Dyer Center

Developer Manual

2019

Joseph Cenci, Gabrielle Martone, Julianna Puleo
Dr. Joann J. Ordille
Dyer Center Developer Manual Release 1.0

Table of Contents

Introduction to the Project **2**

Downloading the Application **2**

IDE & Github Repository 2

Introduction to Structure **2**

Vaadin 2

Maven 3

PostgreSQL & pgAdmin4 3

JDBC 3

Setting Up JARs and Libraries **3**

Vaadin 3

Tomcat 8 Installation 4

PostgreSQL JDBC, & pgAdmin4 4

Database Setup 5

Running in Eclipse 5

Extending the Application for Future Works **6**

UI 6

Database 6

Flows **8**

Email **11**

Current Setup / HardCode **11**

Calendario 2019 Calendar **12**

Introduction to the Project

The Dyer Center is an organization on campus that, “fosters and sustains a culture of innovation and entrepreneurship that increases the creative capacity of Lafayette students to lead and inspire change. The Dyer Center implements several student-focused activities aimed at promoting entrepreneurship, innovation and creativity,” that currently are going relatively unnoticed by the student body. The Dyer Center wants to increase awareness for their organization by having web application that will allow enrolled students and alumni to propose ideas for entrepreneurial projects they have and match based upon their skill sets. From this, students and alumni will be able to discuss their proposals outside of the application and move forward in making their ideas a reality. In turn, this provides work experience for students and builds connections for their future careers. Information on how to use this web application can be found in the Dyer Center User Manual which can be found on https://github.com/cencije/DyerCenter_CS470.

Downloading the Application

IDE & Github Repository

The team utilized Eclipse for its development and management of the Dyer Center web application. The instructions for running and operating the code for the Dyer Center web application will be provided in relation to the Eclipse IDE. The specific version used for the Eclipse IDE is version: 2018-12 (4.10.0).
Once Eclipse is installed, the developer should download the project from the Github repository. The link to the Github is the following:

https://github.com/cencije/DyerCenter_CS470

Once the project is cloned/downloaded to the desired local location, open a workspace in the directory one level above the project directory and import the project into the workspace. The project should appear within Eclipse.

Introduction to Structure

Vaadin

Vaadin is a open source java framework for web application development. The advantage to vaadin is it is solely written in java and can be run on a server in just java, and it is licensed under the Apache 2.0 license. Vaadin has built in components that automatically talk to the server side code and then the automatically built javascript during runtime. It is very simple and easy to learn, and makes web development basic for the programmer.

Our structure jar files for vaadin can be found in
DyerC/target/sd-dyercenter-1.0-SNAPSHOT/WEB-INF/lib

Maven

Maven builds and manages our java files. It simplifies the build process. Maven helps with downloading libraries and reconfiguring the build paths.
Our structure for maven is located in DyerC/target/maven-status

PostgreSQL & pgAdmin4

PostgreSQL is an open source object-relational database system that is known for strong reliability, robustness of features, and high performance. PostgreSQL can handle Web services and data warehousing with many concurrent users and is the default database for macOS Server. It is available for Linux and Windows as well. PostgreSQL was used with pgAdmin 4 to provide a view of our database outside of psql on the server. This allowed us to look at our tables and data without having to run SQL commands. It provides a better and updated view of the database and allows for live edits and management. pgAdmin4 provides a tree structure to breakdown the database into its various components.

JDBC

JDBC stands for Java Database Connectivity and is a Java API that connects and executes queries with the database. The JDBC API can be used to access tabular data stored in any relational database. With JDBC, the web application can save, update, delete, and fetch data from the database. The current version of JDBC is 4.3 and was released September 21, 2017.

Setting Up JARs and Libraries

Vaadin

In order to properly install Vaadin on your local machine when developing, the following steps should be done:

1. Go to toolbar *Help → Eclipse Marketplace…*
2. Search for *Vaadin Plugin Eclipse 4.0..2*
3. Click *Install*

The team used Vaadin 13 to develop their web application. The version of Vaadin is specified in pom.xml file.

