Red Hat Process Automation Manager 7.8

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

Last Updated: 2020-09-18

Red Hat Process Automation Manager 7.8 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

Red Hat Customer Content Services
brms-docs@redhat.com

法律上の通知

概要

本書は、リファレンス実装として Red Hat Process Automation Manager 7.8 に含まれている従業員勤務表スターターアプリケーションを実行する方法および変更する方法を説明します。

目次

前書き

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 **optaweb-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

- Red Hat CodeReady Studio または IntelliJ IDEA などの統合開発環境を使用すること。
- Java 言語を理解していること。
- React と TypeScript について理解していること (この要件は、OptaWeb UI の開発に必要です)。

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、役職が異なれば、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。さらに、従業員によっては、1回に就業できる時間に制限がある契約を交わしている可能性があります。

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、プランニングエンジンはハード制約に違反しない可能性があります。プランニングエンジンは、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) に順守しようとしますが、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、JAR ファイルとして実行します。

または、Eclipse (Red Hat JBoss CodeReady 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。

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 Process Automation Manager 7.8 の Software Downloads ページから **rhpam-7.8.0-reference-implementation.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. **jboss-rhba-7.8.0.GA-maven-repository/maven-repository** サブディレクトリーの内容を **~/.m2/repository** ディレクトリーにコピーします。
4. アドオンアーカイブから展開した **rhpam-7.8.0-optaweb-employee-rostering.zip** ファイルを展開します。
optaweb-employee-rostering-distribution-7.39.0.Final-redhat-00005 フォルダーが作成され、このフォルダーは、本ドキュメントの後続の手順でベースのフォルダーになります。

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. 従業員勤務表スターターアプリケーションの JAR ファイル実行

参照実装ダウンロードにに含まれる JAR ファイルから、従業員勤務表スターターアプリケーションを実行します。

前提条件

- 「デプロイメントファイルの準備」 の記載どおりに、**rhpam-7.8.0-reference-implementation.zip** ファイルをダウンロードして展開している。
- Java 開発キットがインストールされている。
- Maven がインストールされている。
- ホストからインターネットにアクセスできる。ビルドプロセスは、インターネットを使用して、外部のリポジトリーから Maven パッケージをダウンロードします。

手順

1. コマンドターミナルで、**sources** ディレクトリーに移動します。

2. 以下のコマンドを入力します。

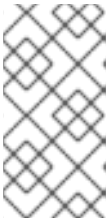
```
mvn clean install -DskipTests
```

3. ビルドプロセスが完了するまで待ちます。

4. **optaweb-employee-rostering-distribution-7.39.0.Final-redhat-00005/sources/optaweb-employee-rostering-standalone/target** ディレクトリーに移動します。

5. 以下のコマンドを実行して、従業員勤務 JAR ファイルを実行します。

```
java -jar optaweb-employee-rostering-standalone-*-exec.jar
```



注記

このコマンドで、非実稼働データベースを使用する従業員勤務表アプリケーションを起動します。実稼働データベースを使用する従業員勤務表を起動するには、上記のコマンドに **--spring.profiles.active=production** の引数を指定してください。

6. アプリケーションにアクセスするには、Web ブラウザーで **http://localhost:8080/** と入力します。

2.3. MAVEN 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ので、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

- 「デプロイメントファイルの準備」 の説明に従ってデプロイメントファイルを準備しておく。
- Java 開発キットがインストールされている。
- Maven がインストールされている。
- ホストからインターネットにアクセスできる。ビルドプロセスは、インターネットを使用して、外部のリポジトリーから Maven パッケージをダウンロードします。

手順

1. **optaweb-employee-rostering-backend** ディレクトリーに移動します。

2. 以下のコマンドを入力します。

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

3. **optaweb-employee-rostering-frontend** ディレクトリーに移動します。
4. 以下のコマンドを入力します。

```
npm start
```

注記

npm を使用してサーバーを起動すると、**npm** によりコードの変更がモニタリングされます。

5. アプリケーションにアクセスするには、Web ブラウザーで **http://localhost:3000/** と入力します。

2.4. 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

- 「デプロイメントファイルの準備」 の説明に従ってデプロイメントファイルを準備しておく。
- Java 開発キットがインストールされている。
- Maven がインストールされている。
- ホストからインターネットにアクセスできる。ビルドプロセスは、インターネットを使用して、外部のリポジトリーから Maven パッケージをダウンロードします。
- MySQL または PostrgeSQL データベースサーバーがデプロイされている。

手順

1. コマンドターミナルで、**optaweb-employee-rostering-standalone/target** ディレクトリーに移動します。
2. 以下のコマンドを実行して、従業員勤務 JAR ファイルを実行します。

```
java -jar optaweb-employee-rostering-standalone-*-exec.jar --
spring.profiles.active=production
spring.datasource.url=<DATABASE_URL> --spring.datasource.username=
<DATABASE_USER> --spring.datasource.password=<DATABASE_PASSWORD>
```

