CISCO- Java Project

****

Disclaimer:

This Document Contains the Project Approach, Problem Statement, and Solution Submitted by Me for CISCO NCH Java Project.

Email: ***devangsharma419@gmail.com***

****

Problem Statement

DESCRIPTION

Create a web application using Java technology platform, that uses the Java Web component technology, capable of predicting the weather in any given town/city in the world.

The application expects the users to be able to design component-based applications that use two or more components, in a distributed environment, to build more serious web-based applications. The application building expects the users to use their knowledge of Java based web applications, using web components, such as Servlets, Java Server Pages, HTML components, ReSTful services, Forms and Validation, Page Navigation, Web Page design etc.

The Web application should have the following features:

- The application should be able to be deployed on a local system
- The application should use the latest Tomcat server
- The application should be developed on Eclipse IDE with Tomcat deployed on the IDE
- The application should present at least three pages

- The ‘About’ page should give a description about the student, his credentials, and description of the application
- The ‘GeoLocation’ page should allow the student to fetch the latitude and longitude of a given geo location in the globe. To do this, the app should make use of a MapBox web service API and the response should be latitude and longitude in the form of JSON data.
- Note: For a given name, the web service api could fetch 0 or more results. If the number of results is zero, then, it should ask you to re-enter the name of the place correctly. If the results are more than 1, the number of results should be limited to 1 only.
- The ‘Weather’ page should use the result from the GeoLocation page and make a request to another web service API call to openweathermap app, which will use the geo location coordinates and fetch the ‘current’ weather details, and tabulate on the page.
- The application should be testable for any city to get the current weather forecast

Assumptions

The project assumes the following pre-requisites on the part of the student.

- Knowledge and experience on Servlets web component technology
- Knowledge and experience on using Web Services and JSON
- Knowledge and hands-on experience on Eclipse IDE
- Knowledge and experience on installing, configuring and using Tomcat server on Eclipse IDE

Time Allocated

4 hours

The time included considers the the installation of Eclipse IDE and Tomcat server installation and configuration, Time for registering and getting the Web Services API Keys

Web Services API

The students should register on the following web services sites to get a free API key to make their weather app work.

MapBox

Sign-up / Registration link: https://account.mapbox.com/auth/signup/

- The student uses the above link to reach the signup page of https://www.mapbox.com/ web site.
- Create the Mapbox account by providing suitable values on username, password, etc. and click on the Get Started button.
- Note: You may have to login into your email, and confirm that you are signing up for the API services. You are now ready for signing into mapbox account
- Login into a mapbox account with your emaIl address and password provided earlier.
- You will be taken into the dashboard of your mapbox for your account.
- Create an access token (Default Public Token) which looks like: pk.eys1Ijoic3VtYXJidiIsImEiIiJjangwY2trbngxb2s4NGFs3jV012xrZmxjIn0.YDUIJjiznQQ2yQugftYt 1A

- Use this token along with the Web Services URL for invoking the web services on MapBox, something like the following (Boston as the place example):
- https://api.mapbox.com/geocoding/v5/mapbox.places/Boston.json?access_token=%20pk.eys1Ijoic3VtYXJidiIsImEiIiJjangwY2trbngxb2s4NGFs3jV012xrZmxjIn0.YDUIJjiznQQ2yQugft%20Yt1A&limit=1'
- The result will be JSON data, and you need to choose ‘centre’ and the key for which there will be two values - ‘latitude’ and ‘longitude’. This value, you will use in the next web services api to get the weather values, at Boston, in the above example.

See the following YouTube link for details for signing up with Mapbox API services provider: https://www.youtube.com/watch?v=A6x_uChn_rk

OpenWeatherMap

Sign-up registration link:https://home.openweathermap.org/users/sign_up

- The student uses the above link to reach the signup page of https://www.mapbox.com/ web site Create New Account page. Create an account by providing suitable values on username, email and passwords. click on I am Not a Robot Captcha button and click on the Create Account button.
- Note: You may have to login into your email and confirm that you are signing up for the API services. You are now ready for signing into OpenWeather account
- Login into OpenWeather account with your email address and password provided earlier.
- In the Dashboard, click on API Keys menu on the Nav bar
- You will be taken to another page in which you will have the key and Name (Default). The api key will look like the following: 2e1e92339c1a941712e97dfd4c67s51b
- Use the following URL for fetching the Weather at a given latitude and longitude: : : https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={your api key}
- Use this key to fetch the weather in a given place.

