CPSC 476

Web Back-End Engineering for Enterprise Applications

Project 3

“URL SHORTENER USING SPRING MVC FRAMEWORK”

Team Members:

Member Name	Email ID	CWID
Dipika Mahashabde	dipika.2204@csu.fullerton.edu	803004274
Niyati Bichu	niyatibichu@csu.fullerton.edu	893471748
Piyusha Zanjad	piyusha_zanjad@csu.fullerton.edu	893578021
Robert Mitchell	r_w_mitchell@csu.fullerton.edu	891307159
Dhaval Shah	dshah8@csu.fullerton.edu	802973719

1. Introduction

1.1 Purpose

The purpose of the document is to present the implementation details of the previous project of URL Shortener which is refactored using Spring MVC Framework. Using Spring's JDBC support, the data is persisted in a relational database called as "HSQLDB".

1.2 Project Description

The project includes a signup page where users can create an account with username and password. After the user is logged in, a private page is displayed where users can submit a Long URL and get back a shortened URL. Also, on the same private page, they can view a list of URLs they have shortened and the number of clicks each shortened URL has received. The application also includes a public front page where users can input a shortened URL and view the original URL without following the link. When we click on the shortened URL, the clicks are recorded before redirecting to the original URL.

1.3 Document Scope

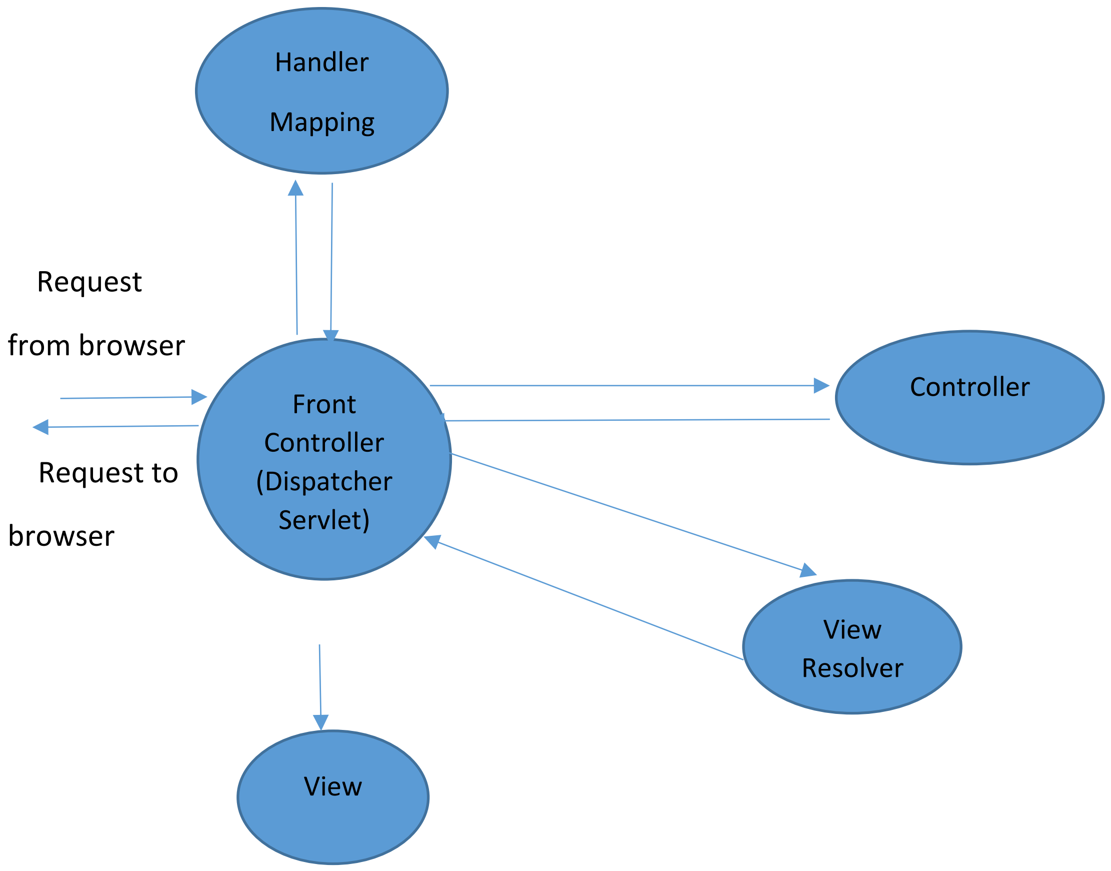
This document describes the functionalities of the URL shortener application. It also describes the configurations and required settings to run it on the system.

2. Project's Architecture

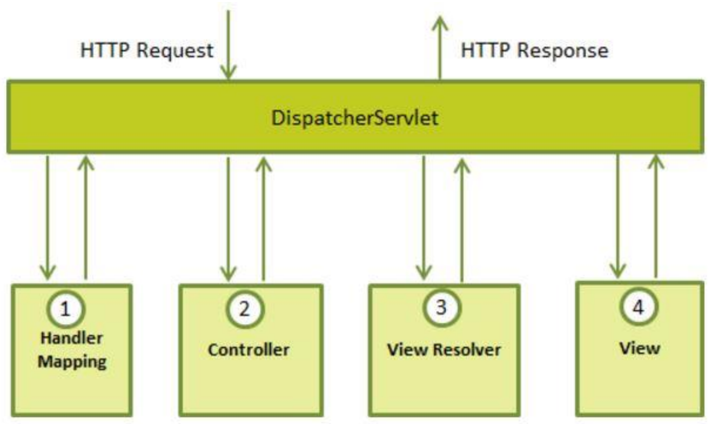
This project is developed using spring MVC framework which provides model-view-controller architecture and ready components which are used to develop flexible and loosely coupled web applications. It separates different aspects of the application such as input logic, business logic and user interface logic. Model encapsulates the application data, view renders the model data and controller process user requests and builds appropriate model and passes it to view for rendering.

