmpty()) {

**for** (Feedback feedback : list) {

**if** (groupId != feedback.getGroupId()) {

list = **null**;

**break**;

}

}

}

}

**if** (list == **null**) {

StringBundler sb = **null**;

**if** (orderByComparator != **null**) {

sb = **new** StringBundler(

3 + (orderByComparator.getOrderByFields().length \* 2));

}

**else** {

sb = **new** StringBundler(3);

}

sb.append(***\_SQL\_SELECT\_FEEDBACK\_WHERE***);

sb.append(***\_FINDER\_COLUMN\_GROUPID\_GROUPID\_2***);

**if** (orderByComparator != **null**) {

appendOrderByComparator(

sb, ***\_ORDER\_BY\_ENTITY\_ALIAS***, orderByComparator);

}

**else** {

sb.append(FeedbackModelImpl.***ORDER\_BY\_JPQL***);

}

String sql = sb.toString();

Session session = **null**;

**try** {

session = openSession();

Query query = session.createQuery(sql);

QueryPos queryPos = QueryPos.*getInstance*(query);

queryPos.add(groupId);

list = (List<Feedback>)QueryUtil.*list*(

query, getDialect(), start, end);

cacheResult(list);

**if** (useFinderCache) {

finderCache.putResult(finderPath, finderArgs, list);

}

}

**catch** (Exception exception) {

**throw** processException(exception);

}

**finally** {

closeSession(session);

}

}

**return** list;

}

**Find by companyId**

@Override

**public** List<Feedback> findBycompanyId(**long** companyId) {

**return** findBycompanyId(

companyId, QueryUtil.***ALL\_POS***, QueryUtil.***ALL\_POS***, **null**);

}

@Override

**public** List<Feedback> findBycompanyId(**long** companyId, **int** start, **int** end) {

**return** findBycompanyId(companyId, start, end, **null**);

}

@Override

**public** List<Feedback> findBycompanyId(

**long** companyId, **int** start, **int** end,

OrderByComparator<Feedback> orderByComparator) {

**return** findBycompanyId(companyId, start, end, orderByComparator, **true**);

}

@Override

**public** List<Feedback> findBycompanyId(

**long** companyId, **int** start, **int** end,

OrderByComparator<Feedback> orderByComparator, **boolean** useFinderCache) {

FinderPath finderPath = **null**;

Object[] finderArgs = **null**;

**if** ((start == QueryUtil.***ALL\_POS***) && (end == QueryUtil.***ALL\_POS***) &&

(orderByComparator == **null**)) {

**if** (useFinderCache) {

finderPath = \_finderPathWithoutPaginationFindBycompanyId;

finderArgs = **new** Object[] {companyId};

}

}

**else** **if** (useFinderCache) {

finderPath = \_finderPathWithPaginationFindBycompanyId;

finderArgs = **new** Object[] {

companyId, start, end, orderByComparator

};

}

List<Feedback> list = **null**;

**if** (useFinderCache) {

list = (List<Feedback>)finderCache.getResult(

finderPath, finderArgs);

**if** ((list != **null**) && !list.isEmpty()) {

**for** (Feedback feedback : list) {

**if** (companyId != feedback.getCompanyId()) {

list = **null**;

**break**;

}

}

}

}

**if** (list == **null**) {

StringBundler sb = **null**;

**if** (orderByComparator != **null**) {

sb = **new** StringBundler(

3 + (orderByComparator.getOrderByFields().length \* 2));

}

**else** {

sb = **new** StringBundler(3);

}

sb.append(***\_SQL\_SELECT\_FEEDBACK\_WHERE***);

sb.append(***\_FINDER\_COLUMN\_COMPANYID\_COMPANYID\_2***);

**if** (orderByComparator != **null**) {

appendOrderByComparator(

sb, ***\_ORDER\_BY\_ENTITY\_ALIAS***, orderByComparator);

}

**else** {

sb.append(FeedbackModelImpl.***ORDER\_BY\_JPQL***);

