École Pour l'Informatique et les Techniques Avancées

PROJECT REPORT

Identity Access Management System

December 2016

Submitted By

SIVARAMAN AROUMOUGAM

Project Guide

Professor. Thomas Broussard

TABLE OF CONTENTS

1. SUBJECT DESCRIPTION

We have a web application; this application has five web pages. The first page displays the login page which has the user-name and password fields.

Once the login is successful, the user is forwarded to the second page which is the welcome page containing links for creating a user and searching a user. Which are the next two pages of the web application.

The first page also contains a link to the registration page where a user can register with their name and e-mail address.

The Authentication Process:

The authentication part of the application allows two things –

- To deny access to unauthorized users
- To adapt the application as per the connected users' rights.

The authentication is successful only if the username and password matches and exists in the IAM system. One can manually check the presence and read a record in the database present in the system.

2. SUBJECT ANALYSIS

2.1 Major Features

The web application makes it possible to create a user who will then be able to login to or logout of a website (iamWeb).

On the website, the major features include creating an identity, modifying an identity, searching an identity and to delete an identity (DAO)

2.2 Application Feasibility

The project uses Tomcat server and Maven, Spring, Derby and Hibernate frameworks.

Maven assists in having a great overview and a definition of what is needed, also known as the dependencies.

Spring framework is used to control the ApplicationContext and assist in managing when to create instances of different classes when needed. This is called dependency injection, and by using the @autowired annotation spring will inject the instances as per the needs.

Derby is used because of the ease of use. By using the embedded mode, a local database is created, which is running in the application. The configuration is easily maintained in the ApplicationContext.xml

Hibernate framework is used help get rid of the complex SQL statements and uses HQL (Hibernate Query Language) instead. This results in more simple and smaller codes and is again controlled in the ApplicationContext.xml file.

The Server used is ***Apache Tomcat*** which is an open source server and is easy to set up with the web application. The web applications' main features will be realized through the user of servlets, Java Classes and JSP files.

2.3 Algorithm Study

The login or logout mechanism is managed by using sessions, where it must check if a user session is stored in the current session. To achieve the functionality of creating users and identities, servlets controlling the request must perform a stepwise algorithm which checks for the correct user input, duplicates and create the identity in the database.

Almost the same approach must be used in the functionality of updating and deleting an identity. Again, a servlet must check for correct user input from the client and update or delete the identity

The search function is achieved by parsing the search criteria from the client to the call of the implemented search method from the IamCore project. The search method looks for identities, that matches the search criteria (Display name)

2.4 Scope of the application

The project is mainly focused on the server side aspect of the website. Even though the project uses the Bootstrap framework to achieve a quick and decent front-end design, the design is limited and no JavaScript is applied.

Furthermore, it is not very flexible nor is it scalable regarding the changing or adding of attributes in the model of identity.

Additionally, a more comprehensive validation of user input could be implemented both on the client side as well as the server side.

3. THE CONCEPT

3.1 Chosen Algorithm

As mentioned earlier, the login process is managed using session. Here, the login servlet calls the SessionFacroty model, to check if the user session is set in the current session. If it's not set, the response will be redirected to the first page, which is the index page or the login page, with a message asking the user to login. If the user is successful in logging in, the corresponding user object will be set in the session. A request to the logout servlet will check if a user is set in the current session and remove it, if it is present. Otherwise, the response will be, once again, redirected to the index page.

When a request is made to create a user, the servlet and identities, servlet RegisterUser performs a validation check of the different inputs sent in the request. The validation checks if the username is valid and if it already exists in the database and if the password is valid. If the requirements are met, a message saying 'Registration Successful' will be displayed.

When a user submits to create an identity, a request to the RegisterUser servlet is made and it will follow a very similar stepwise algorithm as the RegisterUser module. If the requirements are met, a new user identity is created in the database

When a user sends a request to SearchIdentity servlet, it will create a new identity with the given parameter, and searches for identities that matches any of the search criteria. The result is returned in a list which is stored in the session.