2.1 Workflow of Spring MVC

When an end user requests a page by entering some URL through a browser, the request is received by the **Front Controller (**also known as **Dispatcher Servlet)**. It handles the complete flow of the application and has many helpers to get the work done. The front controller's job is to prepare send the response back to the client's browser. After receiving request, it passes the request to **handler mapping**. The front controller will get the information and will invoke the particular **controller**. The controller gets in the information from the **Model** which is a POJO. Then, the front controller will initiate the **view resolver** to build the **view**. The information from the model will be sent to the view then the response is sent back to the front controller.



This framework is designed around **DispatcherServlet** which dispatches the requests to handlers, with configurable handler mappings, view resolution etc. The default handler is based on @Controller and @RequestMapping annotations.



1. Dispatcher Servlet consults the Handler Mapping to call the right controller.
2. Controller takes request and calls the right service methods based on GET/POST methods. The service method will set model data based on business logic and returns view name to dispatcher servlet.
3. Dispatcher servlet takes help from View Resolver to select the defined view for the request.
4. The Dispatcher Servlet passes model data to view which is then rendered on the browser.

For this project, we have used the following annotations:

@Controller indicates that the particular class servers the role of a controller

@RequestMapping maps requests to methods within the controller.

@Service is used in the service layer and annotates that perform service tasks.

@Resource when applied, the container injects the requested resource.

@ModelAttribute refers to a property of Model object. If we have a form with a form object, then Spring MVC supplies this object to the controller using this annotation.

@Autowired annotation injects the required bean.

For this project, Spring MVC form objects are used to handle the jsp forms.

We have specified the name of model class object which acts as the object for this form.

<form:form method="POST" modelAttribute="loginForm" action="/RunningProject/login">

Using this annotation, we make object available to the method body. The name should be same as the name in the attribute of the <form:form> tag.

@ModelAttribute("loginForm")

In this project, we have used different packages to prevent naming conflicts, to control access, to make searching and locating and usage of classes and interfaces easier.

1. **com.project3.controller:** contains all the controller
2. **com.project3.Model:** contains the data access objects which are nothing but POJO's. There are two data access objects User and Links.
3. **com.project3.implementation:** contains two interfaces of User and Links.

4. **com.project3.service:** contains all the definitions of the methods which are in the implementation package. This package also contains a Row Mapper required for converting relational database rows into java objects.

3. Setup Instructions

1. Download and unzip the project on your local drive.
2. Download and keep the build.xml file in web/WEB-INF/Build/build.xml(create a new Build folder in WEB-INF).
3. In the Build folder,
   (For ex: \RunningProject\src\main\webapp\WEB-INF\Build)
   Open command prompt and type: ant clean.
4. Run the build.xml file as Ant Build.
5. After building the file go to hsqldb manager named manage, run the hsql commands given in the hsqldb command.txt file in the manager. This file contains two create commands which will create two tables for User and Links.
6. In the command prompt of the project folder type: mvn clean . This command will clear all the class files. After doing check the target file you should not have anything in it.
7. Run the mvn package this will create a war in the target subdirectory of the project.
8. Start the tomcat server by Startup.bat command.
9. Deploy the war file to the tomcat manager or copy the war file in the webapps folder of tomcat.
10. Go to this url => http://localhost:8080/RunningProject/login

4. Application structure

4.1 Model

1. **User –** This POJO contains member variables and getters and setters for user related information.
   private int id;
   private String username;
   private String password;
   //getters and setters

2. **Links –** This POJO contains member variables and getters and setters for links related information. It also records the number of clicks.
   private int linkId;
   private int userId;
   private String longUrl;
   private String shortUrl;
   private int clicks;
   //getters and setters

4.2 Controllers

1. **LoginController-** contains methods for login as well as signup. After signup, the page is redirected to login. When the user logs in, the page is redirected to shortener controller where users can create shortened links.
2. **ShortenerController-** handles creating short links and displaying them.
3. **RedirectController-** handles redirecting the entered short url to the particular long url.

4.3 Services

1. **UserService –** This interface contains three functions, getUser(), insertUser() and getUserID() which are defined in this this class and has their implementations in UserServiceImplmentation class in the com.project3.implementation package.

2. **LinkService -** This interface contains functions such as, insertLinks(), getUrlByUser(), getLongUrl(), getClicks() and updateClicks() which are defined in this this class and has their implementations in LinkServiceImplmentation class in the com.project3.implementation package.

4.4 Implementation

This project uses Jdbc Template to connect to the database and execute SQL queries.

1. **UserServiceImplementation –** It has the following methods:

- **getUser()** – gets the username and password from the database using select query.
- **insertUser()**- inserts users into the database using insert query.
- **getUserID()**- gets the id from the user table using select query.

2. **LinkServiceImplementation –** It has the following methods:

- **insertLinks()** – inserts links into database using insert query.
- **getUrlByUser()** – gets urls for a particular user using select query.
- **getLongUrl() –** gets longurl for a particular requested short url using select query.
- **getClicks() –** gets clicks for the selected short url using select query.
- **updateClicks()** – update number of clicks using update query.

