AURA Security Plug-in

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Abhishek Soni (151257)

Under the supervision of

Mr. Prashant Thorat

(Lead Consultant)

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234(HP)

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled "AURA SECURITY PLUGIN" in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from February 2019 till now under the supervision of Mr. Prashant Thorat.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Abhishek Soni (151257)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Mr. Prashant Thorat

(Lead Consultant)

Dated:__________

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my mentor Mr. Prashant Thorat (Lead Consultant) who gave me this golden opportunity to do work under this project because he saw spark in me to do such work . His commitment and guidance also generated a high level of interest and gusto in doing the project with full enthusiasm. I am truly grateful for his guidance and great support throughout the project.

The prospect of working in a group with high level of accountability fostered a spirit of teamwork and built in a feeling of oneness which thus, expanded our ken, motivated us to outperform to the level best of our ability and create a report of finest quality.

To perform and give the best quality work with sincerity and precision has been our constant endeavor.

CONTENTS

LIST OF ABBREVATIONS

S. No.	ABBREVIATION	FULLFORM
1.	JVM	Java Virtual Machine
2.	SSO	Single Sign On
3.	UI	User Interface
4.	API	Application Programming Interface
5.	MVC	Model View Controller
6.	ORM	Object Relational Mapping
7.	JSP	Java Server Pages
8.	JSTL	JSP Standard Tag Library

ABSTRACT

The OAuth 2.0 protocol is one of the most broadly deployed authorization protocols and it also fills in as the foundation for the new SSO standard OpenID Connect. Regardless of the popularity of OAuth, so far analysis efforts were for the most part focused at understanding the concept of OAuth and the exact workflow of how does it provides security to our protected resources.

In this report, we carry out the analysis of the OAuth 2.0 standard in an demonstrative web model. Our analysis basically focuses at establishing secure authorization and authentication for which we provide formal definitions. In the analysis, all the four OAuth grant types i.e. authorization code grant, implicit grant, resource owner password credentials grant, and the client credentials grant are covered. They might even run simultaneously in the same and different relying parties and identity providers, where relying parties, identity providers, and browsers are considered as well. Our analysis of the OAuth 2.0 standard assumes that security recommendations and best practices are followed in order to avoid obvious and known attacks.

CHAPTER: 1

INTRODUCTION

1.1 INTRODUCTION

Aura is a platform that provides different aspects of web application development like web UI development, defining services. Core part of Aura is plug-in framework, plug-in framework enforce to develop modular style applications. A plug-in is a way for third party to extend the functionality of an application. A plug-in implements extension points declared by an application or other plug-in. Also a plug-in can define extension points. With Aura you can easily transform a monolithic java application into modular application.

Aura is a solid versatile application design and development solution. The goal is to provide a one-stop application development environment by providing wide range of family of tools and services to plan , build and customize the applications and publish them, with an underlying rich-standard based design and technologies.

Why AURA ?

Problem Area:

Application development requires involvement of various front end and back end technologies which are HTML, CSS and JAVASCRIPT to name a few.

Currently companies require multiple resources and technologies to build a single application. Thus applications can be:

- Distributed multi-tired
- Heterogeneous Multi-technologies
- Composite

Solution:

AURA provides one-stop application development environment to design, develop and publish single multiple page application using built-in widgets, application templates, plug-ins with minimum resources.

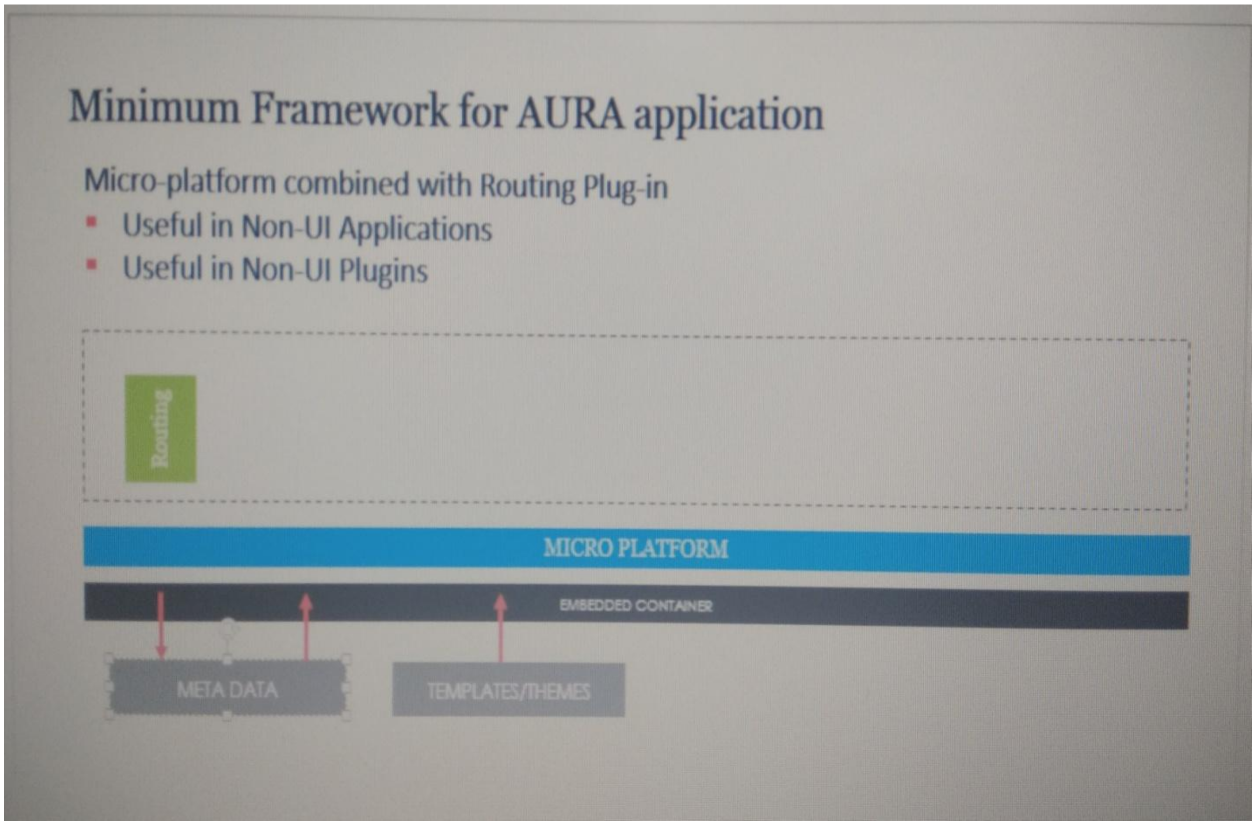
Various Features of AURA project:

- Built on micro platform plug-in architecture
- Caters varieties of scalable, reusable , predefined plug-ins along with provision to add your custom plug-ins
- Offers studio platform with extensive list of components to create customized application.
- Provides readymade generated blocks of code for selected components
- Eliminate time consuming programming task and build scalable , robust web application
- Provision to add custom HTML, JAVASCRIPT, CSS file with their built in editors
- Ensure Consistency irrespective of who is working in project
- Fast , flexible seamless work flow helps to excel the entire development process
- Provides precision layout control and interactive dashboard builder
- Preview changes before publishing the application
- Avails cross-browser compatibility , supporting Google chrome, Internet explorer, Mozilla Firefox, Safari

Micro Platform

- Refers to minimalistic web application frameworks
- It is contrasted to full stack frameworks like enterprise frameworks
- Provides only the components that are absolutely necessary for developer to built an application
- The idea is to keep the core simple but extensible and flexible

- Thus forming a foundation for variety of configurable and customizable components in aura.



Minimum Framework for AURA application

Micro-platform combined with Routing Plug-in

- Useful in Non-UI Applications
- Useful in Non-UI Plugins

Aura Plug-ins

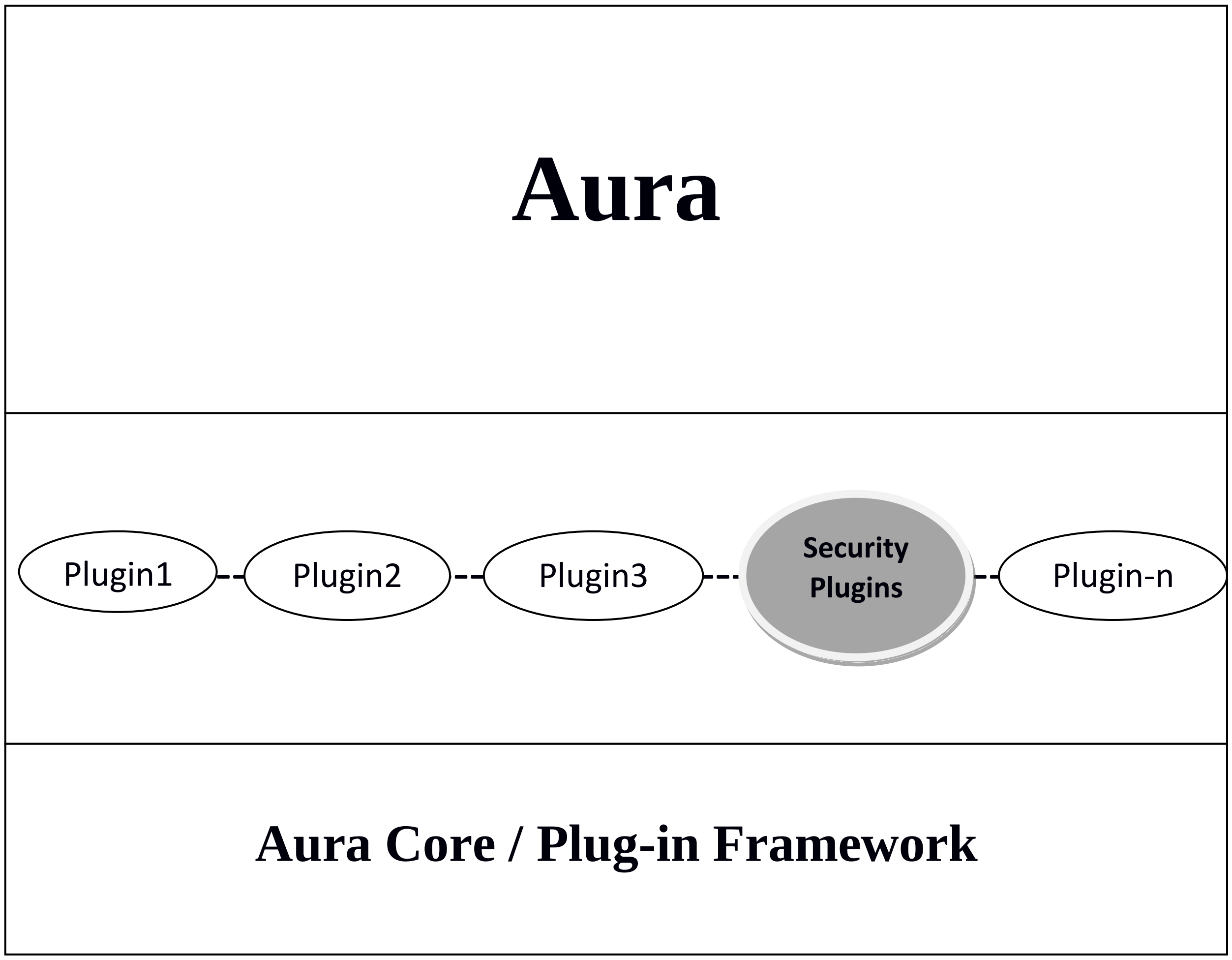
- They are packages or piece of software’s that acts as an add on / extension to the aura application.
- Extend the core functionality of application
- Helps to reduce size of application
- Enable third party developers to create abilities ehich enhance an application
- Modularize the separate subsystems of application , developers seek to emulate

Plug-in features

- Support easy addition of new features to the application
- Can be implemented in variety of ways
    1. A plug-in is incorporated within a plug-in
    2. Set of widgets as a plug-in
    3. An application as a plug-in
- Current plug-ins: Admin, studio , Authentication, authorization , security amd many more to come
- Most would be available as free download
- To install plug-ins you visit the store of AURA plug-ins developer and click on link that will download the installer for the plug-in you have selected
- Will be developed in java with aid of other assets like images, CSS, JavaScript
- Third party developers requires sign off from aura development team to include their developed plug-in in aura plug-in store

Micro Framework for AURA

- Aura micro platform along with routing plugin provides a minimum framework to develop application in aura
- Routing is based on MVC architecture
- MVC basically stands for model view controller
    1. model is responsible for maintaining the applications, data and business login
    2. view is responsible for user interface of application which displays data
    3. controller is request handler and it handles user requests and renders appropriate with model data



1.1 PROBLEM STATEMENT

The project has its motivation come from security purpose of a web application. There is a basic need of users to login into the application. By this system's ability user needs to login just for one time with one set of credentials(username/password) to get access to all corporate apps, websites and data for which they have permission.