For Update Identity validation is made for all user inputs and an update in the database will be requested by the servlet. If no input is given, an update will be made without any side-effects since the identity will not be changed. **In this project, the e-mail is blocked from being edited** (We cannot change the e-mail address of the user)

For the DeleteIdentity, the servlet gets an IF from a hidden form field in the JSP file search-identity, which is associated with the corresponding identity. A new identity is created with the ID requested, and is deleted from the database using the ID

3.2 Data Structures

The data structures used in the project is primarily from the built-in list of Java which is used to store identity objects.An example of its usage is in the IdentityHibernateDAO and AuthenticationServiceDAO implementation of IamCore, where a call to the search method will store all found identities in a List, which is returned from the method.

Through the servlet search_identity, the list is set in the current session, as an attribute. The result will then be displayed on the corresponding view (JSP file), where JSTL will access and print the result.

3.3 Global Application Flow

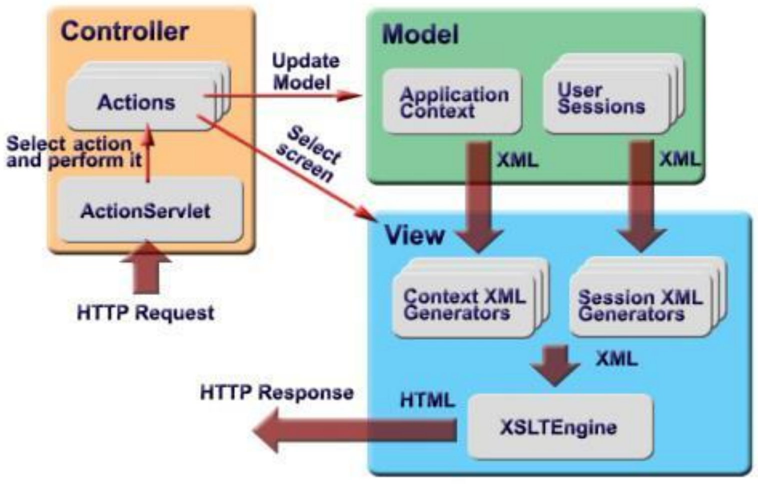
The overall architecture of the web application is implemented using the model-view-controller (MVC) design pattern. The views display information from thee model and shapes what the user views.

The model is the collection of database requests, methods and classes, which defines the program and its functionalities.

Finally, the controller requests and dispatches the content of the model through Hypertext Transfer Protocol (HTTP). The controller handles the user requests and defines the corresponding requests to the model, which subsequently defines the view (MVC Design Pattern)

The different modules in the IamWeb project interact with the Database and the controllers. The model invokes calls to the database and manages methods and functions of the sessions. The model also contains an authenticator which is used as an authentication mechanism (Server side validations of user input).

The application uses the DAO (hibernate implementation) in the IamCore project. The Hibernate implementation is chosen, which is used for creating or searching or editing or deleting identities in the database.



The servlets work as controllers and handles the user-requests and calling the correct models, whether it is in the IdentityHibernateDAO and AuthenticationServiceDAO or in the model package in the IamWeb project and chooses and sends the correct view as the response. The views are displayed using JSP files and by using the JSTL, it is possible to avoid excessive coding within the files.

4. CONFIGURATION INSTRUCTION

4.1 Application Context

The applicationcontext.xml consist of the overall pre-configuration of the database properties and maps the different classes needed for the dependency injection engine (used for auto wiring). By mapping the different classes, it is not needed to create instances, but only indicates where to put instances to make your application run. The application context consists of the following classes –

- User
- SessionFunctions
- HibernateDAO
- AuthenticationService
- SessionFactory
- DataSourceBean
- HibernateProperties

The User, SessionFunctions, AuthenticationService, HibernateDAO are all places as beans in the applicationcontext.xml file because of the mapping needed to use the auto wiring mechanism.

