Red Hat Process Automation Manager 7.8

Deploying and using the vehicle route planning starter application for Red Hat Business Optimizer

Last Updated: 2020-09-07

Red Hat Process Automation Manager 7.8 Deploying and using the vehicle route planning starter application for Red Hat Business Optimizer

Legal Notice

Abstract

This documents describes how to use the OptaWeb Vehicle Routing starter application for Red Hat Business Optimizer.

Table of Contents

PREFACE

As a developer, you can use the OptaWeb Vehicle Routing starter application to optimize your vehicle fleet deliveries.

Prerequisites

- OpenJDK (JDK) 8 is installed. Red Hat build of Open JDK is available from the Software Downloads page in the Red Hat Customer Portal (login required).
- Apache Maven 3.6 or higher is installed. Maven is available from the Apache Maven Project website.

CHAPTER 1. WHAT IS OPTAWEB VEHICLE ROUTING?

The main purpose of many businesses is to transport various types of cargo. The goal of these businesses is to deliver a piece of cargo from the loading point to a destination and use its vehicle fleet in the most efficient way. One of the main objectives is to minimize travel costs which are measured in either time or distance.

This type of optimization problem is referred to as the vehicle routing problem (VRP) and has many variations.

Red Hat Business Optimizer can solve many of these vehicle routing variations and provides solution examples. Red Hat Business Optimizer enables developers to focus on modeling business rules and requirements instead of learning constraint programming theory. OptaWeb Vehicle Routing expands the vehicle routing capabilities of Red Hat Business Optimizer by providing a reference implementation that answers questions such as these:

- Where do I get the distances and travel times?
- How do I visualize the solution on a map?
- How do I build an application that runs in the cloud?

OptaWeb Vehicle Routing uses OpenStreetMap (OSM) data files. For information about OpenStreetMap, see the OpenStreetMap web site.

Use the following definitions when working with OptaWeb Vehicle Routing:

Region: An arbitrary area on the map of Earth, represented by an OSM file. A region can be a country, a city, a continent, or a group of countries that are frequently used together. For example, the DACH region includes Germany (DE), Austria (AT), and Switzerland (CH).

Country code: A two-letter code assigned to a country by the ISO-3166 standard. You can use a country code to filter geosearch results. Because you can work with a region that spans multiple countries (for example, the DACH region), OptaWeb Vehicle Routing accepts a list of country codes so that geosearch filtering can be used with such regions. For a list of country codes, see *ISO 3166 Country Codes*

Geosearch: A type of query where you provide an address or a place name of a region as the search keyword and receive a number of GPS locations as a result. The number of locations returned depends on how unique the search keyword is. Because most place names are not unique, filter out nonrelevant results by including only places in the country or countries that are in your working region.

CHAPTER 2. DOWNLOAD AND BUILD THE OPTAWEB VEHICLE ROUTING DEPLOYMENT FILES

You must download and prepare the deployment files before building and deploying OptaWeb Vehicle Routing.

Procedure

1. Navigate to the Software Downloads page in the Red Hat Customer Portal (login required), and select the product and version from the drop-down options:
   - Product: Red Hat Process Automation Manager
   - Version: 7.8
2. Download **Red Hat Process Automation Manager 7.8 Reference Implementations**(**rhpam-7.8.0-reference-implementation.zip**).
3. Download **Red Hat Process Automation Manager 7.8 Maven Repository**(**rhpam-7.8.0-maven-repository.zip**).
4. Extract the **rhpam-7.8.0-maven-repository.zip** file.
5. Copy the contents of the **rhpam-7.8.0-maven-repository/maven-repository** subdirectory into the **~/.m2/repository** directory.
6. Extract the **rhpam-7.8.0-reference-implementation.zip** file. This archive contains three reference implementation ZIP files.
7. Extract the **rhpam-7.8.0-optaweb-vehicle-routing.zip** file.
8. Navigate to the **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources** directory.
9. Enter the following command to build OptaWeb Vehicle Routing:

```
mvn clean package -DskipTests
```

CHAPTER 3. RUN OPTAWEB VEHICLE ROUTING LOCALLY USING THE RUNLOCALLY.SH SCRIPT

Linux users can use the **runLocally.sh** Bash script to run OptaWeb Vehicle Routing.

NOTE

The **runLocally.sh** script does not run on macOS. If you cannot use the **runLocally.sh** script, see Chapter 4, *Configure and run OptaWeb Vehicle Routing manually* .

The **runLocally.sh** script automates the following setup steps that otherwise must be carried out manually:

- Create the data directory.
- Download selected OpenStreetMap (OSM) files from Geofabrik.
- Try to associate a country code with each downloaded OSM file automatically.
- Build the project if the standalone JAR file does not exist.
- Launch OptaWeb Vehicle Routing by taking a single region argument or by selecting the region interactively.

