Sofia University “St. Kliment Ohridski”

Faculty of Mathematics and Informatics

Weather Forecast

Design and Integration of Software Systems Coursework

Stanislav Peshterliev, 61096

28 January 2011

Table of contents

1. Task

To develop application that uses the web service provided by http://ws.cdyne.com/WeatherWS/Weather.asmx?wsdl. For given US ZIP code, the service provides information about current weather conditions and 7 days forecast. Also an URL to GIF image is returned that can be used for graphical weather indicator.
In order to improve user experience I decided to user two additional web services for converting city name to ZIP code, and displaying the current location on a map.

2. Technologies

The project is implemented using technologies:

- Java[7] and Ruby[4] programming languages.
- HTML, CSS, JavaScript, Servlets, JSP for user interface implementation
- Apache Maven[1] is used for dependency management and build tool.
- Eclipse[3] IDE for Java EE Developer with m2eclipse plugin
- JAX-WS[2] framework for working with SOAP web services
- Apache HTTP client for consuming REST style web services
- Google Maps for displaying a map with current location

3. Implementation

To work with the weather web service JAX-WS client side stub is generated from WSDL file. The following command generates all needed classes.

```
wsimport -version -keep http://ws.cdyne.com/WeatherWS/Weather.asmx?wsdl
```

The generated stub is used in Java web application that renders provided information and GIF image as JSP web page.

The web service for converting from city name to ZIP code is implemented in Ruby programming language using libraries: Sinatra[5] web framework, Sequel[8] database toolkit and Nokogiri[6] XML parser. The source code can be found in the project's sub folder zipcodes. In order to start the web service the above libraries should be installed using the following commands.

```
gem install sinatra
gem install sequel
gem install nokogiri
```

To start the service, execute the following command in zipcodes folder

```
ruby -rubygems app.rb
```

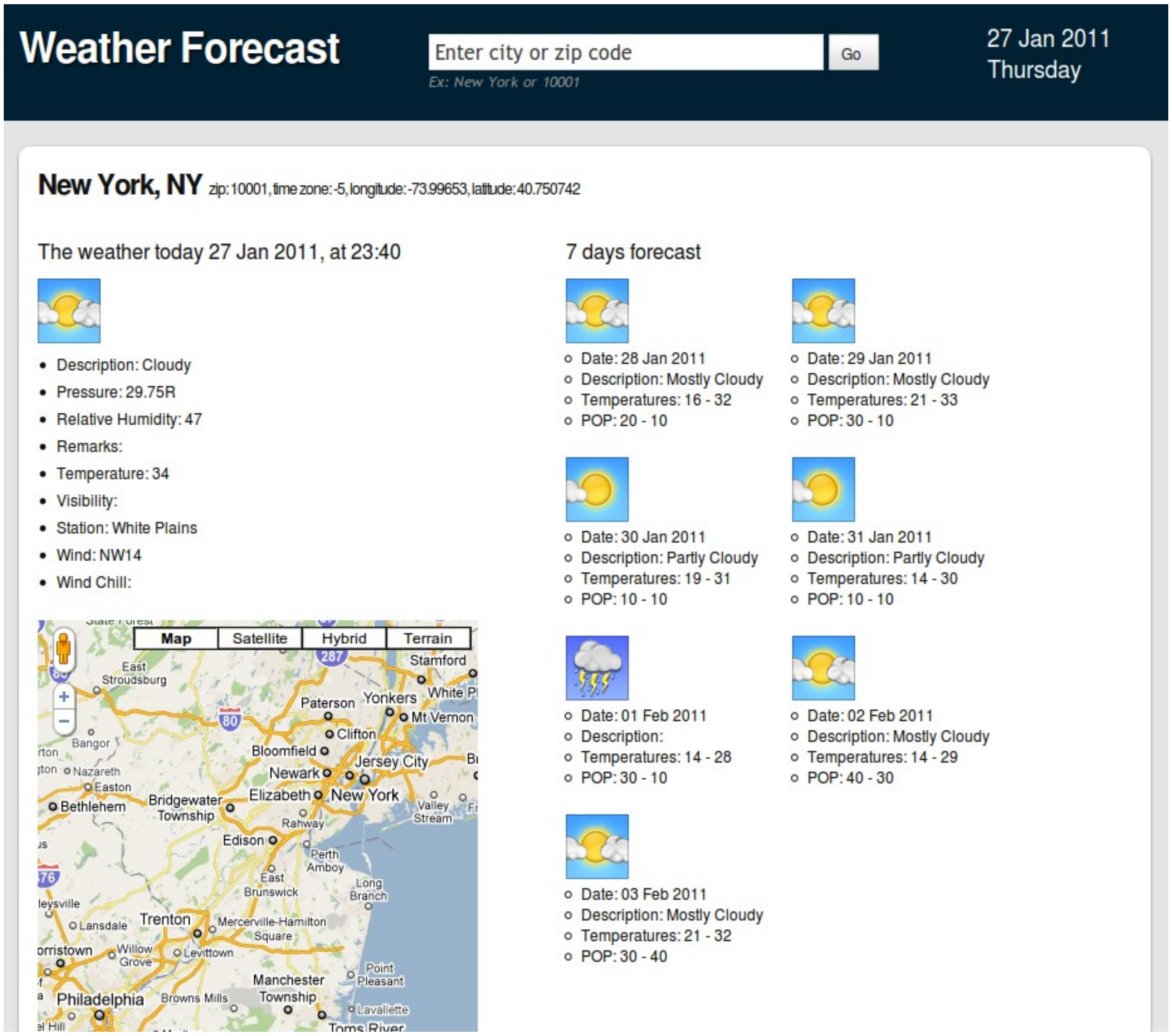
After that you can access the service via url address http://localhost:4567/zipcodes/[City name]. Sample output for New York will be as follows:

```
<?xml version="1.0"?>
<cities>
  <city>
    <longitude>-73.99653</longitude>
    <city>New York</city>
    <zip>10001</zip>
    <state>NY</state>
    <dst>1</dst>
    <timezone>-5</timezone>
    <latitude>40.750742</latitude>
  </city>
</cities>
```

To start the main web application, using Jetty web server, execute the following command in project's root folder.

```
mvn jetty-run
```

The application can be accessed via url address http://localhost:8080/weather. A search for New York would give the following result.