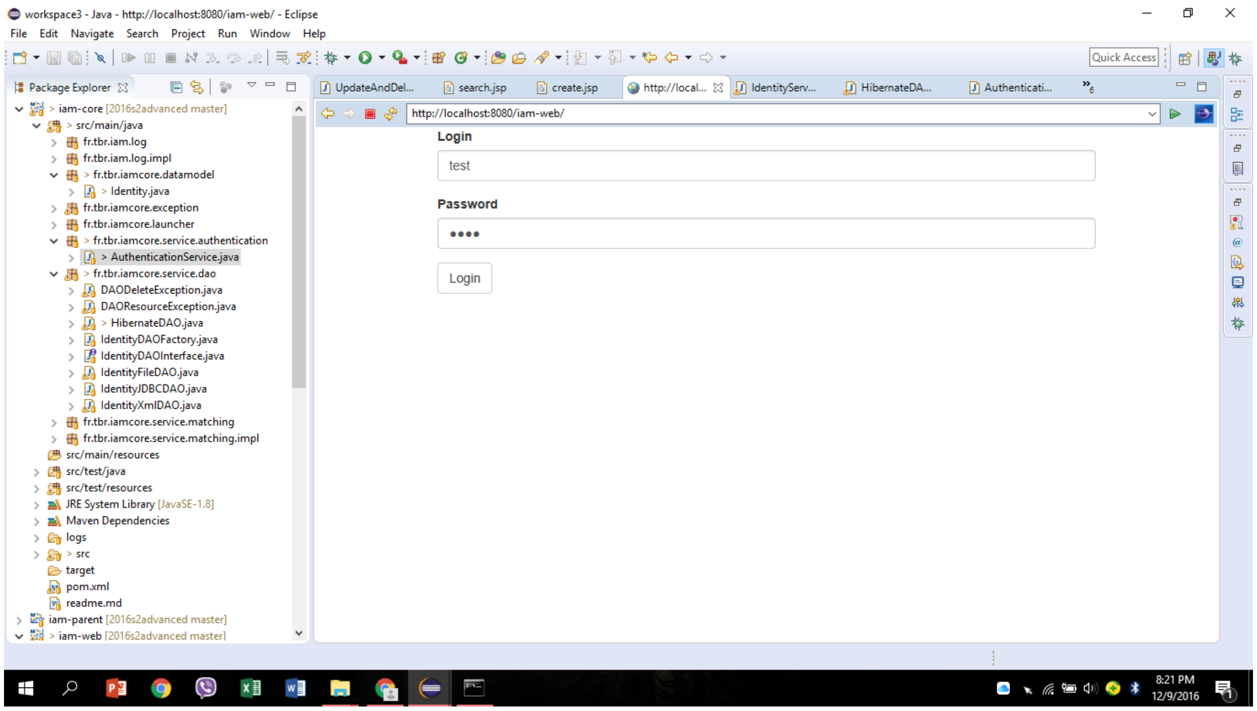
The BeanbasedSessionFactory is mapped as well, but is used as a session factory for the connection of the database. Furthermore, both the identity and user model is mapped using annotation configuration, which means that the classes are defined as entities, and each field or attribute is mapped, to tell hibernate, how to create the entities in the database, with its corresponding columns, DataSourceBean has the properties of the JDBC driver to the database. Properties like DriverClassName, URL, Username are configured here.

The HibernateProperties are the properties of the hibernate configuration to access the database through the framework. In this project, hibernate is set up with the DerbyDialect, which gives the ability to convert HQL into something the database will understand. Furthermore, the configuration is set to display SQL queries in the console.

Additionally, the property hibernate.hbm2ddl.auto is used to endure object representation will synchronize with the database. When starting the application, it is set as create, to start the database from scratch, with no users or identities in the database. After the first execution, the property should be changed to 'update' to keep the created objects in the database.

5. SCREENSHOTS

Login page:



