Migration Toolkit for Applications 5.3

CLI ガイド

Migration Toolkit for Applications を使用してアプリケーションを移行する方法を説明します。

Last Updated: 2022-12-11

Migration Toolkit for Applications 5.3 CLI ガイド

Migration Toolkit for Applications を使用してアプリケーションを移行する方法を説明します。

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

概要

本ガイドでは、Migration Toolkit for Applications CLI を使用して Java アプリケーションの移行を単純化する方法を説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 はじめに

1.1. CLI ガイドについて

Java アプリケーションやその他のコンポーネントを移行するために、Migration Toolkit for Applications (MTA) を使用するエンジニア、コンサルタント、およびその他のユーザーを対象としています。CLI のインストールおよび実行方法、生成されたレポートを確認し、追加機能を活用する方法を説明します。

1.2. MIGRATION TOOLKIT FOR APPLICATIONS の概要

Migration Toolkit for Applications とは？

Migration Toolkit for Applications (MTA) は、Java アプリケーションの移行およびモダナイゼーションを簡素化する拡張可能でカスタマイズ可能なルールベースのツールです。

MTA は、プロジェクトソースディレクトリーやアプリケーションアーカイブを含むアプリケーションアーティファクトを検査し、変更を必要とするエリアを強調表示する HTML レポートを作成します。MTA は、以下の例を含む多くの移行パスをサポートします。

- Red Hat JBoss Enterprise Application Platform の最新リリースへのアップグレード
- Oracle WebLogic または IBM WebSphere Application Server から Red Hat JBoss Enterprise Application Platform への移行
- アプリケーションのコンテナー化とクラウド化
- Java Spring Boot から Quarkus への移行
- Oracle JDK から OpenJDK への更新

ユースケースおよび移行パスの詳細は、開発者向け MTA Web ページを参照してください。

Migration Toolkit for Applications による移行を単純化する方法

Migration Toolkit for Applications は一般的なリソースを探し、アプリケーションを移行する際の既知の問題点を明らかにします。これは、アプリケーションが使用するテクノロジーのハイレベルビューを提供します。

MTA は、移行またはモダナイゼーションパスの評価に関する詳細なレポートを生成します。このレポートは、大規模なプロジェクトに必要な作業を見積もり、関係する作業を減らすのに役立ちます。

詳細情報

Migration Toolkit for Applications の機能、サポートされる設定、システム要件、利用可能なツールの詳細は、**Migration Toolkit for Applications の概要**を参照してください。

1.3. CLI について

CLI は、Migration Toolkit for Applications のコマンドラインツールです。これにより、ユーザーはアプリケーションに対する移行およびモダライゼーションの作業を評価および優先順位付けできます。他のツールのオーバーヘッドなしに分析を強調表示する多数のレポートが提供されます。CLI にはさまざまなカスタマイズオプションが含まれており、MTA 分析オプションを細かく調整したり、外部の自動化ツールと統合したりできます。

第2章 CLI のインストールおよび実行

2.1. CLI のインストール

CLI は、Linux、Windows、または macOS オペレーティングシステムにインストールできます。

前提条件

- Java Development Kit(JDK) がインストールされている。MTA は以下の JDK をサポートします。
  - OpenJDK 11
  - Oracle JDK 11
- 8 GB RAM
- macOS のインストール: **maxproc** の値は **2048** 以上である必要がある。

手順

1. MTA Download ページ に移動し、**Migration Toolkit CLI** ファイルをダウンロードします。
2. **.zip** ファイルを任意のディレクトリーに展開します。

注記

Windows オペレーティングシステムにインストールする場合は、以下を行います。

1. **.zip** ファイルを **mta** という名前のディレクトリーに展開し、**Path too long** エラーを回避します。または、7-Zip のファイルを、任意の名前のディレクトリーに展開します。
2. 抽出中に **Confirm file replace** ウィンドウが表示されている場合は、**Yes to all** をクリックします。

本ガイドで **<MTA_HOME>** が発生した場合は、これを MTA インストールへの実際のパスに置き換えます。

2.2. CLI の実行

アプリケーションに対して MTA を実行できます。

手順

1. ターミナルを開き、**<MTA_HOME>/bin/** ディレクトリーに移動します。
2. **mta-cli** スクリプト (Windows の場合は **mta-cli.bat**) を実行し、適切な引数を指定します。

```
$ ./mta-cli --input /path/to/jee-example-app-1.0.0.ear \
    --output /path/to/output --source weblogic --target eap:6 \
    --packages com.acme org.apache
```

- **--input**: 評価されるアプリケーション。
- **--output**: 生成されたレポートの出力ディレクトリー。
- **--source**: アプリケーション移行元のテクノロジー。
- **--target**: アプリケーション移行先のテクノロジー。
- **--packages**: 評価されるパッケージ。この引数は、パフォーマンスを改善するために強く推奨されます。

3. レポートにアクセスします。

2.2.1. MTA コマンドの例

アプリケーションアーカイブでの MTA の実行
次のコマンドは、JBoss EAP 5 から JBoss EAP 7 に移行するための サンプル EAR アーカイブ jee-example-app-1.0.0.ear の **com.acme** パッケージおよび **org.apache** パッケージを解析します。

```
$ <MTA_HOME>/bin/mta-cli \
    --input /path/to/jee-example-app-1.0.0.ear \
    --output /path/to/report-output/ --source eap:5 --target eap:7 \
    --packages com.acme org.apache
```

ソースコードでの MTA の実行
次のコマンドは、JBoss EAP 6 に移行するための seam-booking-5.2 サンプルソースコードの **org.jboss.seam** パッケージを分析します。

```
$ <MTA_HOME>/bin/mta-cli --sourceMode --input /path/to/seam-booking-5.2/ \
    --output /path/to/report-output/ --target eap:6 --packages org.jboss.seam
```

cloud-readiness ルールの実行
次のコマンドは、JBoss EAP 7 に移行するためのサンプル EAR アーカイブ jee-example-app-1.0.0.ear の **com.acme** パッケージおよび **org.apache** パッケージを解析します。また、クラウドの準備ができるかどうかも評価します。

```
$ <MTA_HOME>/bin/mta-cli --input /path/to/jee-example-app-1.0.0.ear \
    --output /path/to/report-output/ \
    --target eap:7 --target cloud-readiness --packages com.acme org.apache
```

MTA プロパティーの上書き
デフォルトの **Fernflower** デコンパイラーを上書きするには、コマンドラインで **-Dwindup.decompiler** 引数を渡します。たとえば、**Procyon** デコンパイル機能を使用する場合は、以下の構文を使用します。

```
$ <MTA_HOME>/bin/mta-cli -Dwindup.decompiler=procyon \
    --input <INPUT_ARCHIVE_OR_DIRECTORY> --output <OUTPUT_REPORT_DIRECTORY> \
    --target <TARGET_TECHNOLOGY> --packages <PACKAGE_1> <PACKAGE_2>
```

2.2.2. MTA CLI Bash の完了

MTA CLI は、Linux システムの Bash 補完を有効にするオプションを提供し、コマンドを入力した時に Tab キーを押して MTA コマンドライン引数を自動化できます。たとえば、bash 補完を有効にすると、以下のように利用可能な引数のリストが表示されます。

```
$ <MTA_HOME>/bin/mta-cli [TAB]
```

bash 補完の有効化
現在のシェルで bash 補完を有効にするには、以下のコマンドを実行します。

```
$ source <MTA_HOME>/bash-completion/mta-cli
```

永続的な bash 補完の有効化
以下のコマンドを使用すると、再起動後も bash 補完が維持されます。

- システムを再起動しても特定のユーザーの bash 補完を有効にするには、そのユーザーの ~/**.bashrc** ファイルに以下の行を追加します。

  ```
  source <MTA_HOME>/bash-completion/mta-cli
  ```

- システムの再起動後すべてのユーザーの bash 補完を有効にするには、root ユーザーで Migration Toolkit for Applications CLI Bash 補完ファイルを **/etc/bash_completion.d/** ディレクトリーにコピーします。

  ```
  # cp <MTA_HOME>/bash-completion/mta-cli /etc/bash_completion.d/
  ```

2.2.3. MTA ヘルプへのアクセス

mta-cli コマンドで利用可能な引数の完全な一覧を表示するには、ターミナルを開き、**<MTA_HOME>** ディレクトリーに移動して、以下のコマンドを実行します。

```
$ <MTA_HOME>/bin/mta-cli --help
```

2.2.4. OpenRewrite レシピの使用

重要

OpenRewrite レシピのサポートはテクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

テクノロジープレビュー機能のサポート範囲については、Red Hat カスタマーポータルの テクノロジープレビュー機能のサポート範囲 を参照してください。

MTA CLI で OpenRewrite レシピを使用すると、Java アプリケーションのソースコードをリファクタリングできます。

たとえば、OpenRewrite レシピの **org.jboss.windup.JavaxToJakarta** は、インポートされた **javax** パッケージの名前を同等の **jakarta** に変更します。

手順

1. **mta-cli** を実行し、レシピ名、設定ファイルへのパス、およびアプリケーションを指定します。

```
$ ./mta-cli --openrewrite --input </path/to/source/project> \
  "-Drewrite.configLocation=<path/to/rewrite.yaml>  \
  "-DactiveRecipes=<recipe_name>" --goal dryRun
```

- **"-DactiveRecipes=<recipe name&** gt;" : OpenRewrite recipe を指定します (例: **org.jboss.windup.JavaxToJakarta**)。
- **--input**: リファクタリングするアプリケーションを指定します。アプリケーションは、Maven Project Object Model (POM) XML ファイル **pom.xml** を含むソースコードプロジェクトの最上位にある必要があります。
- **-Drewrite.configLocation=<path/to/rewrite.yaml>** : 使用する **rewrite.yaml** 設定ファイルの場所。同梱の **rewrite.yaml** 設定ファイルは、**<MTA_HOME>/rules/openrewrite** サブフォルダーにあります (例: **" -Drewrite.configLocation= <MTA_HOME>/rules/openrewrite/jakarta/javax/imports/rewrite.yaml")**。
- **"-DactiveRecipes=<recipe name&** gt;" : OpenRewrite recipe を指定します (例: **org.jboss.windup.JavaxToJakarta**)。
  activeRecipes パラメーターでそれぞれを指定すると、複数のレシピを含めることができます。たとえば、レシピ **org.jboss.windup.JavaxInjectToJakartaInject** と **org.jboss.windup.JavaxEjbToJakartaEjb"** を含めるには、**"-DactiveRecipes=<recipe name>"** で以下を入力します。

  ```
  "DactiveRecipes=org.jboss.windup.JavaxInjectToJakartaInject, \
    org.jboss.windup.JavaxEjbToJakartaEjb"
  ```

- **--goal**: オプション: 実行する OpenRewrite Maven ゴール。
  - **dryRun**: スクリプトは、提案される変更の一覧を返します。**"Run 'mvn rewrite:run' to apply the recipes"** メッセージは無視します。
  - **run**: スクリプトは変更を適用します。

2. **--goal run** を指定して **mta-cli** を実行して、レシピを適用します。

```
$ ../mta-cli --openrewrite --input </path/to/source/project> \
  "-Drewrite.configLocation=<path/to/rewrite.yaml>  \
  "-DactiveRecipes=<recipe_name>" --goal run
```

2.3. レポートへのアクセス

Migration Toolkit for Applications を実行すると、コマンドラインの **--output** 引数を使用して指定する **<OUTPUT_REPORT_DIRECTORY>** にレポートが生成されます。

output ディレクトリーには、以下のファイルおよびサブディレクトリーが含まれます。

```
<OUTPUT_REPORT_DIRECTORY>/
├── index.html          // Landing page for the report
├── <EXPORT_FILE>.csv   // Optional export of data in CSV format
├── archives/           // Archives extracted from the application
├── mavenized/          // Optional Maven project structure
├── reports/            // Generated HTML reports
├── stats/              // Performance statistics
```

手順

1. MTA の実行後に表示される出力から、レポートの **index.html** ファイルのパスを取得します。

```
Report created: <OUTPUT_REPORT_DIRECTORY>/index.html
        Access it at this URL: file:///<OUTPUT_REPORT_DIRECTORY>/index.html
```

2. ブラウザーを使用して **index.html** ファイルを開きます。
生成されたレポートが表示されます。

第3章 レポートの確認

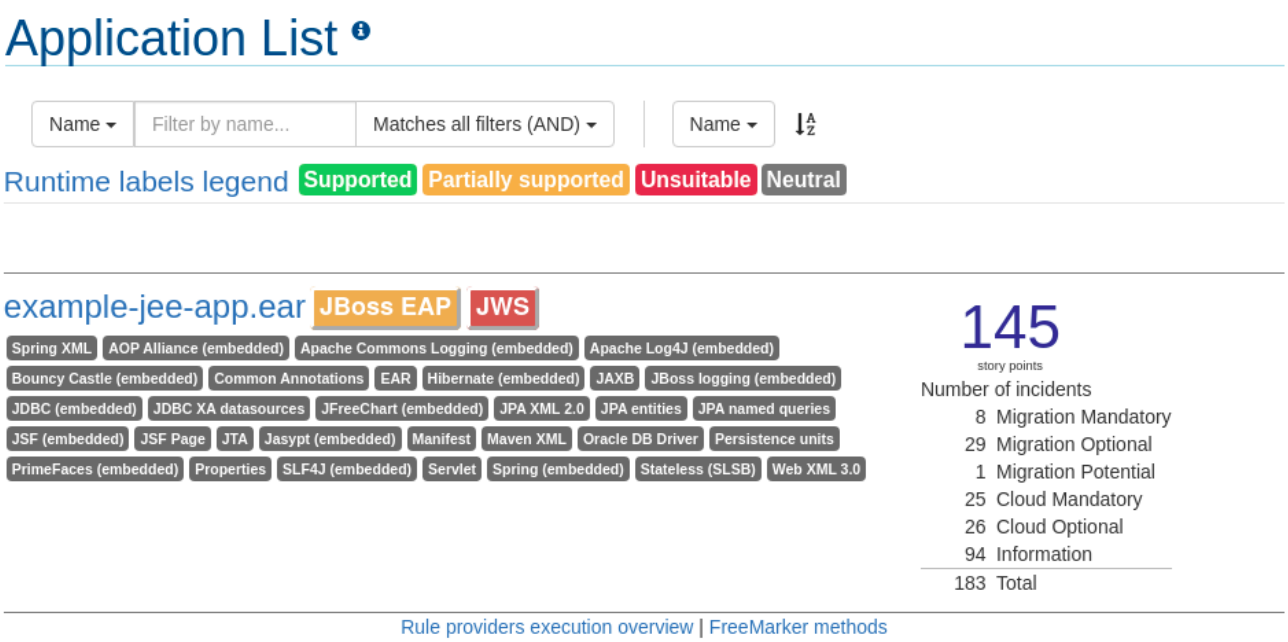
次のセクションに示すレポートの例は、MTA GitHub ソースリポジトリーにある jee-example-app-1.0.0.ear サンプルアプリケーションの **com.acme** パッケージおよび **org.apache** パッケージを分析した結果です。