}

String sql = sb.toString();

Session session = **null**;

**try** {

session = openSession();

Query query = session.createQuery(sql);

QueryPos queryPos = QueryPos.*getInstance*(query);

queryPos.add(companyId);

list = (List<Feedback>)QueryUtil.*list*(

query, getDialect(), start, end);

cacheResult(list);

**if** (useFinderCache) {

finderCache.putResult(finderPath, finderArgs, list);

}

}

**catch** (Exception exception) {

**throw** processException(exception);

}

**finally** {

closeSession(session);

}

}

**return** list;

}

**Find By feedbackText**

@Override

**public** List<Feedback> findByfeedbackText(String feedbackText) {

**return** findByfeedbackText(

feedbackText, QueryUtil.***ALL\_POS***, QueryUtil.***ALL\_POS***, **null**);

}

@Override

**public** List<Feedback> findByfeedbackText(

String feedbackText, **int** start, **int** end) {

**return** findByfeedbackText(feedbackText, start, end, **null**);

}

@Override

**public** List<Feedback> findByfeedbackText(

String feedbackText, **int** start, **int** end,

OrderByComparator<Feedback> orderByComparator) {

**return** findByfeedbackText(

feedbackText, start, end, orderByComparator, **true**);

}

@Override

**public** List<Feedback> findByfeedbackText(

String feedbackText, **int** start, **int** end,

OrderByComparator<Feedback> orderByComparator, **boolean** useFinderCache) {

feedbackText = Objects.*toString*(feedbackText, "");

FinderPath finderPath = **null**;

Object[] finderArgs = **null**;

**if** ((start == QueryUtil.***ALL\_POS***) && (end == QueryUtil.***ALL\_POS***) &&

(orderByComparator == **null**)) {

**if** (useFinderCache) {

finderPath = \_finderPathWithoutPaginationFindByfeedbackText;

finderArgs = **new** Object[] {feedbackText};

}

}

**else** **if** (useFinderCache) {

finderPath = \_finderPathWithPaginationFindByfeedbackText;

finderArgs = **new** Object[] {

feedbackText, start, end, orderByComparator

};

}

List<Feedback> list = **null**;

**if** (useFinderCache) {

list = (List<Feedback>)finderCache.getResult(

finderPath, finderArgs);

**if** ((list != **null**) && !list.isEmpty()) {

**for** (Feedback feedback : list) {

**if** (!feedbackText.equals(feedback.getFeedbackText())) {

list = **null**;

**break**;

}

}

}

}

**if** (list == **null**) {

StringBundler sb = **null**;

**if** (orderByComparator != **null**) {

sb = **new** StringBundler(

3 + (orderByComparator.getOrderByFields().length \* 2));

}

**else** {

sb = **new** StringBundler(3);

}

sb.append(***\_SQL\_SELECT\_FEEDBACK\_WHERE***);

**boolean** bindFeedbackText = **false**;

**if** (feedbackText.isEmpty()) {

sb.append(***\_FINDER\_COLUMN\_FEEDBACKTEXT\_FEEDBACKTEXT\_3***);

}

**else** {

bindFeedbackText = **true**;

sb.append(***\_FINDER\_COLUMN\_FEEDBACKTEXT\_FEEDBACKTEXT\_2***);

}

**if** (orderByComparator != **null**) {

appendOrderByComparator(

sb, ***\_ORDER\_BY\_ENTITY\_ALIAS***, orderByComparator);

}

**else** {

sb.append(FeedbackModelImpl.***ORDER\_BY\_JPQL***);

}

String sql = sb.toString();

Session session = **null**;

**try** {

session = openSession();

Query query = session.createQuery(sql);

QueryPos queryPos = QueryPos.*getInstance*(query);

**if** (bindFeedbackText) {

queryPos.add(feedbackText);

}

list = (List<Feedback>)QueryUtil.*list*(

query, getDialect(), start, end);

cacheResult(list);