http://localhost:8080/iam-web/

Login

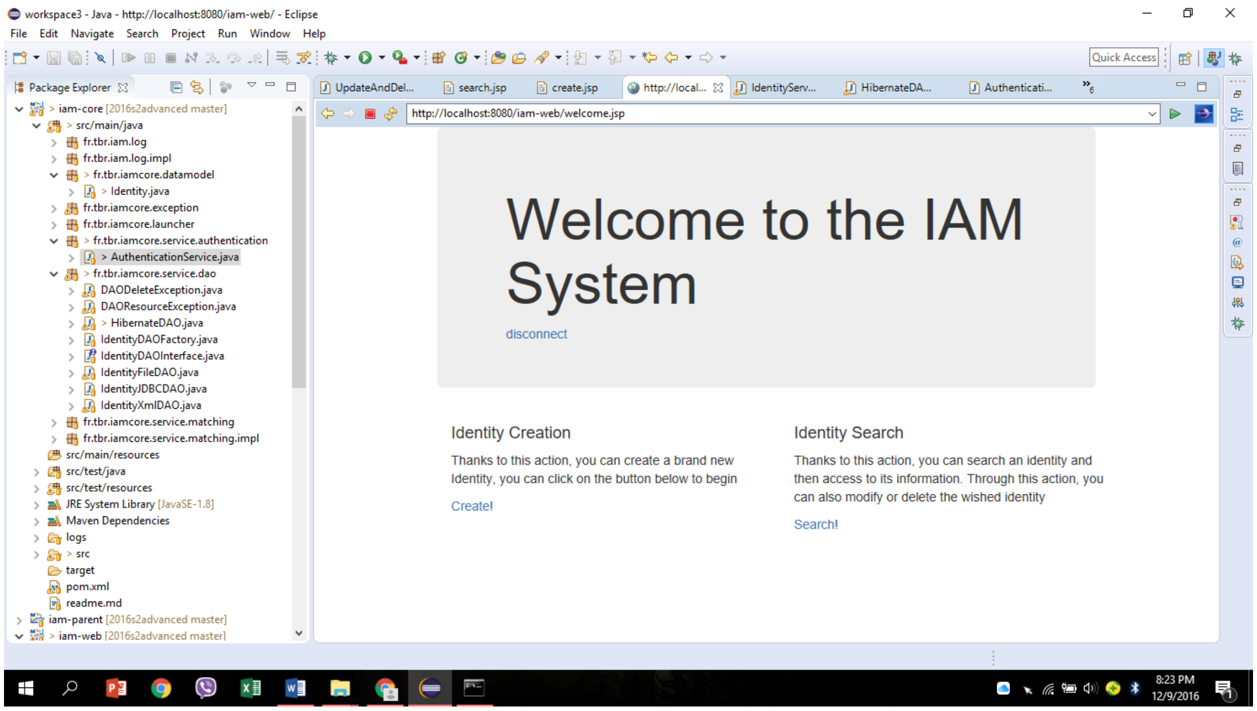
test

Password

••••

Login

Welcome page:



http://localhost:8080/iam-web/welcome.jsp

Welcome to the IAM System

disconnect

Identity Creation

Thanks to this action, you can create a brand new Identity, you can click on the button below to begin

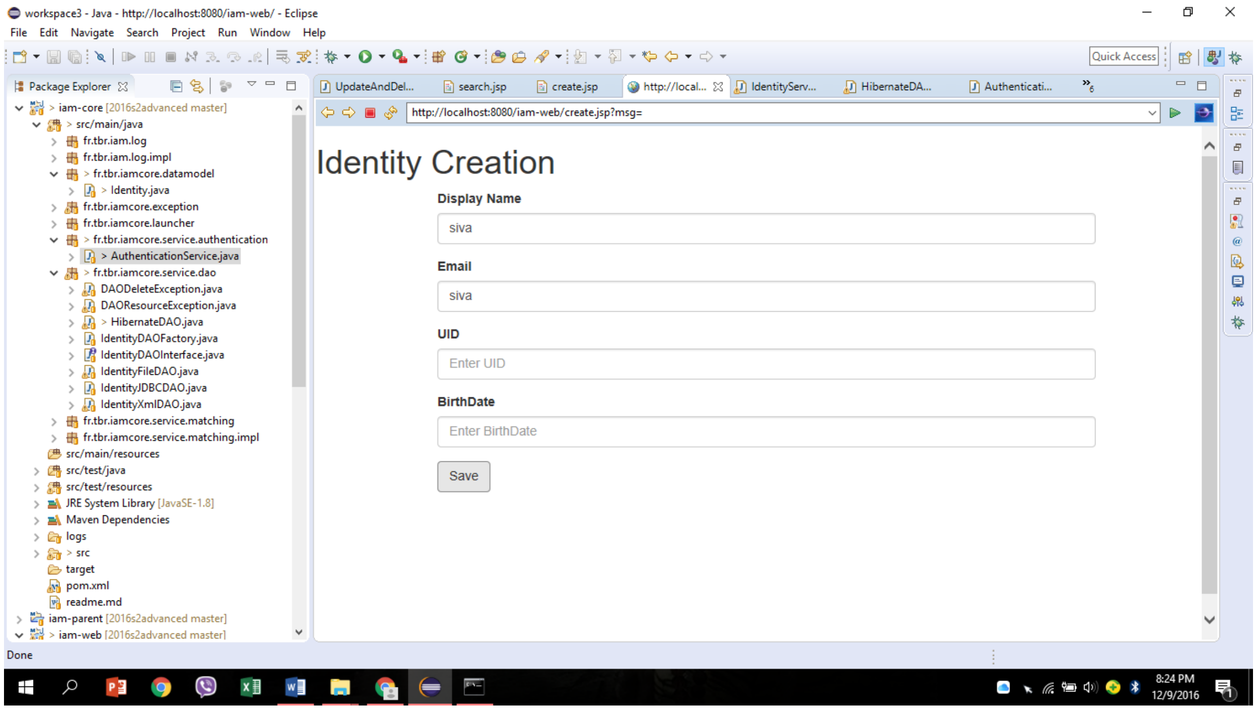
Create!