以下のコマンドを使用してレポートが生成されました。

```
$ <MTA_HOME>/bin/mta-cli --input /home/username/mta-cli-source/test-files/jee-example-app-1.0.0.ear/ --output /home/username/mta-cli-reports/jee-example-app-1.0.0.ear-report --target eap:6 --packages com.acme org.apache
```

ブラウザーを使用して、レポート出力ディレクトリーにある **index.html** ファイルを開きます。これにより、処理されたアプリケーションの一覧が表示されます。各行には、ストーリーポイント、インシデントの数、アプリケーションで発生したテクノロジーの概要が含まれます。

図3.1 アプリケーションリスト



注記

新しいルールが MTA に追加されると、インシデントと予測されるストーリーポイントが変わります。この値は、このアプリケーションをテストする際に表示される値と一致しない場合があります。

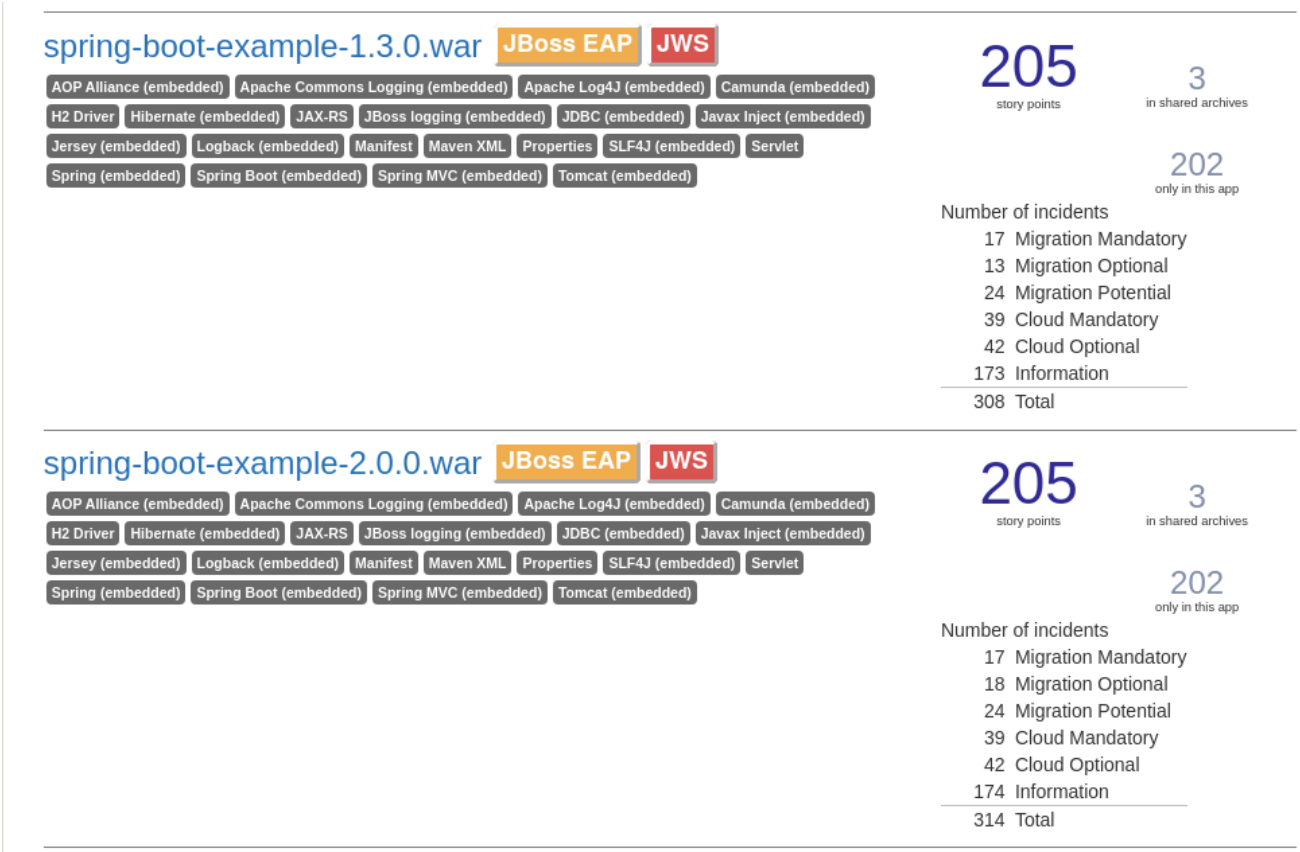
以下の表は、このメインの MTA ランディングページからアクセスできるレポートおよびページの一覧です。アプリケーションの名前 **jee-example-app-1.0.0.ear** をクリックして、アプリケーションレポートを表示します。

ページ	アクセス方法
アプリケーション	アプリケーションの名前をクリックします。
Technologies レポート	ページ上部にある **Technologies** リンクをクリックします。

ページ	アクセス方法
Dependencies Graph レポート	ページ上部の **Dependencies Graph** リンクをクリックします。
複数のアプリケーションで共有されるアーカイブ	**Archives shared by multiple applications** リンクをクリックします。このリンクは、複数のアプリケーションに共有アーカイブがある場合にのみ利用できることに注意してください。
ルールプロバイダー実行の概要	ページの下部にある **Rule providers execution overview** リンクをクリックします。
使用される FreeMarker 機能およびディレクティブ	ページの下部にある **FreeMarker methods** リンクをクリックします。
フィードバックフォームを送信	トップナビゲーションバーの **Send Feedback** リンクをクリックし、MTA チームにフィードバックを送信できるフォームを開きます。

アプリケーションが、他の分析済みアプリケーションとアーカイブを共有している場合は、共有アーカイブからのストーリーポイントの数と、このアプリケーションに固有のストーリーポイントの数が表示されることに注意してください。

図3.2 共有アーカイブ



アプリケーション間で共有されるアーカイブに関する情報は、複数のアプリケーションで共有されるアーカイブレポートを参照してください。

3.1. アプリケーションレポート

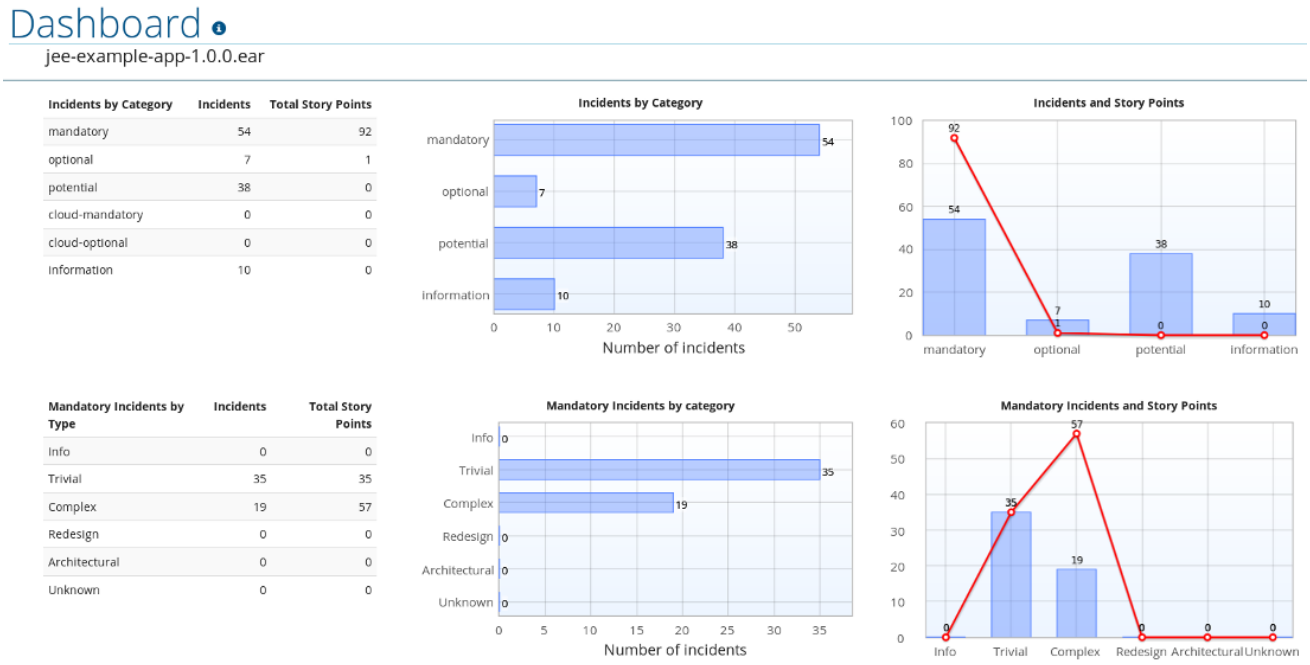
3.1.1. ダッシュボード

アプリケーションのリスト でアプリケーション名をクリックして、レポートランディングページからこのレポートにアクセスします。

ダッシュボードでは、アプリケーションの移行作業全体の概要が表示されます。つまり、以下のようになります。

- カテゴリー別のインシデントおよびストーリーポイント
- 提案された変更の作業レベル別のインシデントおよびストーリーポイント
- パッケージ別のインシデント

図3.3 ダッシュボード



上部のナビゲーションバーには、このアプリケーションの移行に関する追加の詳細を含むさまざまなレポートが一覧表示されます。現在のアプリケーションに適用可能なレポートのみが利用できることに注意してください。

レポート	説明
Issues	注意が必要な問題すべてについて簡潔に説明します。
Application Details	移行時に注意する必要がある可能性のあるアプリケーション内で見つかったすべてのリソースの詳細を説明します。
Technologies	機能を基にグループ化されたすべての埋め込みライブラリーを表示します。これにより、各アプリケーションで使用されるテクノロジーを迅速に表示できます。
Dependencies Graph	分析したアプリケーション内にあるすべての Java パッケージの依存関係のグラフを表示します。このグラフは、各依存関係の関係も示しており、ネストになった複数の依存関係を表示できます。

レポート	説明
Dependencies	アプリケーション内にある Java パッケージの依存関係をすべて表示します。
Unparsable	MTA が想定される形式で解析できなかったすべてのファイルを示しています。たとえば、**.xml** または **.wsdl** 接尾辞が含まれるファイルは XML ファイルであると仮定します。XML パーサーが失敗すると、問題はここに報告され、また個々のファイルがリストされている場所も報告されます。
Remote Services	アプリケーション内で見つかったすべてのリモートサービス参照を表示します。
EJB	アプリケーション内の EJB のリストが含まれます。
JBPM	分析中に見つかった JBPM 関連のリソースすべてが含まれます。
JPA	アプリケーションで見つかったすべての JPA 関連リソースの詳細が含まれます。
Hibernate	アプリケーションで検出されたすべての Hibernate 関連リソースの詳細が含まれています。
Server Resources	入力アプリケーションですべてのサーバーリソース (JNDI リソースなど) を表示します。
Spring Beans	分析中に見つかった Spring Bean の一覧が含まれます。
Hard-Coded IP Addresses	アプリケーションで見つかったすべてのハードコーディングされた IP アドレスの一覧を提供します。
Ignored Files	特定のルールおよび MTA 設定に基づいてアプリケーションに含まれるファイルを一覧表示し、処理されませんでした。詳細は、**--userIgnorePath** オプションを参照してください。
About	現在のバージョンの MTA を説明し、詳細なヘルプリンクを提供します。

3.1.2. Issues レポート

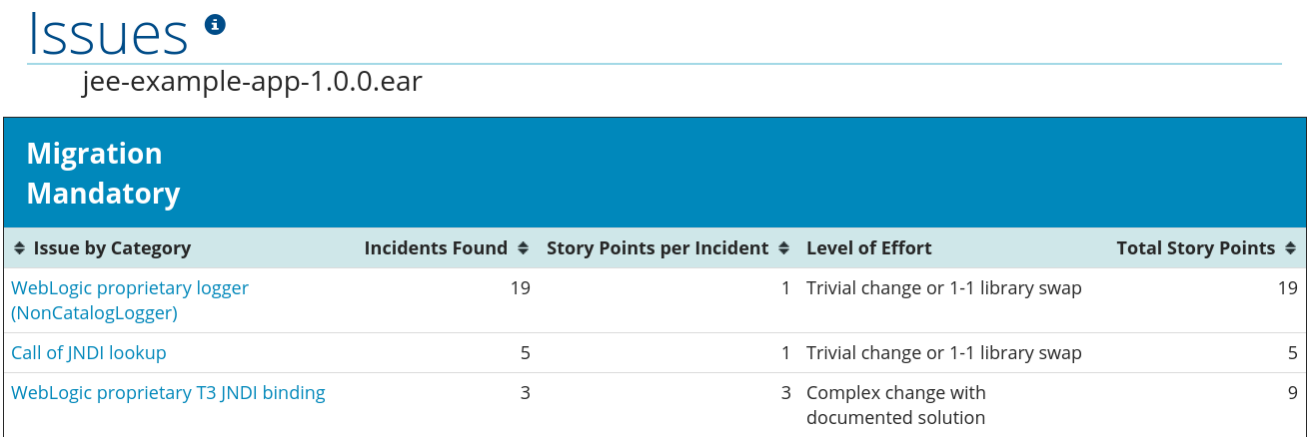
Issues リンクをクリックして、Dashboard からこのレポートにアクセスします。

このレポートには、選択した移行パスによって発生したすべての問題に関する詳細情報が含まれます。発生した問題ごとに以下の情報が提供されます。

- 問題を要約するタイトル。
- インシデントの合計数、または問題の発生回数。
- 問題の 1 つののインスタンスを解決するルールのストーリーポイント。

- この問題を解決するための推定作業量レベル。
- 発生したすべてのインスタンスを解決するための全ストーリーポイント。これは、インシデントごとのストーリーポイントで検出されたインシデントの数を掛けて計算されます。

