OptaWeb Employee Rostering User Guide

The OptaPlanner Team

Version 7.30.0-SNAPSHOT

Table of Contents

Chapter 1. OptaWeb Employee Rostering Introduction

1.1. What is OptaWeb Employee Rostering?

Every organization faces planning problems: providing products or services with a limited set of *constrained* resources (employees, assets, time and money). One such planning problem is employee shift rostering: assigning shifts to employees. OptaWeb is a web application and REST service that solves employee shift rostering problems using the OptaPlanner engine.

1.2. Build and Run the Application

To build the project with Maven, run the following command in the project's root directory:

```
mvn clean install -DskipTests
```

After building the project, run the application with:

```
java -jar employee-rostering-distribution/target/employee-rostering-distribution-*.jar
```

Then open http://localhost:8080/ to see the web application.

Alternatively, run `npm start` in the `employee-rostering-frontend` directory to start the frontend in one terminal, and run `mvn spring-boot:run` in the `employee-rostering-backend` directory to start the backend in another terminal.

To run on another port, use `--server.port=⋯`:

```
java -jar employee-rostering-distribution/target/employee-rostering-distribution-*.jar
--server.port=18080
```

1.3. System Properties

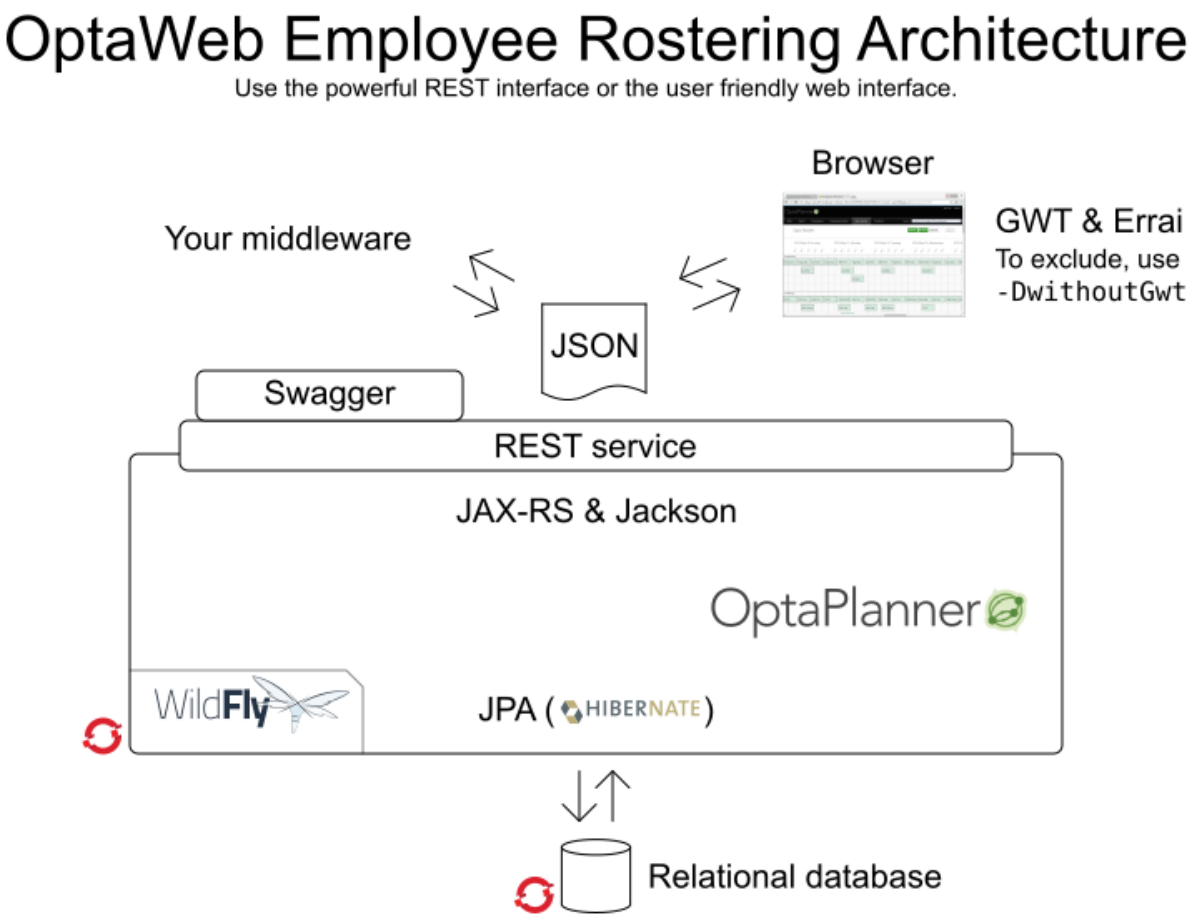
These system properties can overwrite default properties of the application, for example, by passing `-Doptaweb.generator.zoneId="America/New_York"` to Spring Boot. These system properties might also be exposed as OpenShift template parameters.