Tomcat 8 Installation

The team used Tomcat v8.5 to run the application. In order to install, you can download the tar.gz file from:

https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.38/bin/

Once the file is downloaded, in eclipse following these steps:

1. Go to *Windows* → *Show Views* → *Servers*
2. Right click in the server view
3. *New* → *Server*
4. Scroll down to find *Tomcat v8.5 Server* and click *Next*
5. Enter the path of the tomcat file
6. Once it is added, right click on the tomcat server in the server window
7. Click *Add and Remove…* and then add the project into the server

PostgreSQL JDBC, & pgAdmin4

The JAR for PostgreSQL JDBC was downloaded from the following link:

https://jdbc.postgresql.org/download.html

The driver can be downloaded directly by entering the following link in a web browser search bar:

https://jdbc.postgresql.org/download/postgresql-42.2.5.jar

The driver should be placed in 2 locations, within the Build Path of the project and within the directory for tomcat on the server.

- Build Path - To hook up the driver to the build path, the jar must be placed inside the project folder and then linked by doing the following in Eclipse:
    1. Right Click the Project folder → Build Path → Configure Build Path.
    2. Add External JAR → Select the postgresql-42.2.5.jar.
    3. Apply and Close.
- Class Path - To hook up the driver to the build path, the JAR must be placed inside the tomcat folder.
    1. For Mac Users to get to the right folder: Open Finder: Command+Shift+G.
    2. Search: /usr/local/Cellar/tomcat@8/<versionNo>/libexec/lib.
    3. Place the postgresql-42.2.5.jar inside this folder.
- Database - This will be explained under Database Setup.

pgAdmin4 for can be downloaded from the following link:

https://www.pgadmin.org/download/

Upon downloading the application, install it by following the directions on the website. You can make your database username and password according to your own personal preference, but know that you will have to update these variables in the file *config_DB.properties.* After that, try

connecting to the database within pgAdmin4. If it connects, you can now link your project database to that pgAdmin4 database.

Database Setup

To set up the database for postgres perform the following steps:

1. Select Open Perspective in Eclipse on the right side of the IDE
2. Select Database Development
3. Right click on Database Connections → New…
4. Select PostgreSQL and set the name to 'postgres', hit Next
5. Select the Driver by click the New Driver Definition
    a. Select the PostgreSQL JDBC Driver → JAR List
    b. Select the currently shown driver and select Edit JAR/Zip…
    c. Find the postgresql-42.2.5.jar driver and click Open → OK
6. Set Database = postgres
7. At the end of the URL, replace database with postgres
8. User/Name and password according to what you made it when setting up pgAdmin 4.
9. Test connection with a Ping. You will be alerted with a success or failure message.
10. Select Connect if successful.

Once the database is set up, it can be utilized in development to run the application. However, it is important to change the *config_DB.properties* file values to match the username and password that will be used to connect to the database. If not done, an authentication error will be thrown and the application will terminate. This file can be found under *src/main/resources/config_DB/.properties*.

Running in Eclipse

To run the project in Eclipse, right click on the project folder within the project explorer and then select Run As → Run on Server. The web application will begin running on apache tomcat and will output logs to the console. Once the startup line shows writing out the amount of milliseconds, the web application has loaded. The project can be loaded at the localhost of the following link:

localhost:<port-number>/DyerC/

If the build path for the project is not properly configured (build path errors will be shown with small red squares with a white 'x' inside) the project will not load the most recent updates in the code. Instead you will have to right click the project folder, Run As → Maven clean, Run As → Maven install. This will update the project code for the current iteration and will show the most recent code updates upon relaunching the server.

If the build path for the project is properly configured, live updates to the code will be shown when pages are refreshed and when the server is relaunched.

Extending the Application for Future Works

UI

There are various UI additions that can be done to the teams web application. These additional features are written in the Backlog document the team created. The document contains specific details into the clients wants and needs. The document layouts out which classes those implementations lie in and what needs to be done.

Since Vaadin 13 just came out in March 2019, the team felt using this version for the web application took more time than they expected as there is not much tutorials or documentation on this version. The best documentation that the team could find for developing the UI is in this link https://vaadin.com/components/ . This link provides all the prebuilt components that Vaadin 13 provides. The team used built in components because it saved them time and was more efficient for adding them to the different views. The only issue the team ran into was making the web application responsive. Since the components are not responsive going forward this would need to be coded into the CSS. The team worked on some CSS, but with time constraints we decided to focus more on functionality and getting the views to route smoothly.

The way the views route is through the Router class in Vaadin, the documentation for this class is in the link:

https://vaadin.com/docs/flow/routing/tutorial-routing-annotation.html

The team ran into some issues getting a page to initially load when a users view, either student, employee or alumni, was routed to the main class with the navigation buttons. Going forward, using a router may not be the best or most efficient way of implementing view changes. The routing flow for each view starting at the main page to each users view is displayed and explained in the Flows section.

