



Red Hat Decision Manager 7.2

IDE を使用した Red Hat Business Optimizer 向け従業員勤務表スターターアプリケーションの実行および変更

ガイド

Red Hat Decision Manager 7.2 IDE を使用した Red Hat Business Optimizer 向け従業員勤務表スターターアプリケーションの実行および変更

ガイド

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2023 | You need to change the HOLDER entity in the en-US/Running_and_modifying_the_employee_roster_starter_application_for_Red_Hat_Business_Optir file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書は、Red Hat Decision Manager 7.2 にアドオンとして含まれている OptaShift Employee Rostering テンプレートを実行し、変更する方法を説明します。

目次

はじめに	3
第1章 従業員勤務表スターターアプリケーションの概要	4
第2章 従業員勤務表スターターアプリケーションの構築と実行	5
2.1. デプロイメントファイルの準備	5
2.2. コマンドラインからの従業員勤務表スターターアプリケーションの構築と実行	5
2.3. コマンドラインからの永続データストレージを使用した従業員勤務表スターターアプリケーションの構築と実行	6
2.4. ECLIPSE を使用した従業員勤務表スターターアプリケーションの構築と実行	7
第3章 従業員勤務表スターターアプリケーションのソースコードに関する概要	11
第4章 従業員勤務表スターターアプリケーションの変更	13
付録A バージョン情報	14

はじめに

ビジネスルールの作成者は、IDE を使用して Red Hat Business Optimizer 機能を使用する **employee-rostering** スターターアプリケーションのビルド、実行、変更が可能です。

前提条件

- Eclipse (Red Hat Developer Studio を含む) または IntelliJ IDEA などの統合開発環境
- Java 言語の理解

第1章 従業員勤務表スターターアプリケーションの概要

従業員勤務表スターターアプリケーションは、組織内のさまざまな場所に従業員を割り当てます。たとえば、アプリケーションを使用して、病院での看護師のシフト、さまざまな場所での警備勤務シフト、作業者の組み立てラインのシフトを割り当てます。

従業員勤務表を最適化するには、多くの変数を考慮する必要があります。たとえば、業務が異なれば、求められるスキルが異なります。また、従業員の中には、特定の時間帯に勤務できない場合や、特定の時間帯での勤務を希望する場合があります。

このスターターアプリケーションの Red Hat Business Optimizer ルールは、ハード制約およびソフト制約を使用します。最適化時に、従業員が勤務できない (または病欠の) 場合や、ある1つのシフト内の2つのスポットで働くことができない場合など、Red Hat Business Optimizer エンジンがハード制約に違反することができません。Red Hat Business Optimizer エンジンが、ソフト制約 (特定のシフトで勤務しないという従業員の希望など) を順守しようとしませんが、最適なソリューションには違反が必要だと判断した場合は、違反することができます。

第2章 従業員勤務表スターターアプリケーションの構築と実行

ソースコードから従業員勤務表スターターアプリケーションを構築して、JBoss EAP または WildFly アプリケーションサーバーを使用して実行します。

コマンドラインを使用してアプリケーションを構築してから、スタンドアロンサーバーにインストールします。

または、Eclipse (Red Hat JBoss Developer Studio を含む) などの IDE を使用して、アプリケーションを構築し、実行します。

このアプリケーションの使用に関する情報は、[Red Hat OpenShift Container Platform](#) での [Red Hat Business Optimizer 向け従業員勤務表スターターアプリケーションのデプロイと使用](#) を参照してください。

2.1. デプロイメントファイルの準備

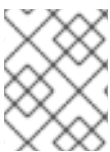
デプロイメントファイルをダウンロードし、準備してから、アプリケーションの構築、デプロイを行う必要があります。

手順

1. Red Hat Decision Manager 7.2 の [Software Downloads](#) ページから **rhdm-7.2.0-add-ons.zip** ファイルをダウンロードします。
2. ダウンロードしたアーカイブを展開します。
3. アドオンアーカイブから展開した従業員勤務表の zip ファイル (**rhdm-7.2.0-employee-rostering.zip**) を展開します。

結果

従業員勤務表の zip ファイルを展開すると、**optashift-employee-rostering-7.14.0.Final-redhat-00002** フォルダが作成されます。このフォルダは、後述する手順でベースとなるフォルダです。



