Migration Toolkit for Runtimes 1.2

CLI 指南

了解如何使用 Migration Toolkit for Runtimes CLI 迁移应用程序。

Last Updated: 2024-06-13

Migration Toolkit for Runtimes 1.2 CLI 指南

了解如何使用 Migration Toolkit for Runtimes CLI 迁移应用程序。

法律通告

摘要

本指南论述了如何使用 Migration Toolkit for Runtimes CLI 简化 Java 应用程序的迁移。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 简介

1.1. 关于 CLI 指南

本指南适用于希望使用 Migration Toolkit for Runtimes (MTR) 迁移 Java 应用程序或其他组件的工程师、顾问和其他组件。它介绍了如何安装和运行 CLI，查看生成的报告，并利用附加功能。

1.2. 关于 MIGRATION TOOLKIT FOR RUNTIMES

什么是 Migration Toolkit for Runtimes?

Migration Toolkit for Runtimes (MTR) 是一个可扩展、可自定义的基于规则的工具，简化了 Java 应用程序的迁移和现代化。

MTR 检查应用程序工件，包括项目源目录和应用程序存档，然后生成 HTML 报告突出显示需要更改的区域。MTR 支持许多迁移路径，包括以下示例：

- 升级至 Red Hat JBoss Enterprise Application Platform 的最新版本
- 从 Oracle WebLogic 或 IBM WebSphere Application Server 迁移到 Red Hat JBoss Enterprise Application Platform
- 容器化应用程序并使之成为云就绪
- 从 Java Spring Boot 迁移到 Quarkus
- 从 Oracle JDK 更新至 OpenJDK
- 从 OpenJDK 8 升级到 OpenJDK 11
- 从 OpenJDK 11 升级到 OpenJDK 17
- 从 OpenJDK 17 升级到 OpenJDK 21
- 将 EAP Java 应用程序迁移到 Azure
- 将 Spring Boot Java 应用程序迁移到 Azure

有关用例和迁移路径的更多信息，请参阅开发人员网页的 MTR。

Migration Toolkit for Runtimes 如何简化迁移？

Migration Toolkit for Runtimes 会查找常见资源和在迁移应用程序时的已知问题。它为应用程序使用的技术提供了高级视图。

MTR 生成详细的报告，评估迁移或现代化路径。此报告可帮助您估算大型项目所需的工作量，并减少涉及的工作。

如何了解更多信息？

请参阅 Migration Toolkit for Runtimes 简介，了解有关 Migration Toolkit for Runtimes 中的功能、支持的配置、系统要求以及可用工具的更多信息。

1.3. MTR CLI

CLI 是 Migration Toolkit for Runtimes 的 Migration Toolkit for Runtimes 中的命令行工具，可用于评估和优先排序应用程序的迁移和现代化工作。它提供了大量报告，突出显示分析而无需使用其他工具。CLI 包括广泛的自定义选项。通过使用 CLI，您可以调整 MTR 分析选项或与外部自动化工具集成。

第 2 章 安装和运行 CLI

2.1. 安装 CLI

您可以在 Linux、Windows 或 macOS 操作系统上安装 CLI。

先决条件

以下是 Migration Toolkit for Runtimes (MTR)安装的先决条件：

- 安装了 Java Development Kit (JDK)。MTR 支持以下 JDK：
  - OpenJDK 11
  - OpenJDK 17
  - Oracle JDK 11
  - Oracle JDK 17
  - Eclipse Temurin™ JDK 11
  - Eclipse Temurin™ JDK 17
- 8 GB RAM
- macOS 安装： **maxproc** 的值必须是 **2048** 或更高版本。

流程

1. 进入 MTR Download 页面 并下载 **Migration Toolkit CLI** 文件。
2. 将 **.zip** 文件提取到您选择的目录。
   在本指南中遇到 **<MTR_HOME>** 时，将其替换为 MTR 安装的实际路径。

2.2. 运行 CLI

您可以针对您的应用程序运行 MTR。

流程

1. 打开一个终端，进入到 **<MTR_HOME>/bin/** 目录。
2. 执行 **windup-cli** 脚本，或为 Windows 执行 **windup-cli.bat**，并指定适当的参数：

```
$ ./windup-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** ：要评估的应用程序。
   - **--output** ：所生成的报告的输出目录。
   - **--source** ：应用程序迁移的源技术。

- **--target** ：应用程序迁移的目标技术。
- **--packages** ：要评估的软件包。强烈建议您使用这个参数提高性能。

3. 访问报告。

2.2.1. MTR 命令示例

在应用程序存档中运行 MTR
以下命令分析 jee-example-app-1.0.0.ear 示例 EAR 存档从 JBoss EAP 5 迁移到 JBoss EAP 7 的 **com.acme** 和 **org.apache** 软件包：

```
$ <MTR_HOME>/bin/windup-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
```

在源代码中运行 MTR
以下命令分析 seam-booking-5.2 示例源代码中的 **org.jboss.seam** 软件包，以迁移到 JBoss EAP 6。

```
$ <MTR_HOME>/bin/windup-cli --sourceMode --input /path/to/seam-booking-5.2/ \

    --output /path/to/report-output/ --target eap:6 --packages org.jboss.seam
```

运行云就绪规则
以下命令分析 jee-example-app-1.0.0.ear 示例 EAR 归档（用于迁移到 JBoss EAP 7）的 **com.acme** 和 **org.apache** 软件包。它还评估了云就绪情况：

```
$ <MTR_HOME>/windup-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
```

覆盖 MTR 属性
要覆盖默认的 *Fernflower* decompiler，请在命令行上传递 **-Dwindup.decompiler** 参数。例如，要使用 *Procyon* decompiler，请使用以下语法：

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

2.2.2. MTR CLI Bash 补全

MTR CLI 提供了一个选项，可为 Linux 系统启用 Bash 补全功能，允许在输入命令时按 Tab 键自动完成 MTR 命令行参数。例如，在启用 Bash 完成时，输入以下内容会显示可用参数的列表：

```
$ <MTR_HOME>/bin/windup-cli [TAB]
```

启用 Bash 完成
要为当前 shell 启用 Bash 完成，请执行以下命令：

```
$ source <MTR_HOME>/bash-completion/windup-cli
```

启用持久性 Bash 完成
以下命令允许 Bash 完成在重启后保留：

- 要在系统重启后为特定用户启用 Bash 完成，请在该用户的 **~/.bashrc** 文件中包括以下行：

```
source <MTR_HOME>/bash-completion/windup-cli
```

- 要为系统重启后所有用户启用 Bash 完成，请以 root 用户身份将运行时 CLI 的 Migration Toolkit for Runtimes CLI completion 文件复制到 **/etc/bash_completion.d/** 目录中。

```
# cp <MTR_HOME>/bash-completion/windup-cli /etc/bash_completion.d/
```

2.2.3. 访问 MTR 帮助

要查看 **windup-cli** 命令可用参数的完整列表，请打开终端，进入 **<MTR_HOME>** 目录，并执行以下命令：

```
$ <MTR_HOME>/bin/windup-cli --help
```

2.2.4. 使用 OpenRewrite methods

重要

OpenRewrite method 支持仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

您可以通过 MTR CLI 使用 OpenRewrite 方法来重构 Java 应用程序的源代码。

例如，OpenRewrite method **org.jboss.windup.JavaxToJakarta** 将导入的 **javax** 软件包重命名为其对于的 **jakarta**。

流程

1. 运行 **windup-cli**，指定方法名称、配置文件的路径和应用程序：

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

   - **"-DactiveRecipes=<recipe name>"**: 指定 OpenRewrite recipe，例如 **org.jboss.windup.JavaxToJakarta**。
   - **--input** ：指定要重构的应用程序。该应用必须是包含 Maven 项目对象模型(POM) XML 文件 **pom.xml** 的源代码项目的顶部。
   - **-Drewrite.configLocation=<path/to/rewrite.yaml>** : 要使用的 **rewrite.yaml** 配置文件的位置。提供的 **rewrite.yaml** 配置文件位于 **<MTR_HOME>/rules/openrewrite** 子文件夹中，例如 **" -Drewrite.configLocation= <MTR_HOME>/rules/openrewrite/jakarta/javax/imports/rewrite.yaml"**。

- **"-DactiveRecipes=<recipe name>"**: 指定 OpenRewrite recipe，例如 **org.jboss.windup.JavaxToJakarta**。
  您可以通过在 **activeRecipes** 参数中指定每个方法包含多个方法。例如，要包含方法 **org.jboss.windup.JavaxInjectToJakartaInject** 和 **org.jboss.windup.JavaxEjbToJakartaEjb"**，为 **"-DactiveRecipes=<recipes=<recipe name>"** 输入以下内容 ：

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

- **--goal** ：可选：要运行的 OpenRewrite Maven 目标。
  - **dryRun** ：该脚本返回提议的更改列表。忽略 **"Run 'mvn rewrite:run' to apply the recipes"** 信息。
  - **run** ： 脚本应用更改。

2. 使用 **--goal run** 运行 **windup-cli** 以应用方法：

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

2.2.4.1. 可用的 OpenRewrite recipes

表 2.1. 可用的 OpenRewrite recipes

迁移路径	用途	rewrite.configLocation	activeRecipes
Java EE 到 Jakarta EE	使用对等的 **jakarta** 软件包替换 **javax** 软件包导入 将 **pom.xml** 文件中声明的 **javax** 工件替换为对等的 **jakarta**	**<MTR_HOME>/rules/openrewrite/jakarta \ /javax/imports/rewrite.yml**	**org.jboss.windup.JavaxToJakarta**
Java EE 到 Jakarta EE	重命名 bootstrap 文件	**<MTR_HOME>/rules/openrewrite/jakarta \ /javax/bootstrapping/rewrite.yml**	**org.jboss.windup.jakarta.javax. \ BootstrappingFiles**
Java EE 到 Jakarta EE	转换 **persistence.xml** 配置	**<MTR_HOME>/rules/openrewrite/jakarta \ /javax/xml/rewrite.yml**	**org.jboss.windup.javax-jakarta. \ PersistenceXML**
Spring Boot 到 Quarkus	在文件中替换 **spring.jpa.hibernate.ddl-auto** 属性来匹配 **application*.properties**	**<MTR_HOME>/rules/openrewrite/quarkus \ /springboot/properties/rewrite.yml**	**org.jboss.windup.sb-quarkus.Properties**

2.3. 访问报告

当您运行 Migration Toolkit for Runtimes 时，会在您在命令行中使用 **--output** 参数指定的 **<OUTPUT_REPORT_DIRECTORY>** 中生成报告。

输出目录包含以下文件和子目录：