1.2 OBJECTIVE

The primary objective of this project is to:

Allow the third party applications or websites to access the users secured data by users permissions. Our system helps user to login to the third party application or website using another account e.g. Facebook, Google, Okta, etc without exposing their passwords. This is known as secure, third party delegated authorization.

1.3 METHODOLOGY

Methodologies used in the project is based on Spring Framework , Spring MVC(Model View Controller), Spring Security and oauth2.0.

Using the concepts of Spring framework including MVC (Model View Controller), spring security and concepts of Oauth2.0 are molded together to built this project.

Firstly I made a small applications to understand the concepts of spring , how does it work , what all annotations are to be used where and when . Then I understood what are web services , how do they work and all. Then I study about the flow of MVC how the services are called and how the view pages are viewed on browser.

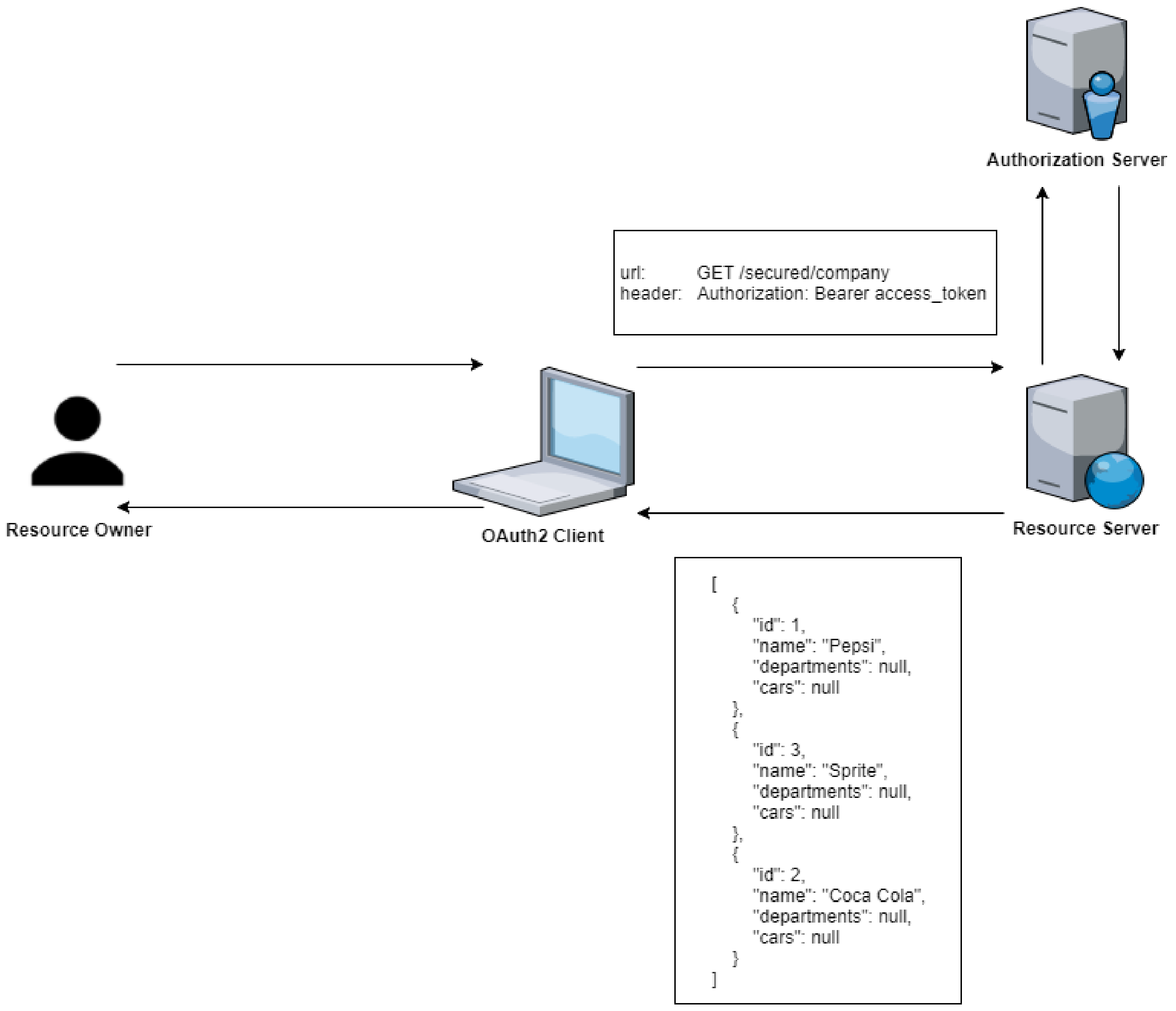
Then I studied about the Spring Security to secure our services as you have to login to experience our services with valid username and password and then how you restrict services to only particular user that have particular roles.

Then I came to OAuth which is used to allow end user to login into our application from applications where user has already logged in like Facebook, Google, Okta, etc.

But I was given task to make our own server to authenticate and authorize the end user.

Actors which are involved in this project are Resource Server, Resource Owner, Authorization Server and Client Application.

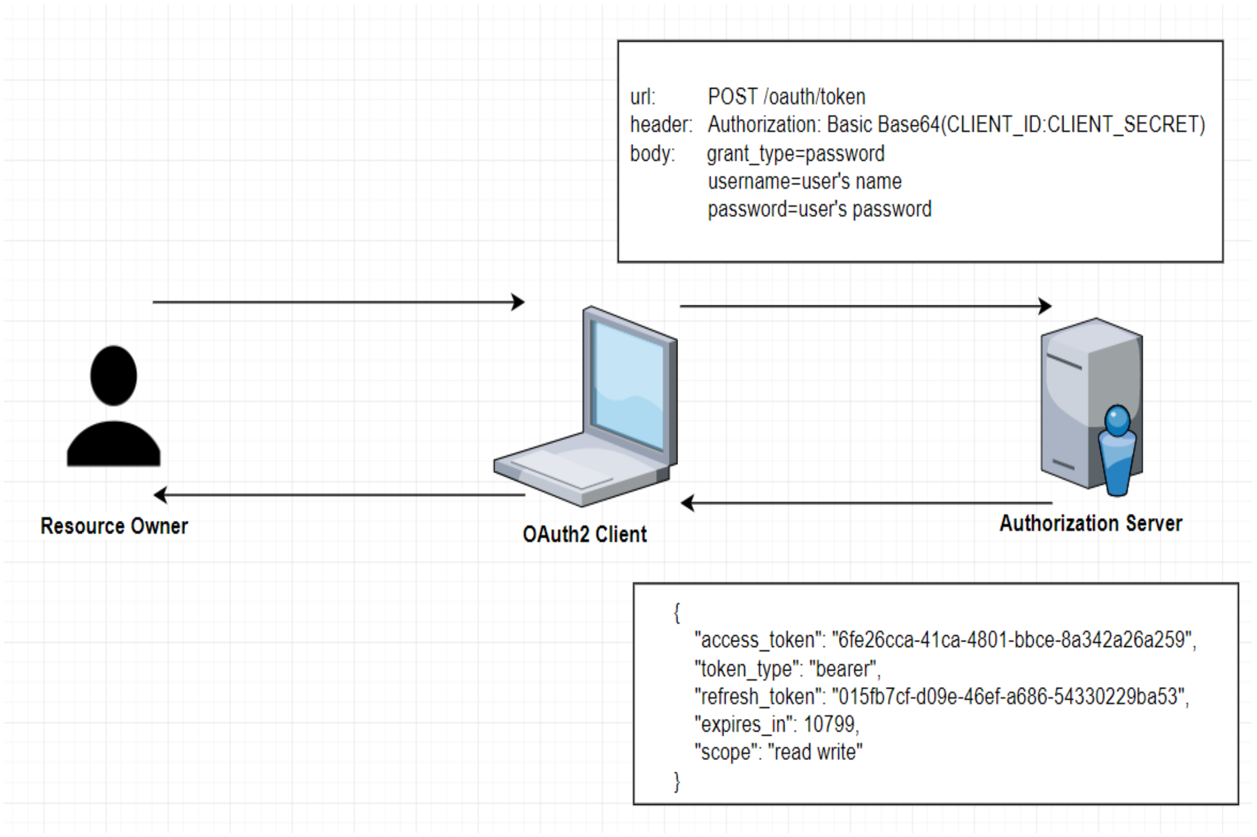
1. **Resource Server –** The server that is hosting the user owned resources is the resource server. These resources are protected by oauth2.0. The resource server only provides the user owned resources by validating the access token that is provided by Authorization server by various authorization grant types.



```
url:      GET /secured/company
header:   Authorization: Bearer access_token
```

```
[
    {
        "id": 1,
        "name": "Pepsi",
        "departments": null,
        "cars": null
    },
    {
        "id": 3,
        "name": "Sprite",
        "departments": null,
        "cars": null
    },
    {
        "id": 2,
        "name": "Coca Cola",
        "departments": null,
        "cars": null
    }
]
```

2. **Authorization Server –** The authorization server gets consent or approval or permission from resource owner (the user) and it issues an access token to client application for accessing the protected resources that are protected by the resource server.