上記の例で、以下の変数を置き換えてください。

- **<DATABASE_URL>**: **jdbc:postgresql://postgresql:5432/MY_DATABASE** などのデータベースに接続する URL
- **<DATABASE_USER>**: データベースに接続するユーザー
- **<DATABASE_PASSWORD>**: **<DATABASE_USER>** のパスワード

2.5. INTELLIJ IDEA 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

IntelliJ IDEA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。

前提条件

- Employee Rostering GitHub ページから従業員勤務表のソースコードをダウンロードしている。
- IntelliJ IDEA、Maven、および Node.js がインストールされている。
- ホストからインターネットにアクセスできる。ビルドプロセスは、インターネットを使用して、外部のリポジトリーから Maven パッケージをダウンロードします。

手順

1. IntelliJ IDEA を起動します。
2. IntelliJ IDEA メインメニューから **File → Open** を選択します。
3. アプリケーションソースのルートディレクトリーを選択し、**OK** をクリックします。
4. メインメニューから **Run → Edit Configurations** を選択します。
5. 表示されるウィンドウで **Templates** を展開し、**Maven** を選択すると、Maven のサイドバーが表示されます。
6. Maven サイドバーの **Working Directory** メニューから **optaweb-employee-rostering-backend** を選択します。
7. **コマンドライン** で **spring-boot:run** と入力します。
8. バックエンドを起動するには、**OK** をクリックします。
9. コマンドターミナルで、**optaweb-employee-rostering-frontend** ディレクトリーに移動します。
10. 以下のコマンドを入力して、フロントエンドを起動します。

```
npm start
```

11. アプリケーションにアクセスするには、Web ブラウザーで **http://localhost:3000/** と入力します。

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構成されています。

- 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 **backend**
- React を使用してユーザーインターフェースを実装し、REST API で **backend** モジュールと対話する **frontend** モジュール

上記のコンポーネントを個別にビルドして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ースを実装して、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生成器 (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。

モジュールと主要なクラス

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。

- **optawebpemployee-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。
- **optaweb-employee-rostering-distribution** モジュール: README ファイルが含まれます。
- **optaweb-employee-rostering-docs** モジュール: ドキュメントファイルが含まれます。
- **optaweb-employee-rostering-frontend** モジュール: React で開発したユーザーインターフェー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。
- **optaweb-employee-rostering-backend** 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/java/org.optaweb.employeerostering.service.roster/rosterGenerator.java**: 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。
  - **src/main/java/org.optaweb.employeerostering.domain.employee/EmployeeAvailability.java**: 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。
  - **src/main/java/org.optaweb.employeerostering.domain.employee/Employee.java**: 従業員を定義します。従業員には名前とスキル一覧があり、契約にしたがって仕事に従事します。スキルは、スキルオブジェクトで表現します。
  - **src/main/java/org.optaweb.employeerostering.domain.roster/Roster.java**: 計算済みの勤務表情報を定義します。
  - **src/main/java/org.optaweb.employeerostering.domain.shift/Shift.java**: 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、2 月 20 日 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。

- **src/main/java/org.optaweb.employeerostering.domain.skill/Skill.java**: 従業員に割り当て可能なスキルを定義します。
- **src/main/java/org.optaweb.employeerostering.domain.spot/Spot.java**: 従業員を配置可能なスポットを定義します。たとえば、**Kitchen** をスポットとして指定できます。
- **src/main/java/org.optaweb.employeerostering.domain.contract/Contract.java**: さまざまな期間の従業員の労働時間を制限する契約を定義します。
- **src/main/java/org.optaweb.employeerostering.domain.tenant/Tenant.java**: 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。
- ***View.java**: ドメインオブジェクト関連のクラス。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。
- ***Service.java**: サービスパッケージに配置されるインターフェースで 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ースの実装を個別に定義します。

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ースに存在しない場合には、ユーザーインターフェー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ビルドするビルド環境がある。
- Java コードの読み取りと変更ができる。

手順

1. 必要な変更をプランニングします。以下の質問に答えてください。
   - 回避する **必要のある** 追加のシナリオにはどのようなものがありますか? このようなシナリオは **ハード制約** になります。
   - できるだけオプティマイザーが **回避を試す** 必要のある 追加のシナリオにはどのようなものがありますか？このようなシナリオは **ソフト制約** になります。
   - 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には、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には、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には、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、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ースを変更してください。
4. 全データが利用できる場合には、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、**optaweb-employee-rostering-backend** モジュールの **src/main/resources/org/optaweb/employeerostering/service/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、「DRL ルールを使用したデシジョンサービスの設計」を参照してください。**optaweb-employee-rostering-backend** モジュールで定義したクラスは、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: 2020 年 9 月 8 日 (木)