See the following sections for instructions about executing the **runLocally.sh** script:

- Section 3.1, “Run the OptaWeb Vehicle Routing runLocally.sh script in quick start mode”
- Section 3.2, “Run the OptaWeb Vehicle Routing runLocally.sh script in interactive mode”
- Section 3.3, “Run the OptaWeb Vehicle Routing runLocally.sh script in non-interactive mode”
- Section 3.4, “Run the OptaWeb Vehicle Routing runLocally.sh script in air distance mode”

3.1. RUN THE OPTAWEB VEHICLE ROUTING RUNLOCALLY.SH SCRIPT IN QUICK START MODE

The easiest way to get started with OptaWeb Vehicle Routing is to run the **runLocally.sh** script without any arguments.

Prerequisites

- OptaWeb Vehicle Routing has been successfully built with Maven as described in Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files* .
- Internet access is available.

Procedure

1. Enter the following command in the **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources** directory.

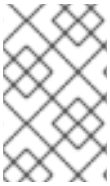
```
./runLocally.sh
```

2. If prompted to create the **.optaweb-vehicle-routing** directory, enter **y**. You are prompted to create this directory the first time you run the script.

3. If prompted to download an OSM file, enter **y**. The first time that you run the script, OptaWeb Vehicle Routing downloads the Belgium OSM file.
   The application starts after the OSM file is downloaded.

4. To open the OptaWeb Vehicle Routing user interface, enter the following URL in a web browser:

```
http://localhost:8080
```



NOTE

The first time that you run the script, it will take a few minutes to start because the OSM file must be imported by GraphHopper and stored as a road network graph. The next time you run the **runlocally.sh** script, load times will be significantly faster.

Next steps

Chapter 6, *Using OptaWeb Vehicle Routing*

3.2. RUN THE OPTAWEB VEHICLE ROUTING RUNLOCALLY.SH SCRIPT IN INTERACTIVE MODE

Use interactive mode to see the list of downloaded OSM files and country codes assigned to each region. You can use the interactive mode to download additional OSM files from Geofabrik without visiting the website and choosing a destination for the download.

Prerequisites

- OptaWeb Vehicle Routing has been successfully built with Maven as described in Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files* .
- Internet access is available.

Procedure

1. Change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources**.

2. Enter the following command to run the script in interactive mode:

```
./runLocally.sh -i
```

3. At the **Your choice** prompt, enter **d** to display the download menu. A list of previously downloaded regions appears followed by a list of regions that you can download.

4. Optional: Select a region from the list of previously downloaded regions:

   a. Enter the number associated with a region in the list of downloaded regions.

   b. Press the Enter key.

5. Optional: Download a region:

a. Enter the number associated with the region that you want to download. For example, to select the map of Europe, enter **5**.

b. To download the map, enter **d** then press the Enter key.

c. To download a specific region within the map, enter **e** then enter the number associated with the region that you want to download, and press the Enter key.

USING LARGE OSM FILES

For the best user experience, use smaller regions such as individual European or US states. Using OSM files larger than 1 GB will require significant RAM size and take a lot of time (up to several hours) for the initial processing.

The application starts after the OSM file is downloaded.

6. To open the OptaWeb Vehicle Routing user interface, enter the following URL in a web browser:

```
http://localhost:8080
```

Next steps

Chapter 6, *Using OptaWeb Vehicle Routing*

3.3. RUN THE OPTAWEB VEHICLE ROUTING RUNLOCALLY.SH SCRIPT IN NON-INTERACTIVE MODE

Use OptaWeb Vehicle Routing in non-interactive mode to start OptaWeb Vehicle Routing with a single command that includes an OSM file that you downloaded previously. This is useful when you want to switch between regions quickly or when doing a demo.

Prerequisites

- OptaWeb Vehicle Routing has been successfully built with Maven as described in Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files* .
- The OSM file for the region that you want to use has been downloaded. For information about downloading OSM files, see Section 3.2, "Run the OptaWeb Vehicle Routing runLocally.sh script in interactive mode".
- Internet access is available.

Procedure

1. Change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources**.
2. Execute the following command where **<OSM_FILE_NAME>** is an OSM file that you downloaded previously:

```
./runLocally.sh <OSM_FILE_NAME>
```

Next steps

Chapter 6, *Using OptaWeb Vehicle Routing*

3.4. RUN THE OPTAWEB VEHICLE ROUTING RUNLOCALLY.SH SCRIPT IN AIR DISTANCE MODE

OptaWeb Vehicle Routing can work in air distance mode that calculates travel times based on the distance between two coordinates. Use this mode in situations where you need to get OptaWeb Vehicle Routing up and running as quickly as possible and do not want to use an OSM (OpenStreetMap) file. Air distance mode is only useful if you need to smoke-test OptaWeb Vehicle Routing and you do not need accurate travel times.

Prerequisites

- OptaWeb Vehicle Routing has been successfully built with Maven as described in Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files* .
- Internet access is available.