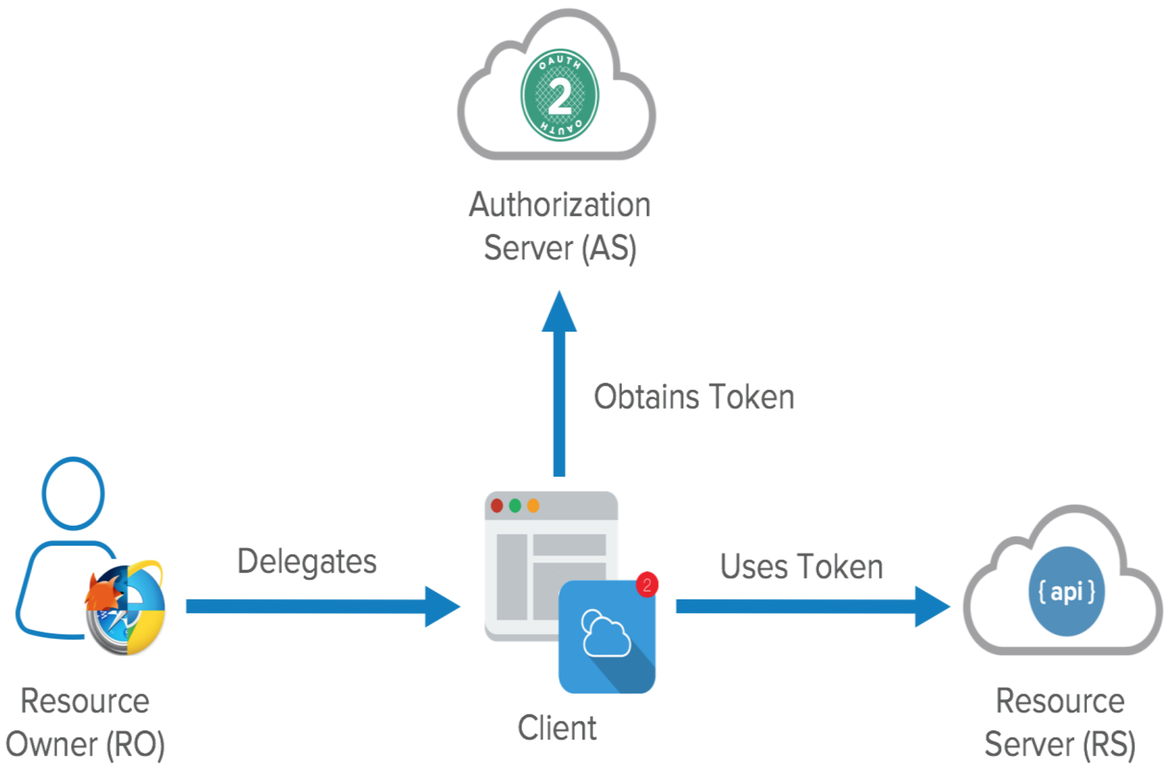


```
url:     POST /oauth/token
header:  Authorization: Basic Base64(CLIENT_ID:CLIENT_SECRET)
body:    grant_type=password
         username=user's name
         password=user's password
```

```
{
   "access_token": "6fe26cca-41ca-4801-bbce-8a342a26a259",
   "token_type": "bearer",
   "refresh_token": "015fb7cf-d09e-46ef-a686-54330229ba53",
   "expires_in": 10799,
   "scope": "read write"
}
```

3. **Resource Owner** – The user that accesses the client application here is resource owner. Only it has the ability to grant the permission or deny the permission to access their own personal data which is hosted by the resource server.

4. **Client –** It is an application which makes API requests to do various actions on protected resources on behalf of resource owner. To perform these actions application should be authorized by the resource owner. Also authorization should be validated / verified by authorization server.



Above diagram shows the methodology or basic roles of actors of this project how the requests are processed.

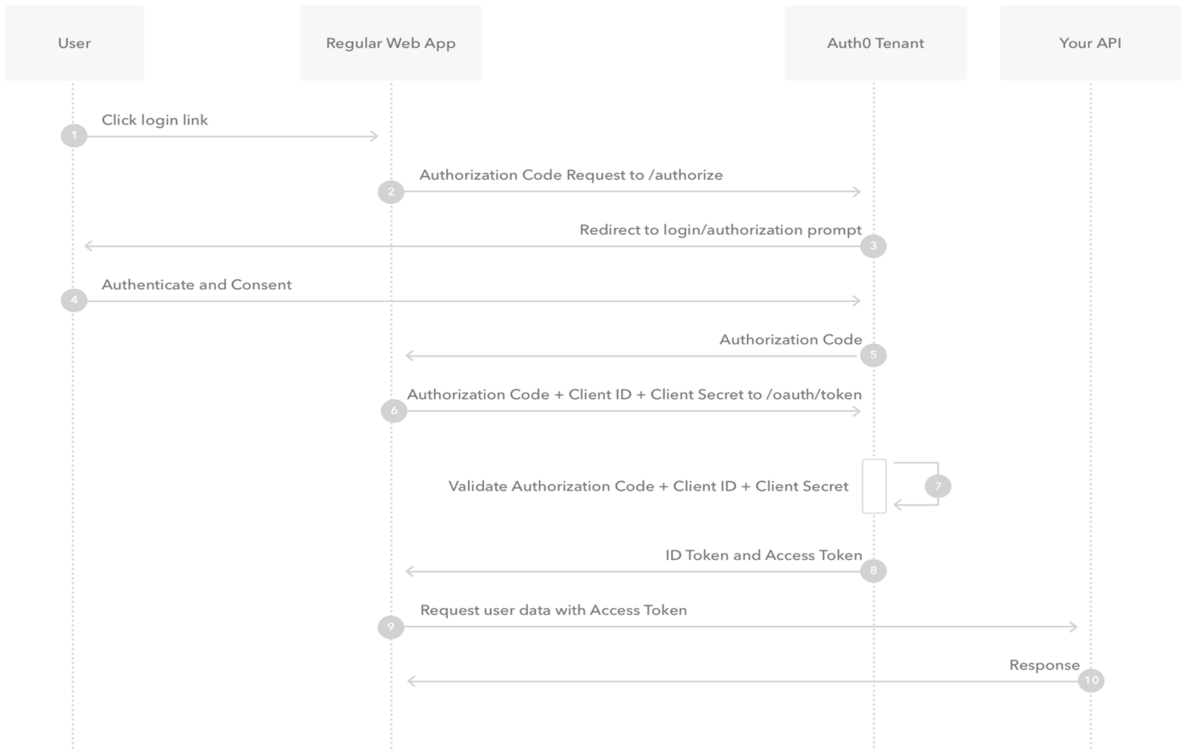
Our system provides various “Grant Types“ for different use cases. By Grant types , we mean various ways of retrieving access tokens. Now which type is most suitable depends mostly on type of application’s type, but other parameters also matter as well, as the level of trust for the application .

The grant types defined are:

- Authorization Code
- Password
- Client Credentials
- Implicit

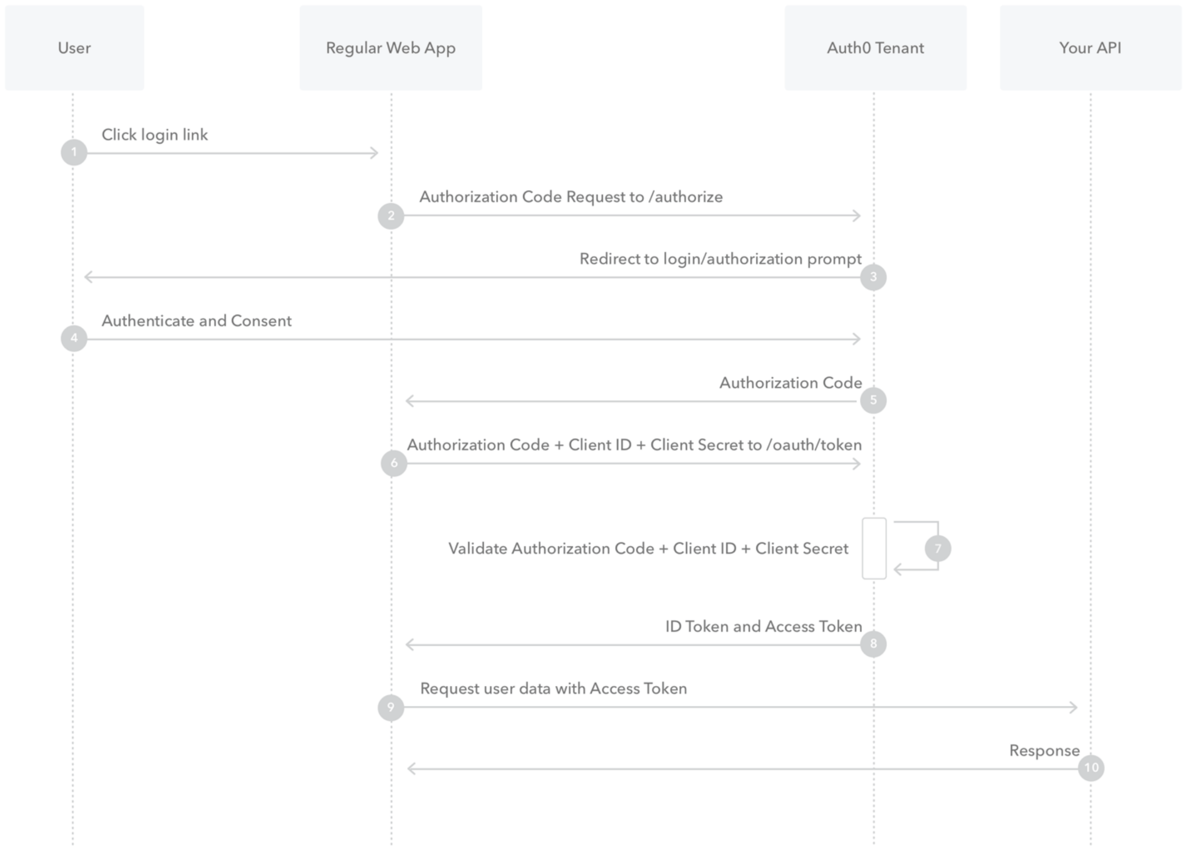
1. Authorization Code Flow

When the client is a third-party server or web application the Authorization Code flow is used, which helps to access to the protected resource protected by resource server. The client does not have access to the resource owner protected credentials. The user connects to a URL hosted by the client. The client redirects the user to the authorization server, including information identifying the client (a client id), the request (scope - the permissions being requested), and a URL pointing back to the client (redirect URL - it is referred to as an URI in the spec though it is always a HTTP URL). The authorization server authorizes the resource owner, and performs authentication such as username and password verification, and confirmation of the action requested. On success of above process, it directs the user back to the client through the provided redirect URL, with an additional authorization code added to the URL. The redirect URL typically points to a server-side script that requests the access token through a POST to the authorization server. The POST is authenticated by the client secret, and provides the authorization code received as proof that the resource owner has indeed authorized the request. The server responds with the access token and an expiration time in the POST response.


User
Regular Web App
Auth0 Tenant
Your API
1 Click login link
2 Authorization Code Request to /authorize
3 Redirect to login/authorization prompt
4 Authenticate and Consent
5 Authorization Code
6 Authorization Code + Client ID + Client Secret to /oauth/token
7 Validate Authorization Code + Client ID + Client Secret
8 ID Token and Access Token
9 Request user data with Access Token
10 Response