注記

ファイルおよびディレクトリーの名前で使用されるバージョン番号は、本書で使用するバージョンよりも新しい場合があります。

2.2. コマンドラインからの従業員勤務表スターターアプリケーションの構築と実行

コマンドラインを使用して、従業員勤務表スターターアプリケーションを構築し、実行することができます。

この手順を使用する場合は、データはメモリーに保存されるため、サーバーが停止するとデータが失われます。データベースサーバーを使用してアプリケーションを構築し、実行して永続的に保存する方法は、「[コマンドラインからの永続データストレージを使用した従業員勤務表スターターアプリケーションの構築と実行](#)」を参照してください。

前提条件

1. Java 開発キットをインストールしておく必要があります。

2. Maven をインストールしておく必要があります。
3. ホストにはインターネットアクセスが必要です (外部のリポジトリから Maven パッケージをダウンロードするため)。

手順

1. ターミナルウィンドウで、**sources** ディレクトリーに移動します。

2. 次のコマンドを実行します。

```
mvn clean install -DskipTests
```

3. ビルドプロセスが完了するまで待ちます。
4. 以下のいずれかの方法を使用して、アプリケーションを実行します。
 - a. ビルドプロセスの一部としてデプロイされた WildFly サーバーを実行します。
 - i. **local/appserver/wildfly-14.0.1-final/standalone/deployments** サブディレクトリーで、**optashift-employee-rostering-webapp-<version>.war.dodeploy** ファイルを作成します。<version> は、同じディレクトリーにある既存の **optashift-employee-rostering-webapp-<version>.war** シンボリックリンクと同じでなければなりません。
 - ii. **local/appserver/wildfly-14.0.1-final/bin** サブディレクトリーで **./standalone.sh** コマンドを実行します。
 - b. 既存の WildFly または JBoss EAP サーバーに **optashift-employee-rostering-webapp/target/optaweb-employee-rostering-*.war** ファイルをデプロイして、アプリケーションサーバーを起動します。
 - c. 以下のコマンドで、Maven を使用してサーバーを実行します。

```
mvn -N wildfly:start wildfly:deploy
mvn gwt:codeserver
```



注記

Maven を使用してサーバーを起動した場合には、UI は gwt コードサーバーを使用して gwtui ソースを監視します。このような場合には、gwtui コードを変更すると、コードサーバーで自動的に変更を検出します。gwtui コードの変更には、アプリケーションの再構築は必要ありません。

後ほど、このサーバーを停止するには、以下のコマンドを使用します。

```
mvn -N wildfly:shutdown
```

5. <http://localhost:8080/gwtui/gwtui.html> でアプリケーションにアクセスします。

2.3. コマンドラインからの永続データストレージを使用した従業員勤務表スターターアプリケーションの構築と実行

コマンドラインで従業員勤務表スターターアプリケーションを構築し、実行する場合には、データベースサーバーを指定して、永続的にデータを保存することができます。

前提条件

1. Java 開発キットをインストールしておく必要があります。
2. Maven をインストールしておく必要があります。
3. ホストにはインターネットアクセスが必要です (外部のリポジトリから Maven パッケージをダウンロードするため)。
4. デプロイした WildFly または Red Hat JBoss EAP アプリケーションサーバーと、MySQL または PostgreSQL データベースサーバーが必要です。
5. データベースサーバー用に、アプリケーションサーバーで JDBC データソースを設定しておく必要があります。

手順

1. ターミナルウィンドウで、**sources** ディレクトリーに移動します。
2. 次のコマンドを実行します。

```
mvn clean install -DproductizedOpenshift -  
Dorg.optaweb.employee rostering.persistence.datasource=<dsname> -  
Dorg.optaweb.employee rostering.persistence.dialect=<dialect>
```

上記のコマンドでは、以下の値に置き換えてください。

- **<dsname>** は、アプリケーションサーバーのデータソース名に置き換えます。
 - **<dialect>** は、データベースサーバーの種類によって以下の文字列のいずれかに置き換えます。
 - MySQL の場合は **org.hibernate.dialect.MySQL5Dialect** を使用します。
 - PostgreSQL の場合は **org.hibernate.dialect.PostgreSQLDialect** を使用します。
3. ビルドプロセスが完了するまで待ちます。
 4. **optashift-employee-rostering-webapp/target/optaweb-employee-rostering-7.11.1-SNAPSHOT.war** ディレクトリーをアプリケーションサーバーにデプロイして、アプリケーションサーバーを起動します。
 5. <http://localhost:8080/gwtui/gwtui.html> でアプリケーションにアクセスします。