Procedure

1. Change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources**.
2. Run the **runLocally.sh** script with the **--air** argument to start OptaWeb Vehicle Routing in air distance mode:

```
./runLocally.sh --air
```

Next steps

Chapter 6, *Using OptaWeb Vehicle Routing*

3.5. UPDATE THE DATA DIRECTORY

You can update the data directory that OptaWeb Vehicle Routing uses if you want to use a different data directory. The default data directory is **$HOME/.optaweb-vehicle-routing**.

Prerequisites

- OptaWeb Vehicle Routing has been successfully built with Maven as described in Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files* .

Procedure

- To use a different data directory, at its absolute path to the **.DATA_DIR_LAST** file in the current data directory.
- To change country codes associated with a region, edit the corresponding file in the **country_codes** directory, in the current data directory.

For example, if you downloaded an OSM file for Scotland and the script fails to guess the country code, set the content of **country_codes/scotland-latest** to GB.

- To remove a region, delete the corresponding OSM file from **openstreetmap** directory in the data directory and delete the region's directory in the **graphhopper** directory.

CHAPTER 4. CONFIGURE AND RUN OPTAWEB VEHICLE ROUTING MANUALLY

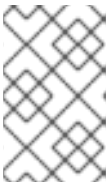
The easiest way to run OptaWeb Vehicle Routing is to use the **runlocally.sh** script. However, if Bash is not available on your system you can manually complete the steps that the **runlocally.sh** script performs.

Prerequisites

- OptaWeb Vehicle Routing has been successfully built with Maven as described in Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files* .
- Internet access is available.

Procedure

1. Download routing data.
   The routing engine requires geographical data to calculate the time it takes vehicles to travel between locations. You must download and store OpenStreetMap (OSM) data files on the local file system before you run OptaWeb Vehicle Routing.

   NOTE

   The OSM data files are typically between 100 MB to 1 GB and take time to download so it is a good idea to download the files before building or starting the OptaWeb Vehicle Routing application.

   a. Open **http://download.geofabrik.de/** in a web browser.

   b. Click a region in the **Sub Region** list, for example **Europe**. The subregion page opens.

   c. In the **Sub Regions** table, download the OSM file (**.osm.pbf**) for a country, for example Belgium.

2. Create the data directory structure.
   OptaWeb Vehicle Routing reads and writes several types of data on the file system. It reads OSM (OpenStreetMap) files from the **openstreetmap** directory, writes a road network graph to the **graphhopper** directory, and persists user data in a directory called **db**. Create a new directory dedicated to storing all of these data to make it easier to upgrade to a newer version of OptaWeb Vehicle Routing in the future and continue working with the data you created previously.

   a. Create the **$HOME/{VRP-DATA-DIR}** directory.

   b. Create the **openstreetmap** directory in the **$HOME/{VRP-DATA-DIR}** directory:

   ```
   $HOME/{VRP-DATA-DIR}
   └── openstreetmap
   ```

   c. Move all of your downloaded OSM files (files with the extension **.osm.pbf**) to the **openstreetmap** directory.
   The rest of the directory structure is created by the OptaWeb Vehicle Routing application when it runs for the first time. After that, your directory structure is similar to the following example:

```
$HOME/{VRP-DATA-DIR}

├── db
│   └── vrp.mv.db
├── graphhopper
│   └── belgium-latest
└── openstreetmap
    └── belgium-latest.osm.pbf
```

3. Change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources/optaweb-vehicle-routing-standalone/target**.

4. To run OptaWeb Vehicle Routing, enter the following command:

```
java -jar optaweb-vehicle-routing-standalone-7.39.0.Final-redhat-00005.jar \
--app.persistence.h2-dir=$HOME/{VRP-DATA-DIR}/db \
--app.routing.gh-dir=$HOME/{VRP-DATA-DIR}/graphhopper \
--app.routing.osm-dir=$HOME/{VRP-DATA-DIR}/openstreetmap \
--app.routing.osm-file=<OSM_FILE_NAME> \
--app.region.country-codes=<COUNTRY_CODE_LIST> \
```

   In this command, replace the following variables:

   - **<OSM_FILE_NAME>**: The OSM file for the region that you want to use and that you downloaded previously
   - **<COUNTRY_CODE_LIST>**: A comma-separated list of country codes used to filter geosearch queries. For a list of country codes, see *ISO 3166 Country Codes*.
     The application starts after the OSM file is downloaded.