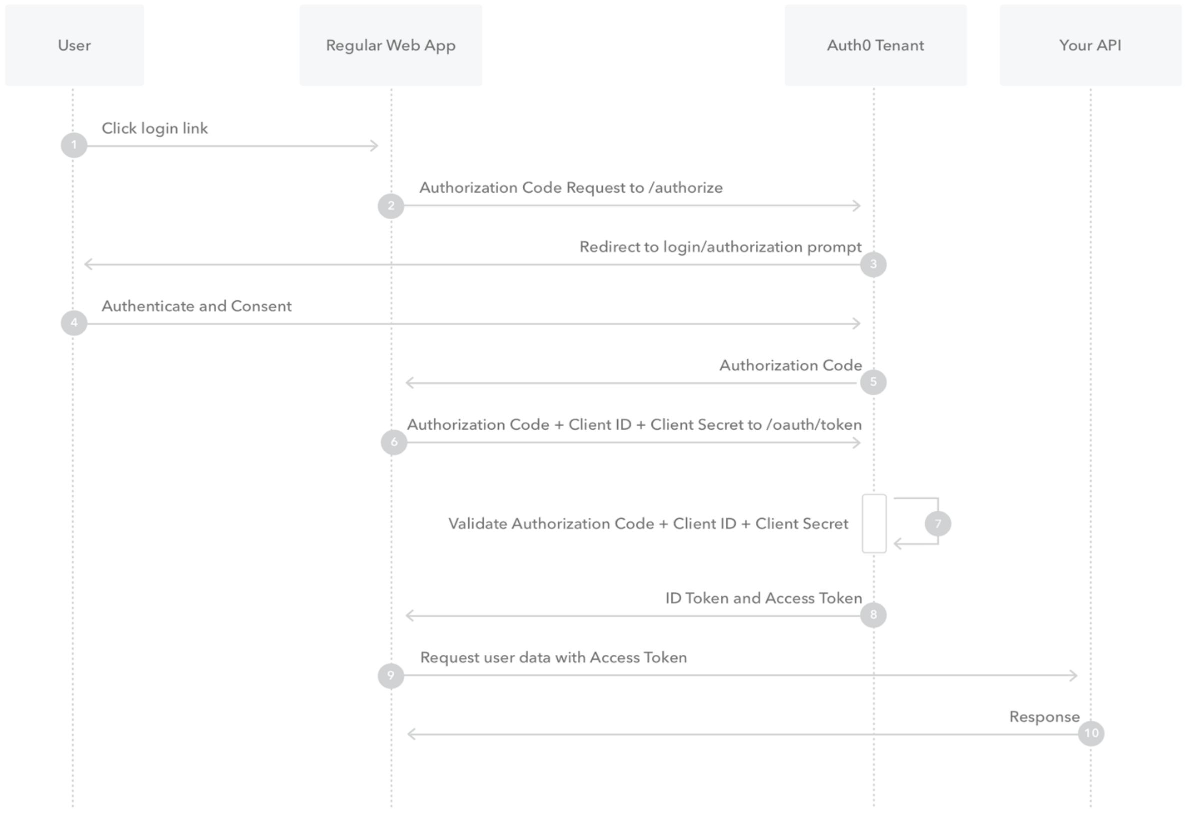
2. Implicit Grant Flow

When the user-agent will access the protected resource directly the Implicit Grant flow is used, such as in a web application or a mobile app. In this the client secret is not used. The user agent connects to the URL on authorization server. This can be either a direct connection or by a redirect made by the client. The request contains the client id, request scope, and redirect URL. If authorization server passes the request, it performs a redirect to redirect URL along with access token and expiration time in the fragments. While the redirect URL points to the client, code inside the client (that is server-side app) does not see it. Instead of that URL may be used to load JavaScript that takes the access token from the URL and uses it for resources. Or, the mobile app can get the redirect, extract the access token from it and use it in the code, in which case URL may just point to static content.


User
Regular Web App
Auth0 Tenant
Your API
1 Click login link
2 Authorization Code Request to /authorize
3 Redirect to login/authorization prompt
4 Authenticate and Consent
5 Authorization Code
6 Authorization Code + Client ID + Client Secret to /oauth/token
7 Validate Authorization Code + Client ID + Client Secret
8 ID Token and Access Token
9 Request user data with Access Token
10 Response

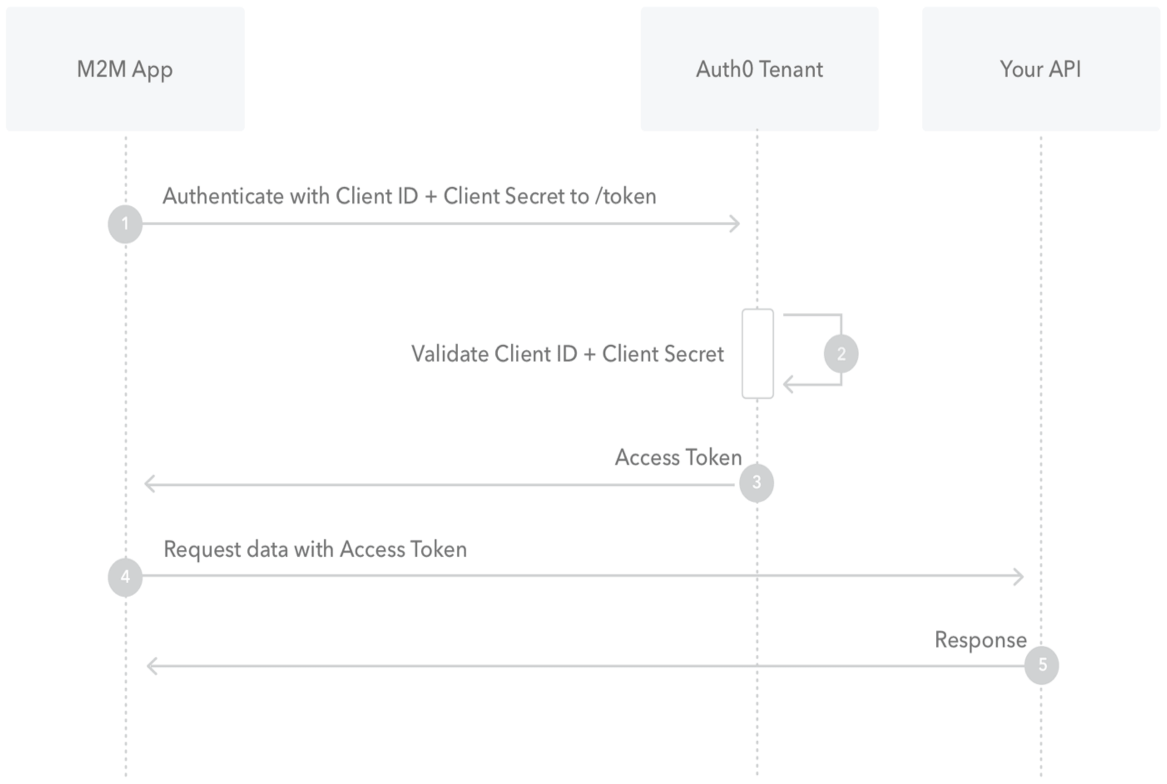
3. Resource Owner Password Credentials Flow

When the resource owner trusts the client to get username and password then there the resource owner password credentials flow is used. The client receives the credentials through other means that are out of scope, then passes them to the authorization server to access token. This flow can be used to migrate traditional username and password authentication schemes.


User
Regular Web App
Auth0 Tenant
Your API
1
Click login link
2
Authorization Code Request to /authorize
3
Redirect to login/authorization prompt
4
Authenticate and Consent
5
Authorization Code
6
Authorization Code + Client ID + Client Secret to /oauth/token
7
Validate Authorization Code + Client ID + Client Secret
8
ID Token and Access Token
9
Request user data with Access Token
10
Response

4. Client Credentials Flow

The client credentials flow is used with Machine-to-Machine (M2M) applications or on the services running on your back-end, the system authenticates or authorizes the client application rather than a user. For this situation, regular authentication schemes just like username and password or social logins doesn’t make sense at all. Rather, M2M applications utilize the client credentials flow in which they pass along their client Id and secret to authenticate them and get access tokens.


M2M App
Auth0 Tenant
Your API
1
Authenticate with Client ID + Client Secret to /token
Validate Client ID + Client Secret
2
Access Token
3
Request data with Access Token
4
Response
5

CHAPTER : 2

LITERATURE SURVEY

This section discusses the information used to build application design and implementation.

2.1 Protect constraint networks using OAuth 2.0 framework

Shamini projected one approach which can be used to protect constrained network from an unknown users with security manager by using OAuth 2.0 framework. In addition to this, it also makes flexible enough the process of managing various IoT networks. The authentication service is provided by the protection manager for multiple IoT networks, which might additionally facilitate to cut back the value overhead to maintain the secure information in constrained networks. Moreover, they have mentioned the procedure to form and manage the database in security manager so it is advantageous to the user to reduce the burden of registering to multiple networks or applications.

2.2 Fire alarm system to notify user using OAuth

Swati proposed a sensible fire alarm system which can be used to notify the user regarding the incidence of a fireplace in the house. Whenever there is fire in the house, user is able to get an alert message via twitter and Gmail account at the same time. The system is made by desegregation IoT technology with temperature sensors. Moreover, in order to use Gmail and twitter account, device ought to authenticate on behalf of the user. For this the OAuth protocol is used to obtain delegated access to the social applications e.g. Gmail and twitter. The result discovered that the proposed system has a low latency as compared to the present system. However, the proposed OAuth based system is able to integrate with many cloud service provider, the notifications are often alleviated to multiple channels. Therefore, the proposed system is able to provide a better notification with more security into multiple channels without compromising on the latency.

2.3 Analysis of OAuth protocol

Feng has analyzed the OAuth protocol and described a systematic root cause analysis of security threats in numerous phases of the OAuth protocol. The attacker model is used to discover that there are number of common network attacks that would be probably carried out by attackers to imitate the users and use their protected resources, like impersonation attacks, forced-login CSRF attacks, network eavesdropping, and replay attacks. Additionally to the present, he has also examined the root cause of these vulnerabilities. In this paper, he centered on the communication

1. Between the user-agent and the authorization server, and

2. Between the user-agent and the consumer application.

CHAPTER : 3

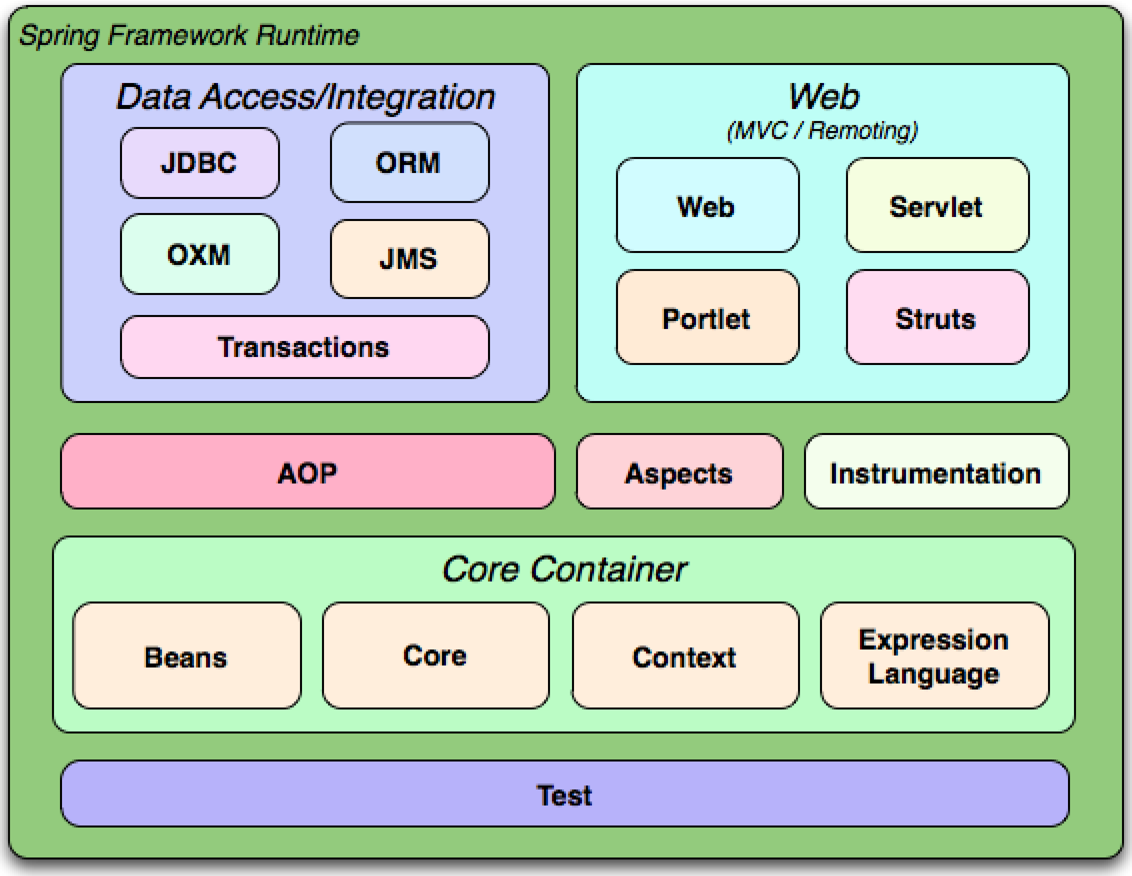
SYSTEM DEVELOPMENT

3.1. TOOLS

3.1.1 SPRING FRAMEWORK

Spring Framework is a Java stage which gives broad establishment support in making Java applications. Spring handles the establishment so that you can focus on the application. Spring empowers you to build applications from "plain old Java objects" (POJOs) and to apply enterprise services non-invasively to POJOs. This ability is applied to the Java SE programming model and to full and partial Java EE.