4.5 Row Mappers

RowMapper is an interface used by Jdbc Template for mapping rows of a result set on a row basis. It maps each row to a result object. This projects contains the following RowMappers which are placed in the same package which is com.project3.implementation:

- **UserRowMapper**
- **UserIdMapper**
- **LinkRowMapper**
- **publicClicksRowMapper**
- **publicLinkRowMapper**

5. Database Schema

User Table

```
create table user(

id INT IDENTITY NOT NULL,

username VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL,

PRIMARY KEY(id)

);
```

Links Table

```
create table Links(

linkId INT IDENTITY NOT NULL,

userId INT NOT NULL,

longUrl VARCHAR(200) NOT NULL,

shortUrl VARCHAR(200) NOT NULL,

clicks INT NOT NULL,

PRIMARY KEY(linkId),

FOREIGN KEY(userId) REFERENCES user(id)

);
```

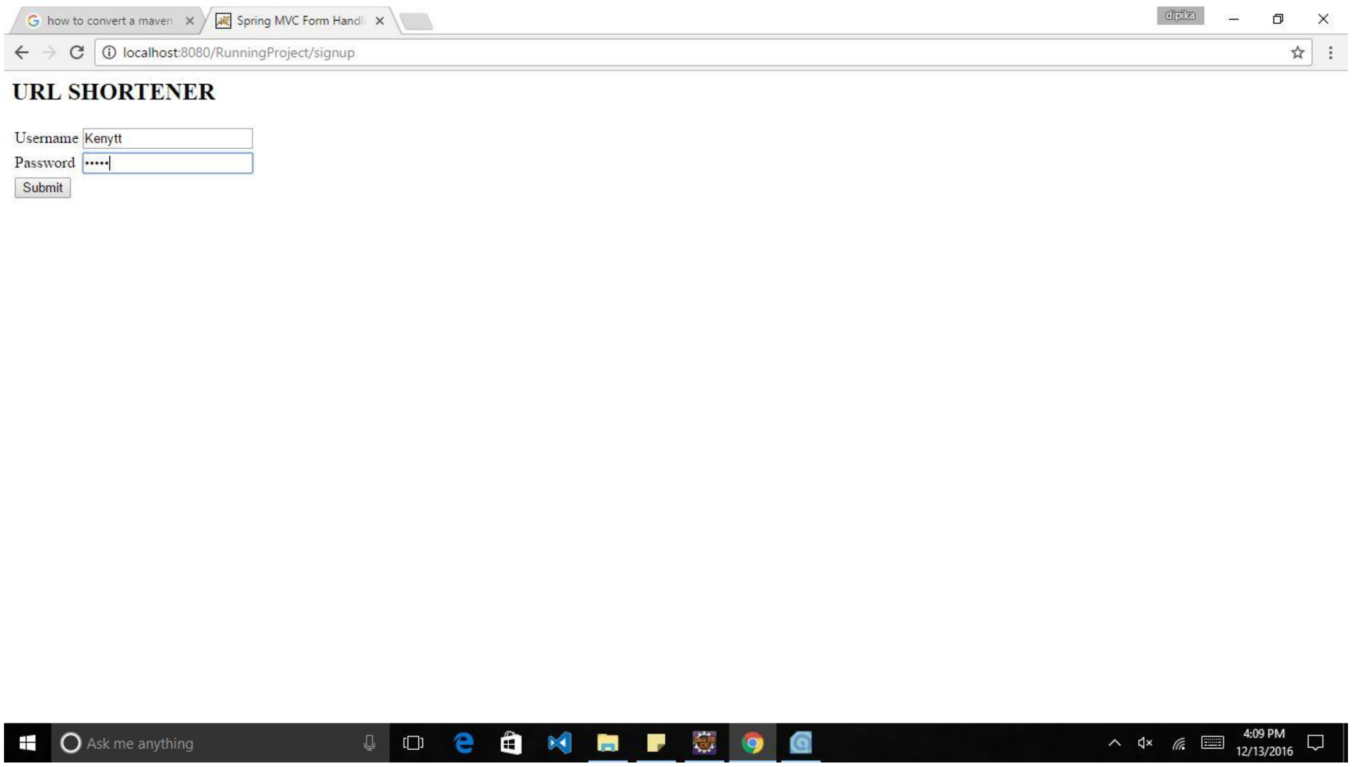
Pom.xml: contains all the dependencies required for

1. Spring framework- **webmvc, core, beans, jdbc, web, context**
2. **Hqsl**
3. **Dbcp**

applicationContext.xml: defines our Data Source and also helps to manage beans. It is also helpful in performing dependency injection. Classpath is >/WEB-INF/classes/applicationContext.xml.

