

**A JAVA**

**Project Report**

**on**

Identity and Access Management System  
[**IAM** Project]

Contents

1.SUBJECT DESCRIPTION 3

[1.1]Introduction 3

[1.1.1]Overview: 3

[1.1.3] Motivation: 3

[1.1.4] Objective 3

[1.1.5] Development Environment 4

2.SUBJECT ANALYSIS 5

[2.1] Major Features 5

[2.2] Application Feasibility 5

[2.3] Data Description 5

[2.4] Expected Results 6

[2.5] Scope and limitations 6

3.CONCEPTION 8

[3.1] Chosen Algorithm 8

[3.2] Data Structures 8

[3.3] Global Application Flow 9

[3.4] Global Schema and major schema features 10

4.CONSOLE OPERATIONS DESCRIPTIONS 11

* Authenticate User 11

User authentication is done by an authenticate method , that takes input as username and password, and calls a validateUser method which connects to the database and authenticates the user if provided credentials are correct, else will not authenticate the user and system stops. 11

* Create an Identity 11

This console operations allows an authenticated user to create a new identity. Identity Uid, name and email are provided by the user.The method used for creating a new identity is createIdentity. 11

* Update an Identity 11

This console operation allows a user to update an existing identity. To update an identity the user must provide the uid of an identity, which is to be updated. The method used is updateIdentity 11

* Delete an Identity 11

5.CONFIGURATION 12

Configuration instructions 12

Prerequisites 12

Get project from GitHub 12

Database Connection Configuration 13

Schema creation 15

Database Configuration 17

BIBLIOGRAPHY 20

# 1.SUBJECT DESCRIPTION

## [1.1]Introduction

### [1.1.1]Overview:

This report contains all the detail from start to end of IAM project development.

**[1.1.2] Background:**

Keeping track of personal information disclosed on the Internet, as well as maintaining high standards of internal control and information security within companies has become increasingly complicated. Identity and Access Management applications lay at the core of resolving these types of problems.

Identity and Access Management (IAM) can be defined as following:

A comprehensive set of processes that enable end users to securely access a broad range of

internal and external IT systems that controls the digital representation of users and manage

information about identities.

### [1.1.3] Motivation:

This report and its subject, namely IAM, has been greatly influenced by our will to learn the concepts of a new programming tool **JAVA**.

What we encountered is that

* Java is Easy to write and more readable and eye catching.
* Most of the concepts are drew from C++ thus making Java learning simpler.

### [1.1.4] Objective

The main goal of this project is to develop an application using Java which can automatically connect to a database and store information in the database and also to retrieve, modify and delete the information in the database.

The following are the objectives that the application needs to satisfy:

1. **Authenticates** a user
2. **Modify** an Identity
3. **Delete** an Identity
4. **Search and Display** identity

### [1.1.5] Development Environment

PLATFORM USED :  **Windows 7**

LANGUAGE USED : **Core Java**

IDE:  **Eclipse**

DATABASE:  **Derby**

# 2.SUBJECT ANALYSIS

## [2.1] OOPS Major Features

* Highly user-friendly
* Platform Independent
* Easy to use
* Robust
* Data entry restricted to avoid errors
* Clean separation of various components
* Easy Modification

## [2.2] Application Feasibility

* This current application is a prototype of a system that can be created for employing a highly secured environment of Identity and access Management.
* The costs are much reduced as we do not depend on graphical interface, instead look for a high system performance
* Most of the components used such as the development platform, servers, and databases are open source.

## [2.3] Data Description

The data description and data access objects are clearly specified below.

The Schema for data is **<IDENTITY\_UID, IDENTITY\_DISPLAYNAME, IDENTITY\_EMAIL>**

IDENTITY\_UID **:** STRING, Unique

IDENTITY\_DISPLAYNAME : STRING

IDENTITY\_EMAIL : STRING

**DAOs**

* **Authenticate :** This module takes user name and password. This module validates auser before login to IAM system

Input parameter : User name, password

Output parameters : Authentication Accepted / Denied

* **createIdentity :** This process is used to create a new identity in the database.

Input parameter : Uid , Name, Email

Output parameters : Entry added to database

* **modifyIdentity** : This process is used to change any record already present in the database and updates the database accordingly.

Input parameter : Uid,Name,Email

Output parameters **:** Identity modified in the database with reference to Uid.

* **deleteIdentity**  : This process is used to delete an identity from the database.

Input parameter :  Uid

Output parameters : Identity deleted from the database.

* **Search Identity** : This process is used to display all identity from the database.

Output parameters **:**Identity display from the database.

## [2.4] Expected Results

* The end result of the application can be looked as a highly sophisticated, user friendly and secure tool created for Identity and access management.
* This tool is capable of authenticating the user, creating a new identity, display all identity, modifying an existing identity, and deleting an identity from the database.
* The database used is a Derby database.
* The tool needs to communicate with the database and return with the results in quick time.

## [2.5] Scope and limitations

**Scope:**

* Privacy: Online transaction, whether financial or exchange of information, could be greatly improved by the adoption of IDM solutions which focus on privacy.
* Improved user experience, Cost savings, security policy enforcement etc.
* Centralization of user administration.

**Limitations:**

* Possibility of decrypting the system password.
* Does not open  the  system  to  manage  permissions  and  attributes  of  users.
* Lack of added feature like in modern address book.

**Evolution:**

* We are working to enhance the IAM as a complete web-based software, users and login and keep details, export details, import to new system, by incorporating an attractive GUI and also send SMS from there and save other information.

# 3.CONCEPTION

## [3.1] Chosen Algorithm

The algorithm that we have used is the exact match algorithm. It can be seen below.

Searching is done with the **uid** and updated

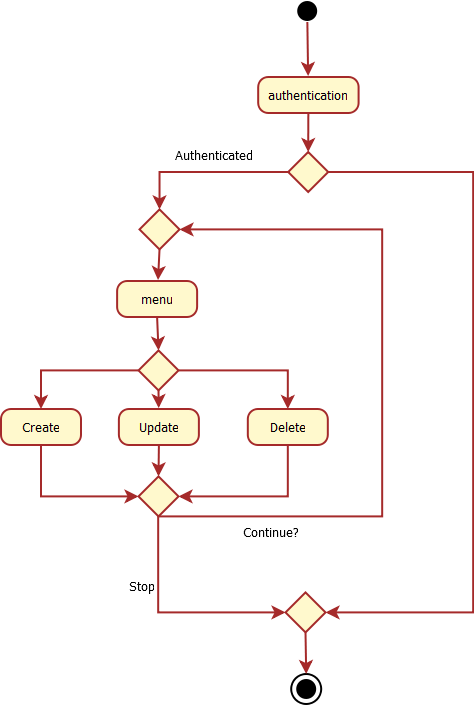
## [3.2] Data Structures

The data structures provided by the Java utility package are very powerful and perform a wide range of functions. These data structures consist of the following interface and classes:

However the data structure used in this project is **LIST ,**which is a collection of elements .List is used in the project to store the collection of identities.

**String** is also a widely used data structure in this project

## [3.3] Global Application Flow



## [3.4] Global Schema and major schema features

# 

# 4.CONSOLE OPERATIONS DESCRIPTIONS

Console Operations Implemented in this Systems are

1. Authenticate user
2. Create an Identity
3. Update an existing Identity
4. Delete an Identity
5. Search Identities

Each Operations are explained below

## Authenticate User

## User authentication is done by an authenticate method , that takes input as username and password, and calls a validateUser method which connects to the database and authenticates the user if provided credentials are correct, else will not authenticate the user and system stops.