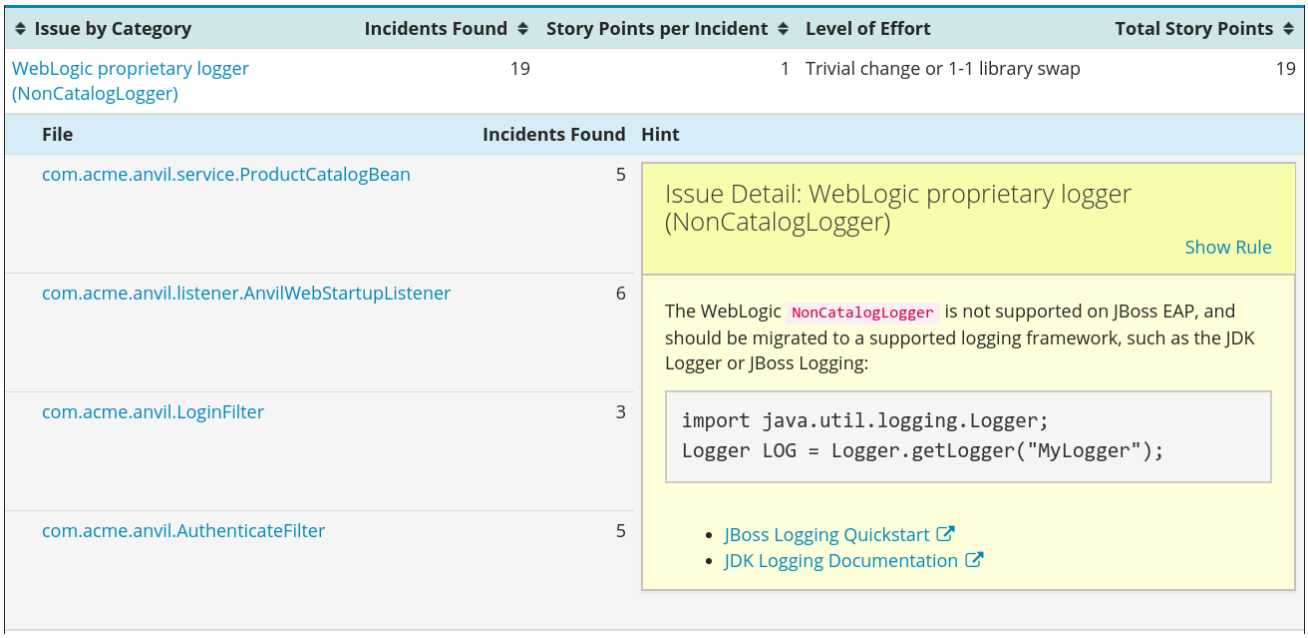
図3.4 Issues レポート



報告された各問題は、タイトルをクリックして追加情報を取得することで拡張できます。以下の情報が含まれています。

- インシデントが発生したファイルの一覧と、各ファイル内のインシデントの数。ファイルが Java ソースファイルの場合は、ファイル名をクリックすると、対応するソースレポートが表示されます。
- 問題の詳細情報。この説明は問題の概要を示し、既知の解決策を提供し、問題または解決策に関するサポートドキュメントを参照します。
- 問題を生成したルールへの **Show Rule** というタイトルの直接リンク。

図3.5 問題の拡張



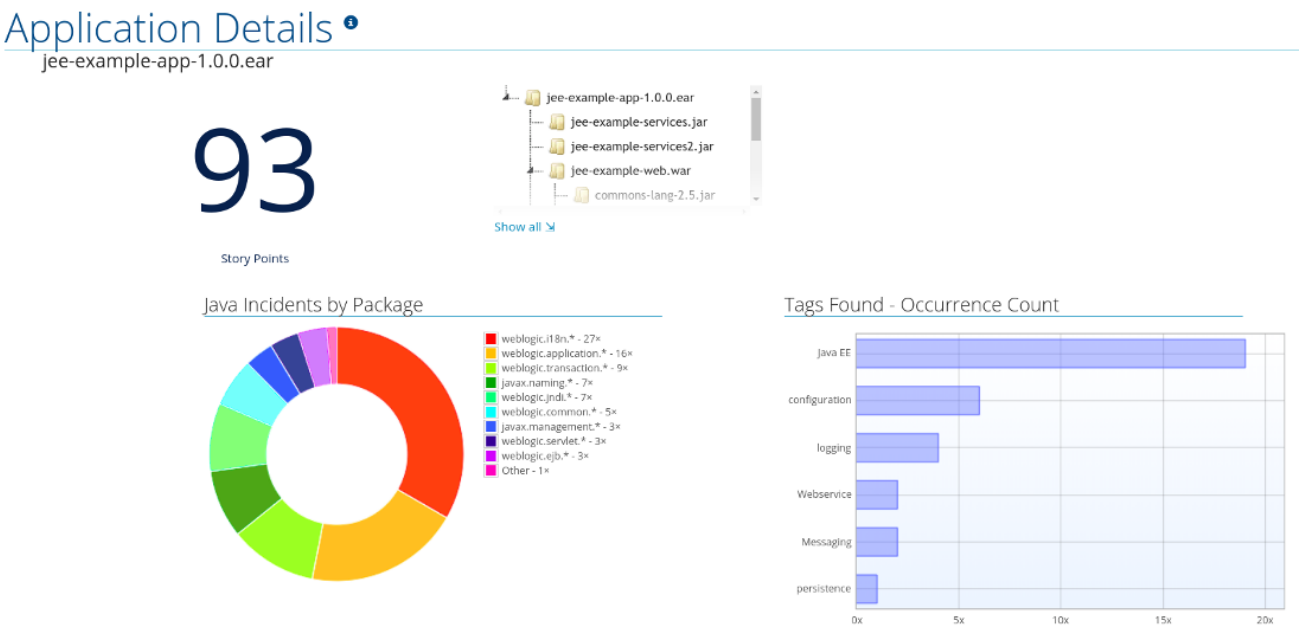
デフォルトでは、問題は 4 つのカテゴリーに分類されます。これらのカテゴリーに関する情報は、カテゴリーを参照してください。

3.1.3. Application Details レポート

Application Details リンクをクリックして、ダッシュボードからこのレポートにアクセスします。

レポートには、ストーリーポイント、パッケージごとの Java インシデント、およびアプリケーションで見つかったテクノロジーの発生回数が記載されます。以下は、移行プロセス中に生成されたアプリケーションメッセージの表示です。最後に、プロセス中に分析される各アーカイブにこの情報の内訳が表示されます。

図3.6 Application Details レポート

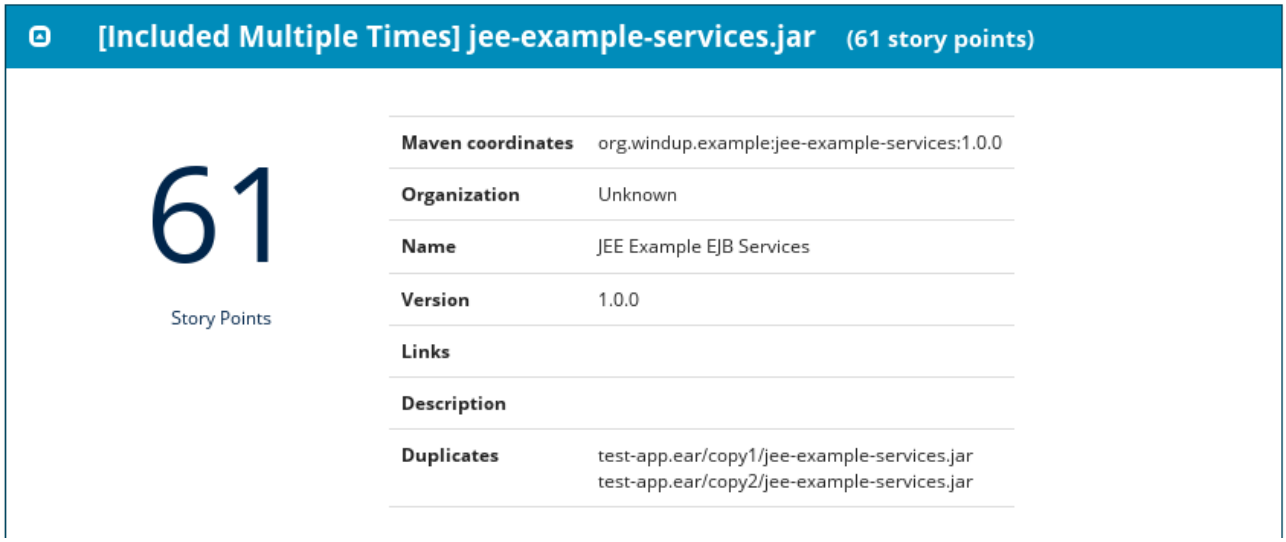


jee-example-app-1.0.0.ear/jee-example-services.jar を展開して、ストーリーポイント、パッケージごとの Java インシデント、およびこのアーカイブで見つかったテクノロジーの発生数を確認します。この概要は、移行に割り当てられたストーリーポイントの合計で始まり、アーカイブ内の各ファイルに必要な変更の詳細を示す表が続きます。レポートには以下のコラムが含まれます。

列名	説明
名前	分析されるファイルの名前。
Technology	分析するファイルのタイプ (例: **Decompiled Java File** または **Properties**)。
Issues	レビューまたは変更が必要なコードのエリアに関する警告。
Story Points	ファイルの移行に必要な作業レベル。

アーカイブがアプリケーションに複数回複製されると、そのアーカイブはレポートで 1 回だけ一覧表示され、**[Included multiple times]** でタグ付けされます。

図3.7 アプリケーションでのアーカイブの重複



アプリケーション内で重複するアーカイブのストーリーポイントは、そのアプリケーションの合計ストーリーポイント数に 1 回だけカウントされます。

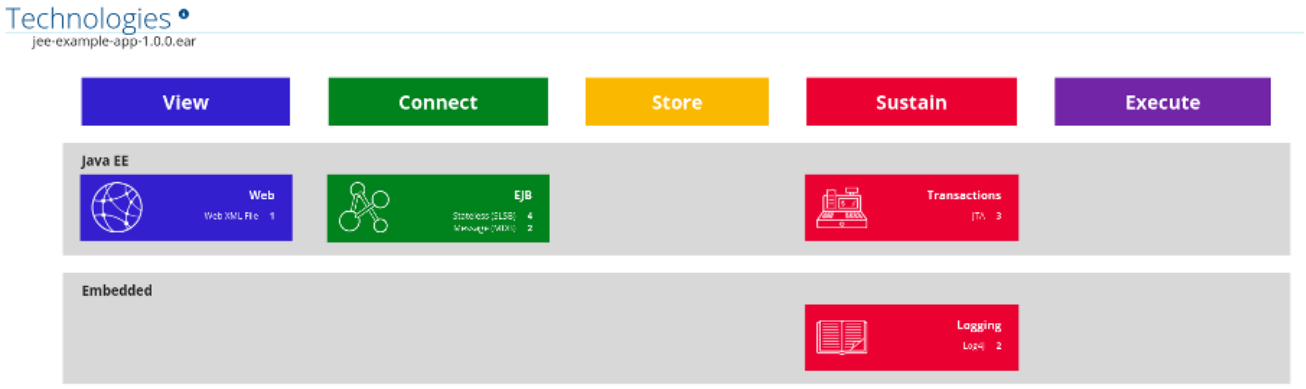
3.1.4. Technologies レポート

Technologies リンクをクリックして、ダッシュボードからこのレポートにアクセスします。

レポートには、解析されたアプリケーションで機能別にグループ化されたテクノロジーが一覧表示されます。これは、アプリケーションに含まれるテクノロジーの概要であり、各アプリケーションの目的を素早く理解できるように設計されています。

以下の図は、**jee-example-app** で使用されるテクノロジーを示しています。

図3.8 アプリケーションのテクノロジー



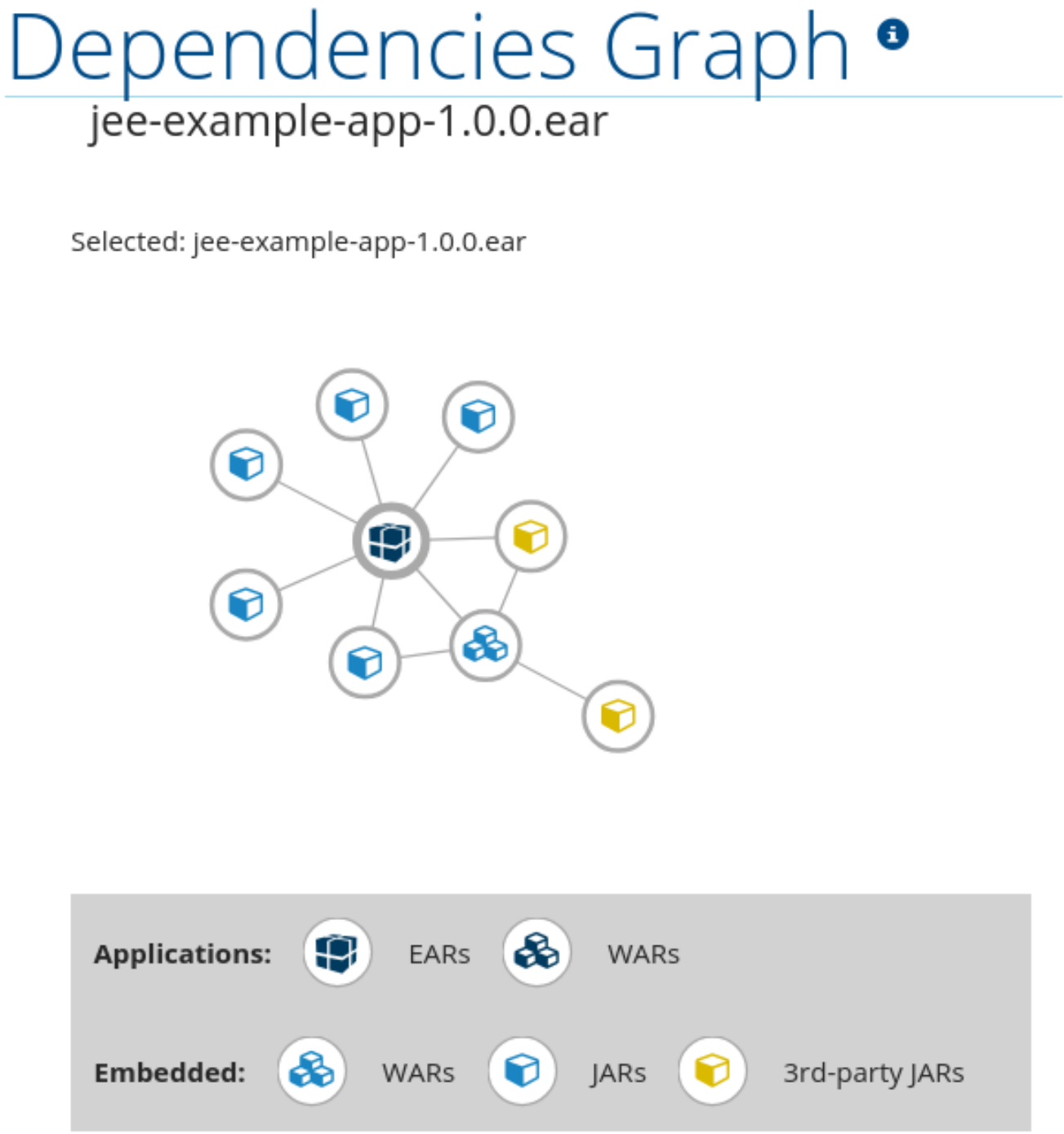
3.1.5. アプリケーションの Dependencies Graph レポート