2.4. ECLIPSE を使用した従業員勤務表スターターアプリケーションの構築と実行

Red Hat JBoss Development Studio を含む Eclipse を使用して、従業員勤務表スターターアプリケーションを構築して実行できます。

前提条件

1. Eclipse をインストールしておく必要があります。
2. ホストにはインターネットアクセスが必要です (外部のリポジトリから Maven パッケージをダウンロードするため)。

3. 推奨される設定でアプリケーションを実行するには、Google Chrome をインストールする必要があります。ただし、別の Web ブラウザーを使用するように設定を変更できます。

手順

1. Eclipse を起動します。
2. メインメニューから **File > Import...** を選択します。
3. **Maven > Existing Maven projects** ウィザードを選択します。
4. root ディレクトリーでは、アプリケーションソースの root ディレクトリーを選択します。
5. **Finish** をクリックします。
6. オプションで、Eclipse で多くのエラーが表示されるのを回避するには、以下を行います。
 - a. アプリケーションソースの root ディレクトリーで、**mvn clean install** コマンドを実行して、ビルドが完了するのを待ちます。
 - b. Eclipse ナビゲーションツリーで、**employee-rostering-shared** を右クリックして、**Build Path > Configure Build Path...** を選択します。
 - c. **Source** タブをクリックしてから、**Add Folder...** をクリックします。
 - d. **employee-rostering-shared/target/generated-sources** フォルダーをクリックして、**OK** をクリックします。
7. メインメニューから **Run > External Tools > External Tools Configurations...** を選択します。
8. **Program** で以下の起動設定を作成します。
 - a. Chrome で OptaWeb 従業員勤務表を開く設定:
 - 名前: **Open OptaWeb Employee Rostering in Chrome**
 - 場所: **/usr/bin/google-chrome**
 - 作業ディレクトリー: **\${workspace_loc:/employee-rostering}**
 - 引数: **--incognito http://localhost:8080/gwtui/gwtui.html**



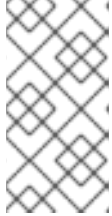
注記

Chrome の代わりに別のブラウザを使用する場合には、名前、場所、引数を変更できます。

- b. コードサーバーを終了する設定:
 - 名前: **Kill Code Server**
 - 場所: **/usr/sbin/fuser**
 - 作業ディレクトリー: **\${workspace_loc:/employee-rostering}**
 - 引数: **fuser -k 9876/tcp**

9. メインメニューから **Run > Run Configurations...** を選択します。
10. **Maven Build** で以下の起動設定を作成します。
 - a. OptaWeb 従業員勤務表のビルド:
 - 名前: **OptaWeb Employee Rostering Build**
 - ベースのディレクトリー: **`${workspace_loc}/employee-rostering`**
 - 目標: **clean install**
 - パラメーター: **gwt:skipCompilation 値: true**
 - b. OptaWeb 従業員勤務表の起動コードサーバー:
 - 名前: **OptaWeb Employee Rostering Start Code Server**
 - ベースのディレクトリー: **`${workspace_loc}/employee-rostering`**
 - 目標: **gwt:codeserver**
 - パラメーター: **gwt:skipCompilation 値: true**
 - c. OptaWeb 従業員勤務表の起動の Webserver:
 - 名前: **OptaWeb Employee Rostering Start Webserver**
 - ベースのディレクトリー: **`${workspace_loc}/employee-rostering`**
 - 目標: **wildfly:start wildfly:deploy**
 - パラメーター: **gwt:skipCompilation 値: true**
 - d. OptaWeb Employee Rostering の停止の Webserver:
 - 名前: **OptaWeb Employee Rostering Stop Webserver**
 - ベースのディレクトリー: **`${workspace_loc}/employee-rostering`**
 - 目標: **wildfly:shutdown**
 - パラメーター: **gwt:skipCompilation 値: true**
11. **Launch Group** の下に、**OptaWeb Employee Rostering Run** という名前の起動グループを作成します。それに次のローンチを追加します。
 - **Program::Kill Code Server** 起動モード: **継承** 起動後のアクション: **中断されるまで待機**
 - **Maven Build::OptaWeb Employee Rostering Stop Webserver** 起動モード: **継承** 起動後のアクション: **中断されるまで待機**
 - **Maven Build::OptaWeb Employee Rostering Build** 起動モード: **継承** 起動後のアクション: **中断されるまで待機**
 - **Maven Build::OptaWeb Employee Rostering Start Webserver** 起動モード: **継承** 起動後のアクション: **なし**