## Create an Identity

## This console operations allows an authenticated user to create a new identity. Identity Uid, name and email are provided by the user.The method used for creating a new identity is createIdentity.

## Update an Identity

## This console operation allows a user to update an existing identity. To update an identity the user must provide the uid of an identity, which is to be updated. The method used is updateIdentity

## Delete an Identity

This console operation allows a user to delete an existing identity and the user is supposed toprovide the uid of the identity which is to be deleted. Method used is **deleteIdentity**

* **Search Identities**

This console operation retrieves all the identities stored in the database and output them to the user. Method used is **listIdentity.**

# 5.CONFIGURATION

* Username : admin(Case sensitive)
* Password : admin(Case Sensitive)
* Database : Derby
* Drivers : Derby JDBC Driver

**6.SNAPSHOTS**

# Configuration instructions

## Prerequisites

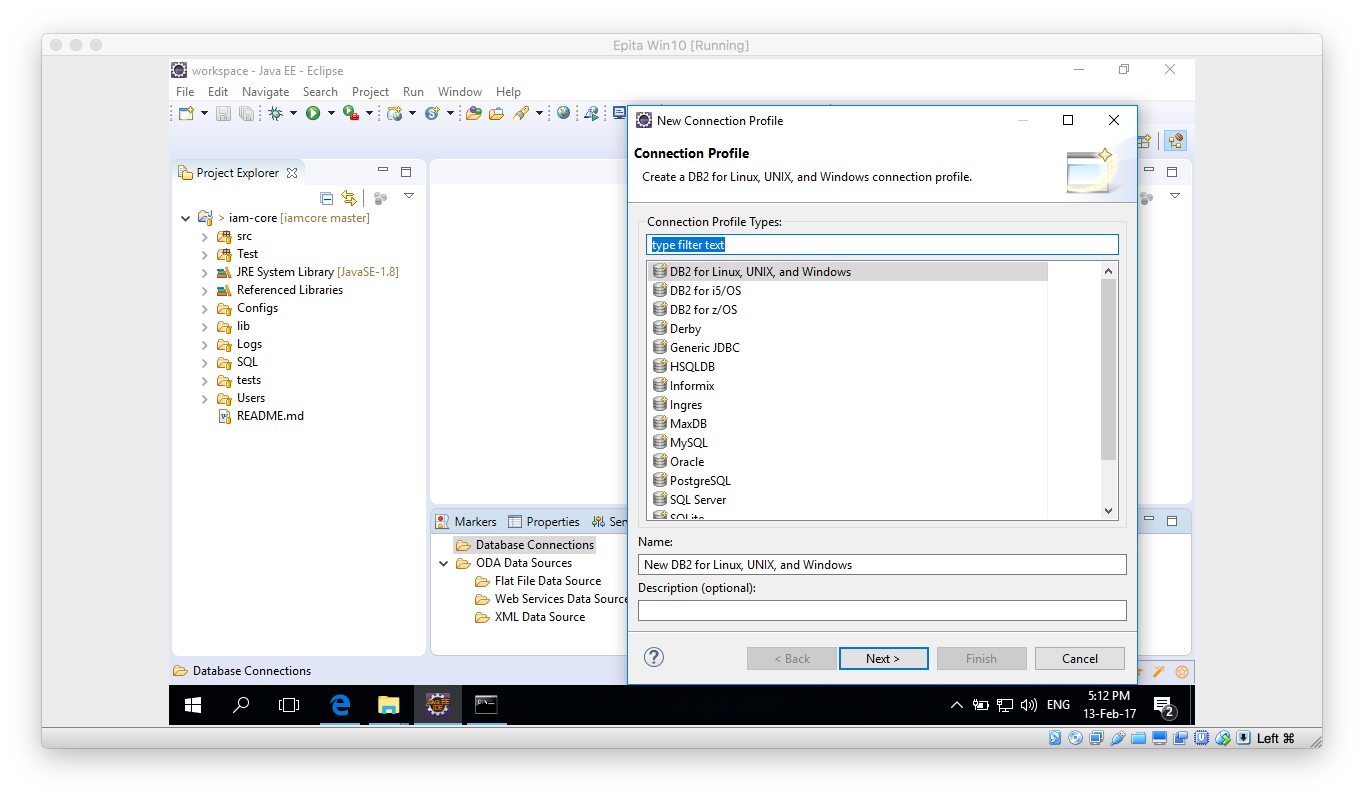
Please make sure you have installed the Java JDK, Eclipse Neon Java EE and the Derby 10.13.

## Get project from GitHub

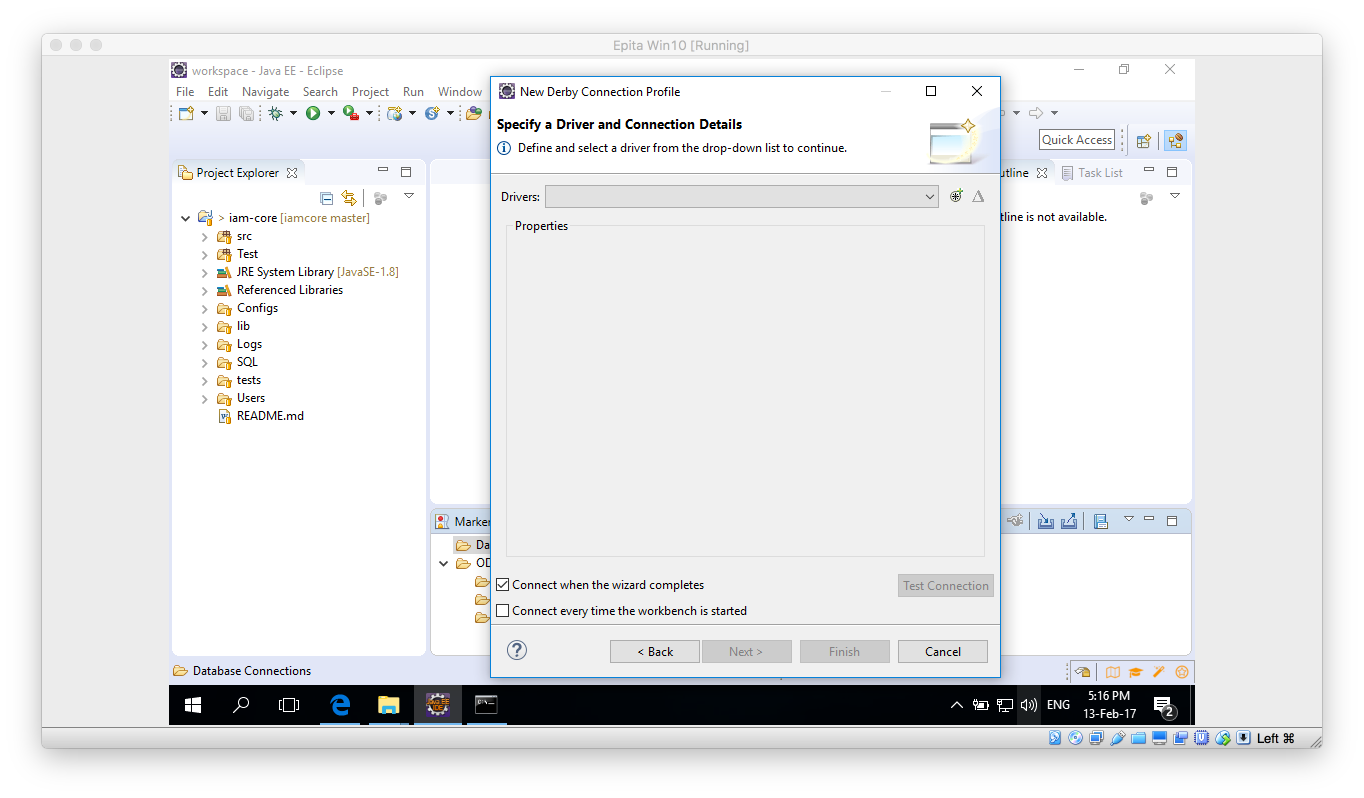
1. Open Eclipse, and go: File -> Import
2. Select Git -> Projects from Git click Next
3. Select Clone URI click Next
4. Paste the following URI: <https://github.com/omkar/iamcore.git> click Next
5. Don’t change anything on this selection and click Next.
6. Select the destination Directory for the project and click next.
7. Do not change any of the selection on this screen. Click Next.
8. No changes and click Finish.