```
<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
```

流程

1. 从运行 MTR 后显示的输出中获取报告的 **index.html** 文件的路径：

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

2. 使用浏览器打开 **index.html** 文件。
   此时会显示生成的报告。

第 3 章 查看报告

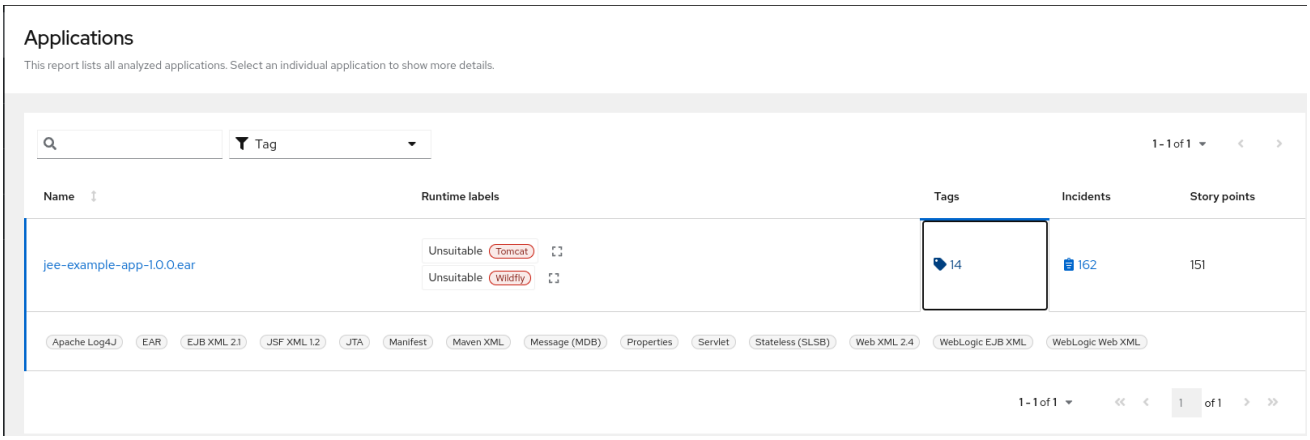
以下部分中显示的报告示例是分析 jee-example-app-1.0.0.ear 示例应用程序中的 **com.acme** 和 **org.apache** 软件包，它们位于 MTR GitHub 源存储库中。

报告是通过以下命令生成的。