Solution

Concepts Used:

Java based web applications, using web components, such as Servlets, Java Server Pages, HTML components, RESTful services, Forms and Validation, Page Navigation, Web Page design etc.ncepts Used:

Tomcat Version Used:
Apache Tomcat Server: 10

API Used:
OpenWeatherMap

*(Result will be JSON data, and you need to choose 'centre' and the key for which there will be two values - **'latitude'** and '**longitude'**. This value, you will use in the next web services api to get the weather values, at Boston, in the above example.)*

Github Repo Link:
https://github.com/Devang-25/CISCO-NCH--Java-Project--Devang-Sharma

Application.Properties

```
server.port = 8081
spring.data.mongodb.database=springmongodb
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
weather.url=api.openweathermap.org/data/2.5/weather
weather.apikey=1c9770dfaf3b327dd03510a4c07b7f2d
```

Weather Configuration

```
package com.spring.restapi.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import
org.springframework.context.support.PropertySourcesPlaceholderConfigurer;

import com.spring.restapi.models.WeatherUrl;

/**
 * @author Devang
 */

@Configuration
@PropertySource("classpath:application.properties")
@ComponentScan
public class WeatherConfigurations {

    @Value("${weather.url}")
    private String url;

    @Value("${weather.apikey}")
    private String apikey;

    @Bean
    public static PropertySourcesPlaceholderConfigurer
propertySourcesPlaceholderConfigurer() {

        PropertySourcesPlaceholderConfigurer c = new
PropertySourcesPlaceholderConfigurer();
        c.setIgnoreUnresolvablePlaceholders(true);
        return c;
```

```
    }

    @Bean
    public WeatherUrl weatherUrl() {

        WeatherUrl weatherUrl = new WeatherUrl();
        weatherUrl.setUrl(url);
        weatherUrl.setApiKey(apikey);
        return weatherUrl;
    }

}
```

REST API Configuration

```
package com.spring.restapi.config;

/**
 * @author Devang
 */

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class RestTemplateConfig {

    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }

}
```

Form City Attribute Model

```java
package com.spring.restapi.models;

/*
 * @author Devang
 */

public class FormCityAttribute {

    private String city;

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public FormCityAttribute(String city) {
        super();
        this.city = city;
    }

    public FormCityAttribute() {
        super();
    }
}
```

Product Module

```
package com.spring.restapi.models;

/**
 * @author Devang
 */

import java.io.Serializable;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "productdb")
public class Product implements Serializable {

    private static final long serialVersionUID = 5604944562275121069L;

    @Id
    private String id;
    private String product;
    private String productType;
    private String description;

    public Product(  String product,String productType, String
description) {

        this.product = product;
        this.description = description;
        this.productType = productType;
    }

    public Product() {
        super();
        // TODO Auto-generated constructor stub
    }
```

```
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getProduct() {
        return product;
    }

    public void setProduct(String product) {
        this.product = product;
    }

    public String getProductType() {
        return productType;
    }

    public void setProductType(String productType) {
        this.productType = productType;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

}
```