Database

The database has some extra functionality and changes that can be made for future iterations. The changes to be updated are listed in the backlog. The following section holds info regarding the database setup. The following list of changes were prospectively going to be implemented by the team but could not be completed within the time frame of the project (Spring 2019 semester).

There are 9 tables currently created within the database.

1. Table_Profile_Students
    a. ID (INT PK)
    b. NAME (TEXT)
    c. EMAIL (TEXT)
    d. PASSWORD (TEXT)

e. PHONENO (TEXT)
f. MAJOR1 (TEXT)
g. MAJOR2 (TEXT)
h. MINOR1 (TEXT)
i. MINOR2 (TEXT)

---

2. Table_Profile_Alumni
   a. ID INT PK
   b. NAME (TEXT)
   c. EMAIL (TEXT)
   d. PASSWORD (TEXT)
   e. PHONENO (TEXT)

---

3. Table_Profile_Employee
   a. ID INT PK
   b. NAME (TEXT)
   c. EMAIL (TEXT)
   d. PASSWORD (TEXT)
   e. PHONENO (TEXT)
4. Table_Events_Master
   a. EVENT_ID (INT PK)
   b. TITLE (TEXT)
   c. LOCATION (TEXT)
   d. DESCRIPTION (TEXT)
   e. URL (TEXT)
   f. DAY (INTEGER)
   g. MONTH (INTEGER)
   h. YEAR (INTEGER)
   i. HOUR (INTEGER)
   j. MINUTE (INTEGER)

---

5. Table_Project_Index
   a. ID (INT PK)
   b. TITLE (TEXT)
   c. START_DATE (TEXT)
   d. END_DATE (TEXT)
   e. LOCATION (TEXT)
   f. DESCRIPTION (TEXT)
   g. PAID (TEXT)
   h. PROPOSER_NAME (TEXT)
   i. DATEPOSTED (TEXT)

6. Table_Project_Skills
   a. PROJECT_ID (INT)
   b. CATEGORY (TEXT)
   c. SKILL_NAME (TEXT)

---

7. Table_Project_Proposed
   a. ID (INT PK)
   b. TITLE (TEXT)
   c. START_DATE (TEXT)
   d. END_DATE (TEXT)
   e. LOCATION (TEXT)
   f. DESCRIPTION (TEXT)
   g. PAID (TEXT)
   h. PROPOSER_NAME (TEXT)
   i. DATEPOSTED (TEXT)

---

8. Table_Skills_Master

a. SKILL_ID (INT PK)
b. CATEGORY (TEXT)
c. SKILL_NAME (TEXT)

---

9. Table_Skills_Student
    a. STUDENT_ID (INT PK)
    b. CATEGORY (TEXT)
    c. SKILL_NAME (TEXT)

The tables marked in red are tables that were created but not used. These tables should be used for future iterations of the project. They are already created and only Table_Project_Proposed needs another column for a status of the proposed project. The tables are linked based upon their IDs they use. The following relationships are used:
Table_Profile_Students ← Share ID → Table_Skills_Student
Table_Project_Index ← Share ID → Table_Project_Skills
This allows projects and students to be matched based upon their skills in the skills' tables. SQL is then used to find projects and students in common based upon Category or Category & Skill_Name.

Flows

UI routing flow:

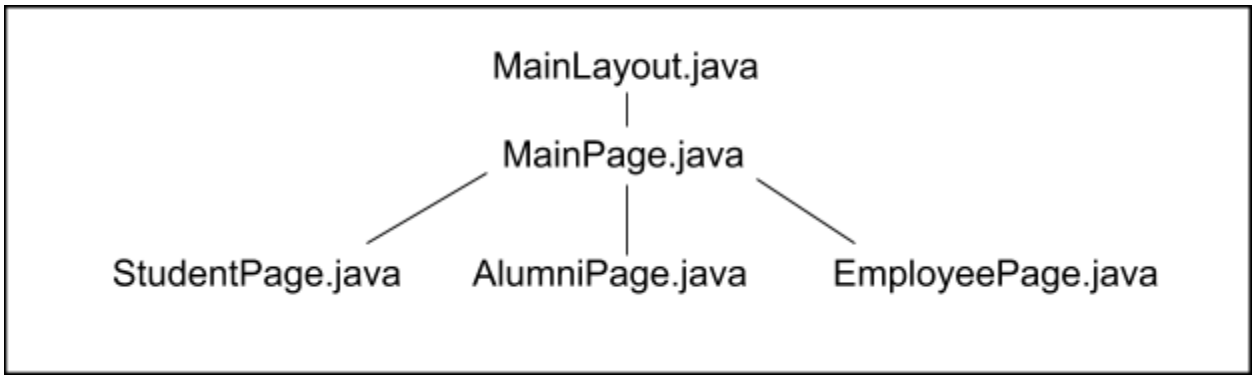

Figure 1: The routing layout from the main page to the 3 user pages

The figure above, Figure 1, show how the routing from the main page to the 3 different user view classes is done. MainLayout.java does not contain anything other than initializing the routing. MainPage.java is the page that controls which view gets set depending upon the user. The MainPage.java class is the view in which the login functionality should be written in, but currently only contains the window with three buttons to click on the button labeled with user they want to view