```
$ <MTR_HOME>/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
```

使用浏览器打开位于报告输出目录中的 **index.html** 文件。这将打开一个登录页面，其中列出了已处理的应用程序。每行包含故事点、事件数以及应用程序里遇到的技术的高级概述。

图 3.1. 应用程序列表



注意

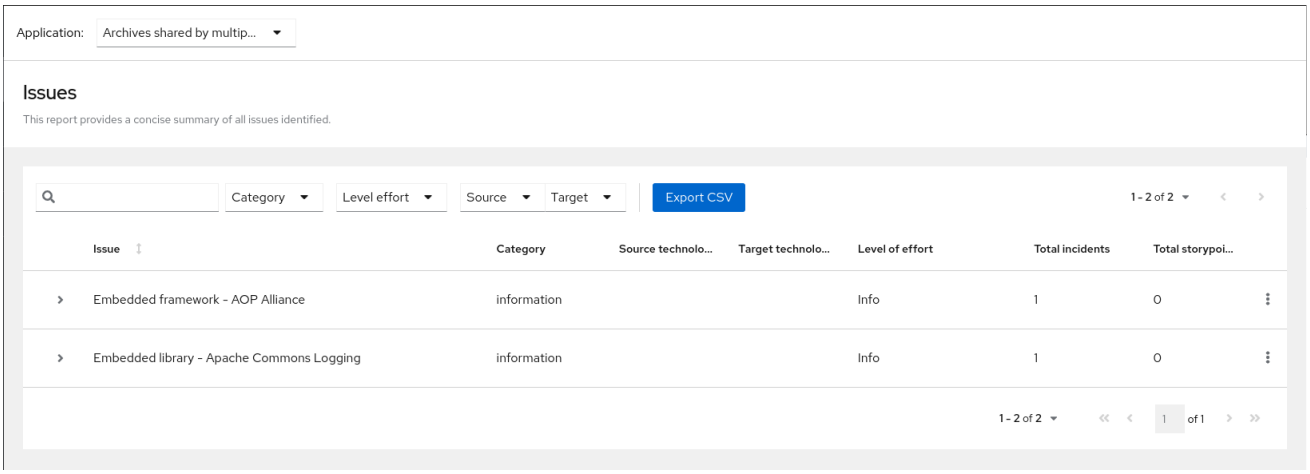
随着新规则添加到 MTR 时，事件和预计的故事点改变。测试此应用程序时，这里的值可能与您看到的值不同。

下表列出了可以从这个主 MTR 登录页访问的所有报告和页面。点应用程序的名称 **jee-example-app-1.0.0.ear** 来查看应用程序报告。

页面	如何访问
Application	点应用程序的名称。
技术报告	单击页面顶部的 **Technologies** 链接。
多个应用程序共享的存档	点 **Archives shared by multiple applications** 链接。请注意，只有在在多个应用程序间有共享存档时，此链接才可用。
规则提供程序执行概述	点页面底部的 **Rule providers execution overview** 链接。

请注意，如果应用程序与其他分析的应用程序共享存档，您会看到有多少故事点来自共享存档，以及此应用程序的独特数量。

图 3.2. 共享归档



有关应用程序间共享的存档的信息，可在多个应用程序报告共享的归档中找到。

3.1. 应用程序报告

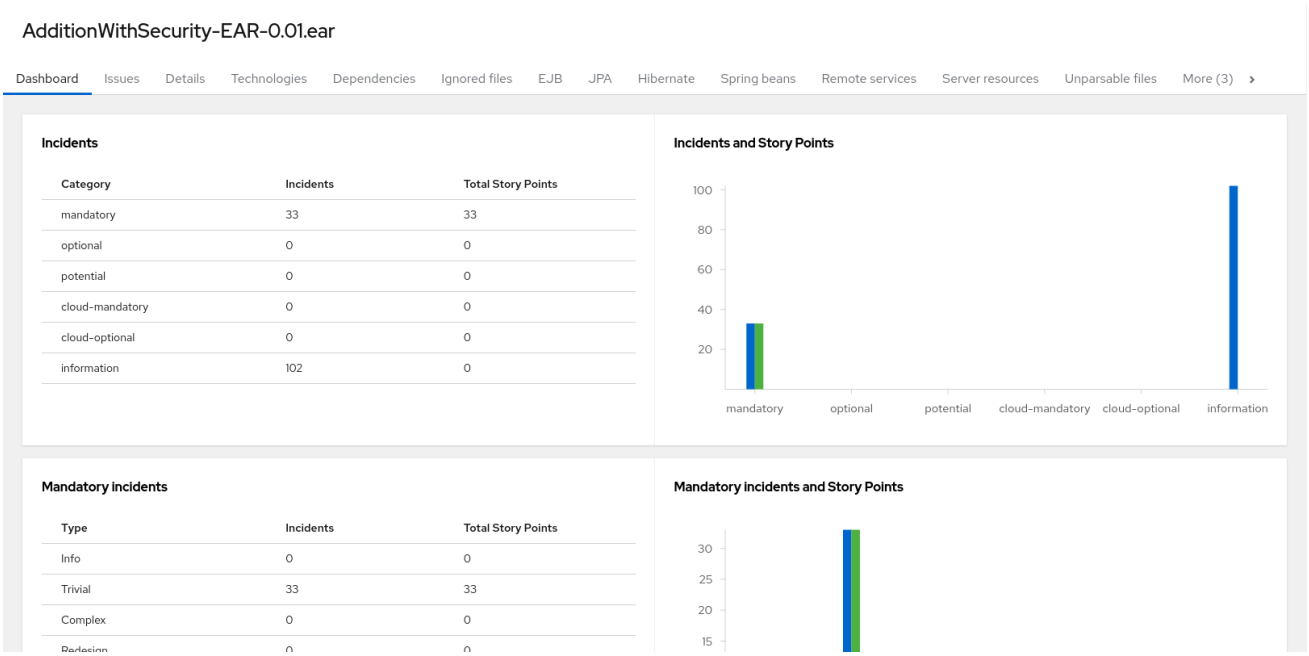
3.1.1. Dashboard

点 Application List 中的应用程序名称，从报告登录页面访问此报告。

控制面板提供整个应用程序迁移工作的概述。总结：

- 事件和故事点按类别分类
- 意外和故事点，按所推荐更改的工作量程度
- 根据软件包的事件

图 3.3. Dashboard



顶部导航栏列出了包含关于迁移此应用程序的更多详情的各种报告。请注意，只有适用于当前应用程序的报告才会可用。

Report	描述
问题	提供有关需要关注的所有问题的简要概述。
应用程序详情	提供了有关应用程序中找到的所有资源的详细信息，这些资源在迁移过程中可能需要注意。
技术	显示所有按功能分组的嵌入式库，允许您快速查看每个应用程序中使用的技术。
依赖项	显示应用程序内找到的所有 Java 打包依赖关系。
unparsable	显示 MTR 无法以预期格式解析的所有文件。例如，假定带有 **.xml** 或 **.wsdl** 后缀的文件是 XML 文件。如果 XML 解析器失败，则会在此处报告问题，以及列出各个文件的位置。
远程服务	显示应用程序内找到的所有远程服务引用。
EJBs	包含应用程序中找到的 EJB 列表。
JBPM	包含分析过程中发现的所有 JBPM 相关资源。
JPA	包含有关应用程序中找到的所有 JPA 相关资源的详细信息。
Hibernate	包含有关应用程序中找到的所有 Hibernate 相关资源的详细信息。
服务器资源	在输入应用中显示所有服务器资源（如 JNDI 资源）。
Spring Beans	包含分析期间找到的 Spring Bean 列表。
硬编码的 IP 地址	提供应用程序中找到的所有硬编码 IP 地址的列表。
忽略的文件	列出应用程序中找到的文件（基于某些规则和 MTR 配置）没有被处理。如需更多信息，请参阅 **--userIgnorePath** 选项。
关于	描述最新版本的 MTR，并为进一步帮助提供有用的链接。

3.1.2. 问题报告

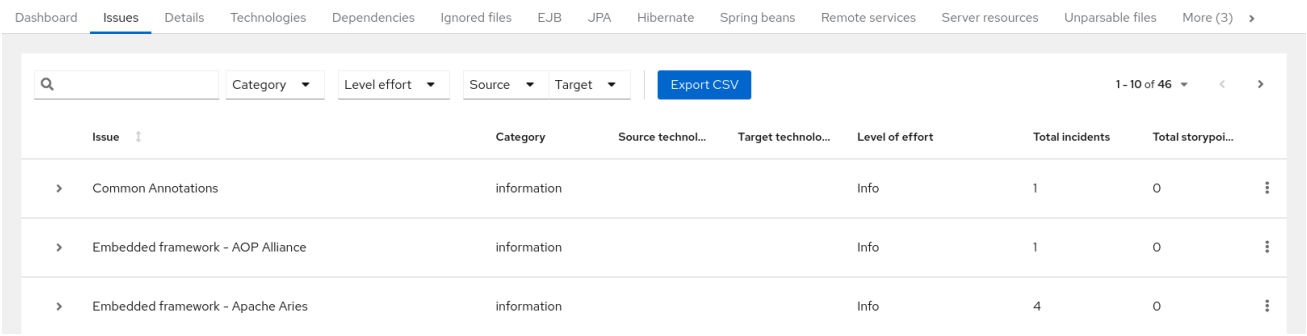
点 Problem 链接，从仪表板 **访问** 此报告。

此报告包括关于所选迁移路径引发的每一个问题的详细信息。遇到的每个问题都会提供以下信息：

- 问题总结的标题。
- 事件总数或问题所遇到的次数。
- 规则故事点可以解决单个问题实例。

- 解决问题的预期工作量。
- 解决遇到的每个实例的总体故事点。这通过乘以每个事件的故事点数来计算得出的。

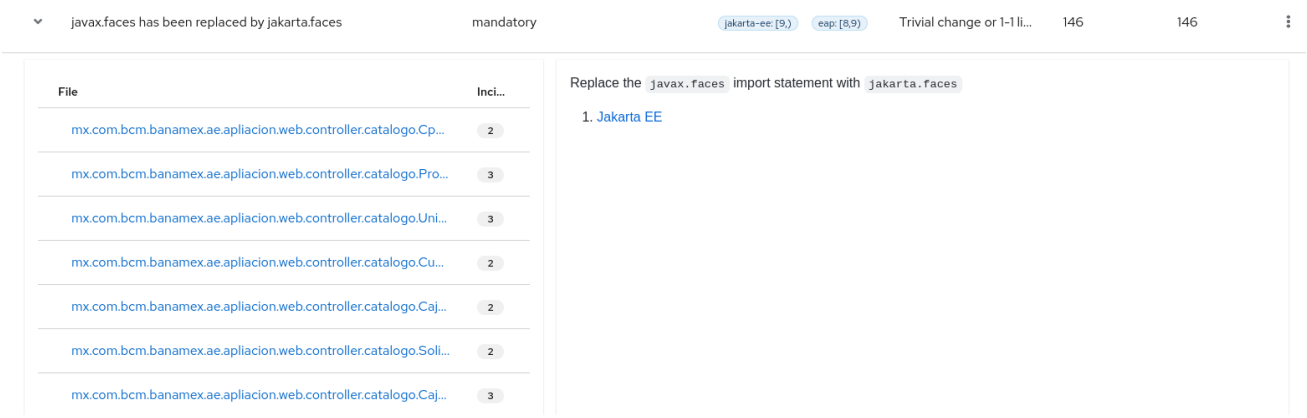
图 3.4. 问题报告



通过单击标题可展开每个报告的问题来获取附加详情。提供了以下信息：

- 发生事件的文件列表，以及各个文件中的事件数。如果文件是 Java 源文件，则点文件名会将您定向到对应的 Source 报告。
- 有关此问题的详细描述。该描述概述了此问题，提供任何已知的解决方案，以及有关问题或解决方案的相关参考文档。
- 向生成该问题的规则授权 **显示** 规则的直接链接。

图 3.5. 扩展的问题



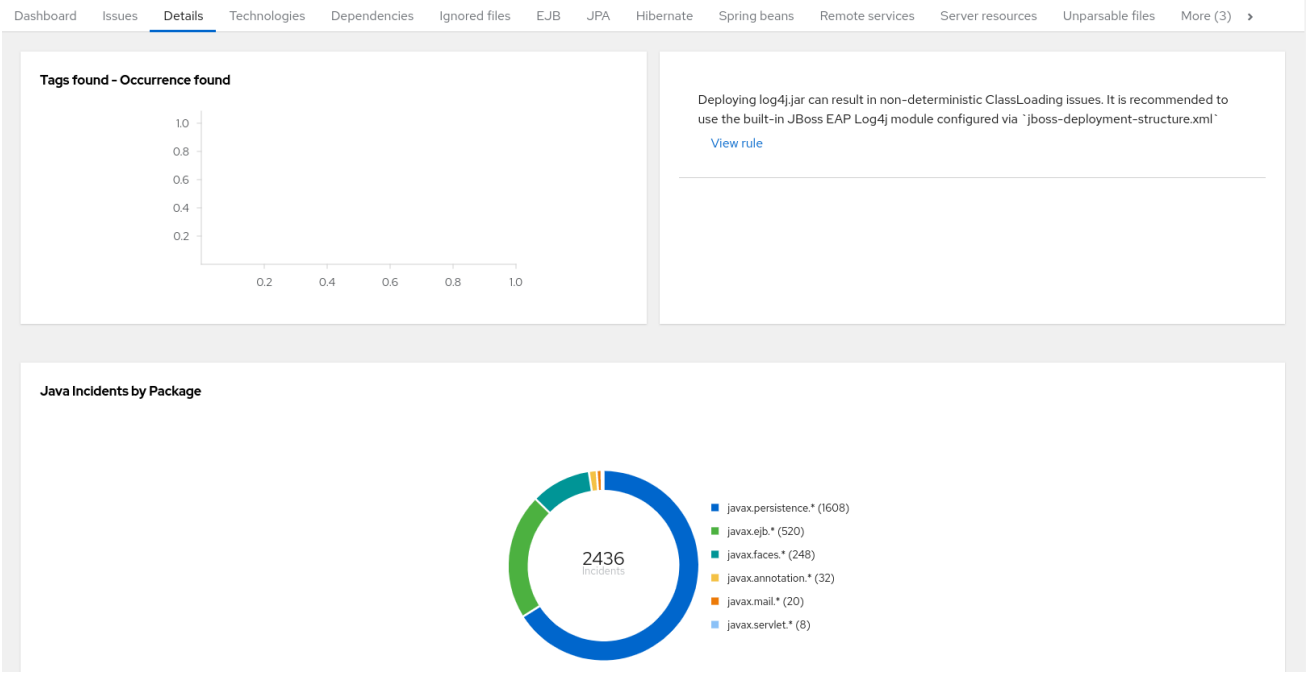
默认情况下，问题按照四个类别排序。有关这些类别的信息，请访问请求类别。

3.1.3. 应用程序详情报告

点 **Application Details** 链接从仪表板访问此报告。

该报告列出了案例点、软件包中的 Java 事件以及应用程序中找到的技术计数。接下来，显示迁移过程中生成的应用程序消息。最后，对过程中分析的每个存档都有此信息分类。

图 3.6. 应用程序详情报告

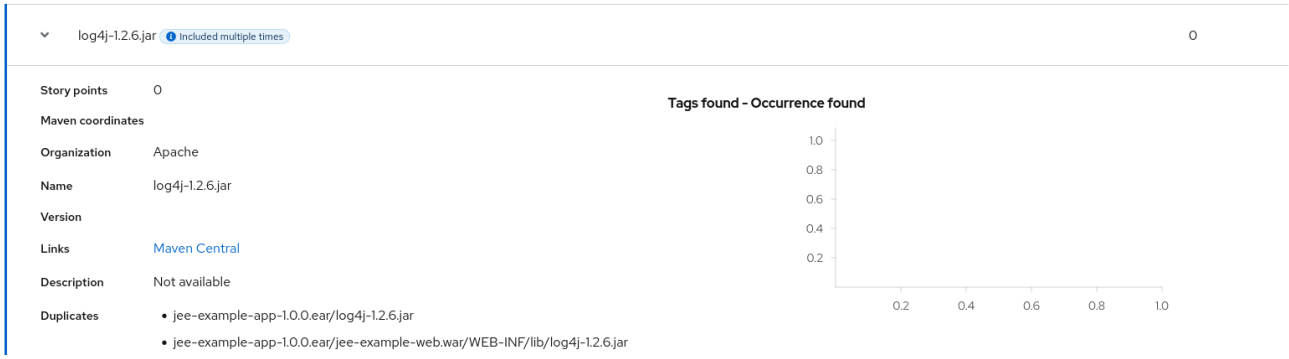


展开 **jee-example-app-1.0.0.ear/jee-example-services.jar** 以检查故事点、Java 事件软件包以及此存档中找到的技术计数。此摘要以分配给其迁移的故事总数开始，然后表详细说明存档中每个文件所需的更改。该报告包含以下几列。

列名称	描述
Name	正在分析的文件的名称。
技术	正在分析的文件的类型，例如 **Decompiled Java File** 或 **Properties**。
问题	有关需要审核或更改的代码区域的警告。
故事点	迁移该文件所需的工作程度。