分析したアプリケーションの依存関係が、ダッシュボードから Dependencies Graph リンクからアクセスできるこのレポートに表示されます。

これには、サードパーティー JAR を含む WAR および JAR の一覧が含まれ、含まれる各ファイル間の関係をグラフ化します。グラフの各円は、アプリケーションで定義されている一意の依存関係を表します。

以下の図は、**jee-example-app** で使用される依存関係と、選択したアプリケーションがグラフの中央にあります。

図3.9 アプリケーションの依存関係グラフ (Dependencies Graph)



Dependencies Graph との対話

依存関係グラフは、以下のいずれかを使用して調整できます。

- 依存関係をクリックすると、左上隅にアプリケーションの名前が表示されます。選択されている間、依存関係には、上のイメージの中央に見られるように、それを識別する影付きの円があります。
- 円をクリックしてドラッグすると、再配置されます。マウスを放すと、依存関係がカーソルの位置に固定されます。
- 固定された依存関係をクリックすると、依存関係がアプリケーションからデフォルトの距離に戻ります。
- どこかをダブルクリックすると、グラフ全体がデフォルトの状態に戻ります。

- 説明の項目をクリックすると、選択したタイプのすべての項目が有効または無効になります。たとえば、埋め込み WAR アイコンを選択すると、そのアイコンが有効な場合はすべての埋め込み WAR が無効になり、無効な場合はこれらの依存関係が有効になります。

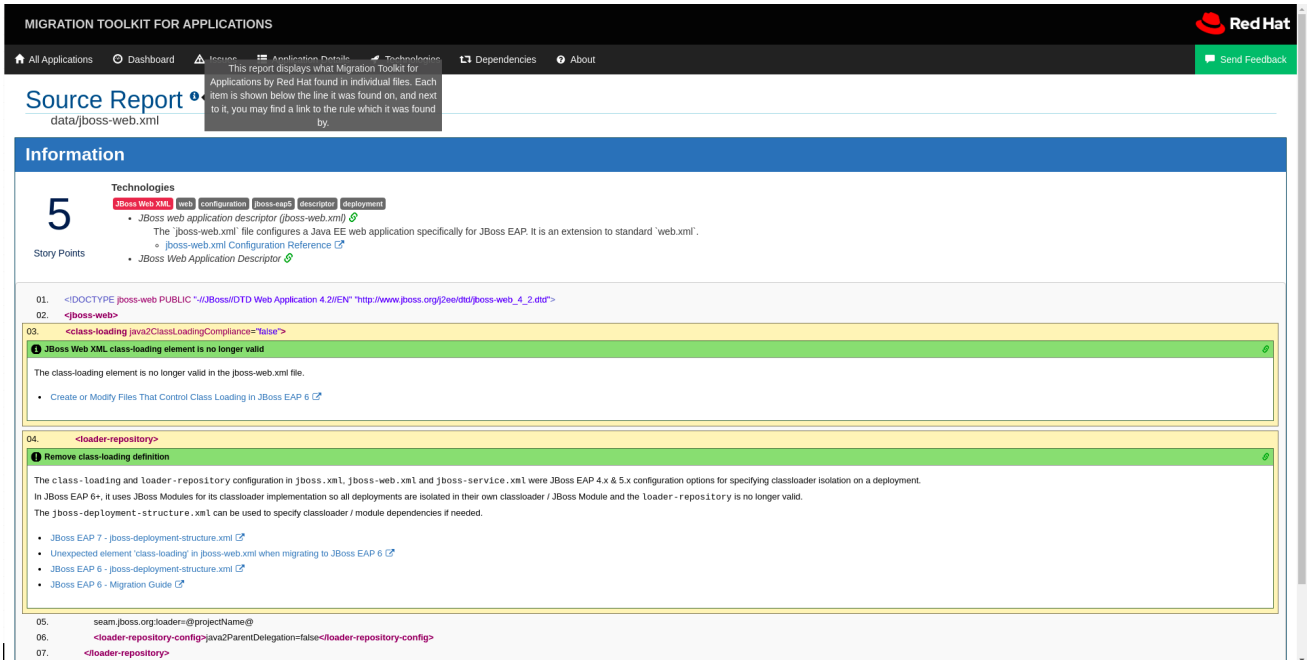
3.1.6. ソースレポート

jee-example-services.jar の分析には、JAR 内のファイルと、各サービスに割り当てられた警告およびストーリーポイントが一覧表示されます。このテスト時の **com.acme.anvil.listener.AnvilWebLifecycleListener** ファイルには 22 個の警告があり、16 個のストーリーポイントが割り当てられています。ファイルリンクをクリックして詳細を表示します。

- **Information** セクションは、ストーリーポイントの概要を説明します。
- ファイルのソースコードが続きます。移行が必要な時点でファイルに警告が表示されます。

この例では、警告がさまざまなインポートステートメント、宣言、およびメソッド呼び出しに表示されます。各警告は、問題と実行すべきアクションを記述します。

図3.10 ソースレポート



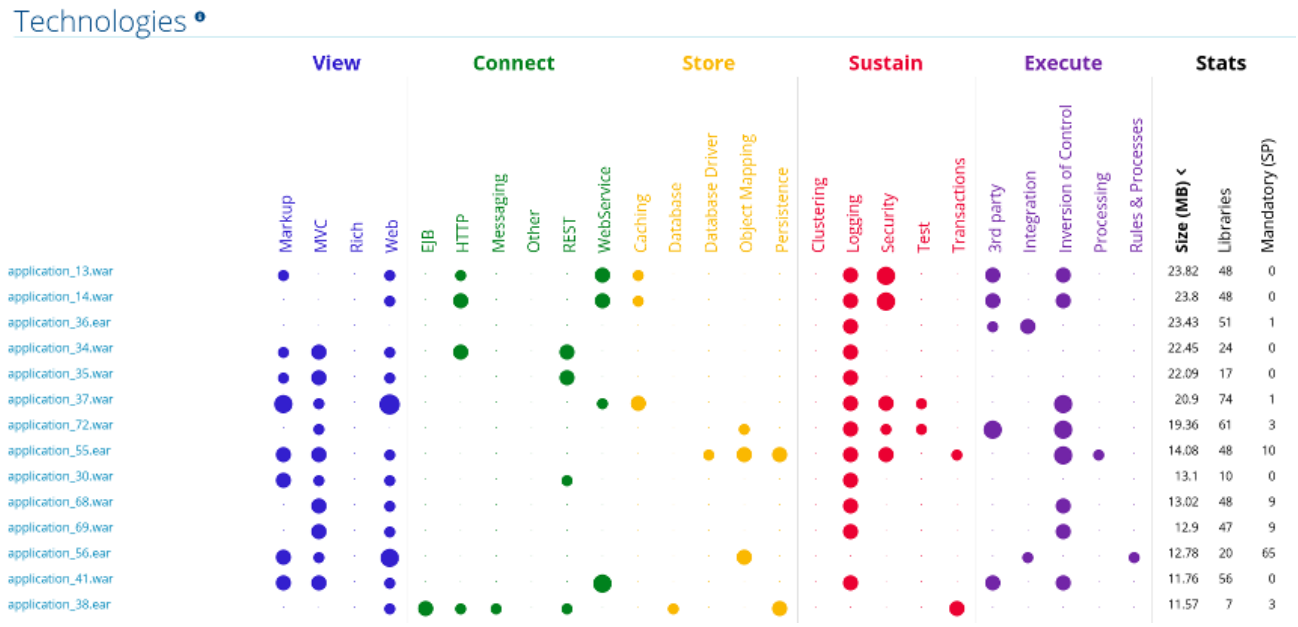
3.2. TECHNOLOGIES レポート

Technologies リンクをクリックして、レポートランディングページからこのレポートにアクセスします。

このレポートは、解析されたアプリケーションに使用されたテクノロジーを機能別にまとめたリストを提供します。これは、テクノロジーがどのように分散されているかを示し、通常は、多数のアプリケーションを分析してアプリケーションをグループ化し、パターンを特定した後にレビューされます。また、各アプリケーションのサイズ、ライブラリー数、およびストーリーポイントの合計も表示されます。

Markup などの任意のヘッダーをクリックすると、結果が降順に並べ替えられます。同じヘッダーを再度選択すると、結果が昇順になります。現在選択されているヘッダーは、並べ替えの方向を示す方向矢印の横に太字で示されます。

図3.11 複数のアプリケーションで使用されるテクノロジー



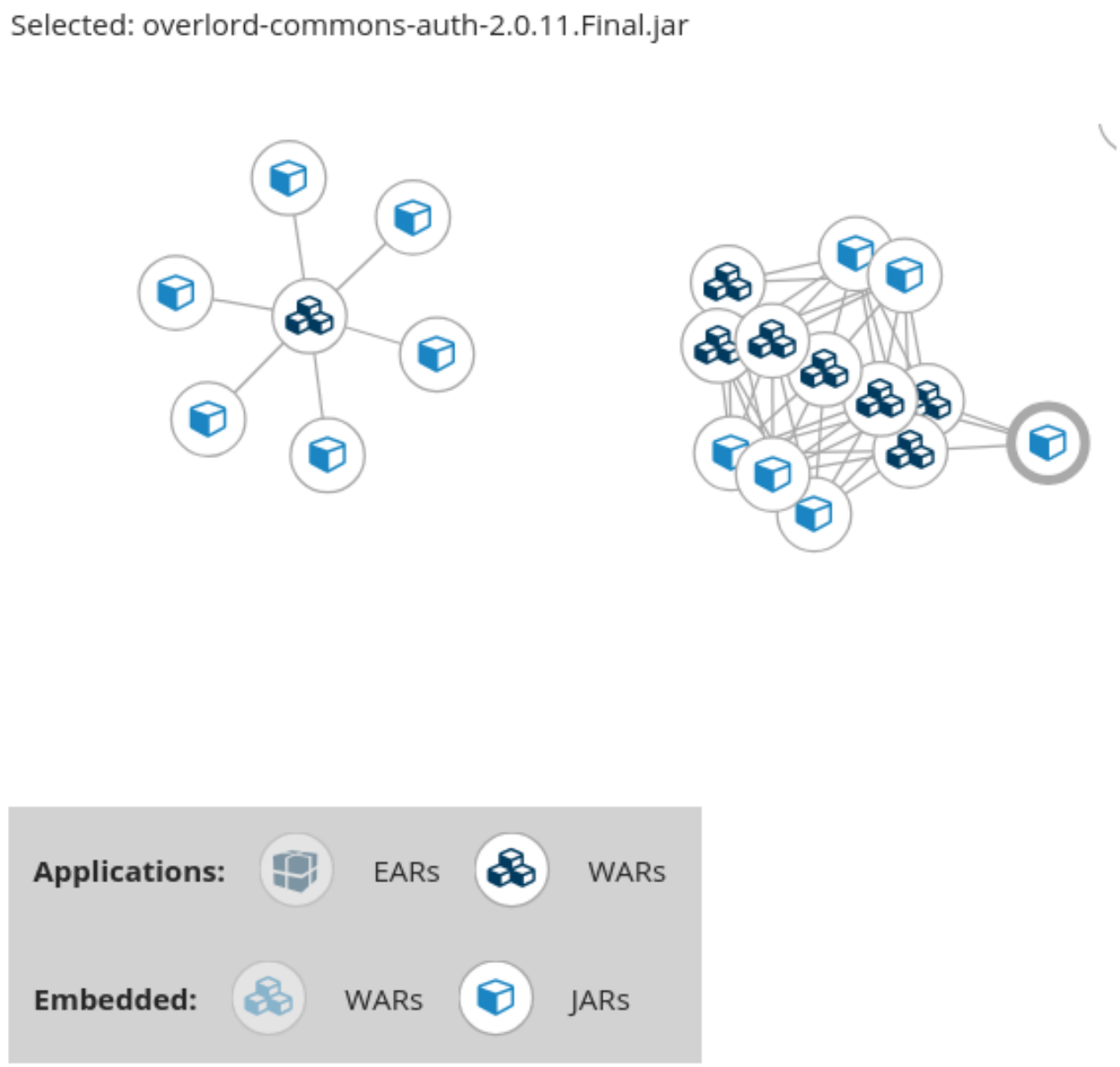
3.3. DEPENDENCIES GRAPH レポート

Dependencies Graph リンクをクリックして、レポートランディングページからこのレポートにアクセスします。

これには、WAR と JAR の一覧が含まれ、含まれる各ファイル間の関係をグラフ化します。グラフの各円は、アプリケーションで定義されている一意の依存関係を表します。ファイルを複数のアプリケーションに依存関係として組み込むと、それらはグラフにリンクされます。

以下の図では、2 つの異なるグループを確認できます。左側には、複数の JAR を依存関係として定義する単一の WAR が表示されます。右側には、複数の WAR で使用されるものと同じ依存関係があります。この 1 つは、選択された **overlord-commons-auth-2.0.11.Final.jar** です。