Weather Module

```java
package com.spring.restapi.models;

/*
 * @author Devang
 */


import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;

import java.io.Serializable;
import java.util.List;
import java.util.Map;

import org.springframework.context.annotation.Bean;

@JsonIgnoreProperties(ignoreUnknown = true)
public class Weather implements Serializable {

    private static final long serialVersionUID = 740621062818244090 2L;

    private String weatherDescription;
    private double lon;
    private String name;
    private double lat;

    @Bean
    public Weather weather() {
        return new Weather();
    }

    public Weather() {
        super();
        // TODO Auto-generated constructor stub
    }
```

```java
public Weather(Weather weather) {
    // TODO Auto-generated constructor stub
}

public double getLat() {
    return lat;
}

@JsonProperty("lat")
public void setLat(double lat) {
    this.lat = lat;
}


public String getName() {
    return name;
}

@JsonProperty("name")
public void setName(String name) {
    this.name = name;
}

public String getWeatherDescription() {
    return weatherDescription;
}

public void setWeatherDescription(String weatherDescription) {
    this.weatherDescription = weatherDescription;
}

@JsonProperty("weather")
public void setWeather(List<Map<String, Object>> weatherEntries) {
    Map<String, Object> weather = weatherEntries.get(0);
    setWeatherDescription((String) weather.get("description"));
}
```

```
    @JsonProperty("lon")
    public double getLon() {
        return lon;
    }

    @JsonProperty("lon")
    public void setLon(double lon) {
        this.lon = lon;
    }

    @JsonProperty("coord")
    public void setCoord(Map<String, Object> coord) {
        setLon((double) coord.get("lon"));
        setLat((double) coord.get("lat"));

    }



}
```

WeatherURL Module

```
package com.spring.restapi.models;

/*
 * @author Devang
 */


public class WeatherUrl {

    private String url;
    private String apiKey;

    public WeatherUrl() {
        super();
    }

    public String getApiKey() {
        return apiKey;
    }

    public void setApiKey(String apiKey) {
        this.apiKey = apiKey;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }


}
```

Product Repository

```java
package com.spring.restapi.repositories;

import java.util.Optional;

import org.springframework.data.mongodb.repository.MongoRepository;

import com.spring.restapi.models.Product;


/*
 * @author Devang
 */



public interface ProductRepository extends MongoRepository<Product,String>
{
    public Optional<Product> findById(String id);
}
```

Product Service

```java
package com.spring.restapi.services;

import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.spring.restapi.models.Product;
import com.spring.restapi.repositories.ProductRepository;

/*
 * @author Devang
 */

@Service
public class ProductService {
    @Autowired
    private ProductRepository productRepository;

    public Iterable<Product> getAllProducts(){
        return productRepository.findAll();
    }

    public  Optional<Product> findProductById(String id) {
        return productRepository.findById(id);
    }

    public void saveProduct(Product product) {
        productRepository.save(product);
    }

    public void deleteProductById(String id) {
        productRepository.deleteById(id);
    }

}
```

Product Controller

```
package com.spring.restapi.controllers;

import java.util.Optional;

/**
 * @author Devang
 */

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import com.spring.restapi.models.Product;
import com.spring.restapi.services.ProductService;

import org.springframework.ui.Model;


@Controller
public class ProductController {

    @Autowired
    private ProductService productService;

    @RequestMapping(method=RequestMethod.GET, value="/")
    public String indexPage() {

         return "redirect:" + "/product";
    }
```

```
    @RequestMapping(method=RequestMethod.GET, value="/product")
    public String product(Model model) {

        model.addAttribute("mongoData", productService.getAllProducts());
        return "mongoList";
    }

    //CRUD operations
    @RequestMapping(method=RequestMethod.GET, value="api/products")
    @ResponseBody
    public Iterable<Product> getAllProducts() {
        return productService.getAllProducts();
    }

    @RequestMapping(method=RequestMethod.GET, value="api/product/{id}")
    public ResponseEntity<Optional<Product>>
getProductById(@PathVariable(value="id") String id) {
        Optional<Product> prod = productService.findProductById(id);
        if(prod == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok().body(prod);
     }

    @RequestMapping(method=RequestMethod.POST, value="api/product")
    @ResponseBody
    public ResponseEntity<String> saveProducts(@RequestBody Product
product) {
        productService.saveProduct(product);
        return ResponseEntity.ok().body(new String("Resource successFully
created!!!"));
    }

    @RequestMapping(method=RequestMethod.DELETE, value="api/product/{id}")
    public ResponseEntity<String>
deleteProductById(@PathVariable(value="id") String id) {
        productService.deleteProductById(id);
```

```
        return ResponseEntity.ok().body(new String("Deleted
successFully"));

    }

}
```