请注意，如果某个归档在应用程序中重复多次，它将仅在报告中列出一次，并会标记 **[Included multiple times]**。

图 3.7. 应用程序中的重复归档



在应用程序内重复归档的案例点仅计算该应用程序的总故事点数一次。

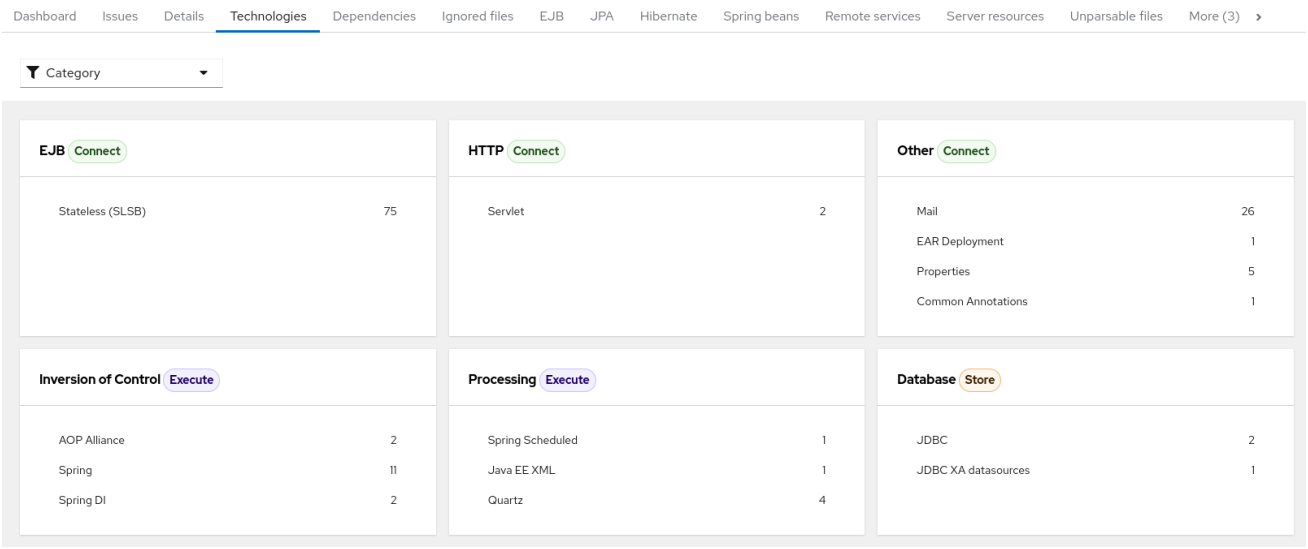
3.1.4. 技术报告

单击 **Technologies** 链接，从仪表板访问此报告。

报告在分析的应用程序中列出了按功能分组的技术。概述了应用程序中找到的技术，旨在帮助用户快速了解每个应用程序的用途。

下图显示了 **jee-example-app** 中使用的技术。

图 3.8. 应用程序中的技术

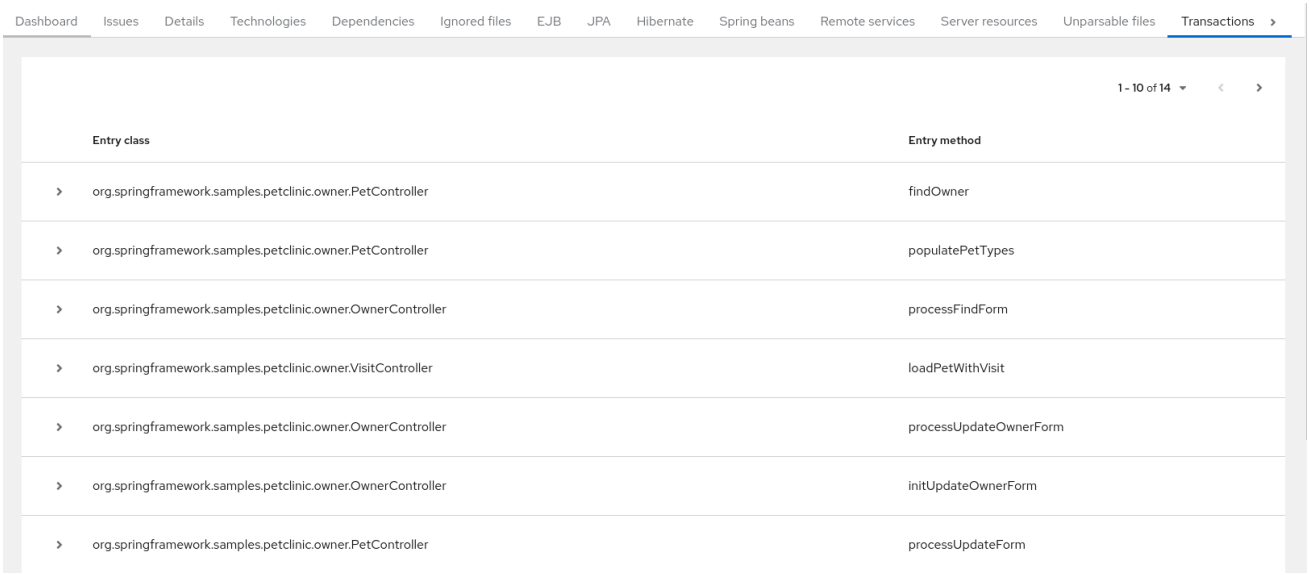


3.1.5. 事务报告

Transactions 报告显示调用堆栈，该堆栈对关系的数据库表执行操作。Enable Transaction Analysis 功能支持 Spring Data JPA，以及用于 SQL 语句执行的传统的 **preparedStatement()** 方法。它不支持 ORM 框架，如 Hibernate。

下图显示了 Transactions 报告的示例。

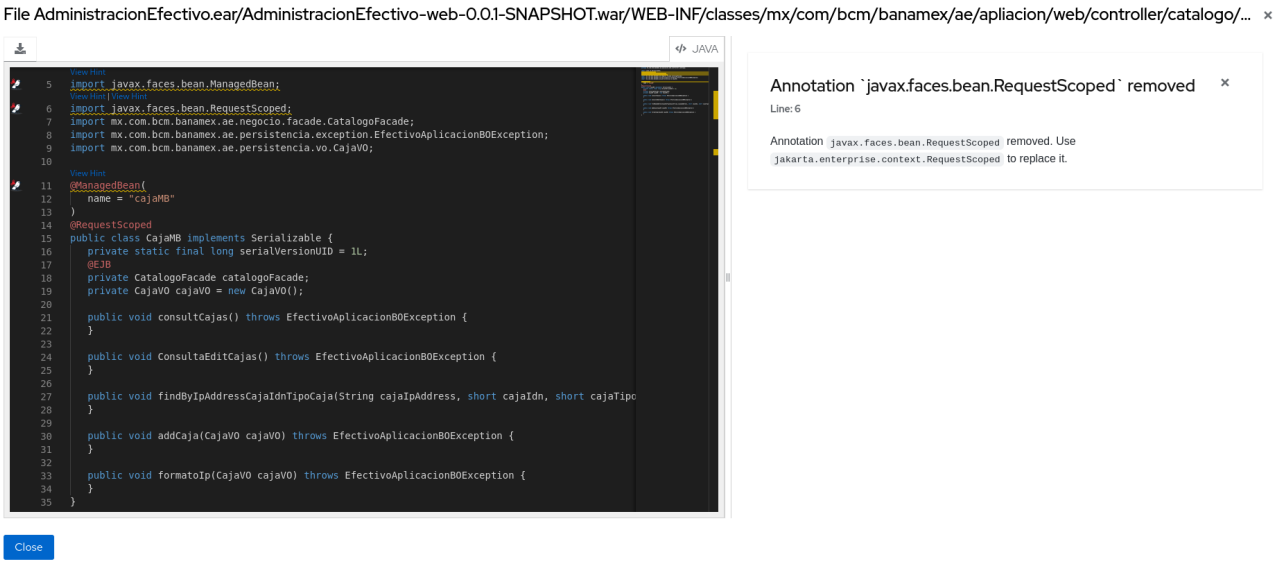
图 3.9. 事务报告



3.1.6. 源报告

Source 报告会在发现它们的源文件上下文中显示迁移问题。

图 3.10. 源报告



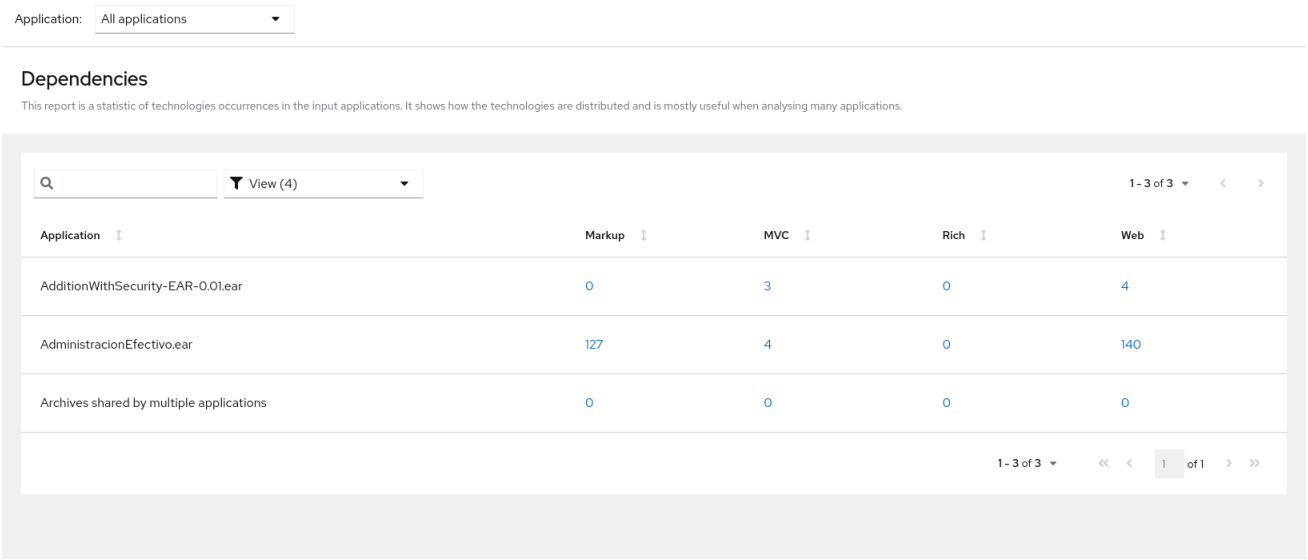
3.2. 技术报告

单击 **Technologies** 链接，从报告登录页面访问此报告。

此报告提供了分析的应用程序所使用的技术的聚合列表，按功能分组。它展示了技术分布方式，通常在分析大量应用程序以对应用程序进行分组并确定模式后，通常会审查技术。它还显示每个应用程序的大小、库数和故事点数。

点击任意标头（如 **Markup** ）以降序排列结果。再次选择同一标头将按顺序选择结果。当前选定的标头以粗体显示，方向箭头旁边表示排序的方向。

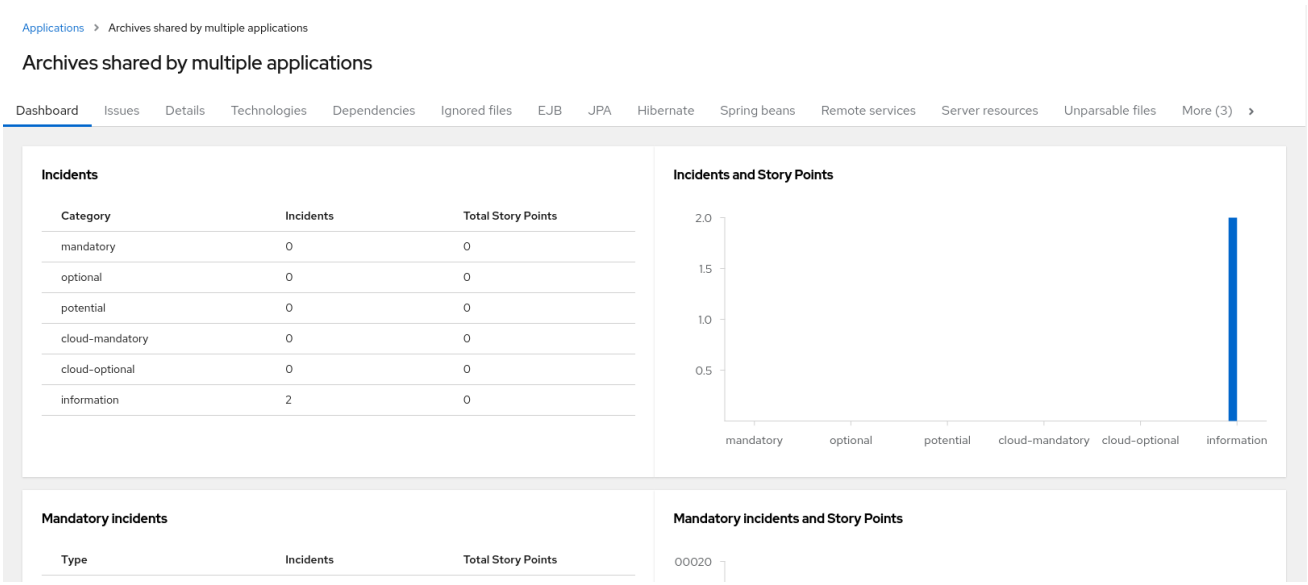
图 3.11. 跨多个应用程序使用的技术



3.3. 多个应用程序共享的存档

点击 **由多个应用程序链接共享的** 归档，从报告登录页面访问这些报告。请注意，只有在有适用的共享存档时，此链接才可用。

图 3.12. 多个应用程序共享的存档



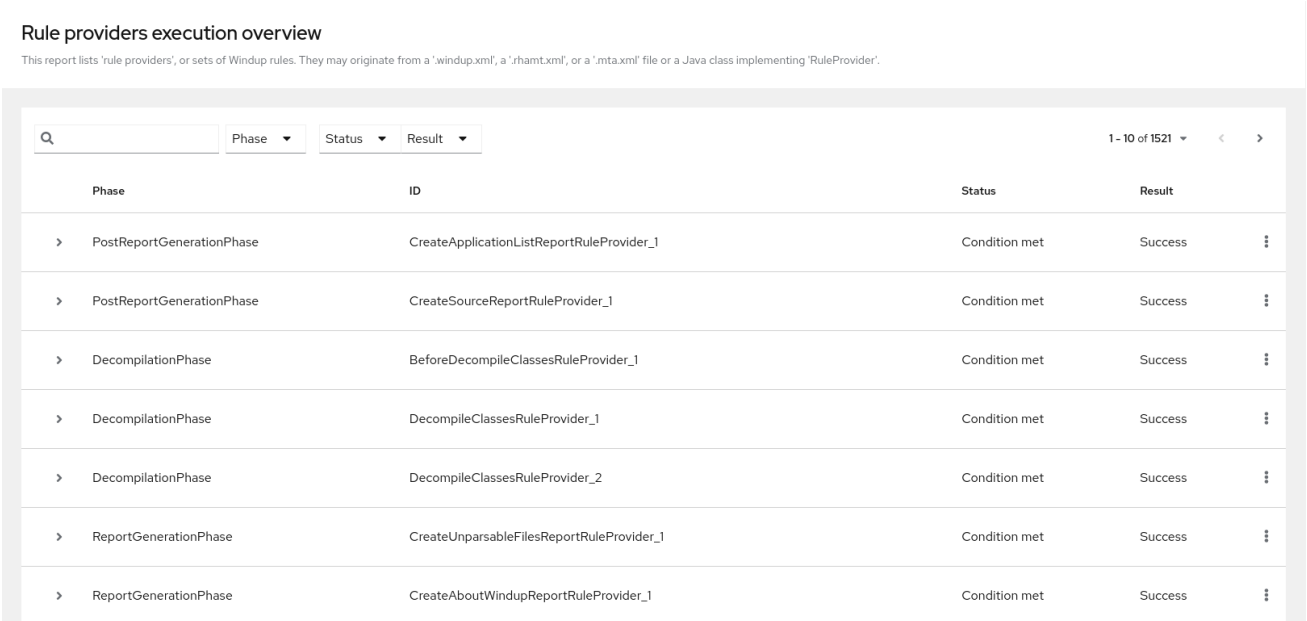
这可让您查看在多个应用程序间共享的所有存档的详细报告。

3.4. 规则提供程序执行概述

点 **规则提供程序执行概述** 链接，从报告登录页面访问此报告。

该报告提供了针对应用程序运行 MTR 迁移命令时运行的规则列表。

图 3.13. 规则提供程序执行概述



第 4 章 以 CSV 格式导出报告

Migration Toolkit for Runtimes (MTR) 提供将报告数据（包括分类和提示）导出到本地文件系统上的平面文件的功能。导出功能当前支持 CSV 文件格式，其报告数据以逗号分开的字段显示(,)。

可以通过电子表格软件（如 Microsoft Excel、OpenOffice Calc 或 libreoffice Calc）导入和操作 CSV 文件。电子表格软件提供了从 MTR 报告排序、分析、评估和管理结果数据的功能。

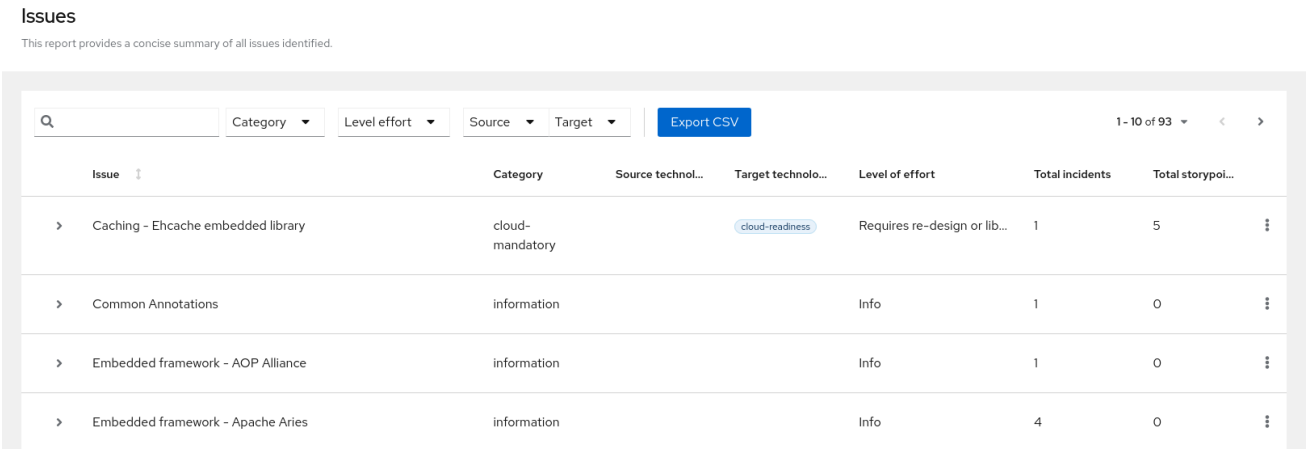
4.1. 导出报告

要将报告导出为 CSV 文件，请使用 **--exportCSV** 参数运行 MTR。在由分析的每个应用程序的 **--output** 参数指定的目录中创建一个 CSV 文件。

所有发现的问题（跨越所有分析的应用程序）都包含在导出至报告根目录下的 **AllIssues.csv** 文件中。

从应用程序报告访问报告
如果您导出了 CSV 报告，您可以在问题报告中下载所有 CSV 问题。要下载这些问题，请点问题报告中的 **Download All issues CSV**。

图 4.1. CSV 下载的问题报告



4.2. 将 CSV 文件导入到电子表格程序中

1. 启动电子表格软件，如 Microsoft Excel。
2. 选择 **File → Open**。
3. 浏览 CSV 导出的文件并选择它。
4. 数据现在已准备好在电子表格软件中进行分析。

4.3. 关于 CSV 数据结构

CSV 格式化输出文件包含以下数据字段：

规则 Id
生成给定项目的规则 ID。

问题类型
hint 或 *classification*

标题

classification 或 *hint* 的标题。此字段总结了给定项目的问题。

描述

给定项目问题的详细描述。

links

提供有关此问题的其他信息的 URL。链接由两个属性组成：链接和描述。

Application

生成此项目的应用程序的名称。

文件名

给定项的文件名。

文件路径

给定项目的文件路径。

行

给定项的文件行号。

故事点

代表给定项目的工作水平的故事点数。

第 5 章 MAVEN 对应用程序进行大小

MTR 提供根据所提供的应用程序生成 Apache Maven 项目结构的功能。这将创建一个目录结构，其中包含指定适当依赖项所需的 Maven 项目对象模型(POM)文件。

请注意，这个功能并不适用于为您的项目创建最终解决方案。它的目的是为您提供一个起点，并确定应用程序所需的依赖项和 API。您的项目可能需要进一步自定义。

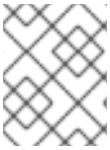
5.1. 生成 MAVEN 项目结构

您可以通过在执行 MTR 时传递 **--mavenize** 标志来为所提供的应用程序生成 Maven 项目结构。

以下示例使用 jee-example-app-1.0.0.ear 测试应用程序运行 MTR：