図3.12 複数のアプリケーション間の依存関係グラフ



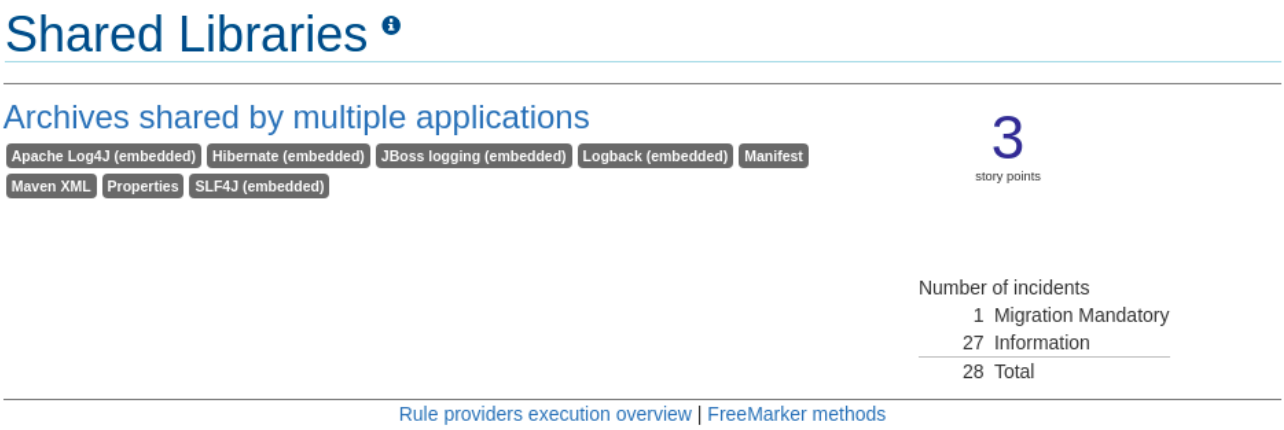
依存関係グラフは、以下のいずれかを使用して調整できます。

- 依存関係をクリックすると、左上隅にアプリケーションの名前が表示されます。選択されている間、依存関係には、上のイメージの中央に見られるように、それを識別する影付きの円があります。
- 円をクリックしてドラッグすると、再配置されます。マウスを放すと、依存関係がカーソルの位置に固定されます。
- 固定された依存関係をクリックすると、依存関係がアプリケーションからデフォルトの距離に戻ります。
- どこかをダブルクリックすると、グラフ全体がデフォルトの状態に戻ります。
- 説明の項目をクリックすると、選択したタイプのすべての項目が有効または無効になります。たとえば、埋め込み WAR アイコンを選択すると、そのアイコンが有効な場合はすべての埋め込み WAR が無効になり、無効な場合はこれらの依存関係が有効になります。

3.4. 複数のアプリケーションで共有されるアーカイブ

複数のアプリケーションによって共有されるアーカイブ リンクをクリックして、レポートランディングページからこれらのレポートにアクセスします。このリンクは、適用可能な共有アーカイブがある場合にのみ利用できることに注意してください。

図3.13 複数のアプリケーションで共有されるアーカイブ



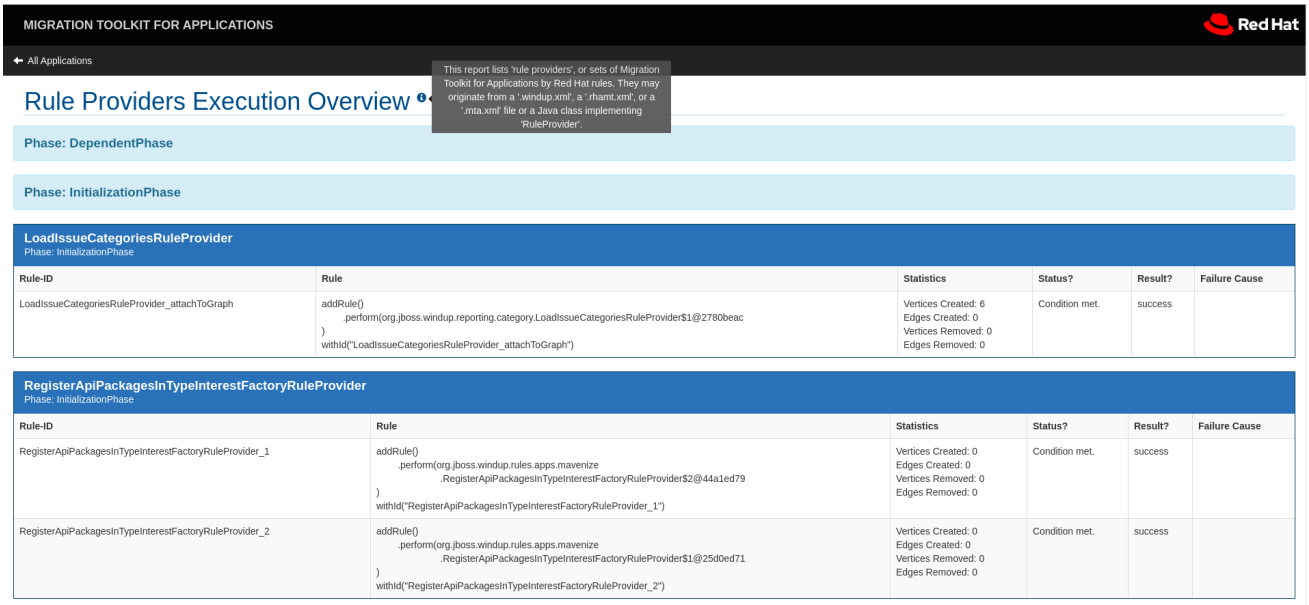
これにより、複数のアプリケーション間で共有されるすべてのアーカイブの詳細なレポートを表示できます。

3.5. ルールプロバイダー実行の概要

Rule providers execution overview リンクをクリックして、レポートランディングページからこのレポートにアクセスします。

このレポートは、アプリケーションに対して MTA 移行コマンドを実行する際に実行するルールの一覧を提供します。

図3.14 ルールプロバイダー実行の概要

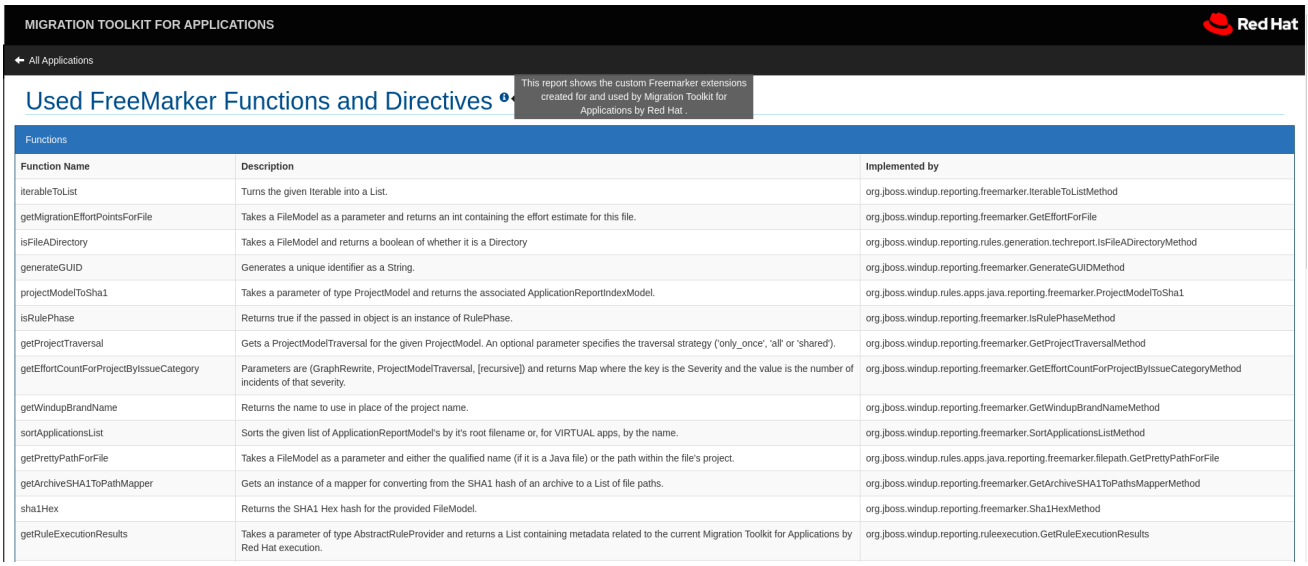


3.6. 使用される FREEMARKER 機能およびディレクティブ

FreeMarker methods のリンクをクリックして、レポートランディングページからこのレポートにアクセスします。

このレポートには、レポートの作成に使用されたすべての登録済み関数およびディレクティブがリストされます。これは、デバッグの目的で、または独自のカスタムレポートを作成する場合に役立ちます。

図3.15 使用される FreeMarker 機能およびディレクティブ



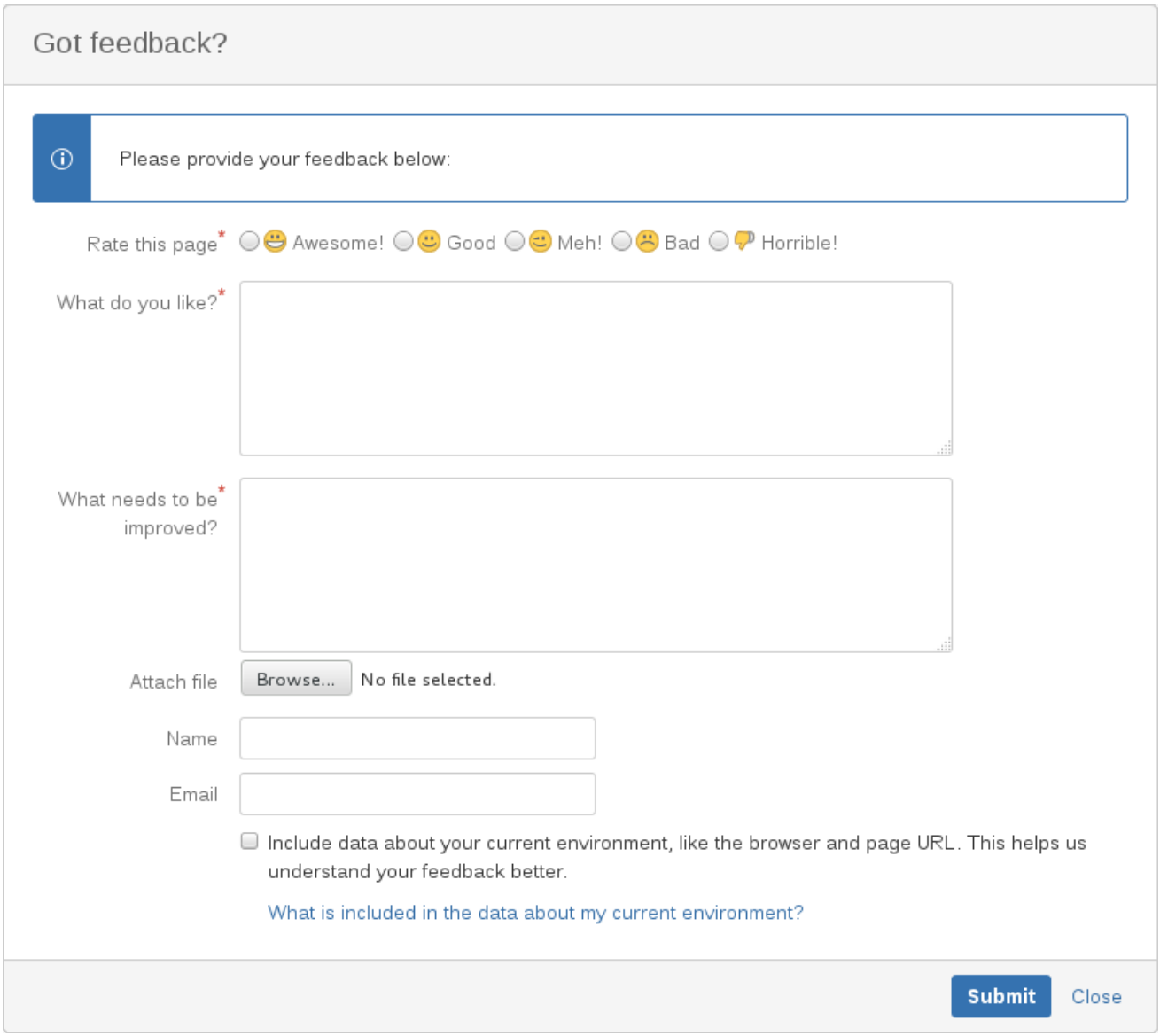
Function Name	Description	Implemented by
iterableToList	Turns the given Iterable into a List.	org.jboss.windup.reporting.freemarker.IterableToListMethod
getMigrationEffortPointsForFile	Takes a FileModel as a parameter and returns an int containing the effort estimate for this file.	org.jboss.windup.reporting.freemarker.GetEffortForFile
isFileADirectory	Takes a FileModel and returns a boolean of whether it is a Directory	org.jboss.windup.reporting.rules.generation.techreport.IsFileADirectoryMethod
generateGUID	Generates a unique identifier as a String.	org.jboss.windup.reporting.freemarker.GenerateGUIDMethod
projectModelToSha1	Takes a parameter of type ProjectModel and returns the associated ApplicationReportIndexModel.	org.jboss.windup.rules.apps.java.reporting.freemarker.ProjectModelToSha1
isRulePhase	Returns true if the passed in object is an instance of RulePhase.	org.jboss.windup.reporting.freemarker.IsRulePhaseMethod
getProjectTraversal	Gets a ProjectModelTraversal for the given ProjectModel. An optional parameter specifies the traversal strategy ('only_once', 'all' or 'shared').	org.jboss.windup.reporting.freemarker.GetProjectTraversalMethod
getEffortCountForProjectByIssueCategory	Parameters are (GraphRewrite, ProjectModelTraversal, [recursive]) and returns Map where the key is the Severity and the value is the number of incidents of that severity.	org.jboss.windup.reporting.freemarker.GetEffortCountForProjectByIssueCategoryMethod
getWindupBrandName	Returns the name to use in place of the project name.	org.jboss.windup.reporting.freemarker.GetWindupBrandNameMethod
sortApplicationsList	Sorts the given list of ApplicationReportModel's by it's root filename or, for VIRTUAL apps, by the name.	org.jboss.windup.reporting.freemarker.SortApplicationsListMethod
getPrettyPathForFile	Takes a FileModel as a parameter and either the qualified name (if it is a Java file) or the path within the file's project.	org.jboss.windup.rules.apps.java.reporting.freemarker.filepath.GetPrettyPathForFile
getArchiveSHA1ToPathMapper	Gets an instance of a mapper for converting from the SHA1 hash of an archive to a List of file paths.	org.jboss.windup.reporting.freemarker.GetArchiveSHA1ToPathsMapperMethod
sha1Hex	Returns the SHA1 Hex hash for the provided FileModel.	org.jboss.windup.reporting.freemarker.Sha1HexMethod
getRuleExecutionResults	Takes a parameter of type AbstractRuleProvider and returns a List containing metadata related to the current Migration Toolkit for Applications by Red Hat execution.	org.jboss.windup.reporting.ruleexecution.GetRuleExecutionResults