     In the following example, OptaWeb Vehicle Routing downloads the OSM map of Central America (**central-america-latest.osm.pbf**) and searches in the countries Belize (BZ) and Guatemala (GT).

```
java -jar optaweb-vehicle-routing-standalone-7.39.0.Final-redhat-00005.jar \
--app.persistence.h2-dir=/home/user/.optaweb-vehicle-routing/db \
--app.routing.osm-dir=/home/user/.optaweb-vehicle-routing/openstreetmap \
--app.routing.gh-dir=/home/user/.optaweb-vehicle-routing/graphhopper \
--app.routing.osm-file=central-america-latest.osm.pbf \
--app.region.country-codes=BZ,GT
```

5. To open the OptaWeb Vehicle Routing user interface, enter the following URL in a web browser:

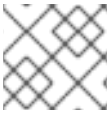
```
http://localhost:8080
```

Next steps

Chapter 6, *Using OptaWeb Vehicle Routing*

CHAPTER 5. RUN OPTAWEB VEHICLE ROUTING ON RED HAT OPENSHIFT CONTAINER PLATFORM

Linux users can use the **runOnOpenShift.sh** Bash script to install OptaWeb Vehicle Routing on Red Hat OpenShift Container Platform.



NOTE

The **runOnOpenShift.sh** script does not run on macOS.

Prerequisites

- You have access to an OpenShift cluster and the OpenShift command-line interface (**oc**) has been installed. For information about Red Hat OpenShift Container Platform, see *Installing OpenShift Container Platform*.
- OptaWeb Vehicle Routing has been successfully built with Maven as described in Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files* .
- Internet access is available.

Procedure

1. Log in to or start a Red Hat OpenShift Container Platform cluster.

a. Enter the following command where **<PROJECT_NAME>** is the name of your new project:

```
oc new-project <PROJECT_NAME>
```

b. If necessary, change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources**.

c. Enter the following command to execute the **runOnOpenShift.sh** script and download an OpenStreetMap (OSM) file:

```
./runOnOpenShift.sh <OSM_FILE_NAME> <COUNTRY_CODE_LIST>
<OSM_FILE_DOWNLOAD_URL>
```

In this command, replace the following variables:

- **<OSM_FILE_NAME>**: The name of a file downloaded from **<OSM_FILE_DOWNLOAD_URL>**.
- **<COUNTRY_CODE_LIST>**: A comma-separated list of country codes used to filter geosearch queries. For a list of country codes, see *ISO 3166 Country Codes*.
- **<OSM_FILE_DOWNLOAD_URL>**: The URL of an OSM data file in PBF format accessible from OpenShift. The file will be downloaded during backend startup and saved as **/deployments/local/<OSM_FILE_NAME>**.
  The following example configures OptaWeb Vehicle Routing to filter geosearch results to Belgium and downloads the latest Belgium OSM extract from Geofabrik:

In the following example, OptaWeb Vehicle Routing downloads the OSM map of Central America (**central-america-latest.osm.pbf**) and searches in the countries Belize (BZ) and Guatemala (GT).

```
./runOnOpenShift.sh central-america-latest.osm.pbf BZ,GT
http://download.geofabrik.de/europe/central-america-latest.osm.pbf
```

NOTE

For help with the **runOnOpenShift.sh** script, enter **./runOnOpenShift.sh --help**.

5.1. UPDATING THE DEPLOYED OPTAWEB VEHICLE ROUTING APPLICATION WITH LOCAL CHANGES

After you deploy your OptaWeb Vehicle Routing application on Red Hat OpenShift Container Platform, you can update the back end and front end.

Prerequisites

- OptaWeb Vehicle Routing has been successfully built with Maven and deployed on OpenShift.

Procedure

- To update the back end, perform the following steps:
  1. Change the source code and build the back end module with Maven.
  2. Change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources/optaweb-vehicle-routing-backend**.
  3. Enter the following command to start the OpenShift build:

     ```
     oc start-build backend --from-dir=. --follow
     ```

- To update the front end, perform the following steps:
  1. Change the source code and build the front end module with the **npm** utility.
  2. Change directory to **sources/optaweb-vehicle-routing-frontend**.
  3. Enter the following command to start the OpenShift build:

     ```
     oc start-build frontend --from-dir=docker --follow
     ```

Next steps

Chapter 6, *Using OptaWeb Vehicle Routing*

CHAPTER 6. USING OPTAWEB VEHICLE ROUTING

In the OptaWeb Vehicle Routing application, you can mark a number of locations on the map. The first location is assumed to be the depot. Vehicles must deliver goods from this depot to every other location that you marked.

You can set the number of vehicles and the carrying capacity of every vehicle. However, the route is not guaranteed to use all vehicles. The application uses as many vehicles as required for an optimal route.

The current version has certain limitations:

- Every delivery to a location is supposed to take one point of vehicle capacity. For example, a vehicle with a capacity of 10 can visit up to 10 locations before returning to the depot.
- Setting custom names of vehicles and locations is not supported.

6.1. CREATING A ROUTE

To create an optimal route, use the **Demo** tab of the OptaWeb Vehicle Routing user interface.

Prerequisites

- OptaWeb Vehicle Routing is running and you have access to the user interface.