The Spring Framework consists of various features organized into nearly 20 modules. These modules are assembled into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test, as should be obvious outline:-



3.1.2 ECLIPSE OXYGEN IDE (Integrated Development Environment)

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most generally utilized Java IDE. It contains a base workspace and an extensible module framework for customizing the environment. Eclipse is composed for the most part in Java and its essential use is for creating Java applications, yet it might likewise be utilized to create applications in other programming dialects through modules.

Also we can say that Eclipse is best development environment (IDE) for Java and other programming dialects like C, C++, PHP, and Ruby and so forth.

3.1.3 JAVA

Java is a programming language made by James Gosling form Sun Micro system (sun) in 1991. The objective of java is to compose a program once and afterward run this program on various working frameworks.

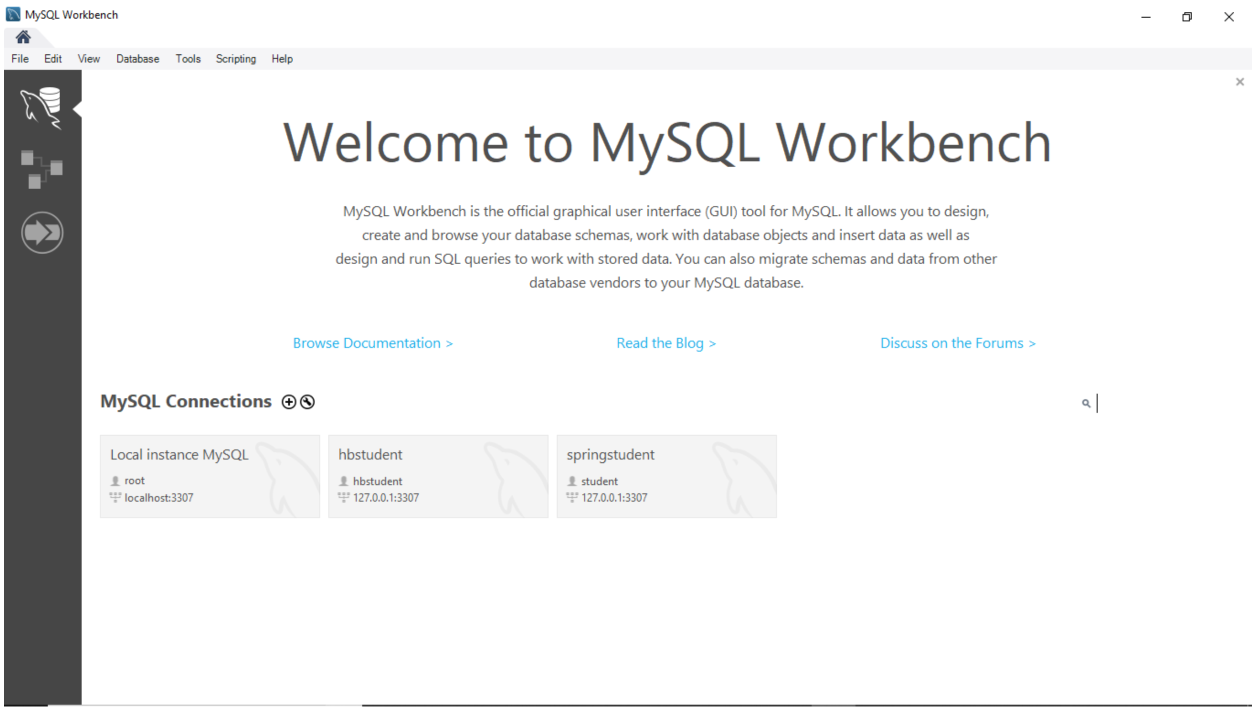
The principal openly accessible adaptation of (Java 1.0) was discharged in 1995. Sun Micro system was obtained by Oracle Corporation in 2010. In 2006 Sun began to make java accessible under the GNU General Public License (GPL). Prophet proceeds with this venture called OpenJDK. Java was initially developed for setup boxes but after that developers found this language can be beneficial for web development. They launched their first browser named NETESCAPE NAVIGATOR which was first java enabled browser.

3.1.4 MySQL WORKBENCH

MySQL is an open source relational database. MySQL is cross stage or cross platform which implies it keeps running on various platform, for example, Windows, Linux and Mac OS etc.

MySQL supports multiple storage engines each with its very own determinations while different frameworks like SQL server just store a single storage engine. It is very famous and efficient relational database and everyone across industry use it because it is also good in terms of performance.

So if we want to use this database in project so there is need of communication and for communication there is need of server access tool and so we can say that server access tool is one of the important tool for communicating with the database.


MySQL Workbench

File Edit View Database Tools Scripting Help

Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Browse Documentation > Read the Blog > Discuss on the Forums >

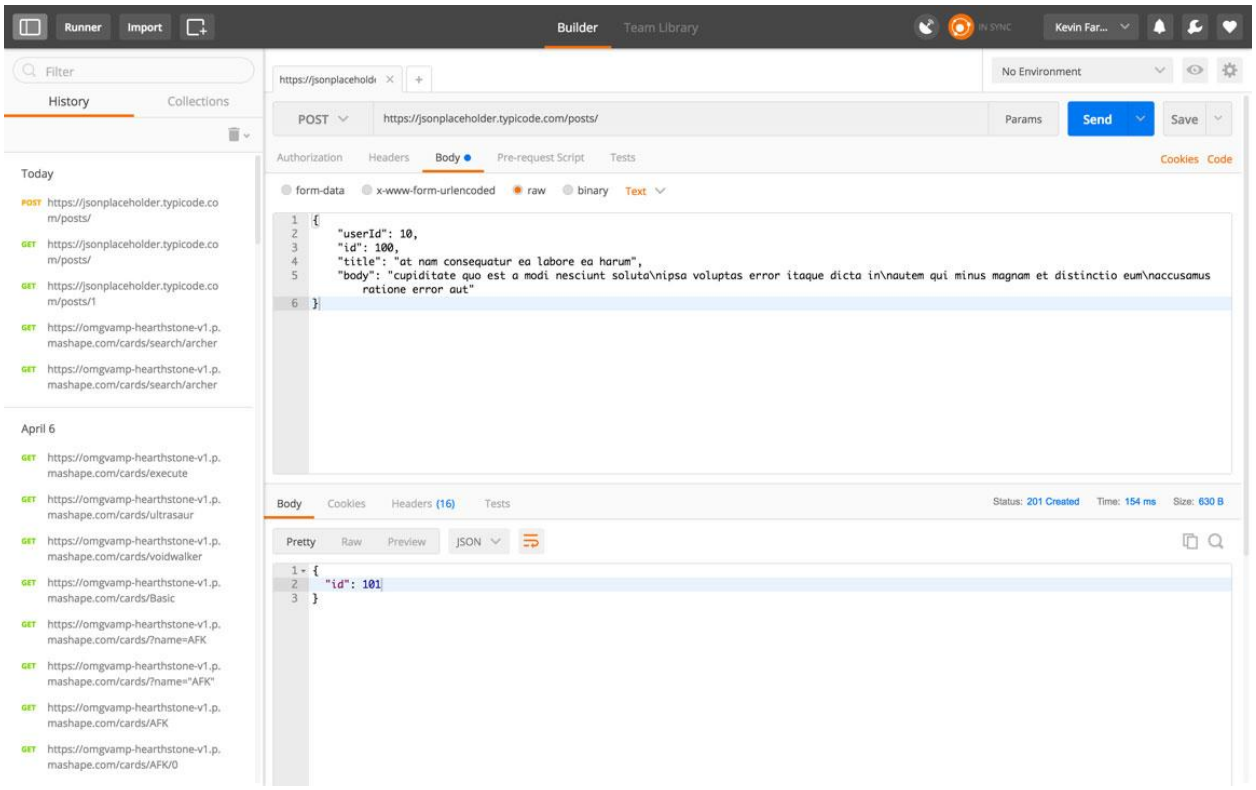
MySQL Connections

Local instance MySQL	hbstudent	springstudent
root	hbstudent	student
localhost:3307	127.0.0.1:3307	127.0.0.1:3307

3.1.5 POSTMAN

Postman is a tool when trying to analyze RESTful APIs made by others or test ones you have made for yourself. It offers a smooth user interface which help to make HTML requests, without the issue of writing a bunch of code just to test an API's functionality.

Assume I expected to make a GET demand against a fan-made API for the PC game Hearthstone to scan for cards with "bowman" in their name. If I needed to test a GET demand against this course without utilizing Postman—rather really working out code in something like Flask—I would need to work out a totally different course and capacity to play out the solicitation, at that point I would need to indicate with more code what I need the reaction to resemble, lastly I would need to print out the reaction to the comfort or give some other method for really seeing the reaction. In truth, I would presumably need to work this out at any rate to make a working application utilizing this API, yet doing this to just test an API's usefulness is superfluously monotonous and tedious when something like Postman exists.



POST https://jsonplaceholder.typicode.com/posts/

```
{
    "userId": 10,
    "id": 100,
    "title": "at nam consequatur ea labore ea harum",
    "body": "cupiditate quo est a modi nesciunt soluta\nipsa voluptas error itaque dicta in\nautem qui minus magnam et distinctio eum\naccusamus ratione error aut"
}
```