- **Maven Build::OptaWeb Employee Rostering Start Code server** 起動モード: **継承** 起動後のアクション: コンソールの出力を待つ (regex): **The code server is ready at**
 - **Program::Open OptaWeb Employee Rostering in Chrome** 起動モード: **継承** 起動後のアクション: なし
12. アプリケーションをビルドして実行し、すぐにアクセスするには、**OptaWeb Employee Rostering Run** 起動グループを実行します。次にアプリケーションを変更して、起動グループに戻り、変更をテストします。



注記

この方法を使用してアプリケーションを起動した場合には、UI は gwt コードサーバーを使用して gwtui ソースを監視します。gwtui コードを変更すると、コードサーバーで自動的に変更を検出します。gwtui コードの変更には、アプリケーションの再構築は必要ありません。

第3章 従業員勤務表スターターアプリケーションのソースコードに関する概要

従業員勤務表スターターアプリケーションは、以下の主要コンポーネントで設定されています。

- Red Hat Business Optimizer を使用して勤務表のロジックを実装して、REST API を提供する **サーバー**
- **gwt** ライブラリーを使用してユーザーインターフェイスを実装し、REST API でサーバーを呼び出す **クライアント**

クライアントとサーバーの間でコードを共有し、これらのコンポーネントを個別にビルドし使用することができます。特に、異なるユーザーインターフェイスを実装して、REST API でサーバーを呼び出すことができます。

主なコンポーネント 2 つに加え、従業員勤務表テンプレートには、ランダムなソースデータのジェネレーター (デモやテスト目的で便利) やベンチマークアプリケーションが含まれます。

モジュールおよび主要なクラス

従業員勤務表テンプレートの Java ソースコードには複数の Maven モジュールが含まれます。これらのモジュールごとに、個別の Maven プロジェクトファイル (**pom.xml**) が含まれていますが、これは共通のプロジェクトで構築するために設計されています。

モジュールには、Java クラスなど複数のファイルが含まれます。このドキュメントでは、全モジュールと、従業員勤務表計算の主な情報を含むその他のファイルとクラスをリストします。

- **employee-rostering-benchmark** モジュール: 乱数データを生成し、ソリューションをベンチマーク化する追加のアプリケーションが含まれます。
- **employee-rostering-distribution** モジュール: readme ファイルが含まれます。
- **employee-rostering-docs** モジュール: ドキュメントファイルが含まれます。
- **employee-rostering-gwtui** モジュール: **gwt** ツールキットを使用して開発したユーザーインターフェイスを使用するクライアントアプリケーションが含まれます。
- **employee-rostering-server** モジュール: Red Hat Business Optimizer を使用して勤務表の計算を行うサーバーをアプリケーションが含まれます。
 - **src/main/resources/org/optaweb/employee rostering/server/solver/employeeRosteringScoreRules.drl** ファイル: Red Hat Business Optimizer の計算用のルールが含まれます。これらのルールは、Drools ルール言語で記述されています。ルールを修正して、従業員勤務表のロジックを変更できます。
 - **src/main/java/org.optaweb.employee rostering.server.roster/rosterGenerator.java** クラス: デモおよびテスト目的でランダムな入力データを生成します。必要な入力データを変更する場合には、生成器も合わせて変更してください。
- **employee-rostering-shared** モジュール: サーバーとクライアントアプリケーション間で共有されるデータ構造が含まれます。特に、**src/main/java/org/optaweb/employee rostering/shared/*** では、このモジュールには以下のクラスを含む勤務表計算の入力データを定義する Java クラスが含まれます。
 - **employee/EmployeeAvailability.java** は、従業員の空き情報を定義します。時間枠ごとに、従業員の空き状況と、従業員の希望する時間枠を指定できます。