Identity Search

Thanks to this action, you can search an identity and then access to its information. Through this action, you can also modify or delete the wished identity

Search!

Identity Creation


workspace3 - Java - http://localhost:8080/iam-web/ - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Package Explorer
UpdateAndDel... search.jsp create.jsp http://local... IdentityServ... HibernateDA... Authenticati...
Done
8:24 PM 12/9/2016

```
iam-core [2016s2advanced master]
  src/main/java
    fr.tbr.iam.log
    fr.tbr.iam.log.impl
    fr.tbr.iamcore.datamodel
      Identity.java
    fr.tbr.iamcore.exception
    fr.tbr.iamcore.launcher
    fr.tbr.iamcore.service.authentication
      AuthenticationService.java
    fr.tbr.iamcore.service.dao
      DAODeleteException.java
      DAOResourceException.java
      HibernateDAO.java
      IdentityDAOFactory.java
      IdentityDAOInterface.java
      IdentityFileDAO.java
      IdentityJDBCDAO.java
      IdentityXmlDAO.java
    fr.tbr.iamcore.service.matching
    fr.tbr.iamcore.service.matching.impl
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  logs
  src
  target
  pom.xml
  readme.md
iam-parent [2016s2advanced master]
iam-web [2016s2advanced master]
```

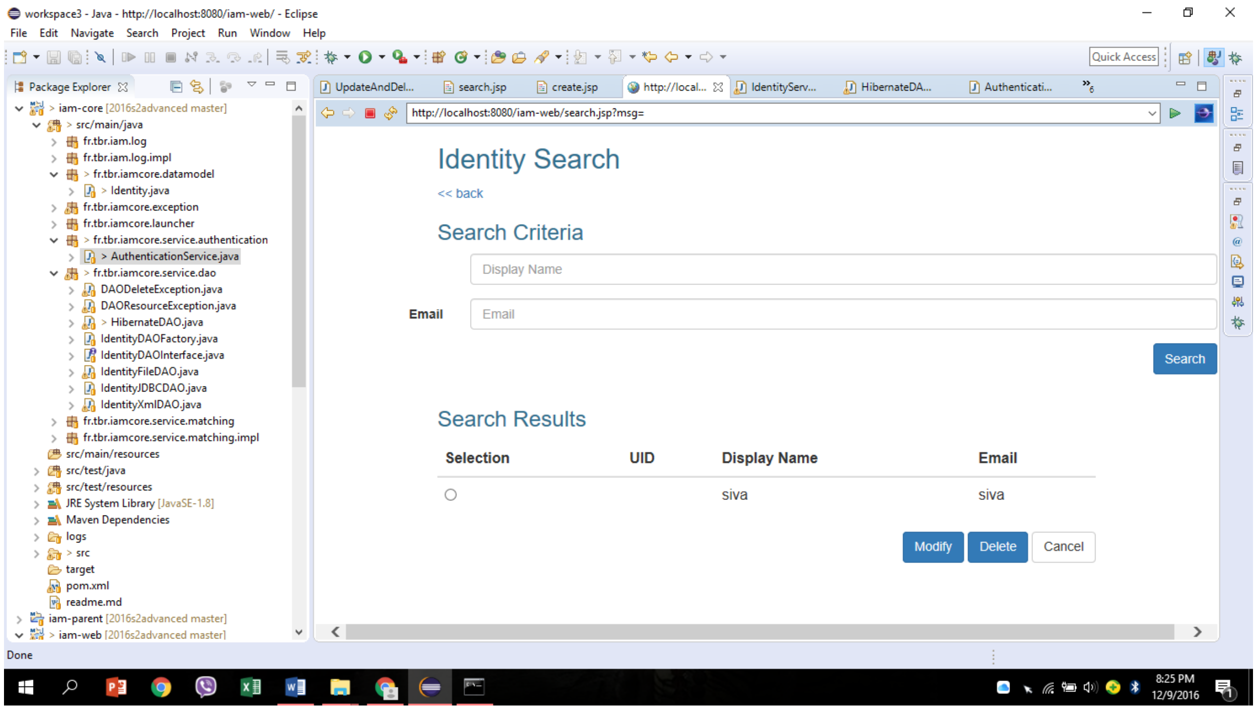
http://localhost:8080/iam-web/create.jsp?msg=