Status: 201 Created Time: 154 ms Size: 630 B

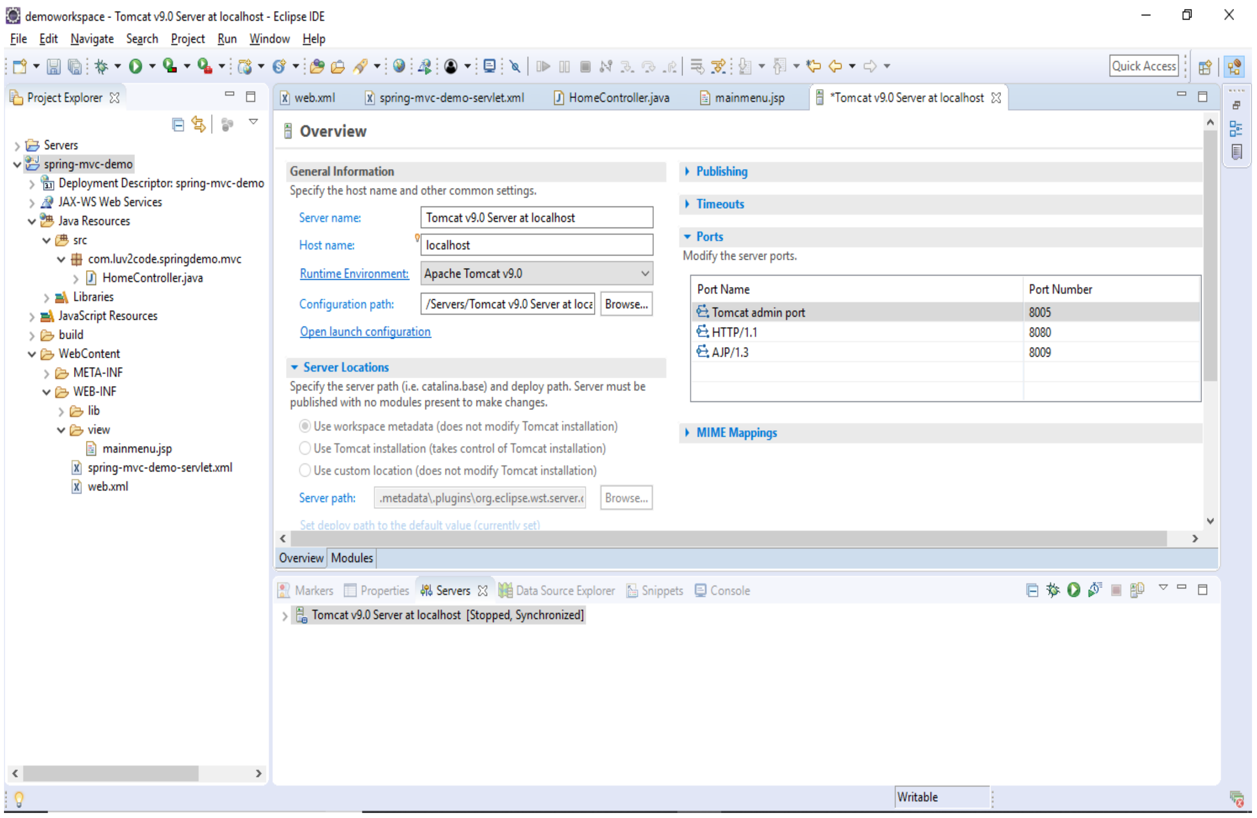
```
{
  "id": 101
}
```

3.1.6 APACHE TOMCAT SERVER

Apache Tomcat (additionally referred to as Tomcat Server) implements several Java Enterprise Edition specifications including Java Server Pages (JSP), Java Servlet,, Java EL, and WebSocket, and gives an "pure Java" HTTP web server condition in which Java code can run.

Tomcat is created and kept by an open network of designers under the support of an Apache Software Foundation, delivered under the Apache License 2.0 permit, and is an open-source programming.

We can add apache tomcat server into eclipse and run our spring applications on the server. But the spring boot applications have in built tomcat server.



General Information

Specify the host name and other common settings.

Server name: Tomcat v9.0 Server at localhost

Host name: localhost

Runtime Environment: Apache Tomcat v9.0

Configuration path: /Servers/Tomcat v9.0 Server at loca Browse...

Open launch configuration

Server Locations

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

- Use workspace metadata (does not modify Tomcat installation)
- Use Tomcat installation (takes control of Tomcat installation)
- Use custom location (does not modify Tomcat installation)

Server path: .metadata\.plugins\org.eclipse.wst.server.c Browse...

Set deploy path to the default value (currently set)

Ports

Modify the server ports.

Port Name	Port Number
Tomcat admin port	8005
HTTP/1.1	8080
AJP/1.3	8009

Tomcat v9.0 Server at localhost [Stopped, Synchronized]

3.2 LIBRARIES USED

3.2.1 SPRING CORE

Spring Core is one of the most important library in the spring framework. It is like the other libraries that are we used in the other languages. As we know it is the core library so it defines all of the necessary imports that related to this framework.

As developers continuously improved this framework but most of the imports remains same in this library and additional functionalities are added to this library.

All of the necessary jar files include by default in referenced library if we add dependency of spring core in pom.xml file in maven project.

3.2.2 ANNOTATIONS

Java annotation is a label that speaks to the metadata for example joined with class, interface, techniques or fields to demonstrate some extra data which can be utilized by java compiler and JVM. Explanation in java are utilized to give extra data , so it is an elective alternative for XML and java marker interfaces. To start with, we will gain proficiency with some inherent explanations then we will proceed onwards making and utilizing custom comments.

3.2.3 HIBERNATE

Hibernate is java object-relational mapping (ORM) and persistence framework by which we map our java class to the database table by using Service and data access object layer. These two layer are very important as we have different requirements in many cases in web applications. By using these two layers when we apply ORM concepts in web application it becomes easier to develop a code and everyone can understand the flow of web application if they know the basic knowledge of web application. The primary objective of hibernate is to assuage the engineer from the regular information perseverance related tasks. It maps the items in the java with the tables in the database in all respects productively and further more you can get greatest utilizing its information question and recovery facilities. Mainly by utilizing hibernate in your activities you can spare staggering time and exertion.

3.2.4 MAVEN

Maven, a Yiddish word that means aggregator of learning, started as endeavor to streamline the manufacture forms in Jakarta Turbine venture. There were a few activities, each with their very own Ant manufacturer documents that were all marginally unique. Containers were registered with CVS. We needed a standard method to construct the undertaking, a reasonable meaning of what the venture comprised of, a simple method to distribute venture data and an approach to share JAR's over a few tasks. The outcome is an apparatus that would now be able to be utilized for structure and dealing with any java-based venture. We trust that we have made something that will make the everyday work of java engineers simpler and for the most part help with the understanding of any java-based project.

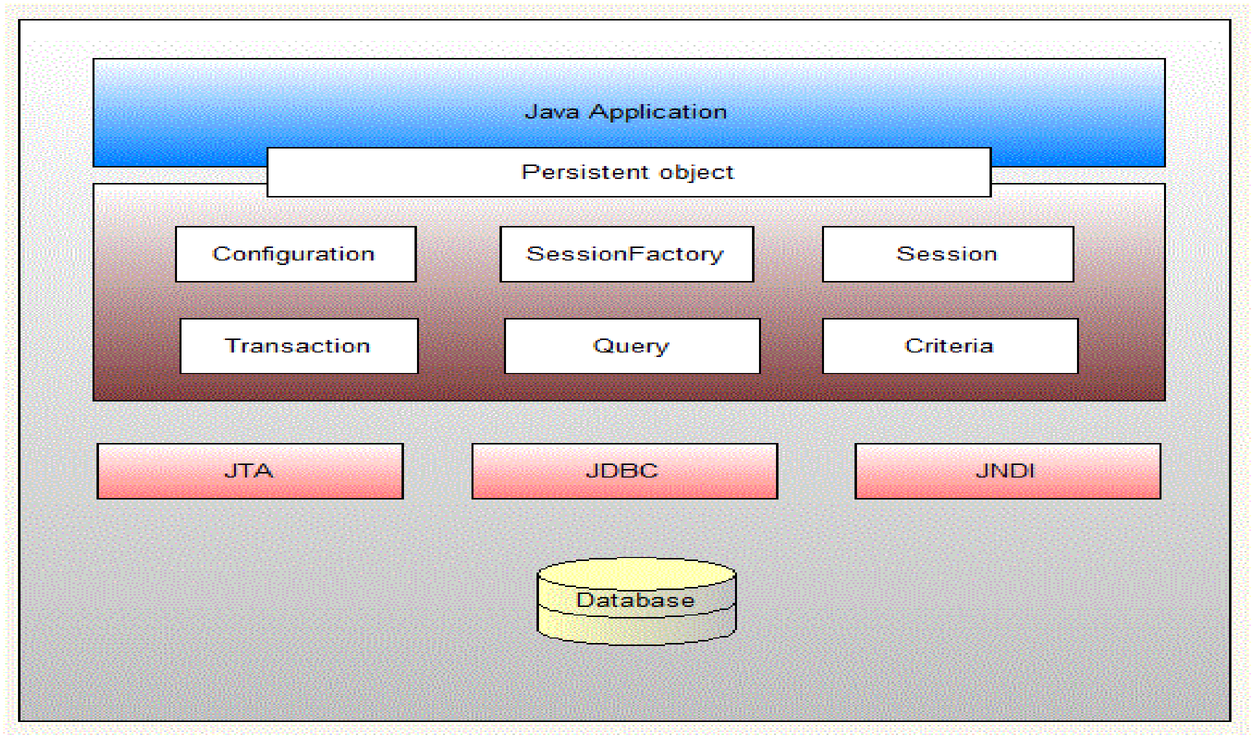
Maven primary objective is to enable an engineer to appropriate the total condition of improvement exertion in the shortest timeframe. So as to accomplish this objective, there are a few region of worry that maven expertise to manage:

1. For making process easy we can use the Maven
2. It can provide us uniform building system
3. It can provide us quality project information

4. Providing guidelines for best practices development
5. Allowing transparent migration to new features
6. It can add jar by default just we need to add dependencies in pom.xml

3.2.5 JPA

JPA is like an interface which have different implementation like hibernate and others. All the persistence information are defined in JPA environment and its implementation is defined in its various smaller ORM framework's like Hibernate. Also we can say that Hibernate is one of the implementations of the JPA. JPA is an open source API, therefore Oracle, Redhat, Eclipse, etc. are the vendors that provide new products by adding the JPA persistence flavor in them and others environment also.



CHAPTER – 4

KEY CONCEPTS or PREREQUISITES REQUIRED FOR PROJECT

4.1Spring MVC

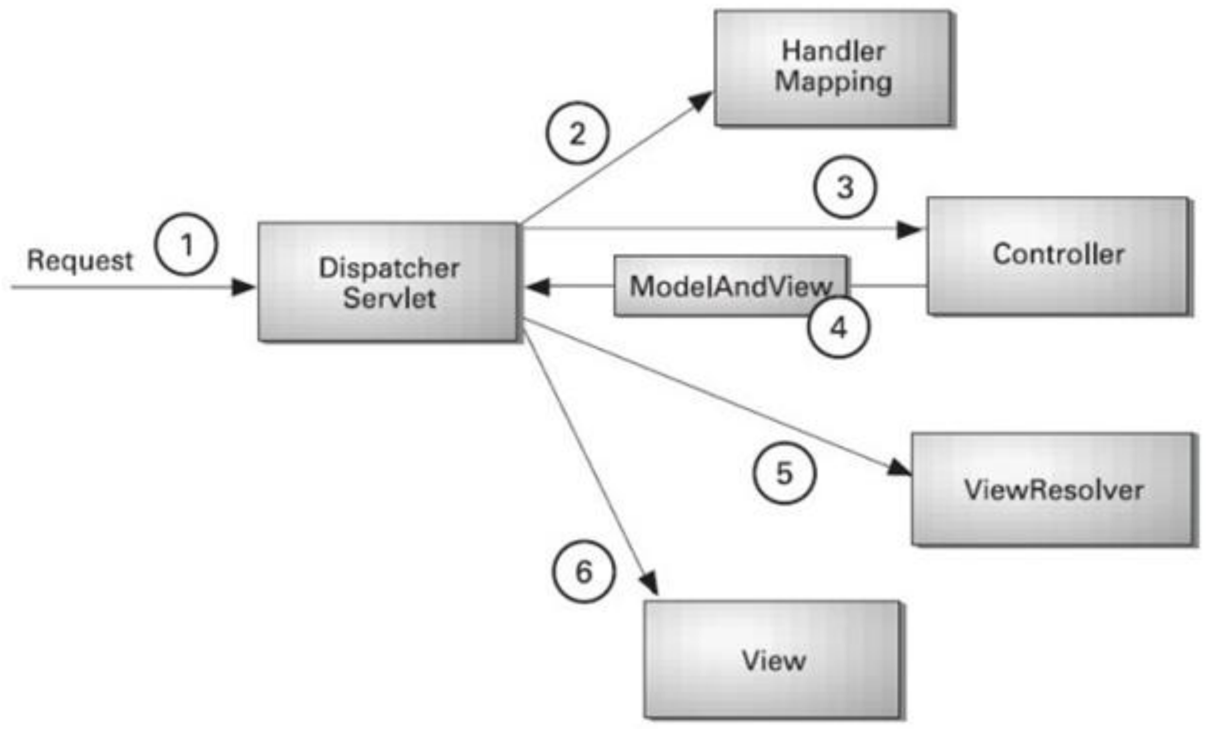
A Spring MVC ie. Model view controller is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.