```
$ <MTR_HOME>/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
```

这会在 **/path/to/output/mavenized** 目录中生成 Maven 项目结构。



注意

您只能在为 **--input** 参数提供编译的应用程序时使用 **--mavenize** 选项。对源代码运行 MTR 时，此功能不可用。

您还可以使用 **--mavenizeGroupId** 选项指定用于 POM 文件的 **<groupId>**。如果未指定，则 MTR 将尝试识别适合应用程序的 **<groupId>**，或者将默认设置为 **com.mycompany.mavenized**。

5.2. 查看 MAVEN 项目结构

The **/path/to/output/mavenized/<APPLICATION_NAME>/** 目录包括以下内容：

- root **POM** 文件。这是顶层目录中的 **pom.xml** 文件。
- A BOM 文件。这是以 **-bom** 结尾的目录中的 **POM** 文件。
- 一个或多个应用程序 **POM** 文件。每个模块的 **POM** 文件在以存档命名的目录中。

示例 **jee-example-app-1.0.0.ear** 应用是包含 WAR 和多个 JAR 的 EAR 存档。每个工件都创建一个单独的目录。以下是为此应用程序创建的 Maven 项目结构。

```
/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
```

检查每个生成的文件，并根据您的项目自定义。要了解更多有关 Maven POM 文件的信息，请参阅 Apache Maven 文档中的 POM 部分介绍。

root POM 文件

jee-example-app-1.0.0.ear 应用程序的 root POM 文件可在 **/path/to/output/mavenized/jee-example-app/pom.xml** 中找到。此文件标识了所有项目模块的目录。

以下模块列在示例 **jee-example-app-1.0.0.ear** 应用程序的 root POM 中。