Identity Creation

Display Name	siva
Email	siva
UID	Enter UID
BirthDate	Enter BirthDate

Save

Identity Search:


workspace3 - Java - http://localhost:8080/iam-web/ - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Package Explorer
UpdateAndDel... search.jsp create.jsp http://local... IdentityServ... HibernateDA... Authenticati...
Done
8:25 PM 12/9/2016

http://localhost:8080/iam-web/search.jsp?msg=

Identity Search

<< back

Search Criteria

Display Name

Email: Email

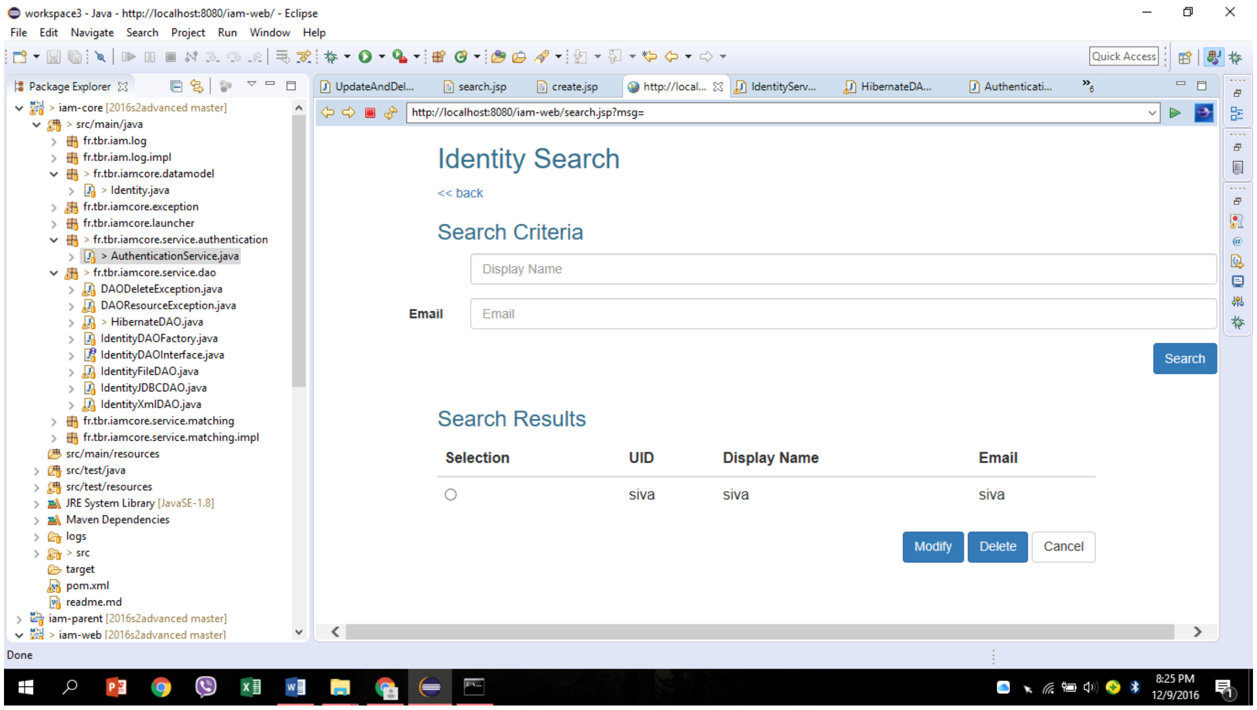
Search

Search Results

Selection	UID	Display Name	Email
○		siva	siva