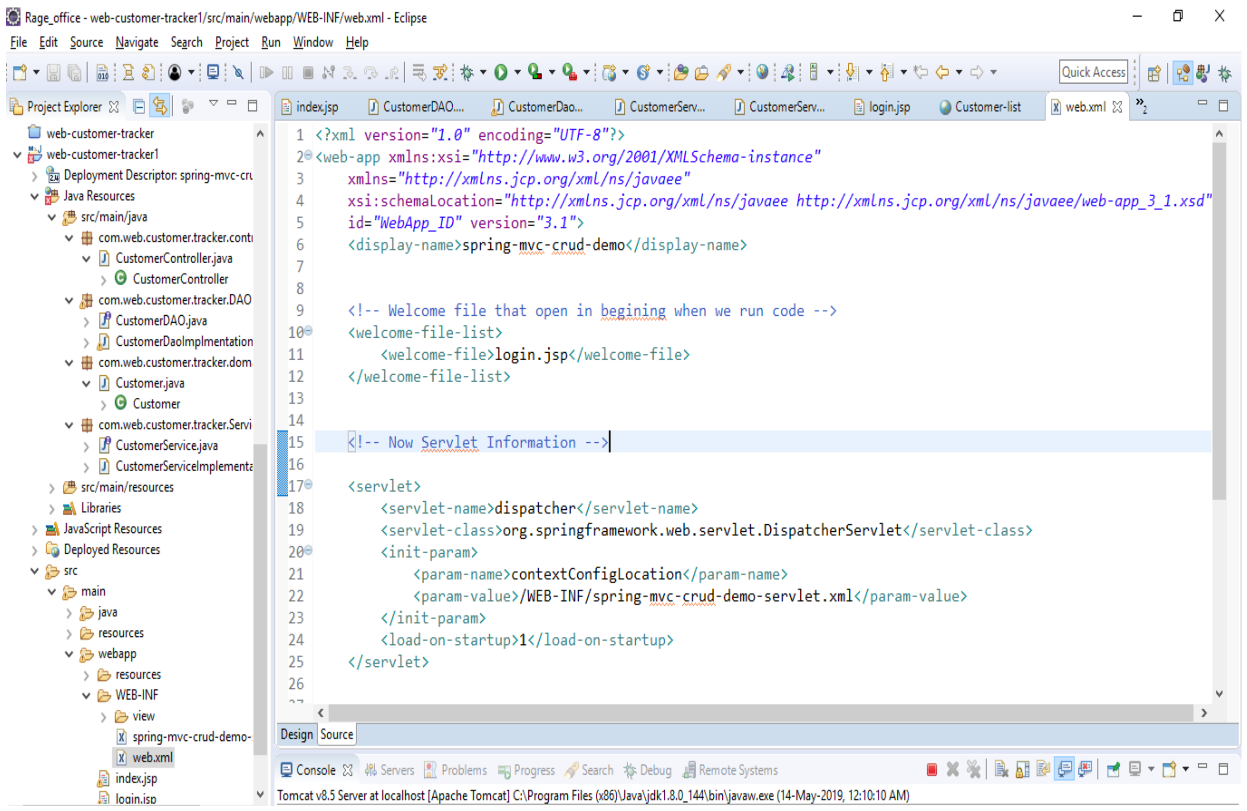
A Spring MVC provides an elegant solution to use MVC in spring framework by the help of Dispatcher Servlet. Here, Dispatcher Servlet is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.

- **Model** - A model is responsible for containing the data of the application. A data can be a single object or a collection of objects.
- **Controller** - A controller is responsible for containing the business logic of an application.
- **View** - A view is responsible to represent the provided information in a particular format. Generally, JSP+JSTL is used to create a view page.
- **Front Controller** - In Spring Web MVC(Model View Controller), the Dispatcher Servlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

You need to load these files:

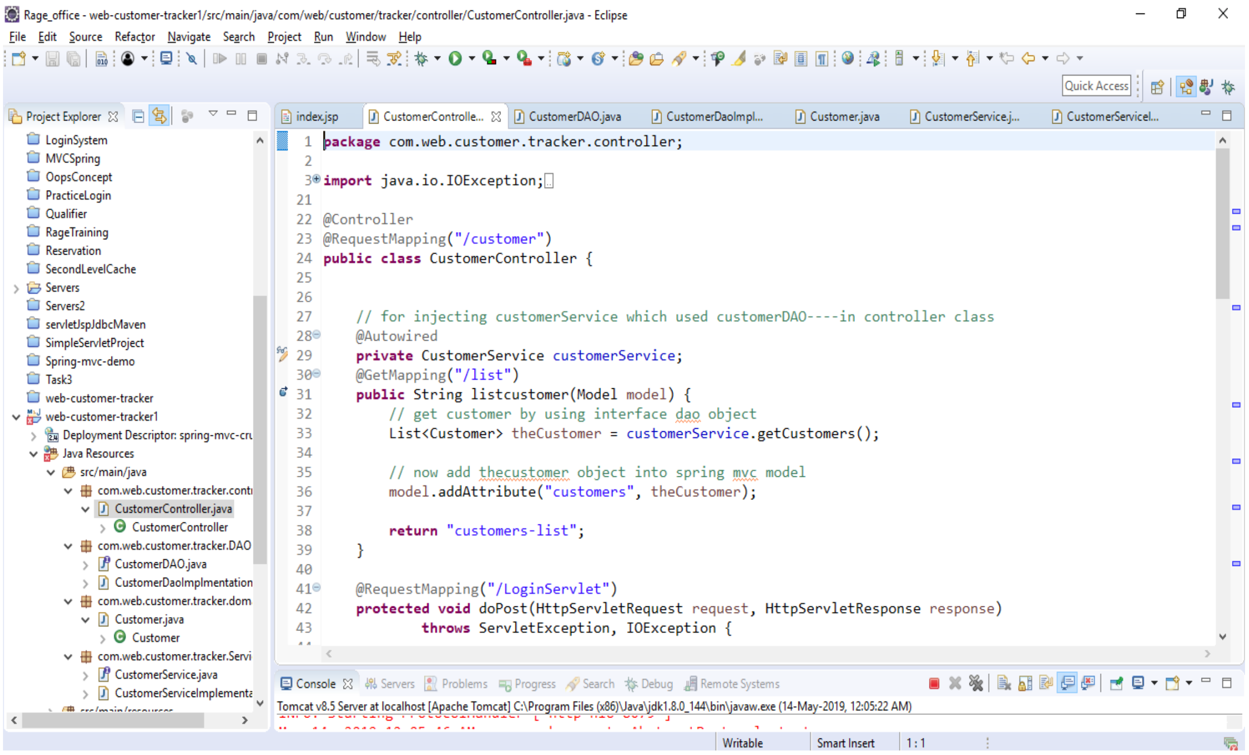
- Spring Core jar files
- Spring Web jar files
- JSP + JSTL jar files





```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    id="WebApp_ID" version="3.1">
    <display-name>spring-mvc-crud-demo</display-name>


    <!-- Welcome file that open in begining when we run code -->
    <welcome-file-list>
        <welcome-file>login.jsp</welcome-file>
    </welcome-file-list>


    <!-- Now Servlet Information -->

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring-mvc-crud-demo-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

```

```
Tomcat v8.5 Server at localhost [Apache Tomcat] C:\Program Files (x86)\Java\jdk1.8.0_144\bin\javaw.exe (14-May-2019, 12:10:10 AM)
```

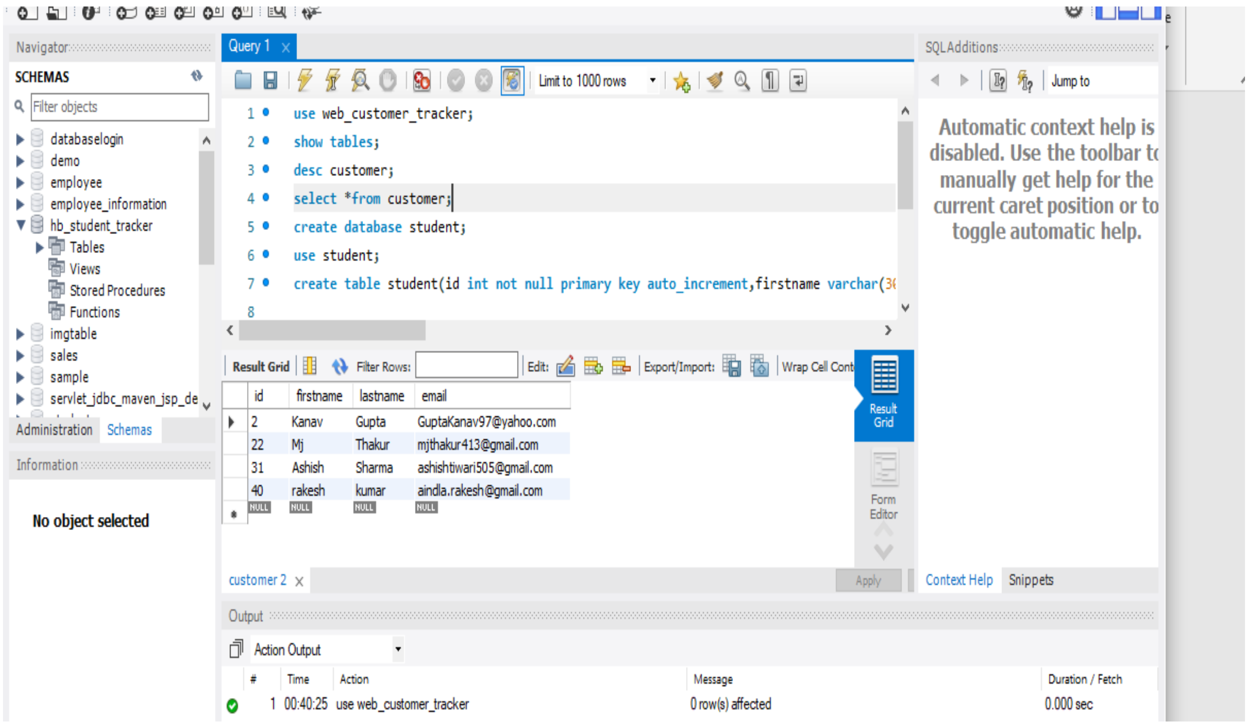


```java
package com.web.customer.tracker.controller;

import java.io.IOException;

@Controller
@RequestMapping("/customer")
public class CustomerController {


    // for injecting customerService which used customerDAO----in controller class
    @Autowired
    private CustomerService customerService;
    @GetMapping("/list")
    public String listcustomer(Model model) {
        // get customer by using interface dao object
        List<Customer> theCustomer = customerService.getCustomers();

        // now add thecustomer object into spring mvc model
        model.addAttribute("customers", theCustomer);

        return "customers-list";
    }

    @RequestMapping("/LoginServlet")
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

```
Tomcat v8.5 Server at localhost [Apache Tomcat] C:\Program Files (x86)\Java\jdk1.8.0_144\bin\javaw.exe (14-May-2019, 12:05:22 AM)
```

4.2 Hibernate

Hibernate is an ORM (Object-Relational Mapping) framework. It maps your database tables to the Java classes, i.e., the entities. It internally uses JDBC. It provides you with the flexibility of changing the database, if needed. For instance, you might start with mysql and at a later point you may want to switch to oracle or postgres or any other database. Since all of your code is in Java and the queries are all in HQL. Native queries will still require to be updated as per the database. Nowadays, JPA is used with Hibernate or any other ORM like Eclipse Link, Top link, etc.


