Maven Plugins: Jersey, Jetty, Metrics, Docker and Web.xml 2

Publish Web-Service 3

JSON, XML Responses and Integration Tests 3

Documentation 4

Data structure and FacilityRateTest 5

Decouple Policy for Evaluating Request from Rate 7

## Maven Plugins: Jersey, Jetty, Metrics, Docker and Web.xml

1. Use Jersey archetype to generate project skeleton. Does not use Spring libraries, DropWizard or Scalatra.

mvn archetype:generate \

-DarchetypeArtifactId=jersey-quickstart-webapp \

-DarchetypeGroupId=org.glassfish.jersey.archetypes \

-DinteractiveMode=true \

2. Use Maven plugin to run project locally.

mvn clean compile exec:java

curl "http://localhost:8080/myapp/spotresource?start=2015-07-01T07:00:00Z&stop=2015-07-01T12:00:00Z"

3. Publish services in WEB.XML. Project also provides Maven Jetty plugin for testing.

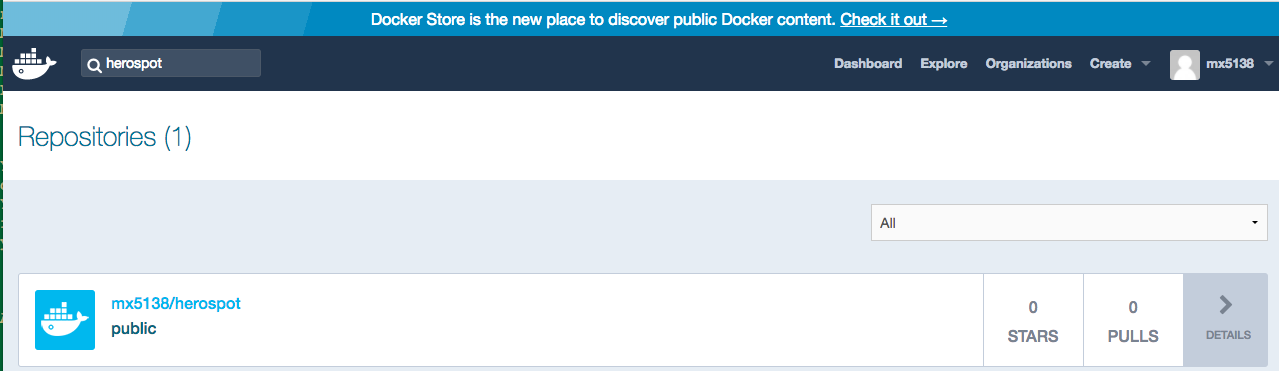
mvn clean compile jetty:run

curl “http://localhost:8080/parking-service/rest/spotresource?start=2015-07-01T07:00:00Z&stop=2015-07-01T12:00:00Z"

4. Annotate Timed and provide Metrics-Servlet in WEB.XML for metrics.

curl <http://localhost:8080/parking-service/metrics>

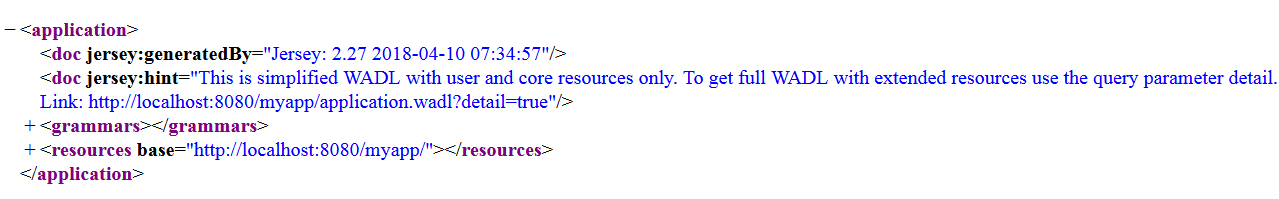
5. Push image to Docker.io. Use image to publish to AWS/Azure/Google.

mvn clean install docker:build

## Publish Web-Service

#curl http://localhost:8080/myapp/application.wadl

Point browser to http://localhost:8080/myapp/application.wadl



## JSON, XML Responses and Integration Tests

#curl "http://localhost:8080/myapp/spotresource?start=2015-07-01T07:00:00Z&stop=2015-07-01T12:00:00Z"

1500

#curl "http://localhost:8080/myapp/spotresource?start=2015-07-04T07:00:00Z&stop=2015-07-04T12:00:00Z"

2000

#curl "http://localhost:8080/myapp/spotresource?start=2015-07-04T07:00:00Z&stop=2015-07-04T20:00:00Z"