Procedure

1. In OptaWeb Vehicle Routing, click **Demo** to open the **Demo** tab.
2. Use the blue minus and plus buttons above the map to set the number of vehicles. Each vehicle has a default capacity of 10.
3. Use the plus button in a square on the map to zoom in as required.

NOTE

Do not double-click to zoom in. A double click also creates a location.

4. Click a location for the depot.
5. Click other locations on the map for delivery points.
6. If you want to delete a location:
   a. Hover the mouse cursor over the location to see the location name.
   b. Find the location name in the list in the left part of the screen.
   c. Click the **X** icon next to the name.

Every time you add or remove a location or change the number of vehicles, the application creates and displays a new optimal route. If the solution uses several vehicles, the application shows the route for every vehicle in a different color.

6.2. VIEWING AND SETTING OTHER DETAILS

You can use other tabs in OptaWeb Vehicle Routing user interface to view and set additional details.

Prerequisites

- OptaWeb Vehicle Routing is running and you have access to the user interface.

Procedure

- Click the **Vehicles** tab to view, add, and remove vehicles, and also set the capacity for every vehicle.
- Click the **Visits** tab to view and remove locations.
- Click the **Route** tab to select each vehicle and view the route for the selected vehicle.

6.3. CREATING CUSTOM DATA SETS WITH OPTAWEB VEHICLE ROUTING

There is a built-in demo data set consisting of a several large Belgian cities. If you want to have more demos available in the **Load demo** menu, you can prepare your own data sets.

Procedure

To do that, follow these steps:

1. In OptaWeb Vehicle Routing, add a depot and a number of visits by clicking on the map or using geosearch.
2. Click **Export** and save the file in the data set_directory.

NOTE

The data set directory is the directory specified in the **app.demo.data-set-dir** property.

If the application is running through the **runLocally.sh** script, the data set directory is set to **$HOME/{VRP-DATA-DIR}/dataset**.

Otherwise, the property is from **application.properties** and defaults to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00001/sources/optaweb-vehicle-routing-standalone/target/local/dataset**.

You can edit the **app.demo.data-set-dir** property to specify a diffent data directory.

3. Edit the YAML file and choose a unique name for the data set.
4. Restart the back end.

After you restart the back end, files in the data set directory appear in the **Load demo** menu.

6.4. TROUBLESHOOTING OPTAWEB VEHICLE ROUTING

If the OptaWeb Vehicle Routing behaves unexpectedly, follow this procedure to trouble-shoot.

Prerequisites

- OptaWeb Vehicle Routing is running and behaving unexpectedly.

Procedure

1. To identify issues, review the back end terminal output log.
2. To resolve issues, remove the back end database:
    a. Stop the back end by pressing kbd:[Ctrl+C] in the back end terminal window.
    b. Remove the directory **optaweb-vehicle-routing/optaweb-vehicle-routing-backend/local/db**.
    c. Restart OptaWeb Vehicle Routing.

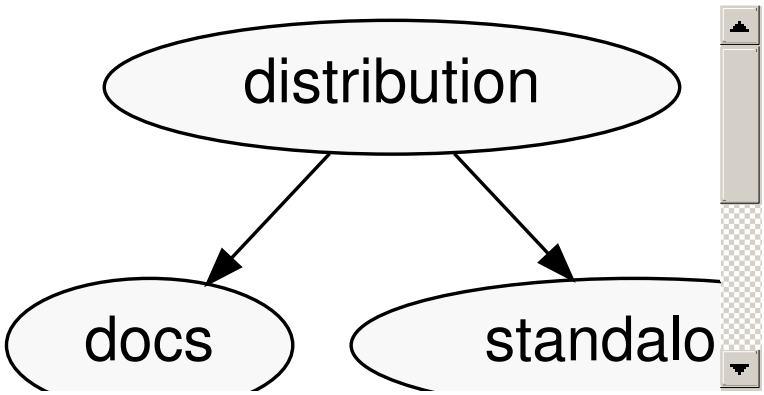
CHAPTER 7. OPTAWEB VEHICLE ROUTING DEVELOPMENT GUIDE

This section describes how to configure and run the back and front end modules in development mode.

7.1. OPTAWEB VEHICLE ROUTING PROJECT STRUCTURE

The OptaWeb Vehicle Routing project is a multi-module Maven project.

Figure 7.1. Module dependency tree diagram



The back end and front end modules are at the bottom of the module tree. These modules contain the application source code.

The standalone module is an assembly module that combines the back end and front end into a single executable JAR file.

The distribution module represents the final assembly step. It takes the standalone application and the documentation and wraps them in an archive that is easy to distribute.

The back end and front end are separate projects that you can build and deploy separately. In fact, they are written in completely different languages and built with different tools. Both projects have tools that provide a modern developer experience with fast turn-around between code changes and the running application.

The next sections describe how to run both back end and front end projects in development mode.