## Database Connection Configuration

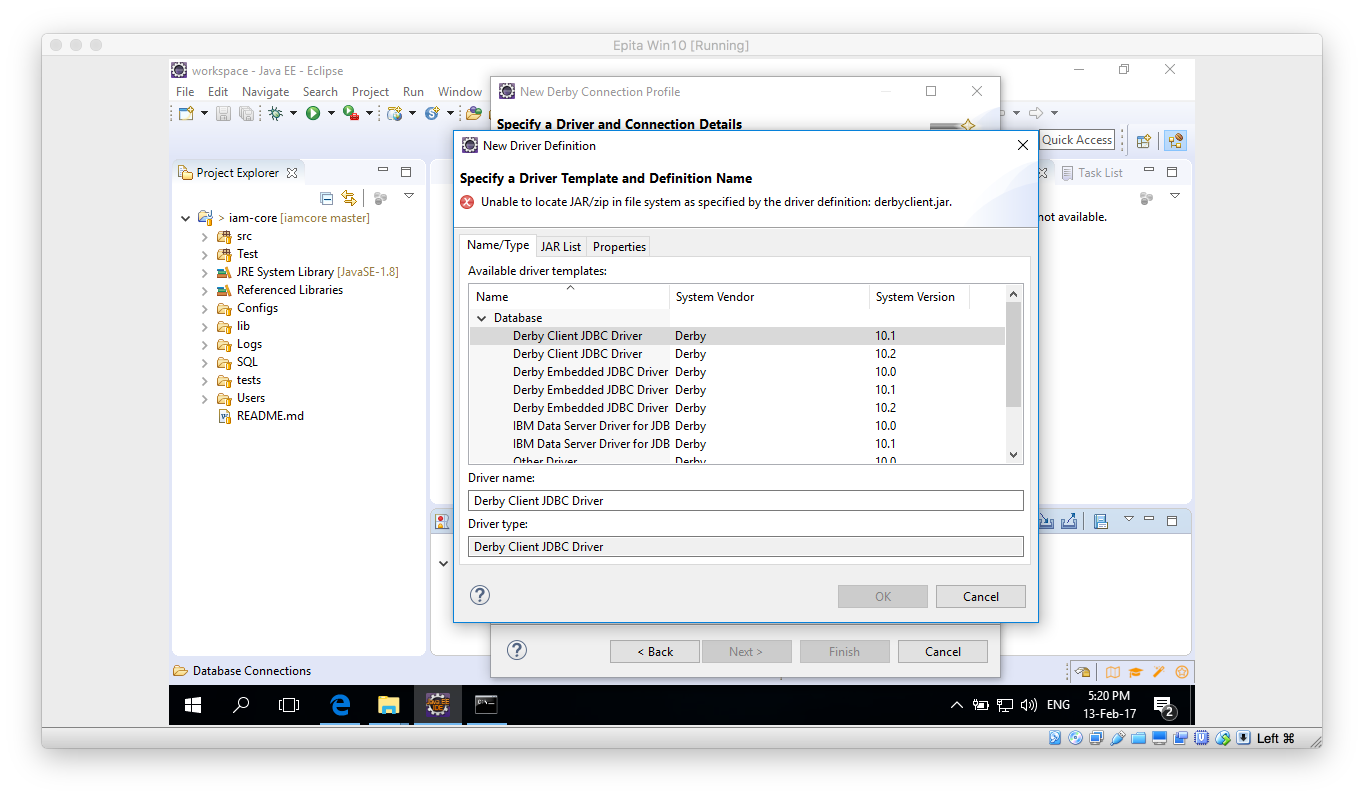
1. Open the Data Source Explorer window on Eclipse.
2. Right Click Database connections, select New.



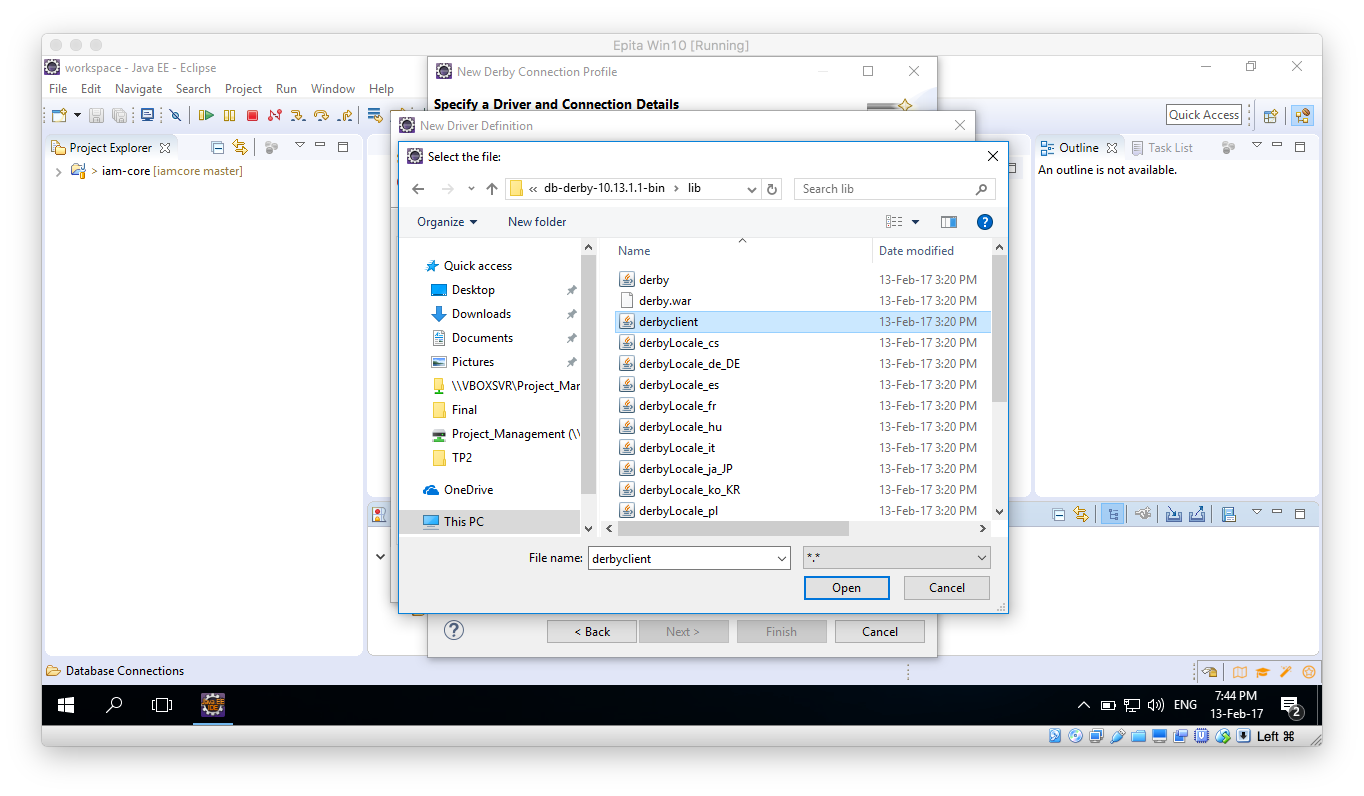
1. In the contextual window select Derby. Click Next.
2. In the new window click on the “New Driver definition button”