3.7. フィードバックフォームを送信

Send feedback リンクをクリックして、レポートランディングページからこのフィードバックフォームにアクセスします。

このフォームでは、製品を評価し、好きなものについて話し、改善のための提案を行うことができます。

図3.16 フィードバックフォームを送信



第4章 CSV 形式でのレポートのエクスポート

MTA には、分類やヒントなどのレポートデータをローカルファイルシステムのフラットファイルにエクスポートする機能があります。export 関数は現在 CSV ファイル形式をサポートし、レポートデータはコンマ (,) で区切られたフィールドとして示されます。

CSV ファイルは、Microsoft Excel、OpenOffice Calc、LibreOffice Calc などのスプレッドシートソフトウェアでインポートおよび操作できます。スプレッドシートソフトウェアは、MTA レポートから結果データを並べ替え、分析、評価、および管理する機能を提供します。

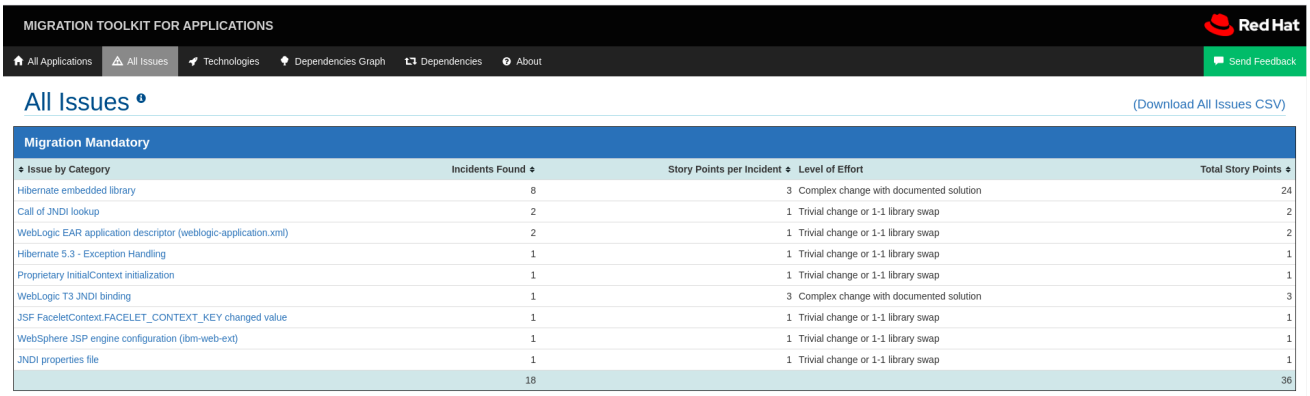
4.1. レポートのエクスポート

レポートを CSV ファイルとしてエクスポートするには、**--exportCSV** 引数を付けて MTA を実行します。CSV ファイルは、解析される各アプリケーションの、**--output** 引数で指定されたディレクトリーに作成されます。

解析されたすべてのアプリケーションにわたって発見されたすべての問題は、レポートのルートディレクトリーにエクスポートされる **AllIssues.csv** ファイルに含まれます。

アプリケーションレポートからのレポートへのアクセス
CSV レポートをエクスポートすると、問題レポートの CSV 問題をすべてダウンロードできます。これらの問題をダウンロードするには、Issues Report の **Download All Issues CSV** をクリックします。

図4.1 CSV ダウンロードに関する問題レポート



4.2. CSV ファイルのスプレッドシートプログラムへのインポート

1. スプレッドシートソフトウェア (例: Microsoft Excel) を起動します。
2. **File → Open** を選択します。
3. CSV でエクスポートされるファイルを参照し、これを選択します。
4. これで、スプレッドシートソフトウェアでデータを分析できるようになりました。

4.3. CSV データ構造について

CSV 形式の出力ファイルには、以下のデータフィールドが含まれます。

ルール ID

指定の項目を生成したルールの ID。

問題のタイプ

ヒント または **分類**

件名

classification または **hint** の件名。このフィールドは、特定の項目の問題を要約します。

説明

指定項目の問題の詳細な説明。

リンク

問題に関する追加情報を提供する URL。リンクは、リンクとリンクの説明という 2 つの属性で設定されます。

アプリケーション

この項目が生成されたアプリケーションの名前。

ファイル名

指定項目のファイルの名前。

ファイルパス

指定項目のファイルパス。

行

指定項目のファイルの行番号。

ストーリーポイント

特定の項目に割り当てられた、努力のレベルを表すストーリーポイントの数。

第5章 アプリケーションの MAVEN 化

MTA は、提供されるアプリケーションに基づいて Apache Maven プロジェクト構造を生成する機能を提供します。これにより、適切な依存関係を指定する必要な Maven Project Object Model (POM) ファイルを使用してディレクトリー構造が作成されます。

この機能は、プロジェクトの最終ソリューションを作成する予定はありません。これは、開始点を示し、アプリケーションに必要な依存関係および API を特定することを目的としています。プロジェクトでは、さらにカスタマイズが必要になる場合があります。

5.1. MAVEN プロジェクト構造の生成

MTA の実行時に **--mavenize** フラグを渡すことにより、提供されたアプリケーションの Maven プロジェクト構造を生成できます。

次の例では、jee-example-app-1.0.0.ear テストアプリケーションを使用して MTA を実行します。

```
$ <MTA_HOME>/bin/mta-cli --input /path/to/jee-example-app-1.0.0.ear --output /path/to/output --target eap:6 --packages com.acme org.apache --mavenize
```

これにより、Maven プロジェクト構造が **/path/to/output/mavenized** ディレクトリーに生成されます。

注記

--input 引数にコンパイルされたアプリケーションを指定する場合に限り、**--mavenize** オプションを使用できます。この機能は、ソースコードに対して MTA を実行する場合は利用できません。

--mavenizeGroupId オプションを使用して、POM ファイルに使用する **<groupId>** を指定することもできます。指定しないと、MTA はアプリケーションに適切な **<groupId>** の識別を試行するか、デフォルトで **com.mycompany.mavenized** になります。

5.2. MAVEN プロジェクト構造の確認

/path/to/output/mavenized/<APPLICATION_NAME>/ ディレクトリーには、以下の項目が含まれます。

- ルート **POM** ファイル。これは、最上位ディレクトリーの **pom.xml** ファイルです。
- BOM ファイル。これは、**-bom** で終わるディレクトリーの **POM** ファイルです。
- 1 つ以上のアプリケーション **POM** ファイル。各モジュールには、アーカイブの名前が付けられたディレクトリーに **POM** ファイルがあります。

サンプルの **jee-example-app-1.0.0.ear** アプリケーションは、WAR と複数の JAR を含む EAR アーカイブです。これらのアーティファクトごとに個別のディレクトリーが作成されます。以下は、このアプリケーション用に作成された Maven プロジェクト構造です。

```
/path/to/output/mavenized/jee-example-app/
    jee-example-app-bom/pom.xml
    jee-example-app-ear/pom.xml
    jee-example-services2-jar/pom.xml
```

```
jee-example-services-jar/pom.xml
jee-example-web-war/pom.xml
pom.xml
```

生成された各ファイルを確認し、プロジェクトに合わせてカスタマイズします。Maven POM ファイルの詳細は、Apache Maven ドキュメントの Introduction to the POM セクションを参照してください。

ルート POM ファイル
jee-example-app-1.0.0.ear アプリケーションのルート POM ファイルは **/path/to/output/mavenized/jee-example-app/pom.xml** にあります。このファイルは、すべてのプロジェクトモジュールのディレクトリーを特定します。

以下のモジュールは、サンプルの **jee-example-app-1.0.0.ear** アプリケーションのルート POM の一覧に記載されています。

```
<modules>
  <module>jee-example-app-bom</module>
  <module>jee-example-services2-jar</module>
  <module>jee-example-services-jar</module>
  <module>jee-example-web-war</module>
  <module>jee-example-app-ear</module>
</modules>
```

注記

必要に応じて、モジュールの一覧がプロジェクトの適切なビルド順序に一覧表示されるように、必ずモジュールの一覧の順番を変更してください。

また、ルート POM は Red Hat JBoss Enterprise Application Platform Maven リポジトリー を使用してプロジェクトの依存関係をダウンロードするように設定されます。

BOM ファイル
BOM (Bill of Materials) ファイルは、**-bom** で終わるディレクトリーに生成されます。サンプルアプリケーション **jee-example-app-1.0.0.ear** の場合、BOM ファイルは **/path/to/output/mavenized/jee-example-app/jee-example-app-bom/pom.xml** にあります。この BOM の目的は、プロジェクトで使用されるサードパーティーの依存関係のバージョンを 1 か所で定義することです。BOM の使用に関する詳細は、Apache Maven ドキュメントの Introduction to the dependencies mechanism セクションを参照してください。

以下の依存関係は、**jee-example-app-1.0.0.ear** アプリケーションの BOM に一覧表示されています。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.6</version>
    </dependency>
    <dependency>
      <groupId>commons-lang</groupId>
      <artifactId>commons-lang</artifactId>
      <version>2.5</version>
```

```
    </dependency>
  </dependencies>
</dependencyManagement>
```

アプリケーション POM ファイル
Maven が可能なアプリケーションモジュールには、POM ファイルが含まれる個別のディレクトリーがあります。ディレクトリー名には、アーカイブタイプに応じてアーカイブの名前と、接尾辞 **-jar**、**-war**、または **-ear** で終わるものが含まれます。

各アプリケーション POM ファイルには、以下を含むモジュールの依存関係が一覧表示されます。

- サードパーティーライブラリー
- Java EE API
- アプリケーションサブモジュール

たとえば、**jee-example-app-1.0.0.ear** EAR (**/path/to/output/mavenized/jee-example-app/jee-example-app-ear/pom.xml**) の POM ファイルには、以下の依存関係の一覧が記載されます。

```
<dependencies>
  <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.6</version>
  </dependency>
  <dependency>
    <groupId>org.jboss.seam</groupId>
    <artifactId>jee-example-web-war</artifactId>
    <version>1.0</version>
    <type>war</type>
  </dependency>
  <dependency>
    <groupId>org.jboss.seam</groupId>
    <artifactId>jee-example-services-jar</artifactId>
    <version>1.0</version>
  </dependency>
  <dependency>
    <groupId>org.jboss.seam</groupId>
    <artifactId>jee-example-services2-jar</artifactId>
    <version>1.0</version>
  </dependency>
</dependencies>
```

第6章 MTA パフォーマンスの最適化

MTA のパフォーマンスは、ハードウェア設定、アプリケーション内のファイルの数と種類、評価するアプリケーションのサイズと数、アプリケーションにソースコードとコンパイル済みコードのどちらが含まれているかなど、多くの要因に依存します。たとえば、10 MB を超えるファイルは、処理に時間がかかる場合があります。

一般に、MTA はクラスの逆コンパイルに約 40%、ルールの実行に 40% の時間を費やし、残りの時間は他のタスクの処理とレポートの生成に費やします。本セクションでは、MTA のパフォーマンスを改善するために実行できる内容を説明します。

6.1. アプリケーションのデプロイおよび実行

ハードウェアをアップグレードする前に、これらの提案を試してください。

- 可能であれば、アーカイブではなくソースコードに対して MTA を実行します。これにより、追加の JAR およびアーカイブをコンパイルする必要がなくなります。
- **<MTA_HOME>/bin/mta-cli** コマンドラインで **--packages** 引数を使用して、MTA によって評価されるパッケージのコンマ区切りリストを指定します。この引数を省略すると、MTA はすべてを逆コンパイルするため、パフォーマンスに大きな影響があります。
- **--excludeTags** 引数を指定すると、処理から除外できます。
- プロプライエタリーパッケージや含まれている依存関係などの不要なパッケージやファイルの逆コンパイルや解析は回避してください。
- 大規模なアプリケーションを分析する際に ulimit を増やします。Red Hat Enterprise Linux でこれを行う方法は、Red Hat Enterprise Linux でオープンファイルの数を制限する を参照してください。
- ラップトップまたはデスクトップマシンよりも優れたリソースを持つサーバーにアクセスできる場合は、そのサーバーで MTA を実行することを検討してください。

6.2. ハードウェアのアップグレード

上記のアプリケーションとコマンドラインの提案がパフォーマンスを改善しない場合は、ハードウェアをアップグレードが必要な場合があります。

- ラップトップまたはデスクトップよりも優れたリソースを持つサーバーにアクセスできる場合は、そのサーバーで MTA を実行することを検討してください。
- 逆コンパイルが必要な非常に大規模なアプリケーションには、大容量のメモリーが必要です。8 GB の RAM が推奨されます。これにより、3 ~ 4 GB の RAM が JVM で使用できるようになります。
- シングルコアまたはデュアルコアからクアッドコアの CPU プロセッサーへのアップグレードにより、パフォーマンスが向上します。
- ディスク領域と断片化はパフォーマンスに影響を及ぼす可能性があります。高速ディスク、特にソリッドステートドライブ (SSD) で、4 GB を超える最適化されたディスク領域があると、パフォーマンスが向上します。

6.3. パッケージおよびファイルを除外する MTA の設定

6.3.1. パッケージの除外

逆コンパイル中および分析中にパッケージを除外して、パフォーマンスを向上させることができます。これらのパッケージへの参照はアプリケーションのソースコードに残りますが、その参照を除外すると、プロプライエタリークラスの逆コンパイルと解析が回避されます。

定義された値に一致するパッケージはすべて除外されます。たとえば、**com.acme** を使用して、**com.acme.example** および **com.acme.roadrunner** の両方を除外できます。

以下のいずれかの方法でパッケージを除外できます。

- **--excludePackages** 引数の使用
- 無視される場所のいずれかに含まれるファイルでパッケージを指定します。各パッケージは別々の行に含める必要があり、このファイルは **.package-ignore.txt** で終わる必要があります。たとえば、**<MTA_HOME>/ignore/proprietary.package-ignore.txt** を参照してください。

6.3.2. ファイルの除外

MTA は、スキャンやレポートの生成中に、含まれるライブラリーや依存関係などの特定のファイルを除外できます。除外されたファイルは、無視されたいずれかの場所で拡張子が **.mta-ignore.txt** または **.windup-ignore.txt** ファイルで定義されています。

これらのファイルには、除外する名前の詳細を示す正規表現文字列が含まれ、1行に1つのファイルがリストされます。たとえば、ライブラリー **ant.jar** と、次の内容を含むファイルを使用して、**Example** で始まる Java ソースファイルを除外できます。