7.2. THE OPTAWEB VEHICLE ROUTING BACK END MODULE

The back end module contains a server-side application that uses Red Hat Business Optimizer to optimize vehicle routes. Optimization is a CPU-intensive computation that must avoid any I/O operations in order to perform to its full potential. Because one of the chief objectives is to minimize travel cost, either time or distance, OptaWeb Vehicle Routing keeps the travel cost information in RAM memory. While solving, Red Hat Business Optimizer needs to know the travel cost between every pair of locations entered by the user. This information is stored in a structure called the *distance matrix*.

When you enter a new location, OptaWeb Vehicle Routing calculates the travel cost between the new location and every other location that has been entered so far, and stores the travel cost in the distance matrix. The travel cost calculation is performed by the GraphHopper routing engine.

The back end module implements the following additional supporting functionality:

- Persistence

- WebSocket connection for the front end
- Data set loading, export, and import

To learn more about the back end code architecture, see Appendix A, *Back end architecture*.

The next sections describe how to configure and run the back end in development mode.

7.2.1. Running the OptaWeb Vehicle Routing back end module using the Spring Boot Maven plugin

You can use the Spring Boot plug-in to run the OptaWeb Vehicle Routing back end module in development mode.

Prerequisites

- OptaWeb Vehicle Routing has been configured as described in Chapter 4, *Configure and run OptaWeb Vehicle Routing manually*.

Procedure

1. Change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources/optaweb-vehicle-routing-backend**.
2. To run the back end in development mode, enter the following command:

```
mvn spring-boot:run
```

7.2.2. Running the OptaWeb Vehicle Routing back end module from IntelliJ IDEA

You can use the IntelliJ IDEA to run the OptaWeb Vehicle Routing back end module to make it easier to develop your project.

Procedure

1. In IntelliJ IDEA, enter **org.optaweb.vehiclerouting.OptaWebVehicleRoutingApplication**. This creates a run configuration that you will edit in the next step.
   a. Open the **OptaWebVehicleRoutingApplication** class in the **Editor** window.
   b. Click the green symbol in the editor window gutter and select **Run OptaWebVehicleRoutingApplication**. The run fails because the working directory is set to the root of the project where the back end module directory is expected.

NOTE

See the *Run Applications* page on the IntelliJ IDEA web site to learn more about running applications in IntelliJ IDEA.

2. Select **Run→Edit Configurations** and then select **Spring Boot→OptaWebVehicleRoutingApplication**.
3. Set **Program arguments** to **--spring.profiles.active=local** to activate the Spring profile called **local**. This directs the application to use configurations in the **application-local.properties** file.

4. Change **Working directory** to the back end module (**optaweb-vehicle-routing-backend**).

5. Set **On Update action** to **Hot swap classes and update trigger file if failed** This enables you to use the **Update** action to quickly restart the application.
For more information, see *Spring and Spring Boot in IntelliJ IDEA 2018.1*.

7.2.3. Spring Boot automatic restart

Automatic restart is provided by Spring Boot DevTools. When you run the OptaWeb Vehicle Routing back end with the Spring Boot Maven plug-in, the application automatically restarts whenever files on the classpath change. Automatic restart scans files on the classpath, so you only need to recompile your changes to trigger application restart. No IDE configuration is needed.

If your IDE has a compile-on-save feature (for example Eclipse or NetBeans), you just need to save the files that have changed since the last compilation.

IntelliJ IDEA saves changes automatically and you need to select either **Build[Recompile]**, which recompiles the file in the active tab, or **Build[Build Project]** which recompiles all changes. For more information, see Compile and build applications with IntelliJ IDEA .

7.2.4. Setting OptaWeb Vehicle Routing back end module configuration properties

There are several ways that you can set OptaWeb Vehicle Routing back end module configuration properties. The methods in this section are useful if you are running OptaWeb Vehicle Routing locally.

Prerequisites

- The OptaWeb Vehicle Routing reference implementation has been downloaded and extracted. For information, see Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files*.

Procedure

1. Set configuration properties in the **application.properties** file:

2. Change directory to **rhpam-7.39.0.Final-redhat-00005-optaweb-vehicle-routing/sources/optaweb-vehicle-routing-backend/src/main/resources**.

3. Open the **application.properties** file in a text editor.

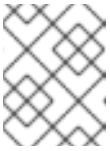
4. Edit or add properties and then save the file.

   - Use a command line argument when running the packaged application. In the following example, **<PROPERTY>** is the name of a property and **<VALUE>** is the value of that property:

     ```
     java -jar optaweb-vehicle-routing-backend.jar --app.<PROPERTY>=<VALUE>
     ```

   - Use an environment variable when running the application with **spring-boot:run**:

     ```
     <PROPERTY>=<VALUE> ./mvnw spring-boot:run
     ```



NOTE

This method requires relaxed binding which only works if the property is defined using **@ConfigurationProperties**.