- **optaweb.generator.timeZoneId**: The time zone ID for the automatically generated tenants. For example `America/New_York`. This defaults to the system default Zone ID.
- **optaweb.generator.initial.data**: What data to initially put in the database. Supported values are: `EMPTY` (no data) and `DEMO_DATA` (several tenants of various sizes). This defaults to `DEMO_DATA`

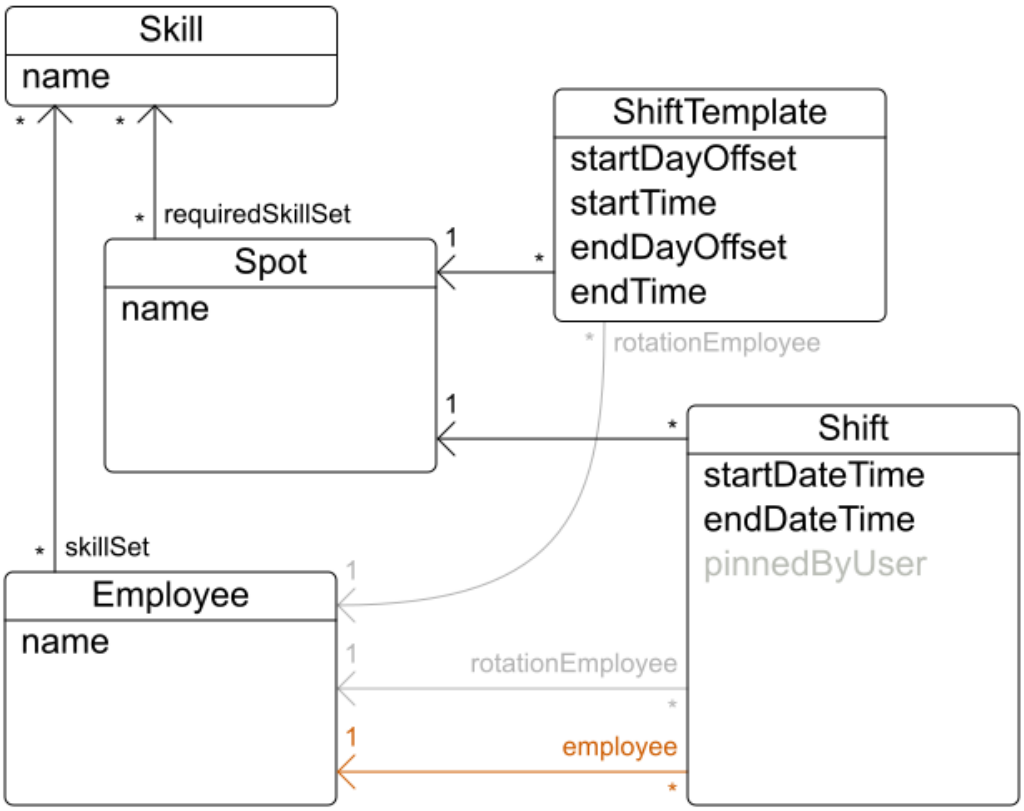
The OpenShift docker image also supports these parameters:

- **org.optaweb.employeerostering.persistence.datasource**
- **org.optaweb.employeerostering.persistence.dialect**
- **org.optaweb.employeerostering.persistence.hbm2ddl.auto**
- **org.optaweb.employeerostering.persistence.id.generator**

Chapter 2. Architecture

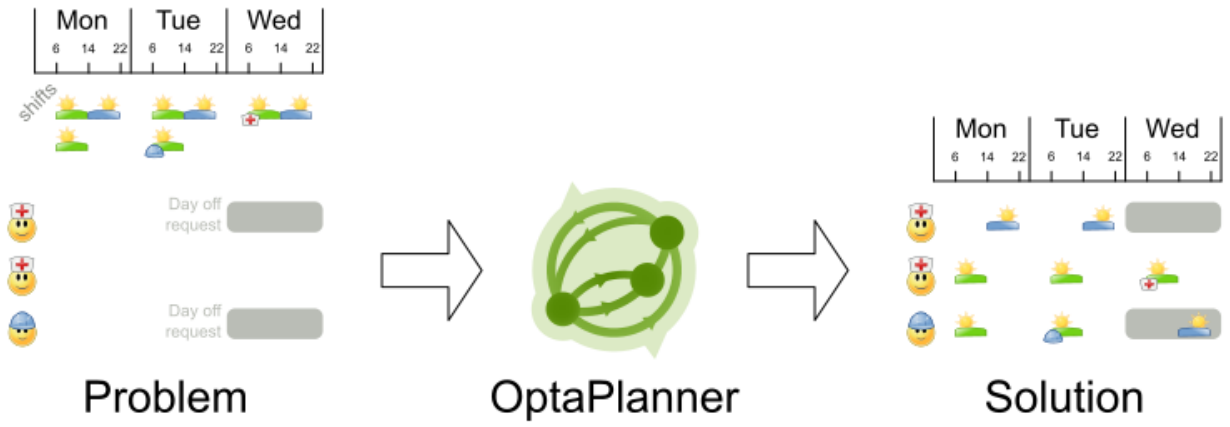


OptaWeb Employee Rostering class diagram



Solving with OptaPlanner

OptaPlanner automatically assigns the shifts, according to our constraints.



Chapter 3. Features in OptaWeb Employee Rostering

3.1. Test the JPA Database with H2

Before testing the database, make sure the application backend is running. If the application isn't running, run the following in the `employee-rostering-backend` directory:

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

Go to http://localhost:8080/h2-console to view the H2 database console. Enter `org.h2.Driver` in the `Driver Class` field and `jdbc:h2:mem:testdb` in the `JDBC URL` field, and keep the other default values. Connect, and click on the entities on the left to run SQL statements. This console allows you to view and modify the application database.

3.2. Test the REST API

As with testing the database, make sure the application backend is running to test the REST API. Go to http://localhost:8080/swagger-ui.html to view documentation and test the REST methods.