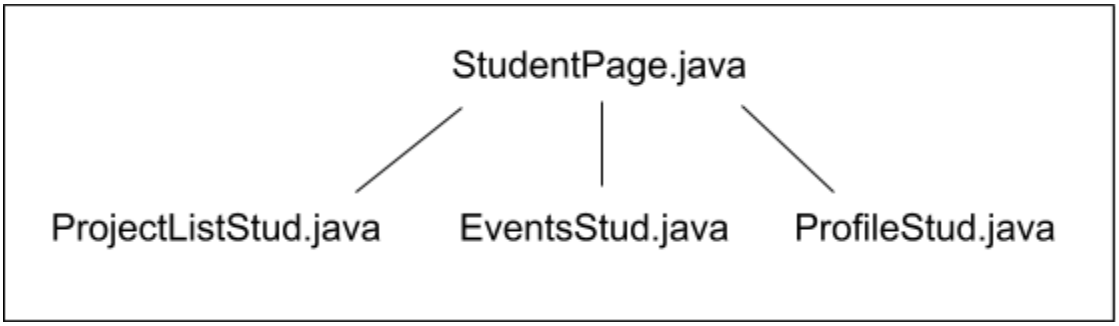

Figure 2: The routing layout for the student view

The figure above shows the student view. The student layout has 3 navigation buttons, projects, events and profile. The projects navigation button routes the projects for the student to view projects or propose a new one. This functionality is in the ProjectListStud.java class. The events button routes the calendar that shows all the Dyer Center events and is in the EventsStud.java class. The profile navigation button routes to the profile page for the student and is in the ProfileStud.java file.

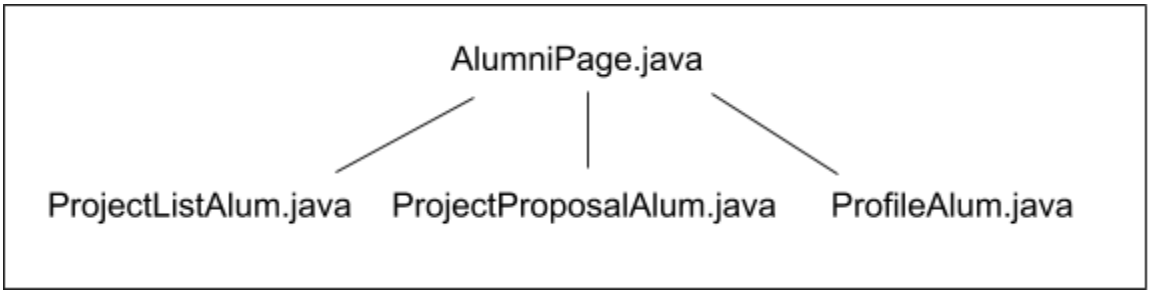

Figure 3: The routing layout for alumni view

The figure above shows the alumni routing. The alumni page navigation also has three buttons for projects, project proposals, and profile. The projects view is where the alumni can view the projects in ProjectListAlum.java class while the project proposals is in the ProjectProposalAlum.java. The profile view for the alumni is in the ProfileAlum.java class.