**if** (useFinderCache) {

finderCache.putResult(finderPath, finderArgs, list);

}

}

**catch** (Exception exception) {

**throw** processException(exception);

}

**finally** {

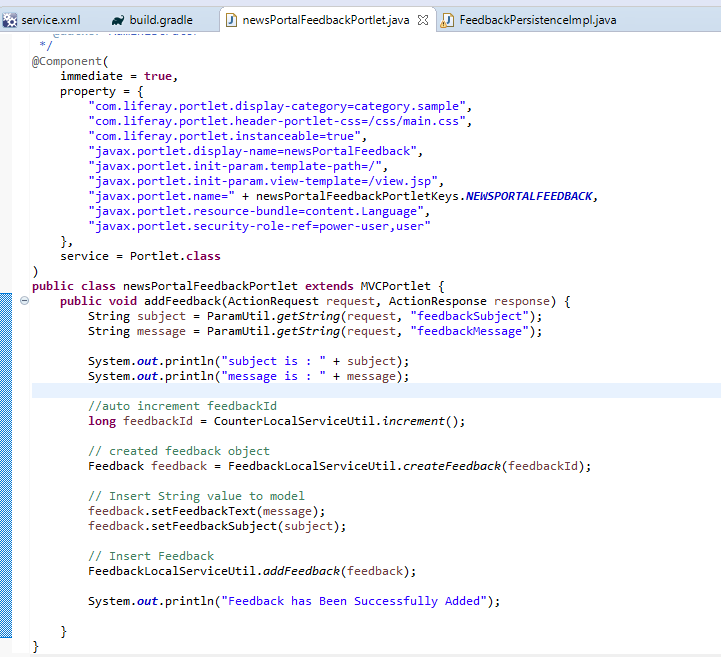
closeSession(session);