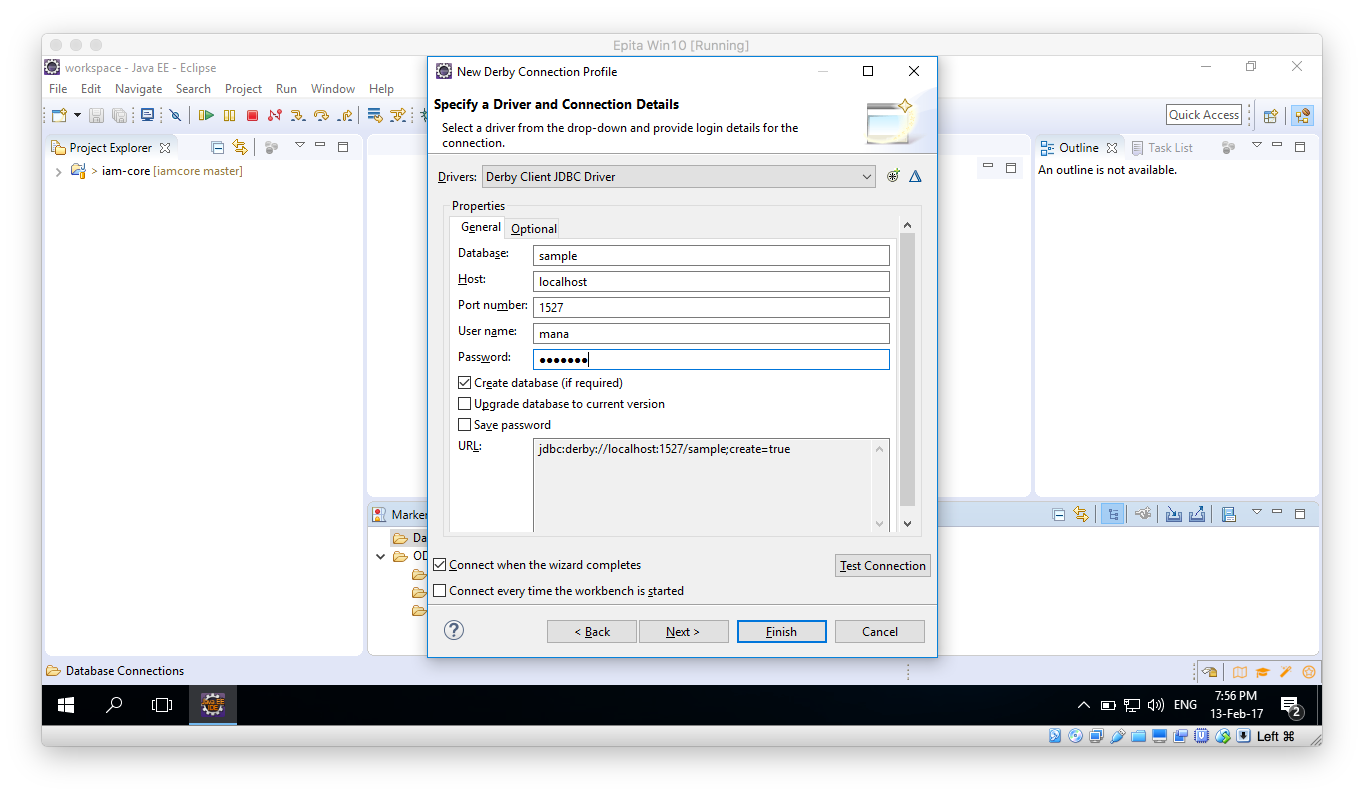
1. Select any Definiton on the Name/Type tab and then move to the JAR List tab.



1. On the JAR List tab click on Clear all button, so no files are listed. Then click on the Add and navigate to the lib folder inside your Derby database. Once there select the derbyclient.jar file.



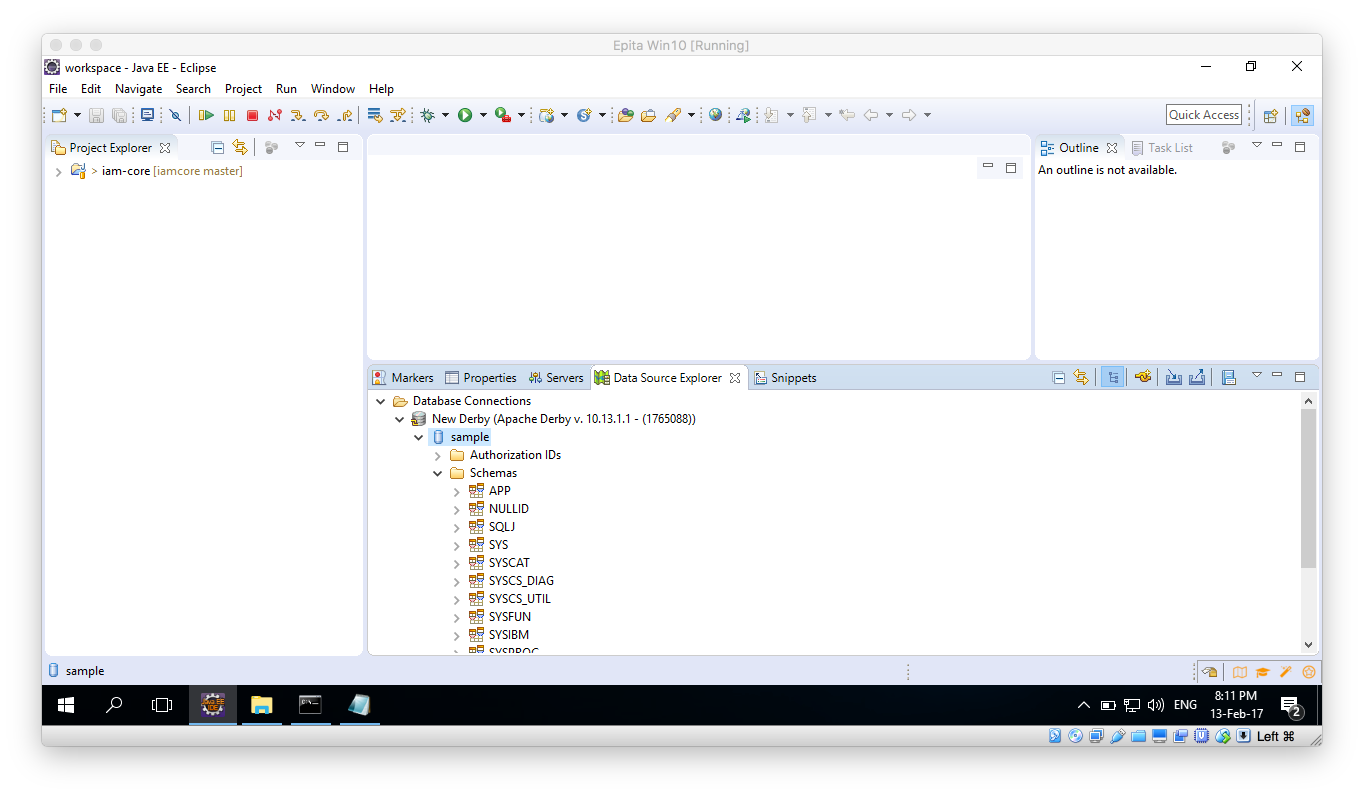
1. Click Open, then the “New Derby Connection Profile” window should be updated. Please take note of the URL and change the user and password if desired. Remember those because we are going to need them later on.