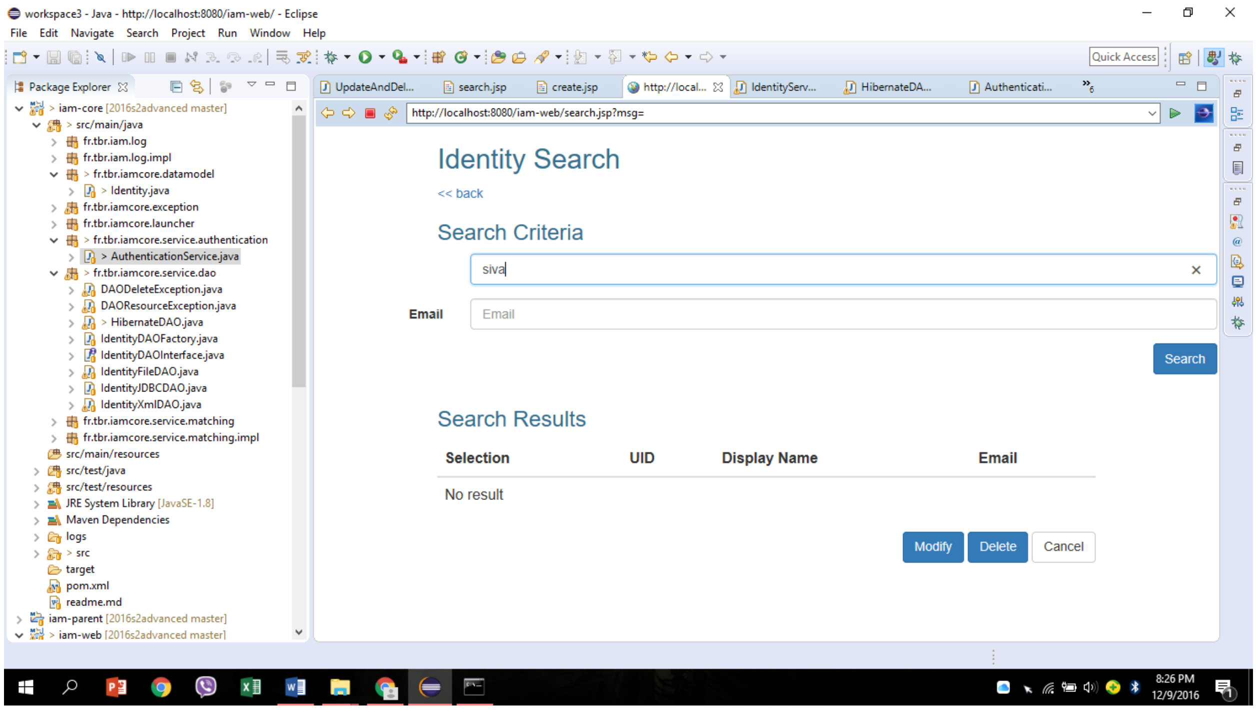
Modify Delete Cancel

Modify:


workspace3 - Java - http://localhost:8080/iam-web/ - Eclipse
http://localhost:8080/iam-web/search.jsp?msg=
Identity Search
<< back
Search Criteria
Display Name
Email
Search
Search Results
Modify
Delete
Cancel

Selection	UID	Display Name	Email
○	siva	siva	siva

Identity Delete:


workspace3 - Java - http://localhost:8080/iam-web/ - Eclipse
http://localhost:8080/iam-web/search.jsp?msg=
Identity Search
<< back
Search Criteria
siva
Email
Search
Search Results
Modify
Delete
Cancel

Selection	UID	Display Name	Email
No result			