}

}

**return** list;

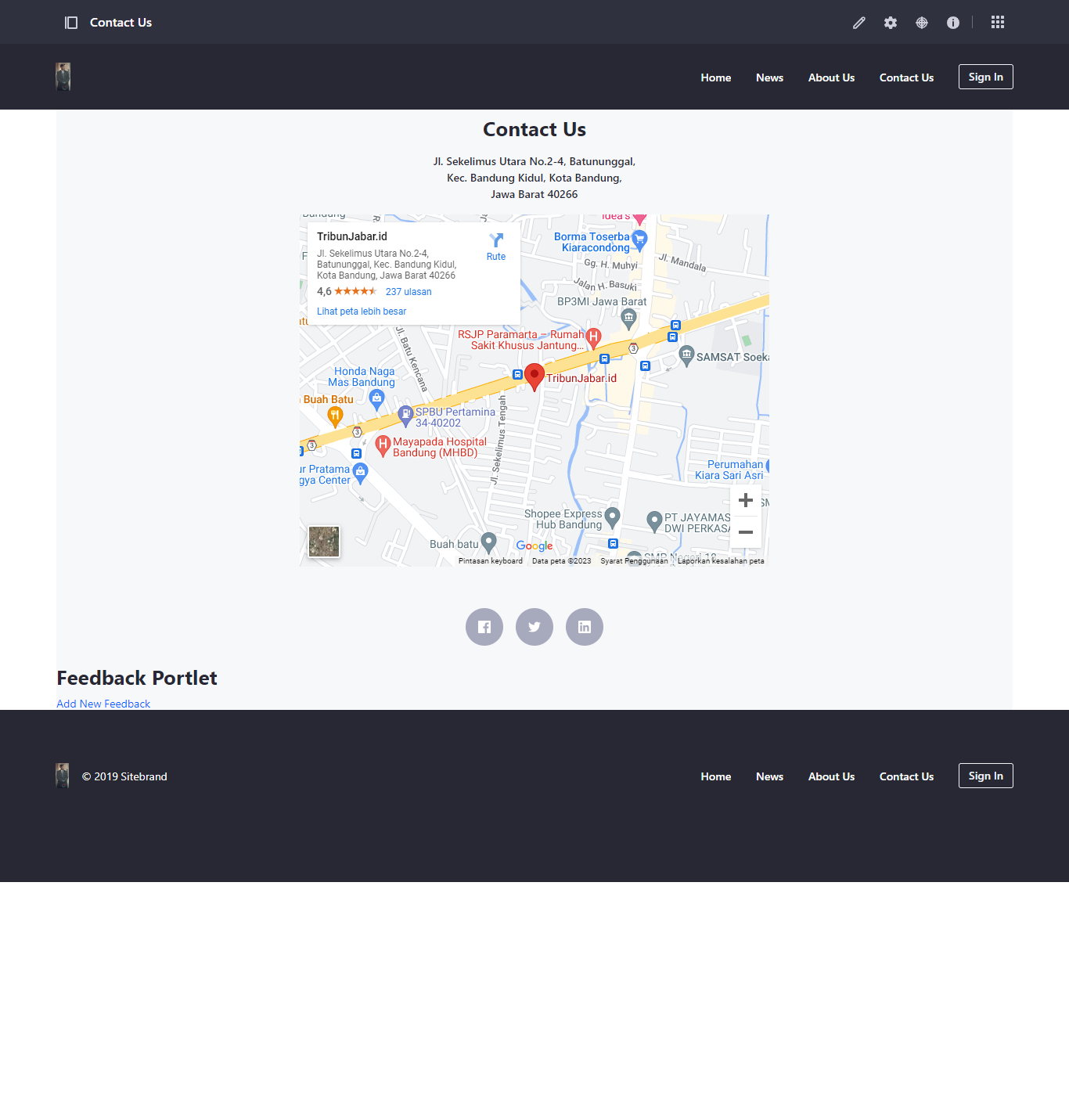
}

1. Update controller (your portlet class) action method to handle the Feedback form.  
   
2. Update view.jsp page to display all the feedback.

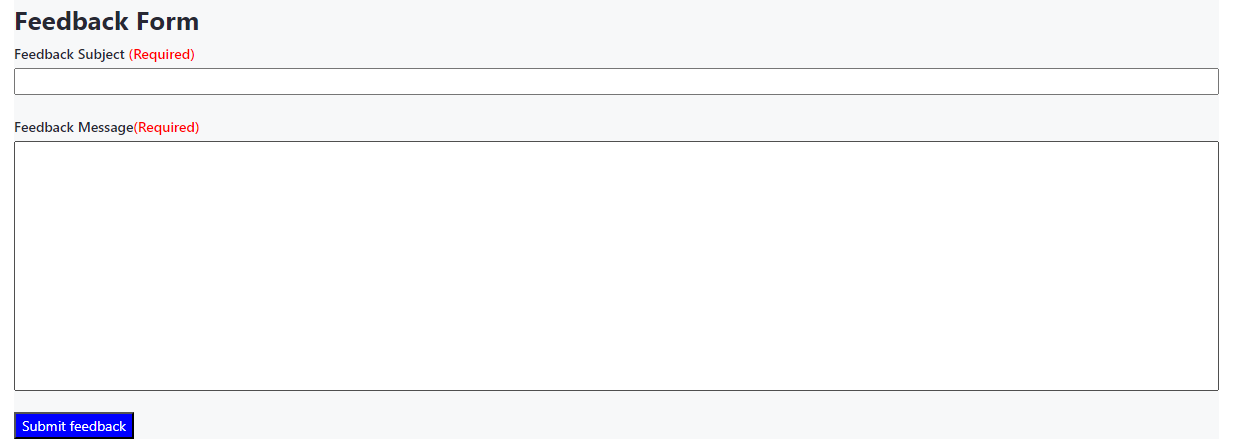