7. Application in Action

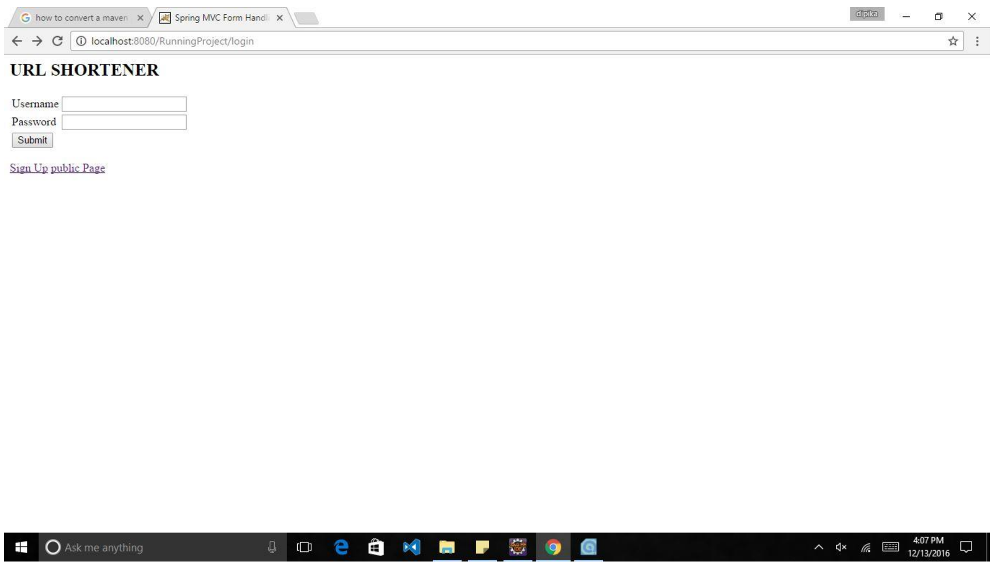
Users can sign up by providing a username and password.



URL SHORTENER

Username Kenytt
Password •••••

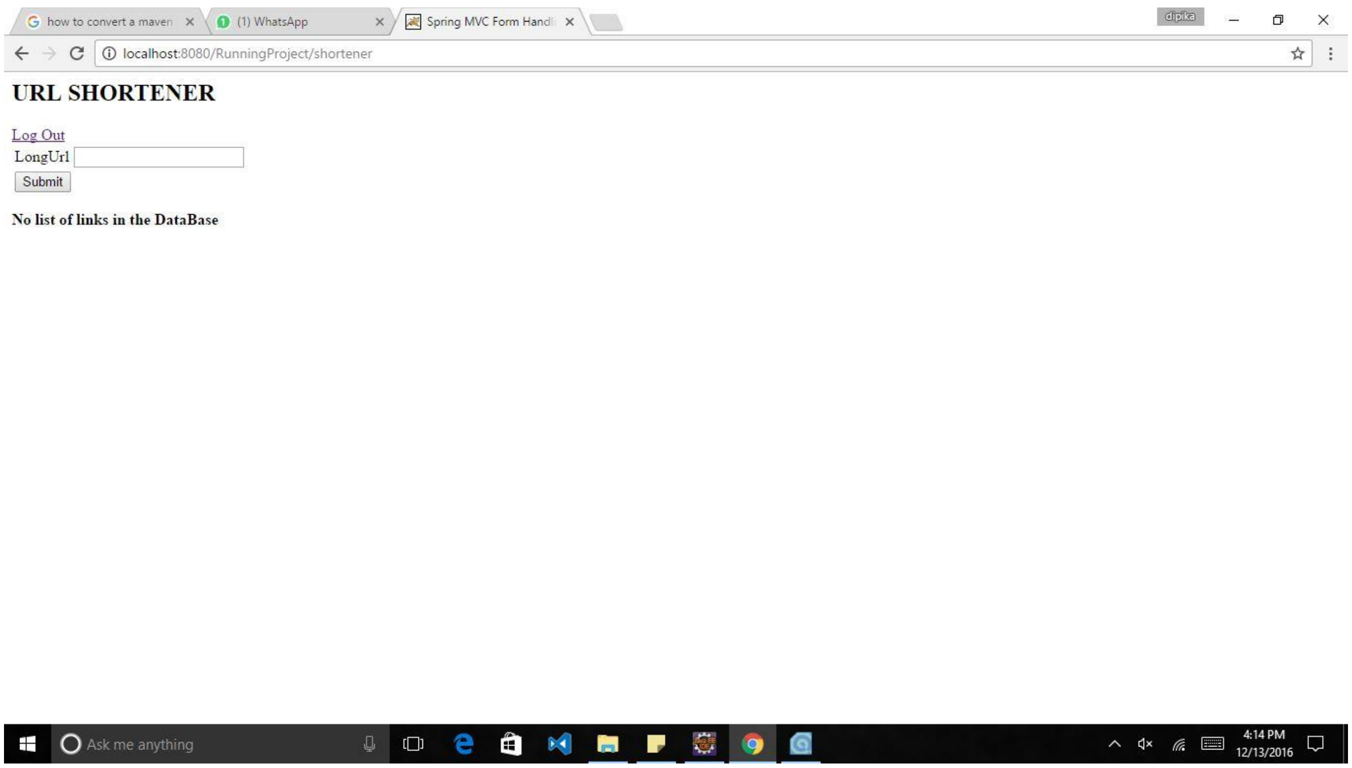
Users can login by entering their username and password. Also there is a public page link where without login, users can see short url for the long url.



URL SHORTENER

Username
Password

After logging in, if the user has no links to display, a message is displayed below saying that there are no links in the database for him.