```
.*ant.jar
.*Example.*\.java
```

6.3.3. 除外の場所の検索

MTA は以下の場所を検索します。

- **~/.mta/ignore/**
- **~/.windup/ignore/**
- **<MTA_HOME>/ignore/**
- **--userIgnorePath** 引数で指定されたファイルおよびディレクトリー

この各ファイルは、除外するコンテンツのタイプに応じて、パッケージまたはファイルを除外するように指定されたルールに準拠する必要があります。

付録A 参考資料

A.1. MTA コマンドライン引数

以下は、利用可能な MTA コマンドライン引数の詳細な説明です。

注記

スクリプトから実行した場合など、プロンプトを表示せずに MTA コマンドを実行するには、以下の引数を使用する必要があります。

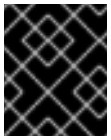
- **--batchMode**
- **--overwrite**
- **--input**
- **--target**

表A.1 MTA CLI 引数

引数	説明
--additionalClassPath	クラスパスに追加する追加の JAR ファイルまたはディレクトリーのスペース区切りリスト。逆コンパイルやその他の解析に使用できます。
--addonDir	指定したディレクトリーをカスタムアドオンリポジトリーとして追加します。
--analyzeKnownLibraries	アプリケーション内に埋め込まれた既知のソフトウェアアーティファクトを分析するフラグ。デフォルトでは、MTA はアプリケーションコードのみを分析します。  **注記** このオプションを使用すると実行時間が長くなり、多数の移行問題が報告される可能性があります。
--batchMode	MTA を非対話モードで実行すべきように指定するフラグ。確認をプロンプトなしで実行します。このモードは、コマンドラインに渡さないパラメーターのデフォルト値を取ります。
--debug	デバッグモードで MTA を実行するフラグ。
--disableTattletale	Tattletale レポートの生成を無効にするフラグ。**enableTattletale** と **disableTattletale** の両方が true に設定されていると、**disableTattletale** は無視されます。また、Tatletale レポートは引き続き生成されます。

引数	説明
--discoverPackages	入力バイナリーアプリケーションで利用可能なパッケージをすべて表示するフラグ。
--enableClassNotFoundAnalysis	クラスパスで利用できない Java ファイルの分析を有効にするフラグ。分析時に一部のクラスが利用できない場合は、これは使用しないでください。
--enableCompatibleFilesReport	Compatible Files レポートの生成を有効にするフラグ。問題が検出されない状態ですべてのファイルを処理するため、このレポートには大規模なアプリケーションの処理に時間がかかる場合があります。
--enableTattletale	各アプリケーションの Tattletale レポートの生成を有効にするフラグ。このオプションは、**eap** が含まれるターゲットにある場合にデフォルトで有効になります。**enableTattletale** と **disableTattletale** の両方が true に設定されていると、**disableTattletale** は無視されます。また、Tatletale レポートは引き続き生成されます。
--excludePackages	評価から除外するパッケージのスペース区切りの一覧。たとえば、**com.mycompany.commonutilities** と入力すると、パッケージ名が **com.mycompany.commonutilities** で始まるクラスをすべて除外します。
--excludeTags	除外するタグのスペースで区切られた一覧。指定されている場合は、これらのタグを持つルールは処理されません。タグの全一覧を表示するには、**--listTags** 引数を使用します。
--explodedApp	指定された入力ディクショナリーに 1 つののアプリケーションのソースファイルが含まれていることを示すフラグ。
--exportCSV	レポートデータをローカルファイルシステムの CSV ファイルにエクスポートするフラグ。MTA は、**--output** 引数で指定されたディレクトリーにファイルを作成します。CSV ファイルは、データの操作および分析のためにスプレッドシートプログラムにインポートできます。
--help	MTA ヘルプメッセージを表示します。
--immutableAddonDir	指定したディレクトリーを、読み取り専用のアドオンリポジトリーとして追加します。
--includeTags	使用するタグのスペースで区切られたリスト。指定されると、これらのタグを持つルールのみが処理されます。タグの全一覧を表示するには、**--listTags** 引数を使用します。

引数	説明
--input	分析する 1 つ以上のアプリケーションを含むファイルまたはディレクトリーへのパスのスペースで区切られたリスト。この引数は必須です。
--install	インストールするアドオンを指定します。構文は **<GROUP_ID>:<ARTIFACT_ID>[:<VERSION>]** です。たとえば、**--install core-addon-x** または **--install org.example.addon:example:1.0.0** です。
--keepWorkDirs	グラフデータベースや展開されたアーカイブファイルなどの一時作業ファイルを削除しないように MTA に指示するフラグ。これはデバッグに役立ちます。
--list	インストールされたアドオンを一覧表示するフラグ。
--listSourceTechnologies	利用可能なすべてのソーステクノロジーを一覧表示するフラグ。
--listTags	使用可能なタグをすべて表示するフラグ。
--listTargetTechnologies	利用可能なすべてのターゲットテクノロジーを一覧表示するフラグ。
--mavenize	アプリケーションの構造および内容に基づいて Maven プロジェクトディレクトリー構造を作成するフラグ。これにより、適切な Java EE API とプロジェクトモジュール間の正しい依存関係を使用して **pom.xml** ファイルが作成されます。**--mavenizeGroupId** オプションも参照してください。
--mavenizeGroupId	**--mavenize** オプションと使用すると、生成されるすべての **pom.xml** ファイルは **<groupId>** に提供された値を使用します。この引数を省略すると、MTA はアプリケーションに基づいて適切な **<groupId>** を判別しようとします。または、デフォルトでは **com.mycompany.mavenized** に設定されます。
--online	フラグは、それを必要とする機能のネットワークアクセスを許可します。現在、外部リソースに対する XML スキーマのみの検証は、インターネットアクセスに依存します。これには、パフォーマンスの低下があることに注意してください。
--output	MTA が生成したレポート情報を出力するディレクトリーへのパスを指定します。

引数	説明
--overwrite	**--output** で指定された既存の出力ディレクトリーを強制的に削除するためのフラグ。この引数を指定せず、**-output** ディレクトリーが存在する場合は、内容を上書きするかどうかを選択するように求められます。  **重要** 重要な情報を含むレポート出力ディレクトリーを上書きしないでください。
--packages	MTA によって評価されるパッケージのスペース区切りの一覧。この引数を使用することは強く推奨されます。
--remove	指定したアドオンを削除します。構文は **<GROUP_ID>:<ARTIFACT_ID>[:<VERSION>]** です。例: **--remove core-addon-x** または **--remove org.example.addon:example:1.0.0**
--skipReports	HTML レポートが生成されないことを示すフラグ。この引数の一般的な用途は、**--exportCSV** を使用してレポートデータを CSV ファイルにエクスポートする場合です。
--source	移行元の 1 つ以上のソーステクノロジー、サーバー、プラットフォーム、またはフレームワークのスペース区切りの一覧。この引数は、**--target** 引数とともに、使用されるルールセットを判断するのに役立ちます。**--listSourceTechnologies** 引数を使用して、利用可能なソースを一覧表示します。
--sourceMode	評価するアプリケーションに、コンパイルされたバイナリーではなくソースファイルが含まれていることを示すフラグ。
--target	移行先の 1 つ以上のターゲットテクノロジー、サーバー、プラットフォーム、またはフレームワークのスペース区切りの一覧。この引数は、**--source** 引数とともに、使用されるルールセットを判断するのに役立ちます。**--listTargetTechnologies** 引数を使用して、利用可能なターゲットを一覧表示します。
--userIgnorePath	**${user.home}/.mta/ignore/** の他に、無視すべきファイルを MTA が特定できるように場所を指定します。
--userLabelsDirectory	MTA がカスタムターゲットランタイムラベルを探す場所を指定します。値には、複数または単数のラベルファイルを含むディレクトリーを使用できます。Target Runtime Label ファイルは、**.windup.label.xml** 接尾辞または **.mta.label.xml** 接尾辞を使用する必要があります。同梱されている Target Runtime Labels は **$MTA_HOME/rules/migration-core/core.windup.label.xml** 内で定義されます。

引数	説明
--userRulesDirectory	**<MTA_HOME>/rules/** および **${user.home}/.mta/rules/** に加えて、MTA がカスタム MTA ルールを探す場所を指定します。値は、単数または複数のルールセットファイルを含むディレクトリーです。ルールセットファイルには **.windup.xml** 接尾辞または **.mta.xml** 接尾辞を使用する必要があります。
--version	MTA バージョンを表示します。

A.1.1. 入力の指定

分析する 1 つ以上のアプリケーションを含むファイルまたはディレクトリーへのパスのスペースで区切られたリスト。この引数は必須です。

使用方法

```
--input <INPUT_ARCHIVE_OR_DIRECTORY> [...]
```

--input 引数に指定された入力ファイルタイプがファイルであるかディクショナリーであるかに応じて、指定された追加の引数に応じて次のように評価されます。

ディレクトリー

--explodedApp	--sourceMode	引数なし
ディレクトリーは 1 つのアプリケーションとして評価されます。	ディレクトリーは 1 つのアプリケーションとして評価されます。	各サブディレクトリーはアプリケーションとして評価されます。

ファイル

--explodedApp	--sourceMode	引数なし
引数は無視されます。ファイルは 1 つのアプリケーションとして評価されます。	ファイルは圧縮プロジェクトとして評価されます。	ファイルは 1 つアプリケーションとして評価されます。

A.1.2. 出力ディレクトリーの指定

MTA が生成したレポート情報を出力するディレクトリーへのパスを指定します。

使用方法

```
--output <OUTPUT_REPORT_DIRECTORY>
```

- 省略すると、レポートは **<INPUT_ARCHIVE_OR_DIRECTORY>.report** ディレクトリーに生成されます。

- 出力ディレクトリーが存在する場合は、次のメッセージが表示されます (デフォルトは N)。

```
Overwrite all contents of "/home/username/<OUTPUT_REPORT_DIRECTORY>" (anything already in the directory will be deleted)? [y,N]
```

ただし、**--overwrite** 引数を指定すると、MTA はディレクトリーの削除と再作成を続行します。詳細は、この引数の説明を参照してください。

A.1.3. ソーステクノロジーの設定

移行元の 1 つ以上のソーステクノロジー、サーバー、プラットフォーム、またはフレームワークのスペース区切りの一覧。この引数は、**--target** 引数とともに、使用されるルールセットを判断するのに役立ちます。**--listSourceTechnologies** 引数を使用して、利用可能なソースを一覧表示します。

使用方法

```
--source <SOURCE_1> <SOURCE_2>
```

--source 引数は、Maven バージョン範囲の構文 に続くバージョンサポートを提供するようになりました。これにより、指定されたバージョンに一致するルールセットのみを実行するように MTA が指示されます。例: **--source eap:5**

警告

JBoss EAP に移行する場合、バージョン (例: **eap:6**) を指定してください。**eap** のみを指定すると、移行パスに関連しないものを含め、すべてのバージョンの JBoss EAP にルールセットが実行されます。

JBoss EAP バージョンに適した **Migration Toolkit for Applications の概要**の サポートされる移行パス を参照してください。

A.1.4. ターゲットテクノロジーの設定

移行先の 1 つ以上のターゲットテクノロジー、サーバー、プラットフォーム、またはフレームワークのスペース区切りの一覧。この引数は、**--source** 引数とともに、使用されるルールセットを判断するのに役立ちます。このオプションを指定しないと、ターゲットを選択するように求められます。**--listTargetTechnologies** 引数を使用して、利用可能なターゲットを一覧表示します。

使用方法

```
--target <TARGET_1> <TARGET_2>
```

--target 引数は、Maven バージョン範囲の構文 に続くバージョンサポートを提供するようになりました。これにより、指定されたバージョンに一致するルールセットのみを実行するように MTA が指示されます。例: **--target eap:7**

警告

JBoss EAP に移行する場合は、必ずバージョンをターゲットに指定してください (例: **eap:6**)。**eap** のみを指定すると、移行パスに関連しないものを含め、すべてのバージョンの JBoss EAP にルールセットが実行されます。

JBoss EAP バージョンに適した **Migration Toolkit for Applications の概要**の サポートされる移行パス を参照してください。

A.1.5. パッケージの選択

MTA によって評価されるパッケージのスペース区切りの一覧。この引数を使用することは強く推奨されます。

使用方法