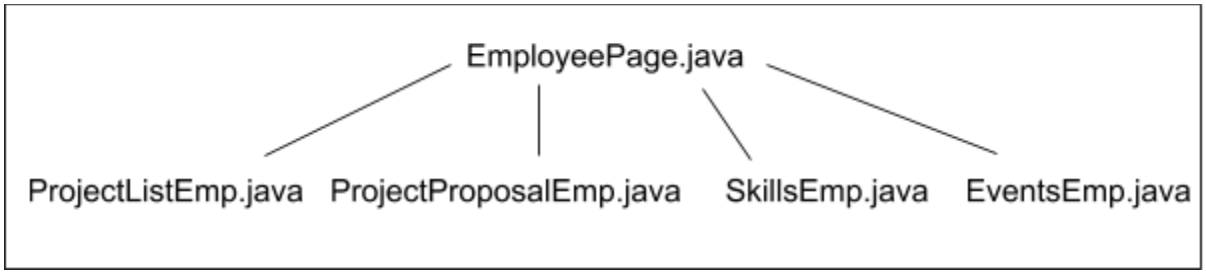

Figure 4: The routing layout for employee view

The figure above shows the routing for the employee page. The employee page has four navigation buttons for routing to projects, project proposals, skills list, and events. The projects view is where the employee can view all the projects which in the ProjectListEmp.java. The route for the project proposals is the ProjectProposalEmp.java file. The skills list button routes to the SkillsEmp.java class which provides the employee to be able to edit the skills list that projects and students match upon. Then, the events button routes to the EventsEmp.java class where the employee can add or remove events on the calendar.

Database Flow for Matching Process:

The goal of the Dyer Center Web Application was to allow students to match on projects according to their skills associated with their profile. Skills are composed of two pieces: the category the skill belongs to and the name of the skill itself. When the student selects and updates the skills they have on their profile page, a call is made to the back end that places their student_id and skills in the Table_Skills_Student table. This data in the Table_Skill_Student table is checked for when the projects page is opened. The list of projects is originally populated with all projects found within the Table_Project_Index table. But when the filter for matching projects only is applied, a sequence of back end calls are made to the database. First, the student_id is found is Table_Profile_Students by using the student's email. Next the student_id is used in Table_Skill_Students to get the list of skills that the student has associated with their profile. After that, a list of those skills is constructed and one by one each skill is looked up in the Table_Project_Skills table. If a project contains that skill it is added to a set to ensure a project is not added multiple times. Once every skill that a student has in their profile is checked in this table, the set of all projects that the student matches with is shown on the projects page.

However, if a student ends up not matching any skills, as they are performed in the aforementioned round by checking if the skill matches based upon its category and name, the check through the Table_Project_Skills table is done again. But this time, if a skill that the student has is within the same category that a skill associated with a project has, then that project is matched. This is so that students are at least encouraged to express interest in a project and do not feel discouraged when they do not have the exact skills that a project calls for.

It is advised that statistics be implemented within this flow so that students can see why they matched to a project, showing which skills they matched on or what categories, if the worst case situation where they do not match exact skills occurs, their relevant skills are within.

Email

The email functionality was set up using the JavaMail API via Gmail SMTP server. That was implemented via a secure connection known as Transport Layer Security (TLS) which is a security protocol that encrypts emails to protect their privacy.

Email for the application has been setup with 2 default emails that were created.

- Username: ignitemDyer@gmail.com | Password: ignitem470
- Username: dummyDyer@gmail.com | Password: dummypassword

These two emails work with the UI to send a project proposal to the Dyer Center Employees. The first email, ignitemDyer@gmail.com, is the one associated with the Web App. All emails sent from the Web App are sent from the ignitemDyer@gmail.com account. The second email, dummyDyer@gmail.com , was created for development and testing. Right now in EmailSender.java, all emails are being sent to dummyDyer@gmail.com. This is a control so that the developer can open the emails within this account, and check on how the email functionality is working.

Current Setup / HardCode

As of our last sprint, we did not have single sign on functionality working. For this reason, we had to use dummy information in our database to demo our front end loading values from the back end. Within our code, calls to database tables are used to populate the front end to simulate what the application would look like for each type of user. With the student pages, the email "goodwayj@lafayette.edu" is used and is stored within the Table_Profile_Students table in order to load values for our demo and testing reasons.

When coding for the future iterations of this application, it is crucial that single sign on is used and implemented before beginning to bother with other functionality. The single sign on feature sets the flow for the application and determines what type of user has logged in to the web application.

As of our last sprint, if "goodwayj@lafayette.edu" is not within a developer's back end and these code calls still exist, the database will throw an error for being unable to find information regarding the user. These code lines should be commented out until the single sign on is completed.

Calendario 2019 Calendar

For the Spring 2019 iteration of CS470: Senior Project, a team was assigned to make a calendar project that worked off of the school calendar. Their functionality of download events from the school calendar and showcasing it on their project can be utilized/mimicked to provide the necessary functionality for the Dyer Center Web Application. It is unnecessary to reinvent the wheel, and using the Calendario 2019 group's code to perform the calendar functionality should help greatly. They were able to not only show events, but include pictures associated with the events, as the current calendar only showcases text regarding the events.