URL SHORTENER

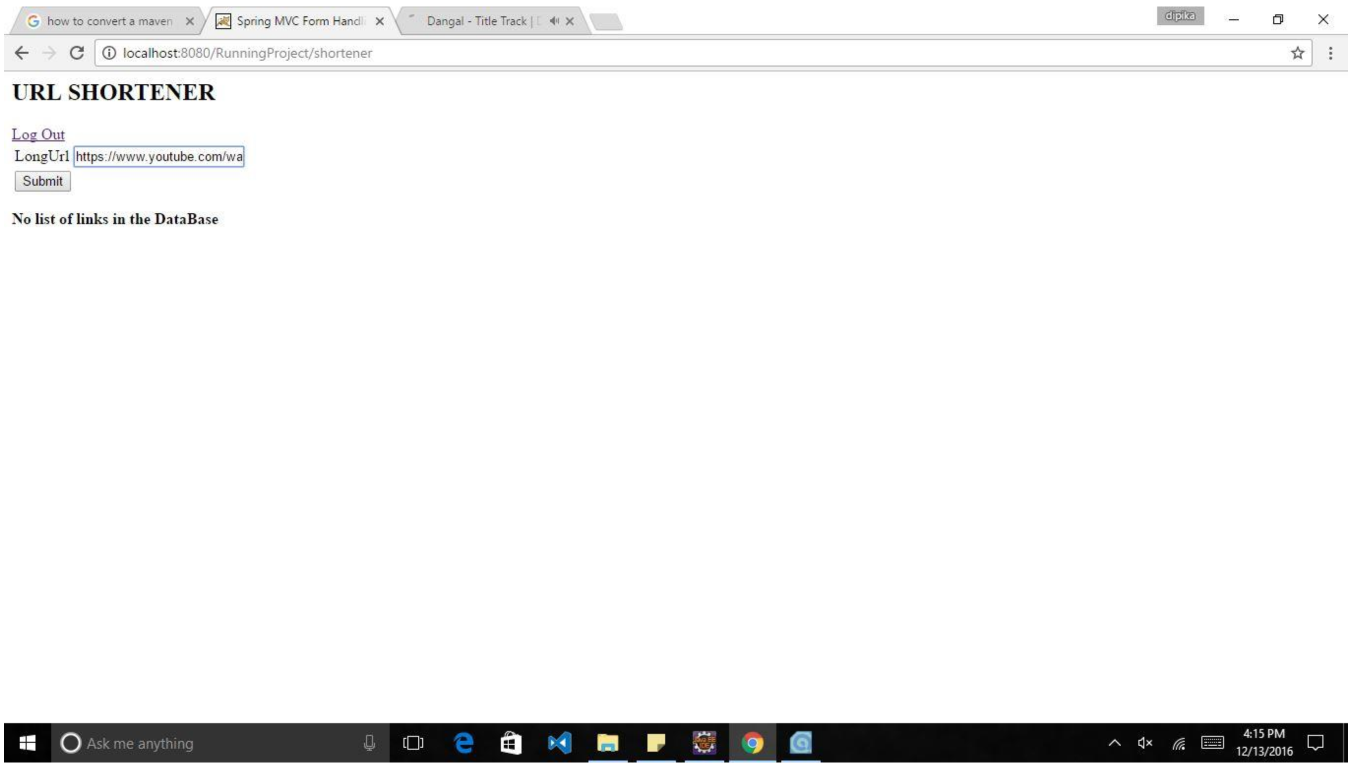
Log Out

LongUrl

Submit

No list of links in the DataBase

Users can create a short url by entering long url by in the textbox.