New York, NY zip:10001,time zone:-5,longitude:-73.99653,latitude:40.750742

The weather today 27 Jan 2011, at 23:40

- Description: Cloudy
- Pressure: 29.75R
- Relative Humidity: 47
- Remarks:
- Temperature: 34
- Visibility:
- Station: White Plains
- Wind: NW14
- Wind Chill:

7 days forecast

- Date: 28 Jan 2011
- Description: Mostly Cloudy
- Temperatures: 16 - 32
- POP: 20 - 10

- Date: 29 Jan 2011
- Description: Mostly Cloudy
- Temperatures: 21 - 33
- POP: 30 - 10

- Date: 30 Jan 2011
- Description: Partly Cloudy
- Temperatures: 19 - 31
- POP: 10 - 10

- Date: 31 Jan 2011
- Description: Partly Cloudy
- Temperatures: 14 - 30
- POP: 10 - 10

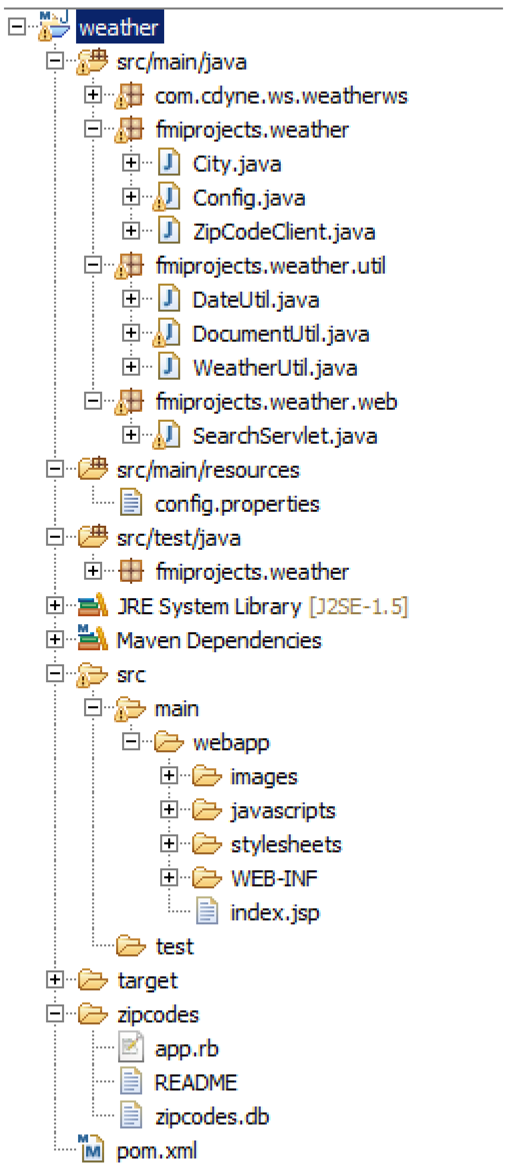
- Date: 01 Feb 2011
- Description:
- Temperatures: 14 - 28
- POP: 30 - 10

- Date: 02 Feb 2011
- Description: Mostly Cloudy
- Temperatures: 14 - 29
- POP: 40 - 30

- Date: 03 Feb 2011
- Description: Mostly Cloudy
- Temperatures: 21 - 32
- POP: 30 - 40

4. Package structure

The project has the following package structure



```
weather
  src/main/java
    com.cdyne.ws.weatherws
    fmiprojects.weather
      City.java
      Config.java
      ZipCodeClient.java
    fmiprojects.weather.util
      DateUtil.java
      DocumentUtil.java
      WeatherUtil.java
    fmiprojects.weather.web
      SearchServlet.java
  src/main/resources
    config.properties
  src/test/java
    fmiprojects.weather
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
    main
      webapp
        images
        javascripts
        stylesheets
        WEB-INF
        index.jsp
    test
  target
  zipcodes
    app.rb
    README
    zipcodes.db
  pom.xml
```

- fmiprojects.weather contains classes for accessing the web service for converting from city name to zipcode and Config class for reading configuration options like web service urls
- fmiprojects.weather.util – utility classes used by other classes to remove repetition and abstract some logic
- fmiprojects.weather.web contains the servlet that calls city name to zip code web service and returns json that is used in text box autocomplte
- src/main/resources contains application resources, currently only configuration file
- src/test/java contains JUnit test cases used to test interaction with web services.
- src/main/webapp contains all code for rendering web frontend - JSP, JS, CSS and images.

- Zipocodes contains the implementation of the web service for converting from city name to zip code and vice versa.
- pom.xml (Project Object Model) is the file that contains Maven configuration for dependency and build.

5. References

1. http://maven.apache.org/
2. http://jax-ws.java.net/.
3. http://eclipse.org/
4. http://www.ruby-lang.org
5. http://www.sinatrarb.com
6. http://nokogiri.org
7. http://www.java.com
8. http://sequel.rubyforge.org/