Weather Controller

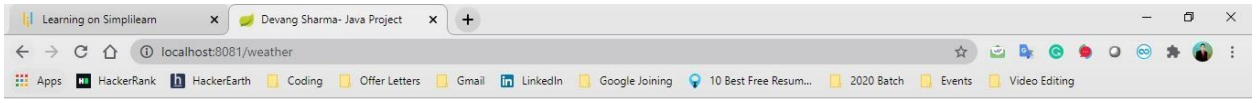
```
package com.spring.restapi.controllers;

/**
 * @author Devang
 */

import java.io.IOException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponents;
import org.springframework.web.util.UriComponentsBuilder;

import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.spring.restapi.models.FormCityAttribute;
import com.spring.restapi.models.Weather;
import com.spring.restapi.models.WeatherUrl;

    @ComponentScan("com.spring.restapi.config")
    @Controller
    public class WeatherController {

        @Autowired
        RestTemplate restTemp;
```

```
        @Autowired
        private WeatherUrl weatherData;

        @RequestMapping(value = "/weather",method=RequestMethod.GET )
        public String CityForm(Model model) {

            model.addAttribute("city", new FormCityAttribute());
            return "formData";
        }

        @RequestMapping(value = "/weather",method=RequestMethod.POST )
        public String getWeather(Model model, @ModelAttribute
FormCityAttribute city)
                throws JsonParseException, JsonMappingException,
IOException {

            UriComponents uriComponents = UriComponentsBuilder
                    .newInstance()
                    .scheme("http")
                    .host(weatherData.getUrl())
                    .path("")
                    .query("q={keyword}&appid={appid}")

.buildAndExpand(city.getCity(),weatherData.getApiKey());
            String uri = uriComponents.toUriString();
            ResponseEntity<String> resp= restTemp.exchange(uri,
HttpMethod.GET, null, String.class);

            ObjectMapper mapper = new ObjectMapper();
            Weather weather = mapper.readValue(resp.getBody(),
Weather.class);
            model.addAttribute("weatherData", weather);

            return "weatherDetails";
        }

    }
```

DEMO Application:

```java
package com.spring.restapi;

/**
 * @author Devang

 */

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@ComponentScan("com.spring.restapi")
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

SCREENSHOTS

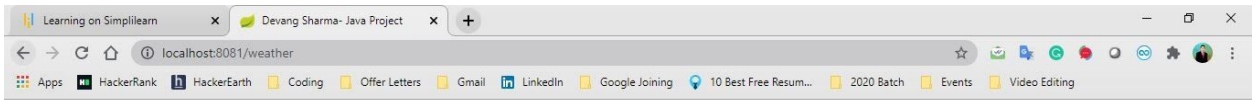


Java Weather Web Application Project By: DEVANG SHARMA

Data fetched from MongoDB

Geo-Location WEATHER ABOUT

City	Latitude	Longitude	Weather
Delhi	28.67	77.22	haze

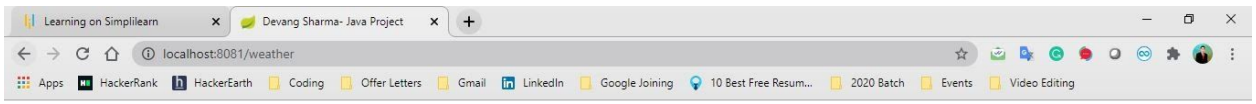


Java Weather Web Application Project By: DEVANG SHARMA

Data fetched from MongoDB

Geo-Location WEATHER ABOUT

City	Latitude	Longitude	Weather
Bengaluru	12.98	77.6	scattered clouds



Java Weather Web Application Project By: DEVANG SHARMA

Data fetched from MongoDB

Geo-Location WEATHER ABOUT

City	Latitude	Longitude	Weather
New York	40.71	-74.01	few clouds