It is not possible to set properties by specifying **-D** when running the application using the Spring Boot Maven plugin (**./mvnw spring-boot:run -D<PROPERTY**). Any system properties to be set by the plugin to the forked Java process in which the application runs must be specified in the **pom.xml** file using the **systemPropertiesVariables** attribute. For information about this attribute, see *Using System Properties* on the Spring web site.

You can learn more about configuring a Spring Boot application on the Spring Boot Externalized Configuration page.

TIP

Use **src/main/resources/application-local.properties** to store your personal configuration without affecting the Git working tree.

For a complete list of OptaWeb Vehicle Routing configuration properties, see Appendix B, *OptaWeb Vehicle Routing back end configuration properties*.

For a complete list of application properties available in Spring Boot, see the *Common Application Properties* page on the Spring web site.

7.2.5. OptaWeb Vehicle Routing backend logging

OptaWeb Vehicle Routing uses the SLF4J API and Logback as the logging framework. The Spring environment enables you to configure most logging aspects, including levels, patterns, and log files, in the same way as other configuration properties. The most common ways to set logging properties are by editing the **application.properties** file or using arguments such as **<PROPERTY>=<VALUE>** where **<PROPERTY>** is the name of a property and **<VALUE>** is the value of that property. See the Spring Boot Logging documentation for more information.

The following examples are properties that you can use to control logging level of some parts of the application:

- **logging.level.org.optaweb.vehiclerouting=debug**: Enables the debug level for the back end code
- **logging.level.org.optaplanner.core=warn**: Reduces Red Hat Business Optimizer logging
- **logging.level.org.springframework.web.socket=trace**: Accesses more details when investigating problems with WebSocket connection

7.3. WORKING WITH THE OPTAWEB VEHICLE ROUTING FRONT END MODULE

The front end project was bootstrapped with Create React App. Create React App provides a number of scripts and dependencies that help with development and with building the application for production.

Prerequisites

- The OptaWeb Vehicle Routing reference implementation has been downloaded and extracted. For information, see Chapter 2, *Download and build the OptaWeb Vehicle Routing deployment files*.

Procedure

1. On Fedora, enter the following command to set up the development environment:

```
sudo dnf install npm
```

   See Downloading and installing Node.js and npm for more information about installing npm.

2. Change directory to **optaweb-vehicle-routing-distribution-7.39.0.Final-redhat-00005/sources/optaweb-vehicle-routing-frontend**.

3. Install **npm** dependencies:

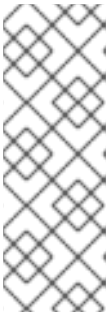
```
npm install
```

   Unlike Maven, the **npm** package manager installs dependencies in **node_modules** under the project directory and does that only when you execute **npm install**. Whenever the dependencies listed in **package.json** change, for example when you pull changes to the master branch, you must execute **npm install** before you run the development server.

4. Enter the following command to run the development server:

```
npm start
```

5. If it does not open automatically, open **http://localhost:3000/** in a web browser.
   By default, the **npm start** command attempts to open this URL in your default browser.



NOTE

If you do not want the **npm start** command to open a new browser tab each time you run it, export the **BROWSER=none** environment variable. You can use **.env.local** file to make this preference permanent. To do that, enter the following command:

```
echo BROWSER=none >> .env.local
```

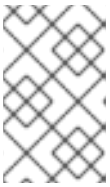
   The browser refreshes the page whenever you make changes in the front end source code. The development server process running in the terminal picks up the changes as well and prints compilation and lint errors to the console.

6. Enter the following command to run tests:

```
npm test
```

7. Change the value of the **REACT_APP_BACKEND_URL** environment variable to specify the location of the back end project to be used by **npm** when you execute **npm start** or **npm run build**, for example:

```
REACT_APP_BACKEND_URL=http://10.0.0.123:8081
```



NOTE

Environment variables are hard coded inside the JavaScript bundle during the **npm** build process, so you must specify the back end location before you build and deploy the front end.

To learn more about the React environment variables, see *Adding Custom Environment Variables*.

8. To build the front end, enter one of the following commands:

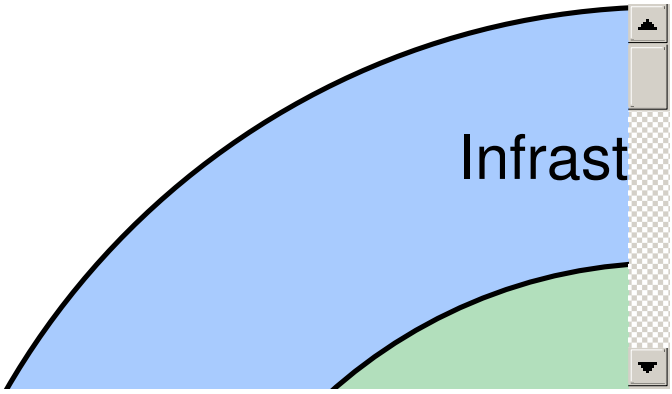
```
./mvnw install
```

```
mvn install
```

APPENDIX A. BACK END ARCHITECTURE

Domain model and use cases are essential for the application. The OptaWeb Vehicle Routing domain model is at the center of the architecture and is surround by the application layer that embeds use cases. Functions such as route optimization, distance calculation, persistence, and network communication are considered implementation details and are placed at the outermost layer of the architecture.