```
<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>
```

注意

如有必要，请务必重新排序模块列表，以便按照项目的适当构建顺序列出这些模块。

root POM 也被配置为使用 Red Hat JBoss Enterprise Application Platform Maven 存储库 下载项目依赖项。

BOM 文件

在目录中生成的 Bill of Materials (BOM) 文件以 **-bom** 结尾。对于示例应用程序 **jee-example-app-1.0.0.ear**，可在 **/path/to/output/mavenized/jee-example-app/jee-example-app/jee-example-app-bom/pom/pom.xml** 中找到 BOM 文件。此 BOM 的目的是，项目使用的第三方依赖项的版本可以在一个位置定义。有关使用 BOM 的更多信息，请参阅 Apache Maven 文档中的 依赖关系机制简介 部分。

以下依赖项列在 **jee-example-app-1.0.0.ear** 应用程序的 BOM 中

```
<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>
```

应用程序 POM 文件

每个可以 mavenized 的应用程序模块都有一个单独的目录，其中包含其 POM 文件。目录名称包含存档的名称，以 **-jar**、**-war** 或 **-ear** 后缀（根据归档类型）结束。

每个应用程序 POM 文件都会列出该模块的依赖项，包括：

- 第三方库
- Java EE API
- 应用程序子模块

例如，**jee-example-app-1.0.0.ear** EAR、**/path/to/output/mavenized/jee-example-app/jee-example-app-ear/pom.xml** 的 POM 文件，列出下列依赖项。

```
<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 优化 MTR 性能

MTR 性能取决于若干因素，包括硬件配置、应用程序中的文件数和类型、要评估的应用程序的大小和数量，以及应用程序是否包含源代码或编译的代码。例如，大于 10 MB 的文件可能需要大量时间进行处理。

一般情况下，MTR 会花费大约 40% 的工作时间处理类，40% 的工作时间执行规则，其他时间用于处理其他任务并生成报告。本节介绍了您可以如何提高 MTR 的性能。

6.1. 部署并运行应用程序

在升级硬件前首先尝试这些建议。

- 如果可能，请针对源代码而不是存档运行 MTR。这消除了编译额外 JAR 和存档的需求。
- 在 **<MTR_HOME>/bin/mtr-cli** 命令行中使用 **--packages** 参数指定一个要被 MTR 评估的、以逗号分隔的软件包列表。如果省略此参数，MTR 将会处理所有内容，这会影响性能。
- 在可能的情况下尽量指定 **--excludeTags** 参数，以将其排除在处理中。
- 避免终止和分析任何不必要的软件包和文件，如专有软件包或包含的依赖项。
- 分析大型应用程序会增大 ulimit。有关如何为 Red Hat Enterprise Linux 执行此操作的说明，请参阅红帽知识库文章。
- 如果您可以使用一个比笔记本电脑或台式机器具有更好资源的服务器，您可能想考虑在该服务器上运行 MTR。

6.2. 升级硬件

如果上述应用程序和命令行建议无法提高性能，您可能需要升级硬件。

- 如果您可以使用一个比笔记本电脑或台式机器具有更好资源的服务器，您可能想考虑在该服务器上运行 MTR。
- 需要进行处理的大型应用程序有较大的内存要求。建议 8 GB RAM。这允许 JVM 使用 3 - 4 GB RAM。
- 从单或双核升级到四核 CPU 处理器可提供更好的性能。
- 磁盘空间和碎片可能会影响性能。一个快速磁盘，特别是固态驱动器 (SSD)，超过 4 GB 的碎片整理磁盘空间应该会提高性能。

6.3. 配置 MTR 以排除软件包和文件

6.3.1. 附加软件包

您可以在编译和分析过程中排除软件包以提高性能。对这些软件包的引用仍保留在应用程序的源代码中，但排除它们可以避免编译和分析专有类。

与定义值匹配的软件包都会被排除。例如，您可以使用 **com.acme** 排除 **com.acme.example** 和 **com.acme.roadrunner**。