- **employee/Employee.java** は従業員を定義します。従業員には名前とスキル一覧を指定します。スキルは、**EmployeeSkillProficiency** オブジェクトで表現します。
- **roster/Roster.java** は計算した勤務表情報を定義します。
- **shift/Shift.java** は、従業員を割り当て可能なシフトを定義します。シフトは、時間枠とスポットで定義します。たとえば、レストランでは、**Kitchen** のスポットで、**February 20 8AM-4PM** の時間枠のシフトなどがあります。複数のシフトを、特定のスポットと時間枠に定義できます。今回の例では、このスポットと時間枠には複数の従業員が必要です。
- **skill/Skill.java** は、従業員が持つスキルを定義します。
- **spot/Spot.java** は、従業員を配置可能なスポットを定義します。たとえば、レストランでは **Kitchen** をスポットとして指定できます。
- **tenant/Tenant.java** はテナントを定義します。テナントごとに、独立したデータセットを表します。あるテナントのデータが変更されても、他のテナントには影響がありません。**employee-rostering-shared** モジュールにも、他の共有アーティファクトが含まれます。
- ***View.java** クラスは、他の情報から計算する値のセットを定義します。クライアントアプリケーションは、REST API 経由でこれらの値を読み取りますが、書き込みはできません。
- ***RestService.java** インターフェイスは REST API を定義します。サーバーとクライアントアプリケーションのいずれも、これらのインターフェイスの実装を個別に定義します。
- **employee-rostering-shared-gwt** モジュール: GWT クライアントアプリケーションに必要なクラスが含まれます。
- **employee-rostering-webapp** モジュール: アプリケーション全体 (クライアントとサーバー) のビルドに必要な HTML と他のファイルが含まれます。

第4章 従業員勤務表スターターアプリケーションの変更

従業員勤務表スターターアプリケーションをニーズに合わせて変更するには、最適化プロセスを統括するルールを変更する必要があります。また、データ構造に必要なデータを含み、ルールに必要な計算が提供されるようにする必要があります。必要なデータがユーザーインターフェイスに存在しない場合は、ユーザーインターフェイスも変更する必要があります。

以下の手順では、従業員勤務表スターターアプリケーションの変更に関する一般的なアプローチを説明しています。

前提条件

- アプリケーションを正常に構築するビルド環境が必要です。
- Java コードの読み取りと変更ができなければなりません。

手順

1. 必要な変更をプランニングします。以下の質問に教えてください。
 - 回避する **必要** がある追加のシナリオは何ですか。これらのシナリオは **ハード制約** です。
 - 可能な場合は、Optimizer が **回避しなければ** ならない追加のシナリオは何ですか。これらのシナリオは **ソフト制約** です。
 - それぞれのシナリオがソリューションで実行されるかどうかを計算するのに必要なデータは何ですか。
 - 既存のバージョンで入力した情報から取得可能なデータはどれですか。
 - ハードコードが可能なデータはどれですか。
 - ユーザー入力が必要なデータと、現在のバージョンで入力されていないデータはどれですか。
2. 必須データを現在のデータから計算するか、ハードコード化できる場合は、計算か、ハードコードを既存のビューまたはユーティリティークラスに追加します。データをサーバー側で計算する必要がある場合は、REST API エンドポイントを追加して読み込みます。
3. ユーザーが必須データを入力する必要がある場合は、データのエントリーを表現するクラスにそのデータを追加 (例: **Employee** クラス) し、データの読み取り、および書き込み用に REST API エンドポイントを追加して、データ入力用にユーザーインターフェイスを変更してください。
4. 全データが利用できる場合は、ルールを変更します。変更の大半は、新規ルールを追加する必要があります。これらのルールは、**employee-rostering-server** モジュールの **src/main/resources/org/optaweb/employeerostering/server/solver/employeeRosteringScoreRules.drl** ファイルに配置されます。
ルールには、Drools 言語を使用します。Drools ルール言語に関する参考情報は、[Rule Language Reference](#) を参照してください。**optashift-employee-rostering-shared** と **optashift-employee-rostering-server** モジュールで定義されるクラスは、デシジョンエンジンで利用できます。
5. アプリケーションの変更後に、ビルドして実行します。

付録A バージョン情報

本書の最終更新日: 2021年11月15日(月)