Figure A.1. Diagram of application layers



A.1. CODE ORGANIZATION

The back end code is organized in three layers, illustrated in the preceding graphic.

```
org.optaweb.vehiclerouting
├── domain
├── plugin          # Infrastructure layer
│   ├── persistence
│   ├── planner
│   ├── routing
│   └── websocket
└── service         # Application layer
    ├── demo
    ├── distance
    ├── location
    ├── region
    ├── reload
    ├── route
    └── vehicle
```

The **service** package contains the application layer that implements use cases. The **plugin** package contains the infrastructure layer.

Code in each layer is further organized by function. This means that each service or plug-in has its own package.

A.2. DEPENDENCY RULES

Compile-time dependencies are only allowed to point from outer layers towards the center. Following this rule helps to keep the domain model independent of underlying frameworks and other implementation details and models the behavior of business entities more precisely. With presentation and persistence being pushed out to the periphery, it is easier to test the behavior of business entities and use cases.

The domain has no dependencies.

Services only depend on the domain. If a service needs to send a result (for example to the database or to the client), it uses an output boundary interface. Its implementation is injected by the Inversion of Control (IoC) container.

Plug-ins depend on services in two ways. First, they invoke services based on events such as a user input or a route update coming from the optimization engine. Services are injected into plug-ins which moves the burden of their construction and dependency resolution to the IoC container. Second, plug-ins implement service output boundary interfaces to handle use case results, for example persisting changes to the database or sending a response to the web UI.

A.3. THE DOMAIN PACKAGE

The **domain** package contains *business objects* that model the domain of this project, for example **Location**, **Vehicle**, **Route**. These objects are strictly business-oriented and must not be influenced by any tools and frameworks, for example object-relational mapping tools and web service frameworks.

A.4. THE SERVICE PACKAGE

The **service** package contains classes that implement *use cases*. A use case describes something that you want to do, for example adding a new location, changing vehicle capacity, or finding coordinates for an address. The business rules that govern use cases are expressed using the domain objects.

Services often need to interact with plug-ins in the outer layer, such as persistence, web, and optimization. To satisfy the dependency rules between layers, the interaction between services and plug-ins is expressed in terms of interfaces that define the dependencies of a service. A plug-in can satisfy a dependency of a service by providing a bean that implements the boundary interface of the service. The Spring IoC container creates an instance of the plug-in bean and injects it to the service at runtime. This is an example of the inversion of control principle.

A.5. THE PLUGIN PACKAGE

The **plugin** package contains infrastructure functions such as optimization, persistence, routing, and network.

APPENDIX B. OPTAWEB VEHICLE ROUTING BACK END CONFIGURATION PROPERTIES

You can set the OptaWeb Vehicle Routing application properties listed in the following table.

Property	Type	Example	Description
app.demo.data-set-dir	Relative or absolute path	**/home/user/{VRP-DATA-DIR}/dataset**	Custom data sets are loaded from this directory. Defaults to **local/dataset**.
app.persistence.h2-dir	Relative or absolute path	**/home/user/{VRP-DATA-DIR}/db**	The directory used by H2 to store the database file. Defaults to **local/db**.
app.region.country-codes	List of **ISO 3166-1 alpha-2** country codes	**US**, **GB,IE**, **DE,AT,CH**, may be empty	Restricts geosearch results.
app.routing.engine	Enumeration	**air**, **graphhopper**	Routing engine implementation. Defaults to **graphhopper**.
app.routing.gh-dir	Relative or absolute path	**/home/user/{VRP-DATA-DIR}/graphhopper**	The directory used by GraphHopper to store road network graphs. Defaults to **local/graphhopper**.
app.routing.osm-dir	Relative or absolute path	**/home/user/{VRP-DATA-DIR}/openstreetmap**	The directory that contains OSM files. Defaults to **local/openstreetmap**.
app.routing.osm-file	File name	**belgium-latest.osm.pbf**	Name of the OSM file to be loaded by GraphHopper. The file must be placed under **app.routing.osm-dir**.
optaplanner.solver.termination.spent-limit	**java.time.Duration**	• **1m** • **150s** • **P2dT21h** (**PnDTnHnMn.nS**)	How long the solver should run after a location change occurs.

Property	Type	Example	Description
server.address	IP address or hostname	**10.0.0.123**, **my-vrp.geo-1.openshiftapps.com**	Network address to which to bind the server.
server.port	Port number	**4000**, **8081**	Server HTTP port.

APPENDIX C. VERSIONING INFORMATION

Documentation last updated on Thursday, September 08, 2020.