您可以使用以下方法之一排除软件包：

- 使用 **--excludePackages** 参数。
- 在其中一个忽略的位置的文件中指定软件包。每个软件包应当包含在单独的行中，文件必须以 **.package-ignore.txt** 结尾。例如，请参阅 **<MTR_HOME>/ignore/proprietary.package-ignore.txt**。

6.3.2. 排除文件

在扫描和报告生成过程中，MTR 可以排除特定的文件，如包括库或依赖项。排除的文件在一个忽略的位置中使用 **.mtr-ignore.txt** 或 **.windup-ignore.txt** 扩展定义。

这些文件包含一个正则表达式字符串，详细描述要排除的名称，每行列出一个文件。例如，您可以排除库 **ant.jar**，以及以 **Example** 开头的任何 Java 源文件，文件包含以下内容：

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

6.3.3. 搜索位置以获取排除

MTR 搜索以下位置：

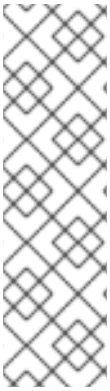
- **~/.mtr/ignore/**
- **~/.windup/ignore/**
- **<MTR_HOME>/ignore/**
- 由 **--userIgnorePath** 参数指定的任何文件和文件夹

根据要排除的内容类型，每个文件都必须符合为排除的软件包或文件指定的规则。

附录 A. 参考材料

A.1. 关于 MTR 命令行参数

以下是可用 MTR 命令行参数的详细描述。



注意

要在不提示的情况下运行 MTR 命令，例如从脚本执行时，您必须使用以下参数：

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

表 A.1. MTR CLI 参数

参数	描述
--additionalClassPath	要添加到类路径的额外 JAR 文件或目录列表（以空格分隔），以便它们可用于处理或进行其他分析。
--addonDir	将指定的目录添加为自定义附加组件存储库。
--analyzeKnownLibraries	用于分析应用程序中嵌入的已知软件工件的标志。默认情况下，MTR 仅分析应用程序代码。  **注意** 这个选项可能会导致执行时间较长，并报告大量迁移问题。
--batchMode	指定 MTR 应该在不提示确认的情况下以非互动模式运行的标志。这个模式采用未传递给命令行的任何参数的默认值。
--debug	在调试模式下运行 MTR 的标志。
--disableTattletale	可禁用 Tattletale 报告的生成标志。如果同时将 **enableTattletale** 和 **disableTattletale** 设定为 true，则 **disableTattletale** 会被忽略，但仍会生成 Tattletale 报告。
--discoverPackages	用于列出输入二进制应用程序中的所有可用软件包的标志。
--enableClassNotFoundAnalysis	标志，以启用对类路径上不可用的 Java 文件进行分析。如果一些类在分析时不可用，则不应使用。

参数	描述
--enableCompatibleFilesReport	启用生成可组合文件报告的标志。由于处理的所有文件都未发现问题，对于大型应用程序，此报告可能需要很长时间才能完成。
--enableTattletale	用于为每个应用程序生成 Tattletale 报告的标志。当 **eap** 位于包含的目标中时，默认启用这个选项。如果同时将 **enableTattletale** 和 **disableTattletale** 设定为 true，则 **disableTattletale** 会被忽略，但仍会生成 Tattletale 报告。
--enableTransactionAnalysis	[技术预览] 标记来启用生成交易报告，该报告显示调用堆栈，该堆栈对关系数据库表执行操作。Enable Transaction Analysis 功能支持 Spring Data JPA，以及用于 SQL 语句执行的传统的 **preparedStatement()** 方法。它不支持 ORM 框架，如 Hibernate。 **注意** enableTransactionAnalysis 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。
--excludePackages	要排除在评估中排除的软件包列表。例如，输入 **com.mycompany.commonutilities** 排除所有软件包名称以 **com.mycompany.commonutilities** 开头的类。
--excludeTags	要排除的以逗号分隔的标签列表。指定后，不会处理具有这些标签的规则。要查看完整的标记列表，请使用 **--listTags** 参数。
--explodedApp	指定提供的输入目录包含单个应用程序的源文件的标志。
--exportCSV	标志，将报告数据导出到本地文件系统中的 CSV 文件。MTR 在由 **--output** 参数指定的目录中创建文件。CSV 文件可导入到电子表格程序，以便进行数据操作和分析。
--exportSummary	指示 MTR 在输出目录中生成 **analysisSummary.json** 导出文件的标记。文件包含了对每个被分析的应用程序的分析摘要信息，包括按类别划分的事件和故事点数，以及与分析的应用程序关联的所有技术标签。
--exportZipReport	此参数在输出文件夹中创建一个 **report.zip** 文件。该文件包含分析输出，通常是报告。如果请求，它也可以包含 CSV 导出文件。
--help	显示 MTR 帮助信息。
--immutableAddonDir	将指定目录添加为自定义只读附加组件存储库。

参数	描述
--includeTags	要使用的空格分隔的标记列表。指定后，仅处理具有这些标签的规则。要查看完整的标记列表，请使用 **--listTags** 参数。
--input	到要分析的一个或多个应用程序的文件或目录的路径列表。此参数是必需的。
--install	指定要安装的附加组件。语法为 **<GROUP_ID>:<ARTIFACT_ID>[:<VERSION>]**。例如：**--install core-addon-x** 或 **--install org.example.addon:example:1.0.0**。
--keepWorkDirs	标志指示 MTR 不会删除临时文件，如图形数据库和提取的存档文件。这对于调试非常有用。
--legacyReports	指示 MTR 生成旧格式报告而不是新的格式报告的标记。
--list	列出已安装的附加组件的标志。
--listSourceTechnologies	用于列出所有可用的源技术的标志。
--listTags	用于列出所有可用的标签的标志。
--listTargetTechnologies	用于列出所有可用目标技术的标记。
--mavenize	标志，基于应用的结构和内容创建 Maven 项目目录结构。这将使用适当的 Java EE API 和项目模块之间依赖关系创建 **pom.xml** 文件。另请参阅 **--mavenizeGroupId** 选项。
--mavenizeGroupId	与 **--mavenize** 选项一起使用时，所有生成的 **pom.xml** 文件都使用为它们的 **<groupId>** 提供的值。如果省略了这个参数，则 MTR 将尝试根据应用程序确定适当的 **<groupId>** 参数，或者将默认为 **com.mycompany.mavenized**。
--online	标志，允许对需要的功能进行网络访问。目前，根据外部资源验证 XML 模式是否依赖于互联网访问。请注意，这附带了一个性能损失。
--output	指定输出 MTR 生成的报告信息的目录路径。
--overwrite	用于强制删除 **--output** 指定的现有输出目录的标志。如果没有指定此参数，且存在 **--output** 目录，则会提示您选择是否覆盖内容。  **重要** 不要覆盖包含重要信息的报告输出目录。

参数	描述
--packages	MTR 要评估的软件包用空格分隔的列表。强烈建议您使用此参数。
--remove	删除指定的附加组件。语法为 **<GROUP_ID>:<ARTIFACT_ID>[:<VERSION>]**。例如： **--remove core-addon-x** 或 **--remove org.example.addon:example:1.0.0**。
--skipReports	表示 HTML 报告不应生成的标志。此参数的常见用途是使用 **--exportCSV** 将报告数据导出到 CSV 文件时。
--source	要迁移的一个或多个源技术、服务器、平台或框架的空格分隔列表。此参数与 **--target** 参数结合使用有助于确定使用哪个规则集。使用 **--listSourceTechnologies** 参数列出所有可用的源。
--sourceMode	用于表示要评估的应用包含源文件而非编译的二进制文件的标志。sourceMode 参数已弃用。现在，不再需要指定它。MTR 可以直观地处理向其呈现的任何输入。另外，可以使用同一分析执行中的二进制输入来分析项目源文件夹。
--target	要迁移到的一个或多个目标技术、服务器、平台或框架的空格分隔列表。此参数与 **--source** 参数结合使用有助于确定使用哪个规则集。使用 **--listTargetTechnologies** 参数列出所有可用的目标。
--userIgnorePath	除了 **${user.home}/.mtr/ignore/** 之外，为 MTR 指定位置，以识别应忽略的文件。
--userLabelsDirectory	指定 MTR 来查找自定义 Target Runtime Labels 的位置。该值可以是含有标签文件或单个标签文件的目录。Target Runtime Label 文件必须使用 **.windup.label.xml** 后缀。提供的 Target Runtime Labels 在 **$MTR_HOME/rules/migration-core/core.windup.label.xml** 中定义。
--userRulesDirectory	除 **<MTR_HOME>/rules/** 和 **${user.home}/.mtr/rules/** 外为 MTR 指定一个位置来查找自定义 MTR 规则。该值可以是包含规则集文件或单个规则集文件的目录。ruleset 文件必须使用 **.windup.xml** 后缀。
--version	显示 MTR 版本。
--skipSourceCodeReports	此选项会在生成应用程序分析报告时跳过生成*源代码报告*。当需要关注应用程序分析中的源代码信息时，请使用此高级选项。

A.1.1. 指定输入

到要分析的一个或多个应用程序的文件或目录的路径列表。此参数是必需的。

使用方法

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

根据提供给 **--input** 参数提供的输入文件类型是否为文件或目录，它将根据提供的附加参数进行评估。

目录

--explodedApp	--sourceMode	没有参数
目录作为单个应用进行评估。	目录作为单个应用进行评估。	每个子目录都作为应用进行评估。

File

--explodedApp	--sourceMode	没有参数
参数将被忽略；该文件将评估为单个应用。	该文件作为压缩的项目进行评估。	该文件作为单个应用进行评估。