```
use web_customer_tracker;
show tables;
desc customer;
select *from customer;
create database student;
use student;
create table student(id int not null primary key auto_increment,firstname varchar(3
```

id	firstname	lastname	email
2	Kanav	Gupta	GuptaKanav97@yahoo.com
22	Mj	Thakur	mjthakur413@gmail.com
31	Ashish	Sharma	ashishtiwari505@gmail.com
40	rakesh	kumar	aindla.rakesh@gmail.com
NULL	NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

No object selected

```
#  Time      Action                    Message            Duration / Fetch
1  00:40:25  use web_customer_tracker  0 row(s) affected  0.000 sec
```

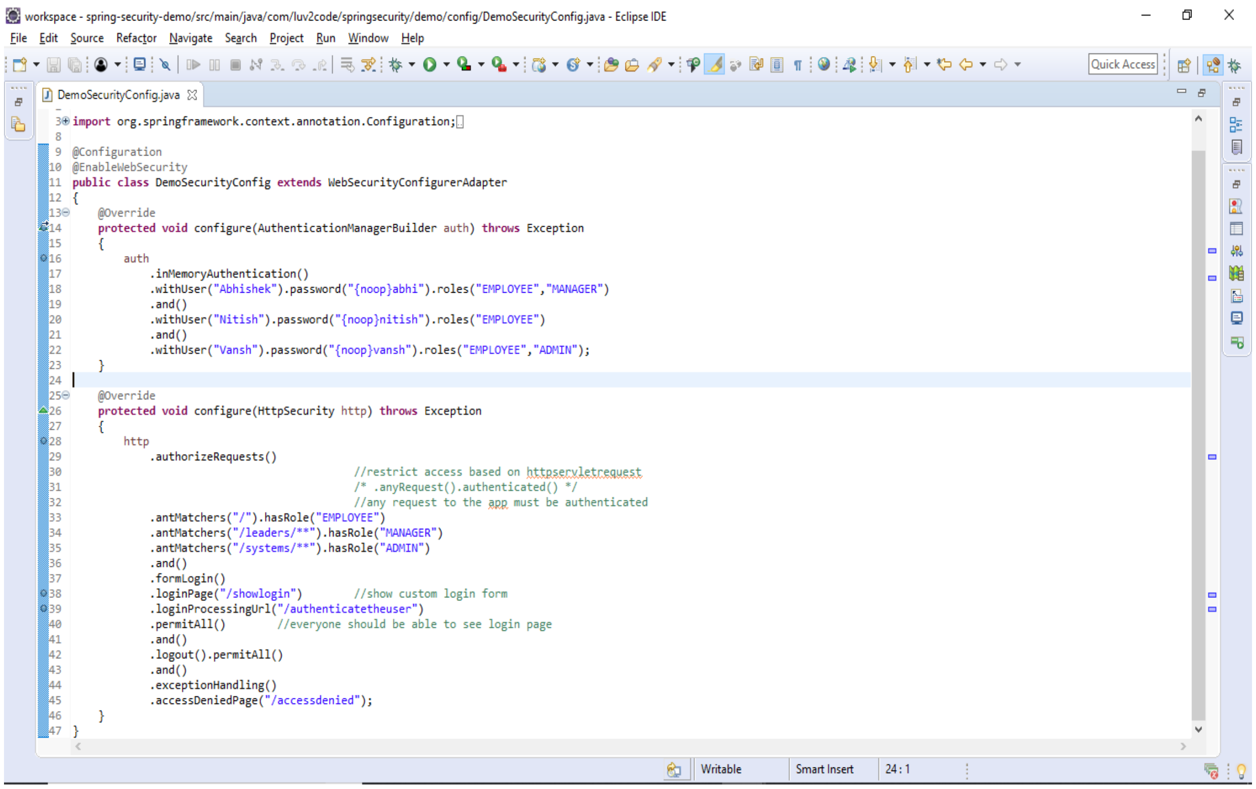
4.3 Spring Security

It is a very powerful and very highly authentication and access-control framework. It is the standard for securing spring based application.

Spring Security is a framework that basically focuses on providing both authentication and authorization to java applications. Like all the spring projects, the real power of spring security is found in how easily it can be extended to meet custom requirements.

Features:

- Comprehensive and extensible support for both authentication and authorization
- Protection against attacks like cross site request forgery, etc
- Servlet API integration
- Optional integration with spring web MVC


workspace - spring-security-demo/src/main/java/com/luv2code/springsecurity/demo/config/DemoSecurityConfig.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
DemoSecurityConfig.java
Writable
Smart Insert
24 : 1

```java
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter
{
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
    {
        auth
            .inMemoryAuthentication()
            .withUser("Abhishek").password("{noop}abhi").roles("EMPLOYEE","MANAGER")
            .and()
            .withUser("Nitish").password("{noop}nitish").roles("EMPLOYEE")
            .and()
            .withUser("Vansh").password("{noop}vansh").roles("EMPLOYEE","ADMIN");
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception
    {
        http
            .authorizeRequests()
                                            //restrict access based on httpservletrequest
                                            /* .anyRequest().authenticated() */
                                            //any request to the app must be authenticated
            .antMatchers("/").hasRole("EMPLOYEE")
            .antMatchers("/leaders/**").hasRole("MANAGER")
            .antMatchers("/systems/**").hasRole("ADMIN")
            .and()
            .formLogin()
            .loginPage("/showlogin")        //show custom login form
            .loginProcessingUrl("/authenticatetheuser")
            .permitAll()        //everyone should be able to see login page
            .and()
            .logout().permitAll()
            .and()
            .exceptionHandling()
            .accessDeniedPage("/accessdenied");
    }
}
```

CHAPTER – 5

WORKFLOW OF PROJECT

4.1 Register Client and Get Client Credentials

OAuth necessitates that the customer should be registered with authorization server. The server then requests some essential information about the client, for example, name and redirect_uri (this is the URL where the authorization server will redirect you to when the resource owner grants permission) and it returns client credentials (client-id and client-secret) to the client. These credentials are very critical in securing the authenticity of requests when performing various activities like exchanging authorization codes for access tokens and refreshing access tokens.

For example, Facebook requires us to register our client on the Facebook Developers Portal. Go to Facebook Developers portal and register the client application and get client credentials (username and password).

4.2 Get an Access Token Step-by-Step:

The client application need to get an access token by Facebook to access users personal data. In order to get an access token , the client application redirects the user to Facebook's login page. Upon successful login, it redirects to redirect_uri (as registered when we register the client application) along with short-lived authorization code. This code is used only to get access token to access protected resources. Client Application exchanges the short lived code to get the long lived access token. The access token is used to access the user personal data. The access token here is the key or permission given to Facebook by client to access the protected information. This is the the most popular flow in OAuth2.0, called the Authorization code grant flow.

Below is sequence diagram to get a token in the authorization code grant:

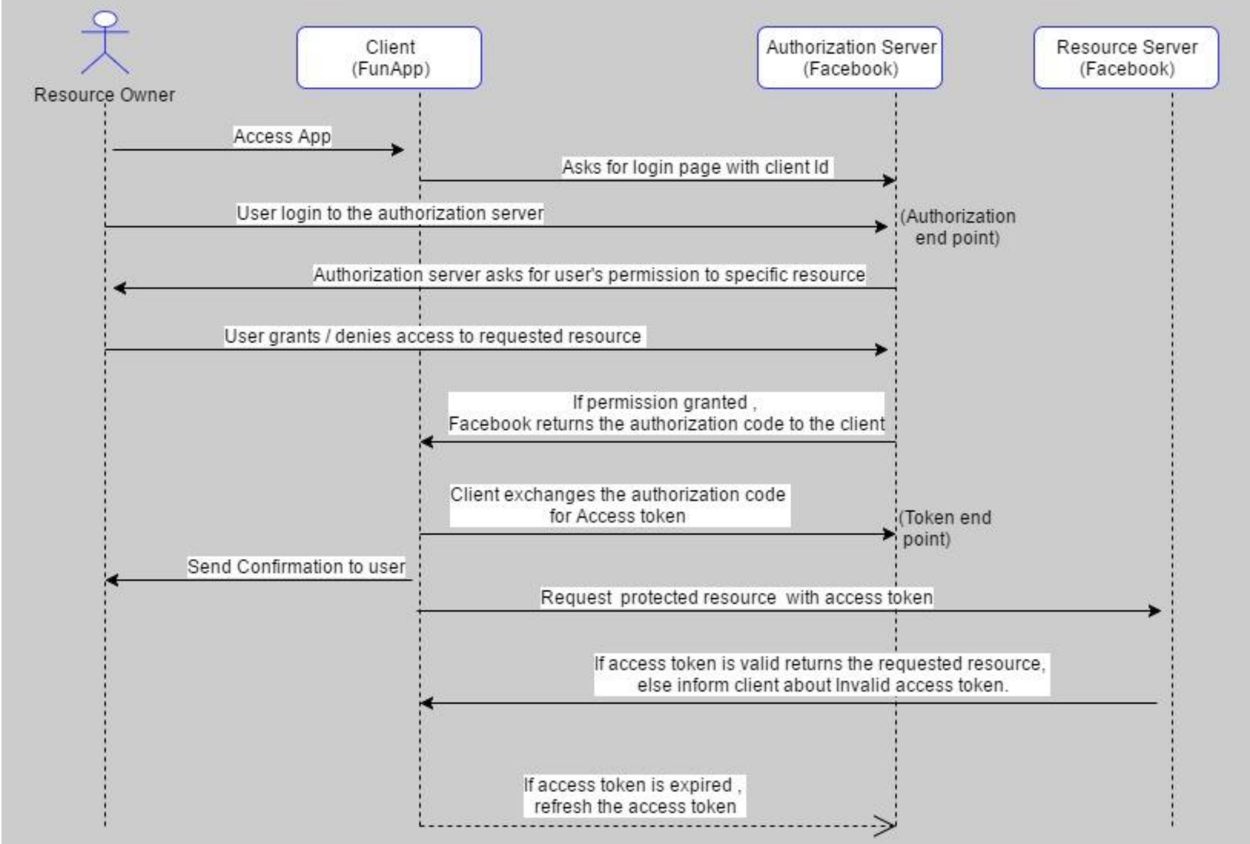

Resource Owner
Client (FunApp)
Authorization Server (Facebook)
Resource Server (Facebook)
Access App
Asks for login page with client Id
User login to the authorization server
(Authorization end point)
Authorization server asks for user's permission to specific resource
User grants / denies access to requested resource
If permission granted , Facebook returns the authorization code to the client
Client exchanges the authorization code for Access token
(Token end point)
Send Confirmation to user
Request protected resource with access token
If access token is valid returns the requested resource, else inform client about Invalid access token.
If access token is expired , refresh the access token

Fig: Authorization code sequence diagram

CHAPTER - 6

CONCLUSIONS

6.1 Conclusions

Hence far in our task we have done the integration of our system with Authorization server, that is providing us the authorization code and that code is utilized by the client application to get access_token from the authorization server. Now that access token is used by client application to use the protected resources of resource owner that are protected by resource server.

6.2 Future Scope

After building up this project, this can be integrated into various web application to secure the web services by configuring them with authorization server and validating the user (resource owner) by third party application and then providing the client with protected resources of resource owner.

CHAPTER -7

REFRENCES

1. The Spring documentation - https://spring.io/docs
2. Udemy - https://www.udemy.com
3. OAuth2.0 documentation - https://oauth.net/2/
4. Apache Maven - http://maven.apache.org/
5. Johnson, R. J2EE development frameworks. Computer Volume 38, Issue 1, Jan. 2005 Page(s):107 – 110.