URL SHORTENER

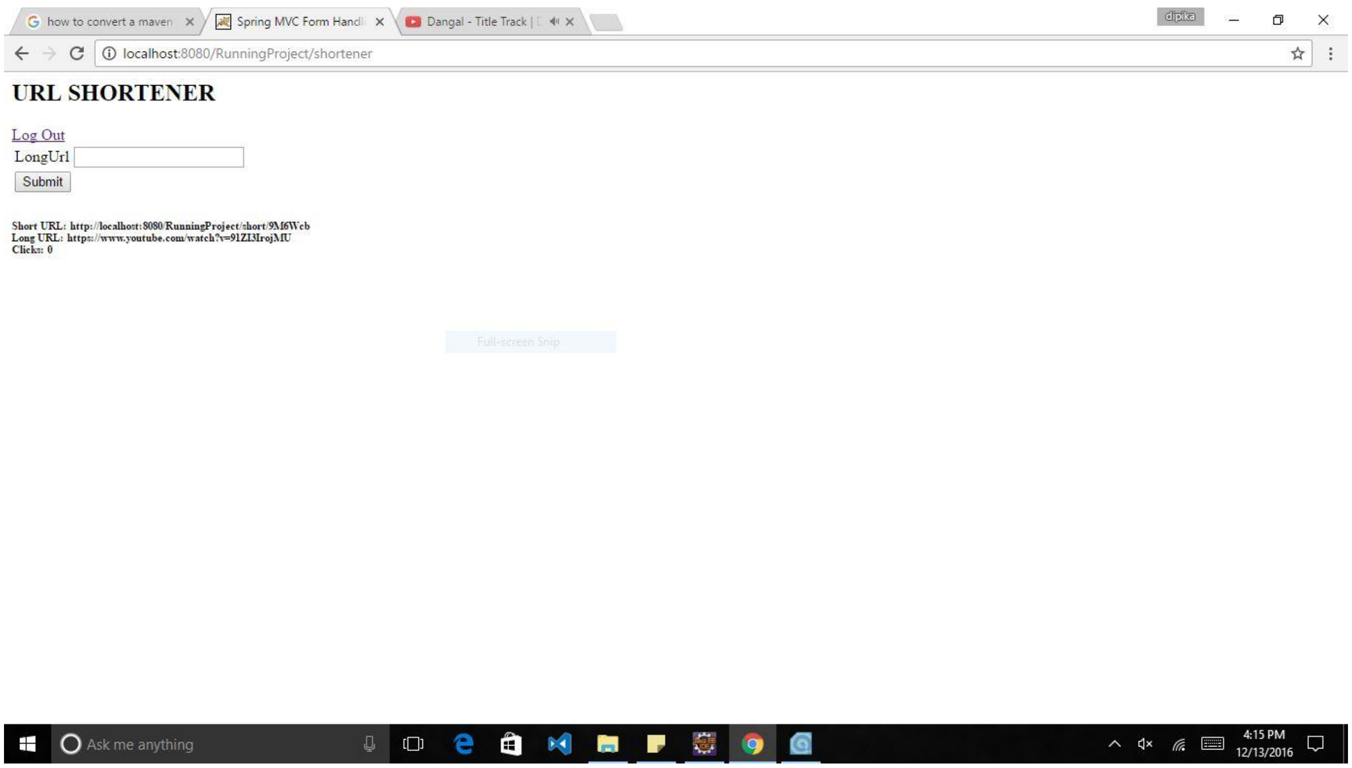
Log Out

LongUrl https://www.youtube.com/wa

Submit

No list of links in the DataBase

After entering a long url and a corresponding short url is generated. The created short url is displayed as follows. The no of clicks is 0 since the link is not visited.



URL SHORTENER

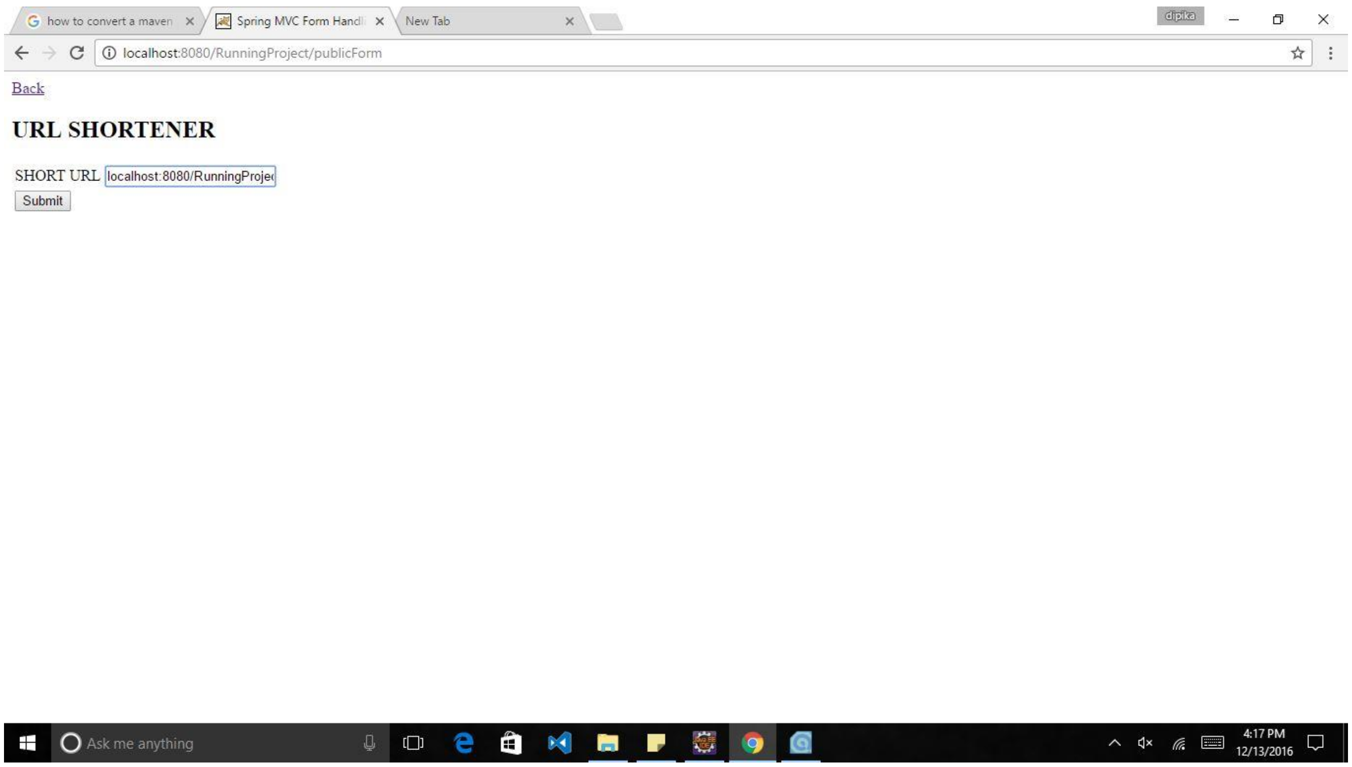
Log Out

LongUrl

Submit

Short URL: http://localhost:8080/RunningProject/short/9M6Wcb
Long URL: https://www.youtube.com/watch?v=91ZI3IrojMU
Clicks: 0

This is the public page where user can enter a short url and get back the corresponding long url.