**view.jsp**  


1. Build your service application, deploy your portlet, and try adding some Feedback for your portal.

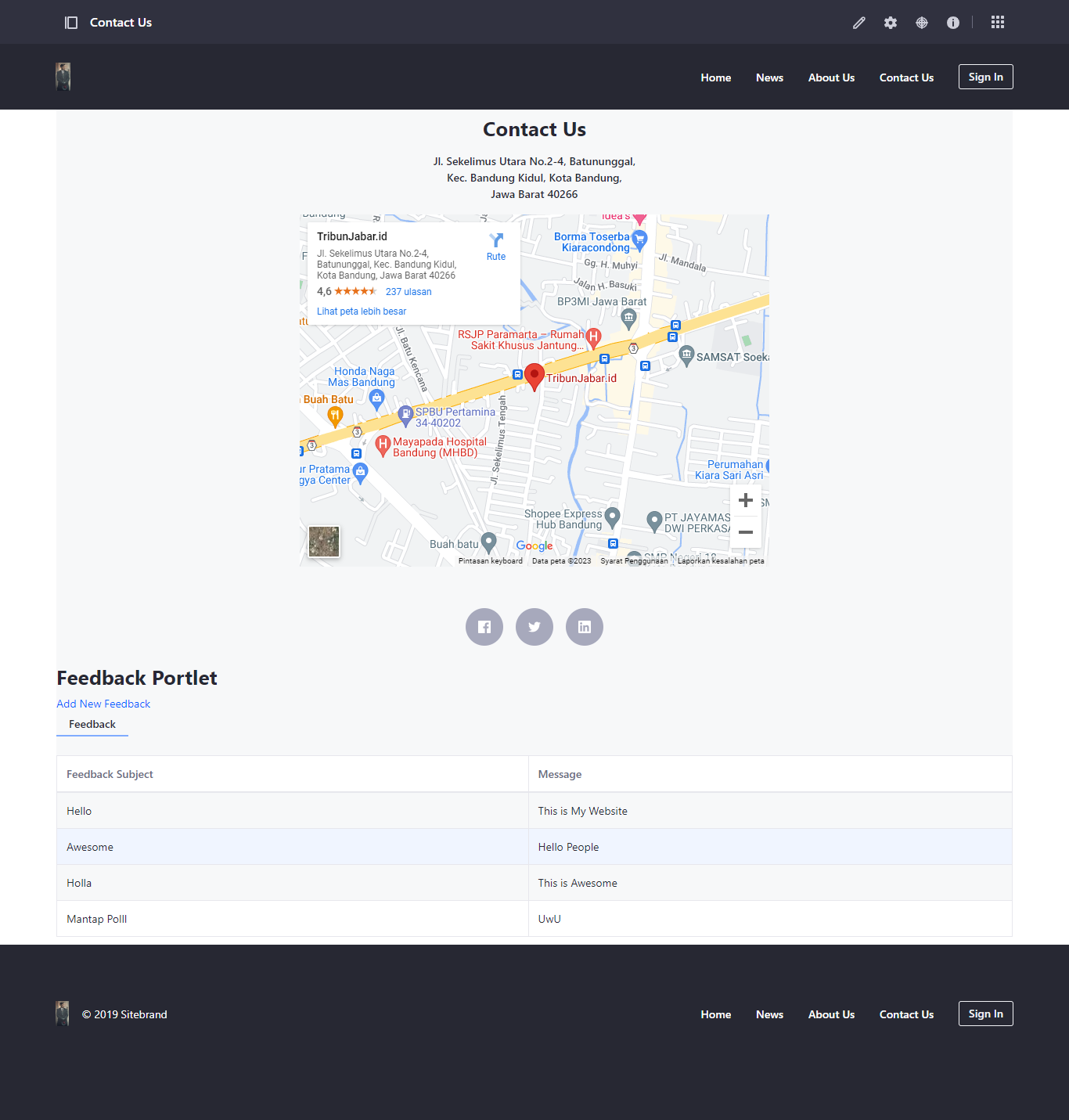
**Contact Us Before adding Feedback**



**Feedback Form**



**Contact Us After adding Feedback**

****