unavailable

# Complete code in Git

**public** **class** SpotResourceTest {

@Test

**public** **void** testSpotResource() {

}

@Test

**public** **void** testSpotResourceJson() {

}

@Test

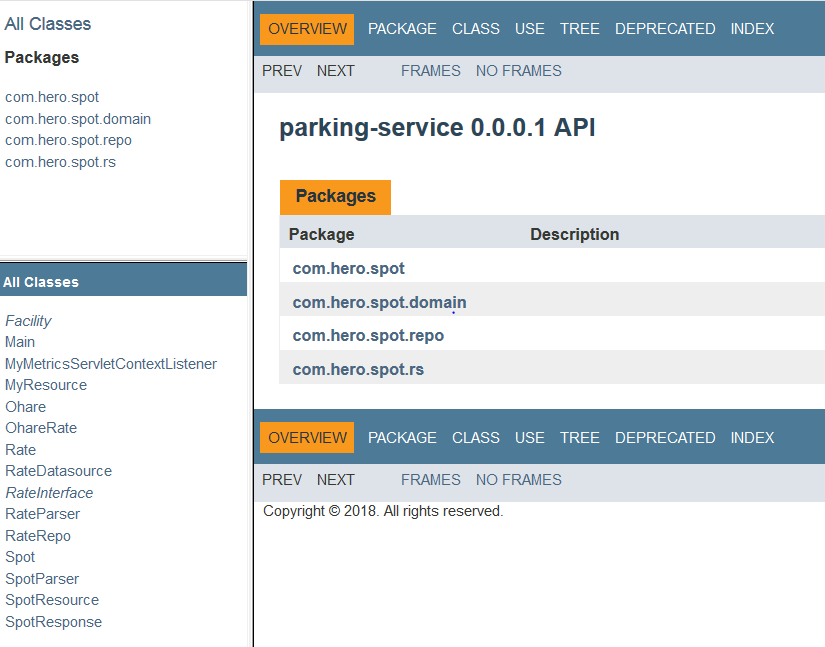
**public** **void** testSpotResourceXML() {

}

}

## Documentation

mvn javadoc:javadoc generates API.



## Data structure and FacilityRateTest

Facility available spots are contained inside a two-dimensional array. Spot provides information about end-time, start-time and price. The caller class can then use the information to determine if the spot is available.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Mon |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Tues |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Wed |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Thurs |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fri |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Sat |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Sun |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

package com.hero.spot.domain;

import java.time.DayOfWeek;

import java.time.LocalTime;

/\*\*

\* Facility's spots are in two dimensional array that use DayOfWeekIndx and LocalTimeHourIndx

\*

\* Monday 0100 = availableSpots[0][1]

\*

\*/

public interface Facility {

/\*\*

\* Get Spot given the day of the week and the hour.

\* If the spot is available for booking, it will contain the

\* start-time, end-time and price.

\*

\* @param d

\* @param hr

\* @return

\*/

Spot getSpot(DayOfWeek d, LocalTime hr);

Spot[][] getAvailableSpots();

}

**package** com.hero.spot.repo;

/\*\*

\* Main test to check the validity of the algorithm. Rates are from the assignment

\*/

**public** **class** FacilityRateTest {

@Before

**public** **void** setUp() {

rates = **new** ArrayList();

Rate r1 = **new** Rate();

r1.setDays("mon,tues,thurs");

r1.setTimes("0900-2100");

r1.setPrice(1500);

rates.add(r1);

Rate r2 = **new** Rate();

r2.setDays("fri,sat,sun");

r2.setTimes("0900-2100");

r2.setPrice(2000);

rates.add(r2);

Rate r3 = **new** Rate();

r3.setDays("wed");

r3.setTimes("0600-1800");

r3.setPrice(1750);

rates.add(r3);

Rate r4 = **new** Rate();

r4.setDays("mon,wed,sat");

r4.setTimes("0100-0500");

r4.setPrice(1000);

rates.add(r4);

Rate r5 = **new** Rate();

r5.setDays("sun,tues");

r5.setTimes("0100-0700");

r5.setPrice(925);

rates.add(r5);

}

@Test

**public** **void** testSundayAvailableHours() {}

@Test

**public** **void** testSundayUnavailableHours() {}

@Test

**public** **void** testMidNiteHour() {}

@Test

**public** **void** testEndTime() {}

@Test

**public** **void** testAvailableTime() {}

@Test

**public** **void** testStartTime() {}

}

## Decouple Policy for Evaluating Request from Rate

Improve testability.

/\*\*

\* SpotPolicy evaluates the request's start-time and end-time against found spot.

\*

\* Decouples business policy from spot and rate.

\*

\*/

public interface SpotPolicyInterface {

String evaluateRequest(LocalTime startTime, LocalTime endTime, Spot found);

}

package com.hero.spot.rs;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertTrue;

import java.time.DayOfWeek;

import java.time.LocalTime;

import org.junit.Test;

import com.hero.spot.domain.Spot;

public class SpotPolicyTest {

@Test

public void testAvailableSpot() {

SpotPolicyInterface policy = new SpotPolicy();

Spot found = new Spot();

LocalTime spotStartTime = LocalTime.of(1, 0);

LocalTime spotEndTime = LocalTime.of(3, 0);

found.setDayOfWeek(DayOfWeek.MONDAY);

found.setHourOfDay(spotStartTime);

found.setStartTime(spotStartTime);

found.setEndTime(spotEndTime);

found.setPrice(Integer.valueOf(100));

LocalTime requestStartTime = LocalTime.of(1, 0);

LocalTime requestEndTime = LocalTime.of(2, 59);

String response = policy.evaluateRequest(requestStartTime, requestEndTime, found);

assertEquals(100, Integer.parseInt(response));

}

@Test

public void testExcludeEndTime() {

SpotPolicyInterface policy = new SpotPolicy();

Spot found = new Spot();

LocalTime spotStartTime = LocalTime.of(1, 0);

LocalTime spotEndTime = LocalTime.of(3, 0);

found.setDayOfWeek(DayOfWeek.MONDAY);

found.setHourOfDay(spotStartTime);

found.setStartTime(spotStartTime);

found.setEndTime(spotEndTime);

found.setPrice(Integer.valueOf(100));

LocalTime requestStartTime = LocalTime.of(1, 0);

LocalTime requestEndTime = LocalTime.of(3, 0);

String response = policy.evaluateRequest(requestStartTime, requestEndTime, found);

assertTrue("unavailable".equalsIgnoreCase(response));

}

}