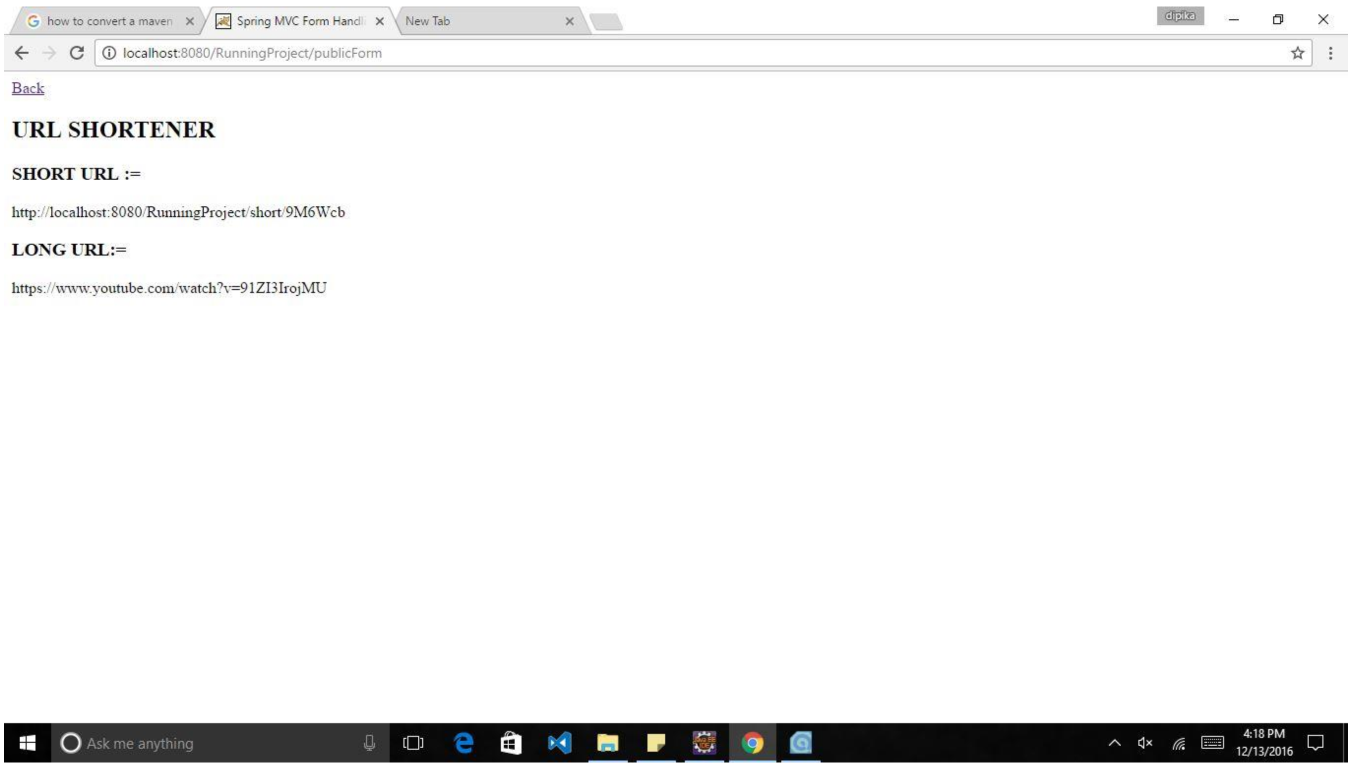
Back

URL SHORTENER

SHORT URL localhost:8080/RunningProje

Submit

After entering the short url, this page is displayed showing the long url for the short url.