```
--packages <PACKAGE_1> <PACKAGE_2> <PACKAGE_N>
```

- 多くの場合、関心があるのは、カスタムアプリケーションクラスパッケージの評価で、標準の Java EE パッケージやサードパーティーのパッケージではありません。**<PACKAGE_N>** 引数はパッケージ接頭辞で、すべてのサブパッケージがスキャンされます。たとえば、パッケージ **com.mycustomapp** および **com.myotherapp** をスキャンするには、コマンドラインで **--packages com.mycustomapp com.myotherapp** 引数を使用します。
- **org.apache** などの標準の Java EE サードパーティーソフトウェアにパッケージ名を指定することはできますが、通常は移行作業に影響しないため、追加しないことが推奨されます。

警告

--packages 引数を省略すると、アプリケーションのすべてのパッケージがスキャンされ、パフォーマンスに影響を及ぼす可能性があります。

A.2. サポート対象のテクノロジータグ

以下のテクノロジープレビュータグは MTA 5.3.0 でサポートされています。

- 0MQ Client (組み込み)
- 3scale (組み込み)
- Acegi Security (組み込み)
- AcrIS Security (組み込み)
- ActiveMQ (組み込み)

- Airframe (組み込み)
- Airlift Log Manager (組み込み)
- AKKA JTA (組み込み)
- Akka Testkit (組み込み)
- Amazon SQS Client (組み込み)
- AMQP Client (組み込み)
- Anakia (組み込み)
- AngularFaces (組み込み)
- ANTLR StringTemplate (組み込み)
- AOP Alliance (組み込み)
- Apache Accumulo Client
- Apache Aries (組み込み)
- Apache Axis (組み込み)
- Apache Axis2 (組み込み)
- Apache Camel (組み込み)
- Apache Commons JCS (組み込み)
- Apache Commons Logging (組み込み)
- Apache Commons Validator (組み込み)
- Apache CXF (組み込み)
- Apache Flume (組み込み)
- Apache Geronimo (組み込み)
- Apache Hadoop (組み込み)
- Apache HBase Client
- Apache Ignite (組み込み)
- Apache Karaf (組み込み)
- Apache Log4J (組み込み)
- Apache Mahout (組み込み)
- Apache Meecrowave JTA (組み込み)
- Apache Santuario (組み込み)

- Apache Shiro (組み込み)
- Apache Sirona JTA (組み込み)
- Apache Struts (組み込み)
- Apache Synapse (組み込み)
- Apache Tapestry (組み込み)
- Apache Wicket (組み込み)
- Apiman (組み込み)
- Arquillian (組み込み)
- AspectJ (組み込み)
- Atomikos JTA (組み込み)
- Avalon Logkit (組み込み)
- Axion Driver
- BabbageFaces (組み込み)
- Bean Validation
- BeanInject (組み込み)
- Blaze (組み込み)
- Blitz4j (組み込み)
- BootsFaces (組み込み)
- Bouncy Castle (組み込み)
- ButterFaces (組み込み)
- Cache API (組み込み)
- Cactus (組み込み)
- Camel Messaging Client (組み込み)
- Camunda (組み込み)
- Cassandra Client
- CDI
- CDI (組み込み)
- Cfg Engine (組み込み)
- Chunk Templates (組み込み)

- Cloudera (組み込み)
- Clustering EJB
- Clustering Web Session
- Coherence (組み込み)
- Common Annotations
- Composite Logging JCL (組み込み)
- Concordion (組み込み)
- Cucumber (組み込み)
- Dagger (組み込み)
- DbUnit (組み込み)
- Debugging Support for Other Languages
- Decompiled Java File
- Demoiselle JTA (組み込み)
- Derby Driver
- Drools (組み込み)
- DVSL (組み込み)
- Dynacache (組み込み)
- EAR
- Easy Rules (組み込み)
- EasyMock (組み込み)
- EclipseLink (組み込み)
- EJB
- EJB XML
- Ehcache (組み込み)
- Elasticsearch (組み込み)
- Enterprise Web Services
- Entity Bean
- EtlUnit (組み込み)
- Everit JTA (組み込み)

- Evo JTA (組み込み)
- FreeMarker (組み込み)
- Geronimo JTA (組み込み)
- GFC Logging (組み込み)
- GIN (組み込み)
- GlassFish JTA (組み込み)
- Google Guice (組み込み)
- Grails (組み込み)
- Grapht DI (組み込み)
- Guava Testing (組み込み)
- GWT (組み込み)
- H2 Driver
- Hamcrest (組み込み)
- Handlebars (組み込み)
- HavaRunner (組み込み)
- Hazelcast (組み込み)
- Hdiv (組み込み)
- Hibernate (組み込み)
- Hibernate Cfg
- Hibernate Mapping
- Hibernate OGM (組み込み)
- HighFaces (組み込み)
- HornetQ Client (組み込み)
- HSQLDB Driver
- HTTP Client (組み込み)
- HttpUnit (組み込み)
- ICEfaces (組み込み)
- Ickenham (組み込み)
- Ignite JTA (組み込み)

- Ikasan (組み込み)
- iLog (組み込み)
- Infinispan (組み込み)
- Injekt for Kotlin (組み込み)
- Iroh (組み込み)
- Istio (組み込み)
- JACC
- Jamon (組み込み)
- Jasypt (組み込み)
- Java EE
- Java EE Batch
- Java EE Batch API
- Java EE JSON-P
- Java EE Security
- Java Source
- Java Transaction API (組み込み)
- JavaMail
- Javax Inject (組み込み)
- JAX-RPC
- JAX-RS
- JAX-WS
- JAXB
- JAXR
- JayWire (組み込み)
- JBehave (組み込み)
- JBoss Cache (組み込み)
- JBoss EJB XML
- JBoss logging (組み込み)
- JBoss Transactions (組み込み)

- JBoss Web XML
- JBossMQ Client (組み込み)
- JBPM (組み込み)
- JCA
- Jcabi Log (組み込み)
- JCache (組み込み)
- JCunit (組み込み)
- JDBC (組み込み)
- JDBC datasources
- JDBC XA datasources
- Jersey (組み込み)
- Jetbrick Template (組み込み)
- Jetty (組み込み)
- JFreeChart (組み込み)
- JFunk (組み込み)
- JMock (組み込み)
- JMockit (組み込み)
- JMS
- JMS Connection Factory
- JMS Queue
- JMS Topic
- JMustache (組み込み)
- JPA
- JPA entities
- JPA Matchers (組み込み)
- JPA named queries
- JPA XML
- JSecurity (組み込み)
- JSF (組み込み)

- JSF Page
- JSilver (組み込み)
- JSON-B
- JSP Page
- JSTL (組み込み)
- JTA
- Jukito (組み込み)
- JUnit (組み込み)
- Ka DI (組み込み)
- Keyczar (組み込み)
- Kibana (組み込み)
- KLogger (組み込み)
- Kodein (組み込み)
- Kotlin Logging (組み込み)
- KouInject (組み込み)
- KumuluzEE JTA (組み込み)
- LevelDB Client
- Liferay (組み込み)
- LiferayFaces (組み込み)
- Lift JTA (組み込み)
- Log.io (組み込み)
- Log4s (組み込み)
- Logback (組み込み)
- Logging to file system
- Logging to Socket Handler
- Logging Utils (組み込み)
- Logstash (組み込み)
- Lumberjack (組み込み)
- Macros (組み込み)

- Manifest
- MapR (組み込み)
- Maven XML
- MckoiSQLDB Driver
- MEJB
- Memcached client (組み込み)
- Message (MDB)
- Micro DI (組み込み)
- Microsoft SQL ドライバー
- MinLog (組み込み)
- Mixer (組み込み)
- Mockito (組み込み)
- MongoDB Client
- Monolog (組み込み)
- Morphia
- MRules (組み込み)
- Mule (組み込み)
- Mule Functional Test Framework (組み込み)
- MultithreadedTC (組み込み)
- Mycontainer JTA (組み込み)
- MyFaces (組み込み)
- MySQL Driver
- Narayana Arjuna (組み込み)
- Needle (組み込み)
- Neo4j (組み込み)
- NLOG4J (組み込み)
- Nuxeo JTA/JCA (組み込み)
- OACC (組み込み)
- OAUTH (組み込み)

- OCPsoft Logging Utils (組み込み)
- OmniFaces (組み込み)
- OpenFaces (組み込み)
- OpenPojo (組み込み)
- OpenSAML (組み込み)
- OpenWS (組み込み)
- OPS4J Pax Logging Service (組み込み)
- Oracle ADF (組み込み)
- Oracle DB Driver
- Oracle Forms (組み込み)
- Orion EJB XML
- Orion Web XML
- Oscache (組み込み)
- OTR4J (組み込み)
- OW2 JTA (組み込み)
- OW2 Log Util (組み込み)
- OWASP CSRF Guard (組み込み)
- OWASP ESAPI (組み込み)
- Peaberry (組み込み)
- Pega (組み込み)
- Persistence units
- Petals EIP (組み込み)
- PicketBox (組み込み)
- PicketLink (組み込み)
- PicoContainer (組み込み)
- Play (組み込み)
- Play Test (組み込み)
- Plexus Container (組み込み)
- Polyforms DI (組み込み)

- Portlet (組み込み)
- PostgreSQL Driver
- PowerMock (組み込み)
- PrimeFaces (組み込み)
- Properties
- Qpid Client (組み込み)
- RabbitMQ Client (組み込み)
- RandomizedTesting Runner (組み込み)
- Resource Adapter (組み込み)
- REST Assured (組み込み)
- Restito (組み込み)
- RichFaces (組み込み)
- RMI
- RocketMQ Client (組み込み)
- Rythm Template Engine (組み込み)
- SAML (組み込み)
- Scalate (組み込み)
- Scaldi (組み込み)
- Scribe (組み込み)
- Seam (組み込み)
- ServiceMix (組み込み)
- Servlet
- ShiftOne (組み込み)
- Silk DI (組み込み)
- SLF4J (組み込み)
- Snippetory Template Engine (組み込み)
- SNMP4J (組み込み)
- SOAP (SAAJ)
- Spark (組み込み)

- Specsy (組み込み)
- Spock (組み込み)
- Spring (組み込み)
- Spring Batch (組み込み)
- Spring Boot (組み込み)
- Spring Data (組み込み)
- Spring Integration (組み込み)
- Spring Messaging Client (組み込み)
- Spring MVC (組み込み)
- Spring Security (組み込み)
- Spring Test (組み込み)
- Spring Transactions (組み込み)
- Spring XML
- SQLite Driver
- SSL (組み込み)
- Stateful (SFSB)
- Stateless (SLSB)
- Sticky Configured (組み込み)
- Stripes (組み込み)
- SubCut (組み込み)
- Swagger (組み込み)
- SwarmCache (組み込み)
- SwitchYard (組み込み)
- Syringe (組み込み)
- Talend ESB (組み込み)
- Teiid (組み込み)
- TensorFlow (組み込み)
- Test Interface (組み込み)
- TestNG (組み込み)

- Thymeleaf (組み込み)
- TieFaces (組み込み)
- tinylog (組み込み)
- Tomcat (組み込み)
- Tornado Inject (組み込み)
- Trimou (組み込み)
- Trunk JGuard (組み込み)
- Twirl (組み込み)
- Twitter Util Logging (組み込み)
- UberFire (組み込み)
- Unirest (組み込み)
- Unitils (組み込み)
- Vaadin (組み込み)
- Velocity (組み込み)
- Vlad (組み込み)
- Water Template Engine (組み込み)
- Web XML
- WebLogic Web XML
- Webmacro (組み込み)
- WebSphere EJB
- WebSphere EJB Ext
- WebSphere Web XML
- WebSphere WS Binding
- WebSphere WS Extension
- Weka (組み込み)
- Weld (組み込み)
- WF Core JTA (組み込み)
- Winter (組み込み)
- WS Metadata

- WSDL (組み込み)
- WSO2 (組み込み)
- WSS4J (組み込み)
- XACML (組み込み)
- XFire (組み込み)
- XMLUnit (組み込み)
- Zbus Client (組み込み)

A.3. ルールのストーリーポイントについて

A.3.1. ストーリーポイントとは

ストーリーポイント は、アジャイルソフトウェア開発で一般的に使用される抽象メトリクスで、機能や変更を実装するのに必要な **作業量** を予測します。

Migration Toolkit for Applications はストーリーポイントを使用して、特定のアプリケーションコンストラクトとアプリケーション全体を移行するために必要な作業のレベルを表現します。必ずしも工数に変換される訳ではありませんが、この値はタスク全体で一貫性を持たせる必要があります。

A.3.2. ルールにおけるストーリーポイントの見積方法

ルールのストーリーポイントの作業レベルを見積もることは複雑です。以下は、ルールに必要な作業レベルを見積もる際に MTA が使用する一般的なガイドラインです。

作業レベル	Story Points	説明
Information	0	移行の優先度が非常に低いか、優先度のない情報警告。
Trivial	1	移行は、些細な変更または単純なライブラリースワップであり、API の変更はないか、最小限となります。
Complex	3	移行タスクに必要な変更は複雑ですが、解決策が文書化されています。
Redesign	5	移行タスクでは、API が大幅に変更され、再設計または完全なライブラリーの変更が必要になります。
Rearchitecture	7	移行には、コンポーネントまたはサブシステムの完全な再アーカイブが必要です。
Unknown	13	移行ソリューションは不明なため、完全な再書き込みが必要になる場合があります。

A.3.3. タスクカテゴリー

作業量レベルに加えて、移行タスクを分類してタスクの重大度を示すことができます。移行作業の優先順位付けに役立つ問題をグループ化するために、以下のカテゴリーが使用されます。

Mandatory

移行を成功させるには、タスクを完了する必要があります。変更が行われないと、生成されるアプリケーションはビルドまたは実行に成功しません。たとえば、ターゲットプラットフォームでサポートされないプロプライエタリー API の置き換え例が含まれます。

Optional

移行タスクが完了しない場合、アプリケーションは動作しますが、結果が最適になるとは限りません。移行時に変更が行われない場合は、移行の完了後すぐにスケジュールに配置することが推奨されます。これには、EJB 2.x コードの EJB 3 へのアップグレードが挙げられます。

Potential

移行プロセス中にタスクを調べる必要があります。しかし、移行を成功させるためにタスクが必須かどうかを判断するのに十分な詳細情報がありません。これの例は、直接互換性のあるタイプがないサードパーティーのプロプライエタリタイプの移行です。

Information

タスクは、特定のファイルの存在を通知するために含まれています。これらは、モダナイゼーション作業の一部として検証または変更する必要がある場合がありますが、通常は変更が必要ありません。これには、ロギング依存関係または Maven **pom.xml** があります。

タスクの分類に関する詳細は、カスタムルールカテゴリーの使用 を参照してください。

A.4. 関連情報

A.4.1. 関わること

Migration Toolkit for Applications が、自分のものを含むほとんどのアプリケーション設定とサーバー設定に対応できるように、以下の項目のいずれかを支援できます。

- jboss-migration-feedback@redhat.com にメールを送信し、MTA 移行ルールが対象とすべき内容をご連絡ください。
- 移行ルールをテストするためのアプリケーションの例を指定します。
- 移行が困難なアプリケーションコンポーネントおよび問題の領域を特定してください。
  - これらの問題がある移行領域について簡単な説明を記入する。
  - 問題の移行領域を解決する方法を説明する簡単な概要を記述します。
- アプリケーションで Migration Toolkit for Applications を試行します。発生している問題を必ず報告してください。
- Migration Toolkit for Applications ルールリポジトリーへの貢献。
  - Migration Toolkit for Applications ルールを記述して、移行プロセスを識別または自動化します。
  - 新規ルールのテストを作成します。
  - 詳細は、**ルール開発ガイド** を参照してください。
- プロジェクトのソースコードへの貢献。

- コアルールを作成します。
- MTA のパフォーマンスまたは効率が向上します。
- 環境の設定およびプロジェクトの設定方法に関する詳細は、**Core Development Guide** を参照してください。

あらゆるレベルの貢献が大きく評価されます。

A.4.2. リソース

- MTA フォーク: https://developer.jboss.org/en/windup
- MTA Jira 問題トラッカー
  - コア MTA: https://issues.redhat.com/projects/WINDUP
  - MTA ルール: https://issues.redhat.com/projects/WINDUPRULE
- MTA メーリングリスト: jboss-migration-feedback@redhat.com
- MTA IRC チャネル: Server FreeNode (**irc.freenode.net**) (チャンネル **#windup** (transcripts))

A.4.3. 問題の報告

MTA は Jira を問題追跡システムとして使用します。MTA の実行で問題が発生した場合は、Jira issue を作成してください。

改訂日時：2022-12-10 15:43:53 +1000