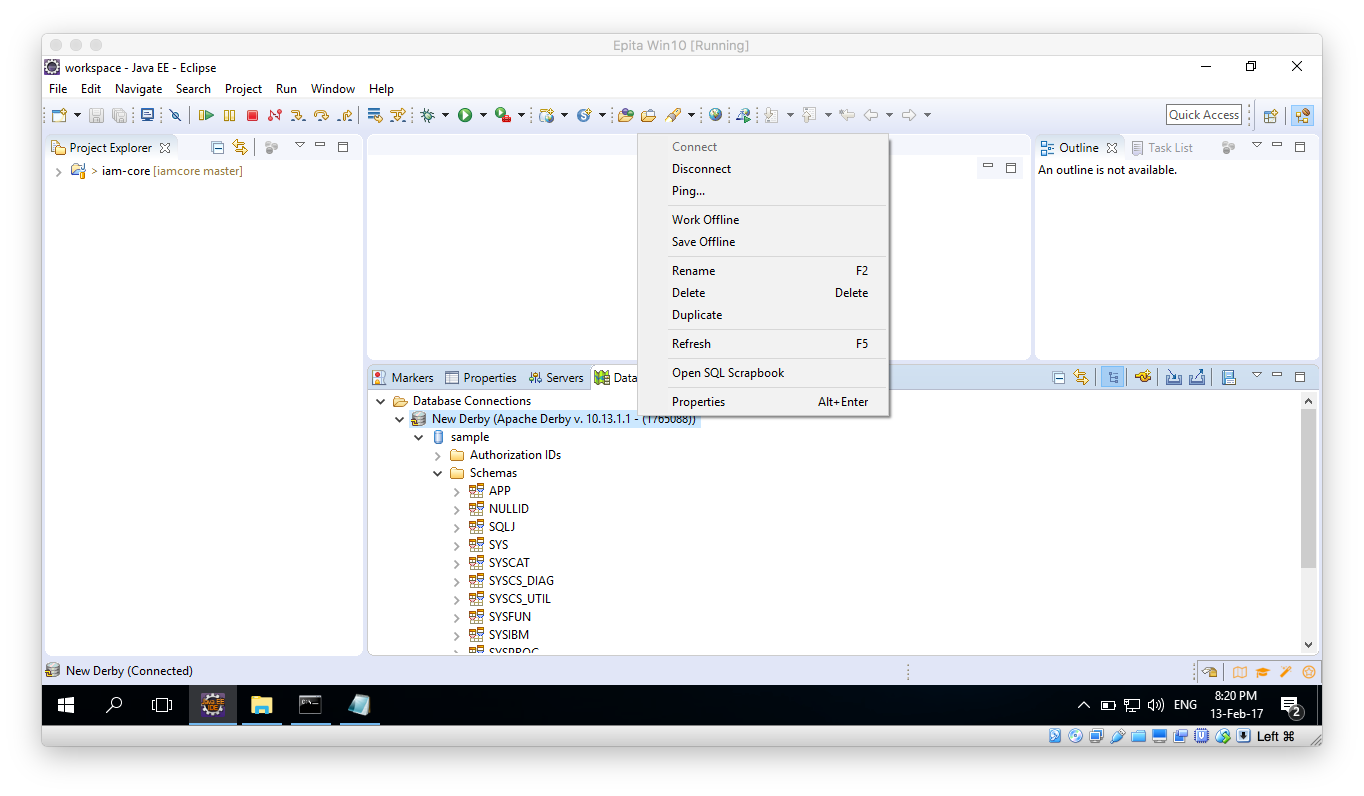
1. Make sure the Derby Database is running an click on Test Connection button, it should succeed.
2. Click on Finish button.

## Schema creation

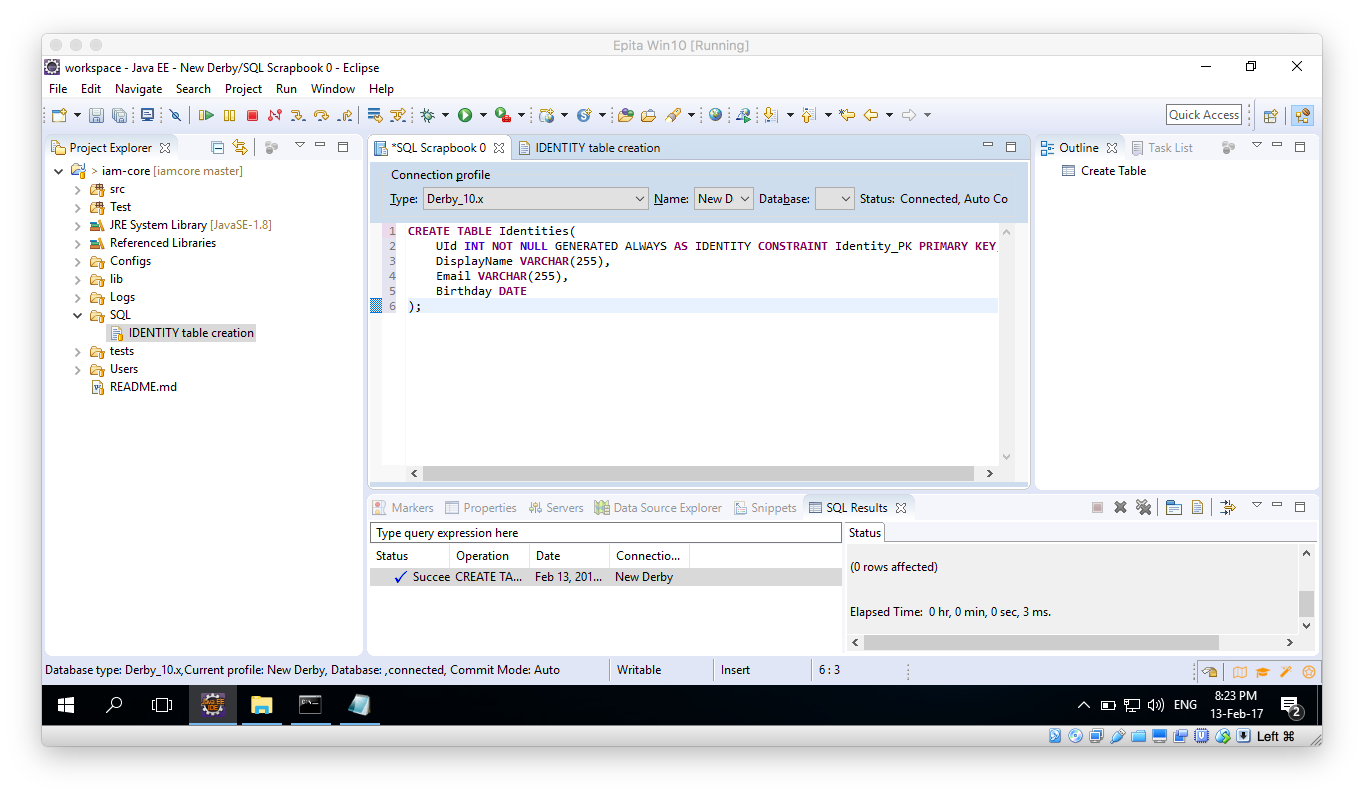
1. Once the above process is done, within the Data Source Explorer window you should be able to expand all the way to Schemas:



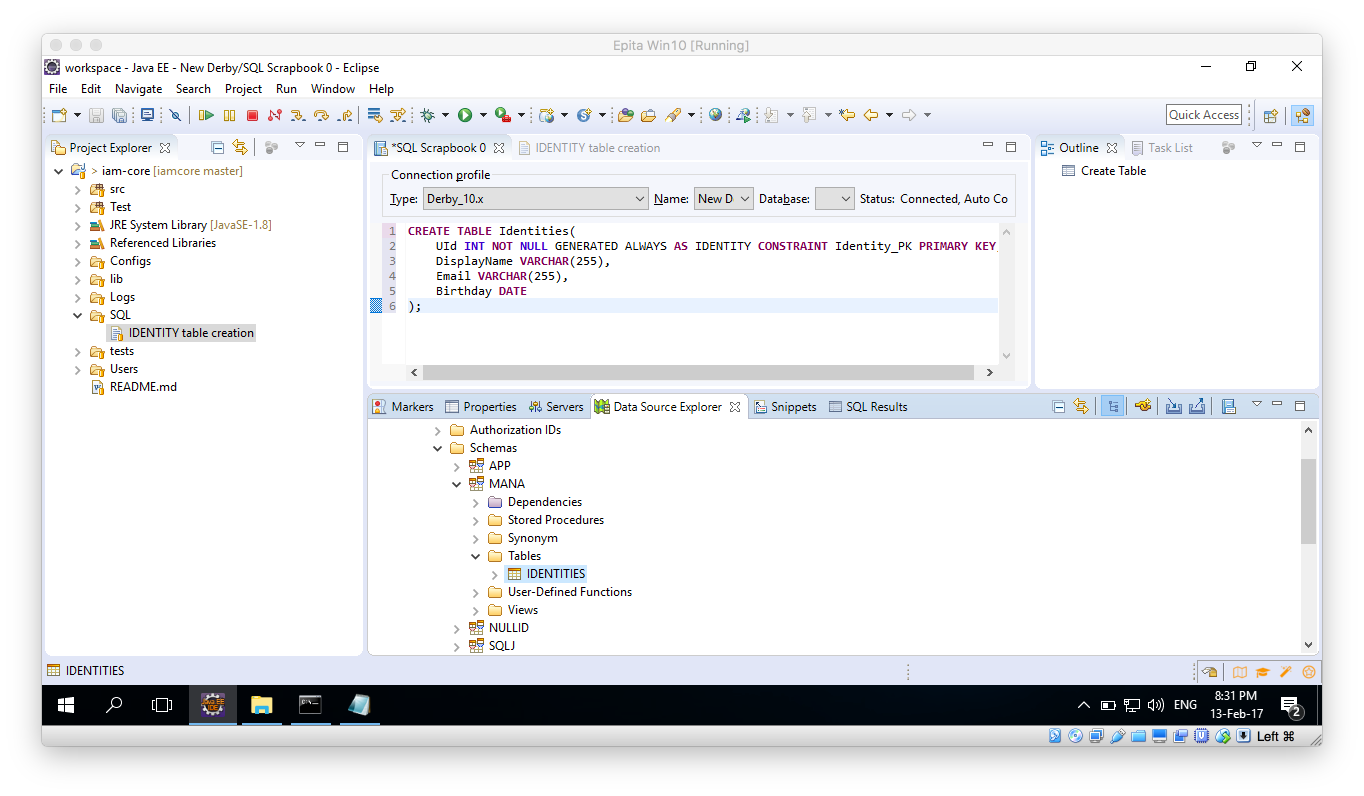
1. Right click over the New Derby and select: Open SQL Scrapbook:



1. Once the new window opens, in the Project Explorer navigate: iam-core/SQL/IDENTITY table creation and open the file. Copy the file contents to the Scarpbook window and execute.
2. Right click anywhere on the Scrapbook window and select “Execute All”.
3. Execution should be successful:



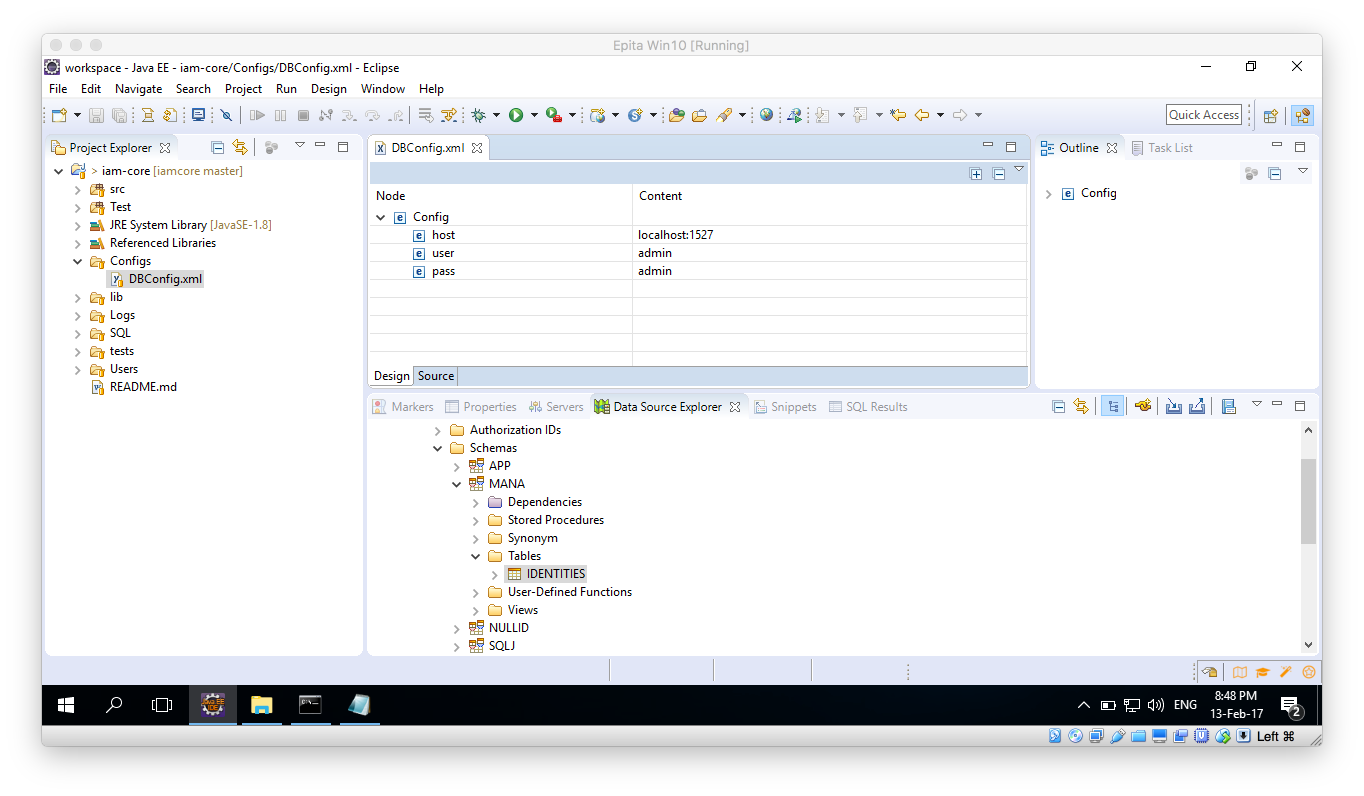
1. Go back to the Data Source Explorer window and right click on the Schemas folder, select Refresh. You should now see and Schema with a name that matches your user name, inside it the Identities table should have been created:



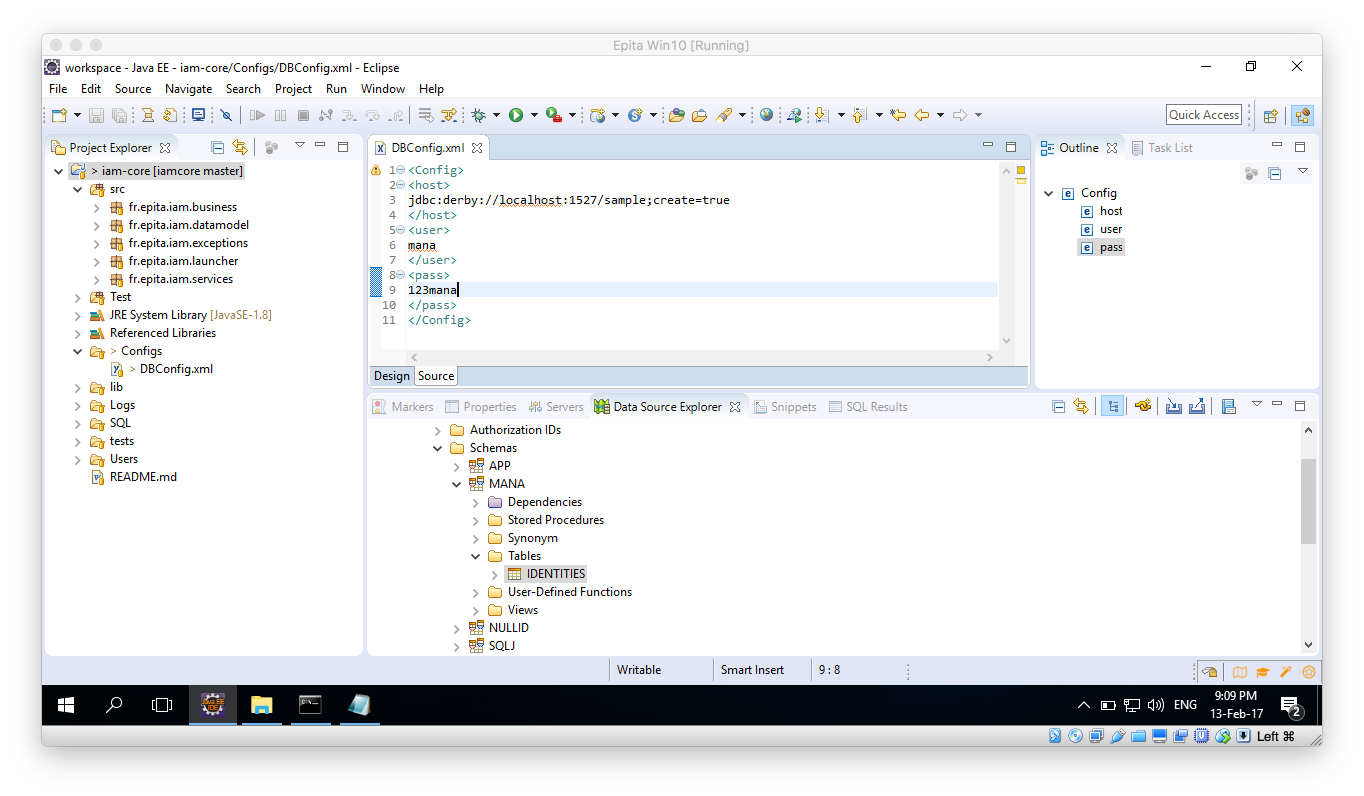
1. Once all of this is done close the Scrapbook and the Identity table creation sq. windows.

## Database Configuration

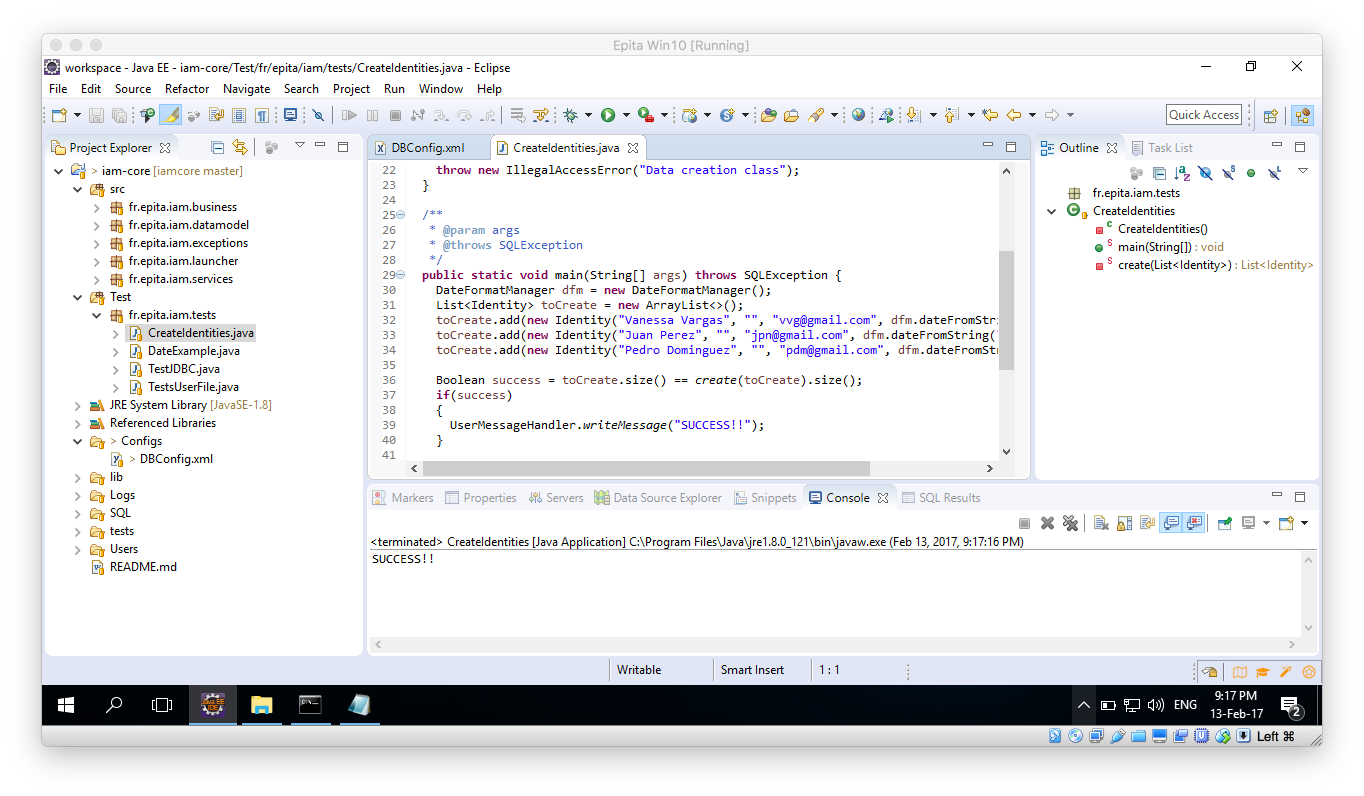
1. Open the XML found under: iam-core/Configs/DBConfig.xml
2. Select the Source view:



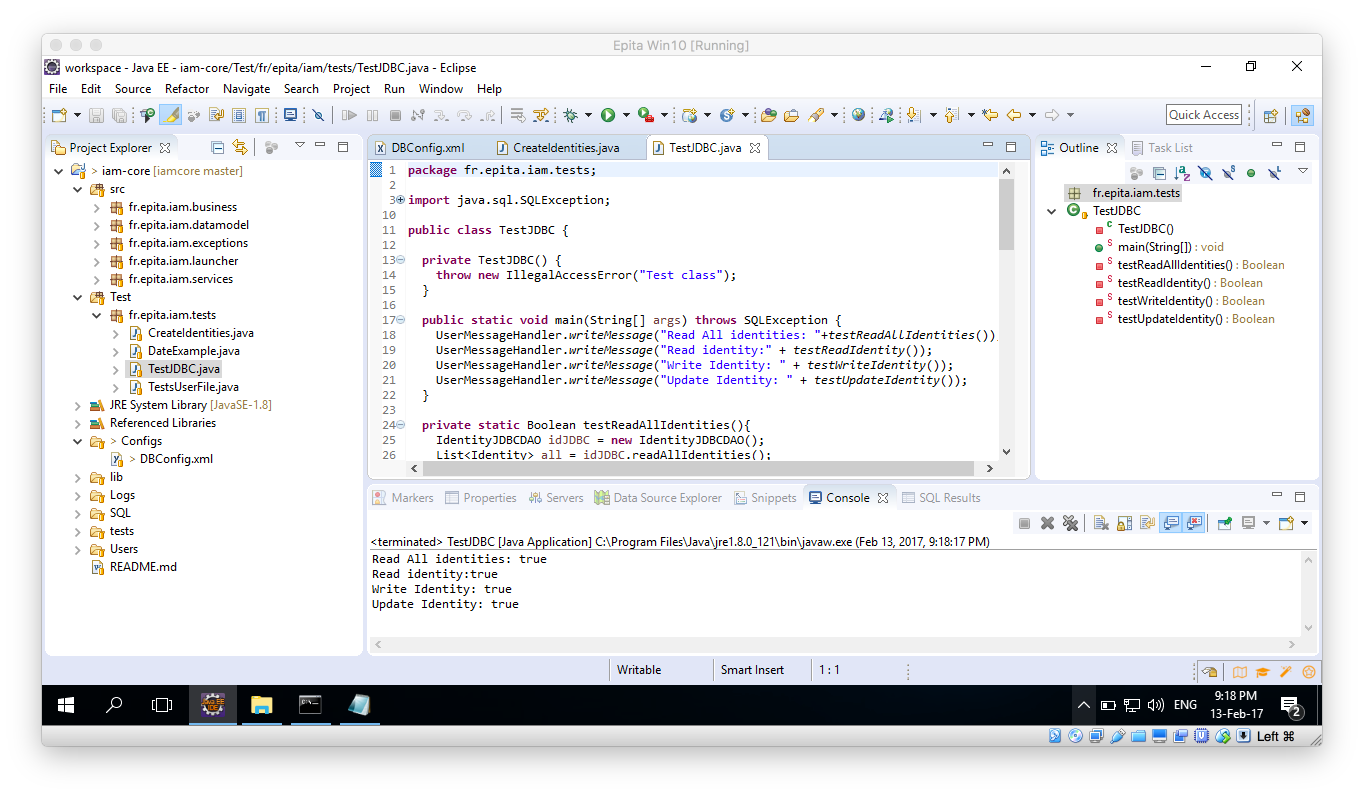
1. Modify the values with those according to your URL and user name and password:



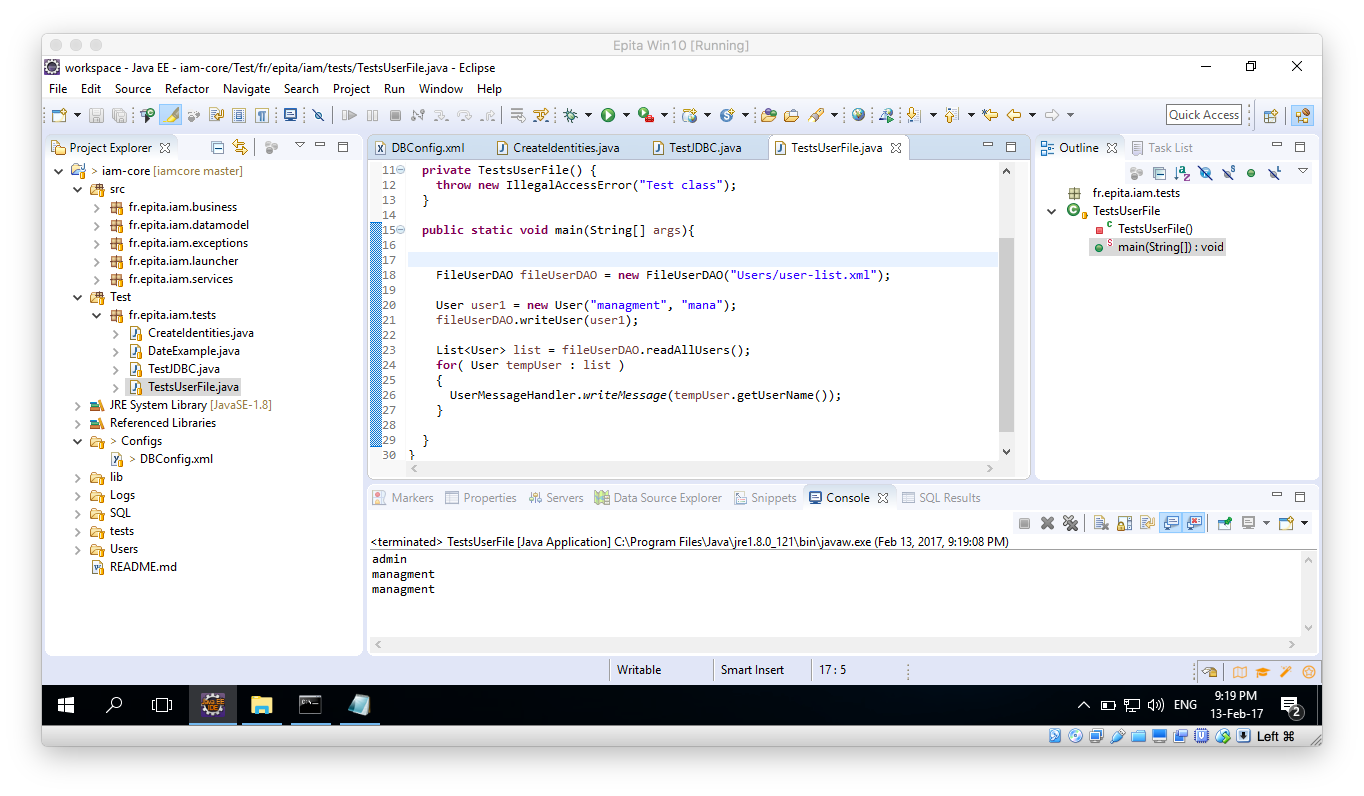
1. Save the file.
2. To verify everything is ok, Run the Test file: CreateIdentities.java (iam-core/Tests/fr.epita.iam.tests/CreateIdentities.java)



1. Further on you could try and run also the TestJDBC.java, all results should be true.



1. Lastly, to make sure everything is working properly, run the TestUserFile.java, the output should be like the following:



1. As explained above, there is no way to manage the Users on the current application, so they are all stored on an XML file located under: iam-core/Users/user-list.xml.
2. In there the name and password can be clearly seen, you can add your own users by modifying that file.

Now you should be able to run the IAM-Core project from the Launcher. Just look for the ConsoleLauncher.java class under the iam-core/sac/fr.epita.iam.launcher package.

# BIBLIOGRAPHY

* <http://thomas-broussard.fr/>
* <http://www.tutorialspoint.com/java/java_data_structures.htm>
* <http://www.tutorialspoint.com/>