localhost:8080/RunningProject/publicForm

Back

URL SHORTENER

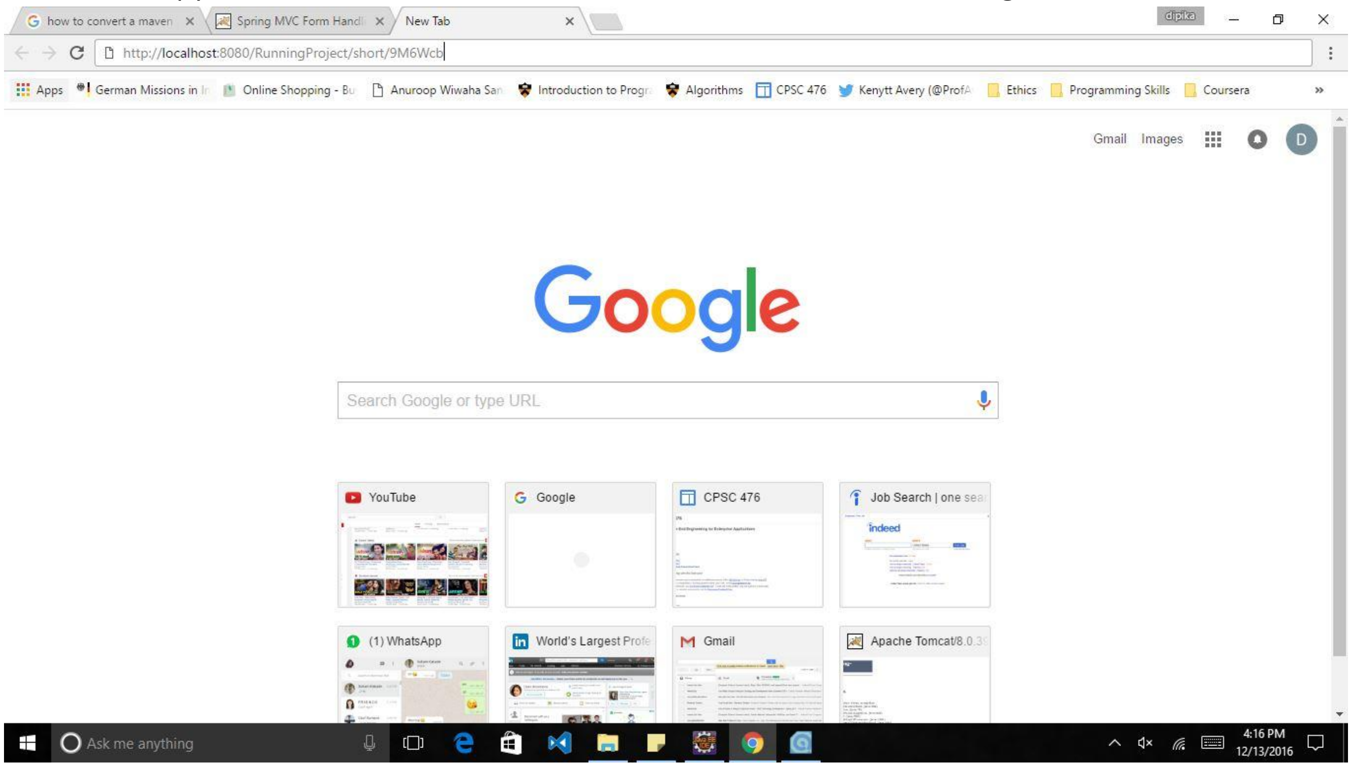
SHORT URL :=

http://localhost:8080/RunningProject/short/9M6Wcb

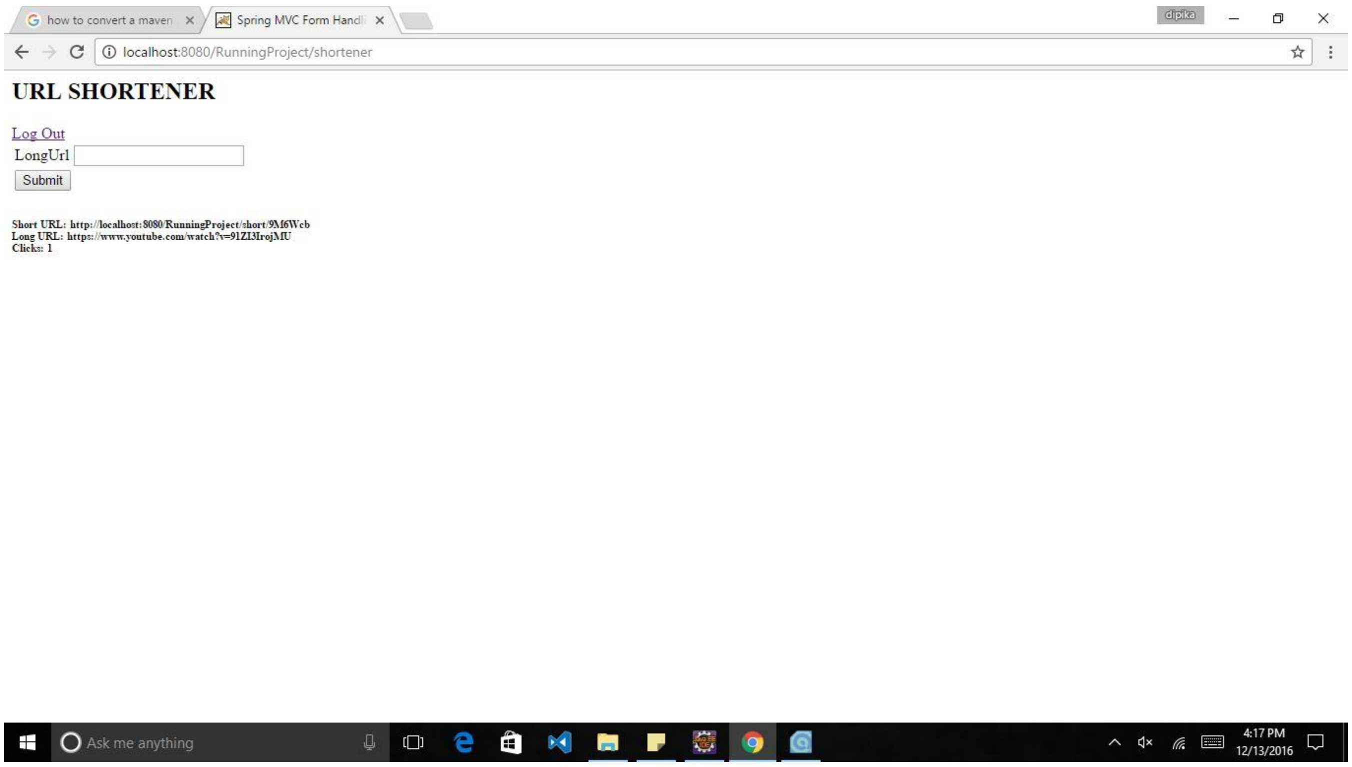
LONG URL:=

https://www.youtube.com/watch?v=91ZI3IrojMU

Also, If we copy the short url link in the browser, we are redirected to the original website.


http://localhost:8080/RunningProject/short/9M6Wcb

If we again visit the same short link, the count gets updated each time.



URL SHORTENER

Log Out
LongUrl
Submit

Short URL: http://localhost:8080/RunningProject/short/9M6Wcb
Long URL: https://www.youtube.com/watch?v=91ZI3IrojMU
Clicks: 1

6. Challenges

The challenges faced during the project were:

1. Understanding the structure of Spring MVC was a difficult task.
2. In the redirecting section, the website was redirecting infinite times. So, we did the following changes:
   return "redirect:/file.jsp"
   file.jsp is the name of the jsp file we want to redirect

7. References

1. What is the architecture of a scalable URL shortener?

https://www.quora.com/What-is-the-architecture-of-a-scalable-URL-shortener?share=1

2. To create random string with random characters

http://www.java2novice.com/java-collections-and-util/random/string/

3. Web MVC Framework

http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html