A.1.2. 指定输出目录

指定输出 MTR 生成的报告信息的目录路径。

使用方法

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

- 如果省略，则会在 **<INPUT_ARCHIVE_OR_DIRECTORY>.report** 目录中生成报告。
- 如果存在输出目录，系统将提示您（默认值为 N）。

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

但是，如果您指定 **--overwrite** 参数，则 MTR 将继续删除并重新创建 目录。如需更多信息，请参阅此参数的描述。

A.1.3. 设置源技术

要迁移的一个或多个源技术、服务器、平台或框架的空格分隔列表。此参数与 **--target** 参数结合使用有助于确定使用哪个规则集。使用 **--listSourceTechnologies** 参数列出所有可用的源。

使用方法

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

--source 参数现在提供版本支持，它遵循 Maven 版本范围语法。这指示 MTR 只运行与指定版本匹配的规则集。例如，**--source eap:5**。

警告

迁移到 JBoss EAP 时，请务必指定版本，如 **eap:6**。仅指定 **eap** 将针对所有版本的 JBoss EAP 运行规则集，包括与您的迁移路径无关。

有关相应 JBoss EAP 版本的 *Migration Toolkit for Runtimes 简介*，请参阅支持的迁移路径。

A.1.4. 设置目标技术

要迁移到的一个或多个目标技术、服务器、平台或框架的空格分隔列表。此参数与 **--source** 参数结合使用有助于确定使用哪个规则集。如果没有指定这个选项，系统会提示您选择一个目标。使用 **--listTargetTechnologies** 参数列出所有可用的目标。

使用方法

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

--target 参数现在提供版本支持，它遵循 Maven 版本范围语法。这指示 MTR 只运行与指定版本匹配的规则集。例如，**--target eap:7**.

警告

迁移到 JBoss EAP 时，请务必在目标中指定版本，例如 **eap:6**。仅指定 **eap** 将针对所有版本的 JBoss EAP 运行规则集，包括与您的迁移路径无关。

有关相应 JBoss EAP 版本的 *Migration Toolkit for Runtimes 简介*，请参阅支持的迁移路径。

A.1.5. 选择软件包

MTR 要评估的软件包用空格分隔的列表。强烈建议您使用此参数。

使用方法

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

- 在大多数情况下，您只想评估自定义应用程序类软件包而不是标准 Java EE 或第三方软件包。**<PACKAGE_N>** 参数是一个软件包前缀；所有子软件包都会被扫描。例如，要扫描命令行上的 **com.mycustomapp** 和 **com.myotherapp** 和 **com.myotherapp com.mycustomapp com.myotherapp** 参数。
- 虽然您可以为标准 Java EE 第三方软件（如 **org.apache** ）提供软件包名称，但通常最好不要包含它们，因为它们不会影响迁移工作。

警告

如果省略 **--packages** 参数，应用程序中的每个软件包都会被扫描，这可能会影响性能。

A.2. 支持的技术标签

MTR 1.2.6 支持以下技术标签：

- 0MQ Client
- 3scale
- Acegi Security
- AcrIS Security
- ActiveMQ library
- Airframe
- Airlift Log Manager
- AKKA JTA
- Akka Testkit
- Amazon SQS Client
- AMQP Client
- Anakia
- AngularFaces
- ANTLR StringTemplate
- AOP Alliance
- Apache Accumulo Client
- Apache Aries
- Apache Commons JCS
- Apache Commons Validator
- Apache Flume
- Apache Geronimo
- Apache Hadoop

- Apache HBase Client
- Apache Ignite
- Apache Karaf
- Apache Mahout
- Apache Meecrowave JTA
- Apache Sirona JTA
- Apache Synapse
- Apache Tapestry
- Apiman
- Applet
- Arquillian
- AspectJ
- Atomikos JTA
- Avalon Logkit
- Axion Driver
- Axis
- Axis2
- BabbageFaces
- Bean Validation
- BeanInject
- Blaze
- Blitz4j
- BootsFaces
- Bouncy Castle
- ButterFaces
- Cache API
- Cactus
- Camel
- Camel Messaging Client

- Camunda
- Cassandra Client
- CDI
- Cfg Engine
- Chunk Templates
- Cloudera
- Coherence
- 常见注解
- Composite Logging
- Composite Logging JCL
- Concordion
- CSS
- Cucumber
- Dagger
- DbUnit
- Demoiselle JTA
- Derby Driver
- Drools
- DVSL
- Dynacache
- EAR Deployment
- Easy Rules
- EasyMock
- Eclipse RCP
- EclipseLink
- Ehcache
- EJB
- EJB XML
- Elasticsearch

- Entity Bean
- EtlUnit
- Eureka
- Everit JTA
- Evo JTA
- Feign
- File system Logging
- FormLayoutMaker
- FreeMarker
- Geronimo JTA
- GFC Logging
- GIN
- GlassFish JTA
- Google Guice
- Grails
- Grapht DI
- Guava Testing
- GWT
- H2 Driver
- Hamcrest
- Handlebars
- HavaRunner
- Hazelcast
- Hdiv
- 休眠
- Hibernate Cfg
- Hibernate Mapping
- Hibernate OGM
- HighFaces

- HornetQ Client
- HSQLDB Driver
- HTTP Client
- HttpUnit
- ICEfaces
- Ickenham
- Ignite JTA
- Ikasan
- iLog
- Infinispan
- Injekt for Kotlin
- Iroh
- Istio
- Jamon
- Jasypt
- Java EE Batch
- Java EE Batch API
- Java EE JACC
- Java EE JAXB
- Java EE JAXR
- Java EE JSON-P
- Java Transaction API
- JavaFX
- JavaScript
- javax Inject
- JAX-RS
- JAX-WS
- JayWire
- JBehave

- JBoss Cache
- JBoss EJB XML
- JBoss logging
- JBoss Transactions
- JBoss Web XML
- JBossMQ Client
- JBPM
- JCA
- Jcabi Log
- JCache
- JCunit
- JDBC
- JDBC datasources
- JDBC XA datasources
- Jersey
- Jetbrick Template
- Jetty
- JFreeChart
- JFunk
- JGoodies
- JMock
- JMockit
- JMS Connection Factory
- JMS Queue
- JMS Topic
- JMustache
- JNA
- JNI
- JNLP

- JPA entities
- JPA Matchers
- JPA named queries
- JPA XML
- JSecurity
- JSF
- JSF Page
- JSilver
- JSON-B
- JSP Page
- JSTL
- JTA
- Jukito
- JUnit
- Ka DI
- Keyczar
- Kibana
- KLogger
- Kodein
- Kotlin Logging
- KouInject
- KumuluzEE JTA
- LevelDB Client
- Liferay
- LiferayFaces
- Lift JTA
- Log.io
- Log4J
- Log4s

- Logback
- Logging Utils
- Logstash
- Lumberjack
- Macros
- Magicgrouplayout
- Mail
- Management EJB
- MapR
- MckoiSQLDB Driver
- Memcached
- Message (MDB)
- Micro DI
- Micrometer
- Microsoft SQL Driver
- MiGLayout
- MinLog
- Mixer
- Mockito
- MongoDB Client
- Monolog
- Morphia
- MRules
- Mule
- Mule Functional Test Framework
- MultithreadedTC
- Mycontainer JTA
- MyFaces
- MySQL Driver

- Narayana Arjuna
- Needle
- Neo4j
- NLOG4J
- Nuxeo JTA/JCA
- OACC
- OAUTH
- OCPsoft Logging Utils
- OmniFaces
- OpenFaces
- OpenPojo
- OpenSAML
- OpenWS
- OPS4J Pax Logging Service
- Oracle ADF
- Oracle DB Driver
- Oracle Forms
- Orion EJB XML
- Orion Web XML
- Oscache
- OTR4J
- OW2 JTA
- OW2 Log Util
- OWASP CSRF Guard
- OWASP ESAPI
- Peaberry
- Pega
- Persistence units
- Petals EIP

- PicketBox
- PicketLink
- PicoContainer
- Play
- Play Test
- Plexus Container
- Polyforms DI
- Portlet
- PostgreSQL Driver
- PowerMock
- PrimeFaces
- Properties
- Qpid Client
- RabbitMQ Client
- RandomizedTesting Runner
- Resource Adapter
- REST Assured
- Restito
- RichFaces
- RMI
- RocketMQ Client
- Rythm Template Engine
- SAML
- Santuario
- Scalate
- Scaldi
- Scribe
- Seam
- Security Realm

- ServiceMix
- Servlet
- ShiftOne
- Shiro
- Silk DI
- SLF4J
- Snippetory Template Engine
- SNMP4J
- Socket handler logging
- Spark
- Specsy
- Spock
- Spring
- Spring Batch
- Spring Boot
- Spring Boot Actuator
- Spring Boot Cache
- Spring Boot Flo
- Spring Cloud Config
- Spring Cloud Function
- Spring Data
- Spring Data JPA
- Spring DI
- Spring Integration
- Spring JMX
- Spring Messaging Client
- Spring MVC
- Spring Properties
- Spring Scheduled

- Spring Security
- Spring Shell
- Spring Test
- Spring Transactions
- Spring Web
- SQLite Driver
- SSL
- Standard Widget Toolkit (SWT)
- Stateful (SFSB)
- Stateless (SLSB)
- Sticky Configured
- Stripes
- Struts
- SubCut
- Swagger
- SwarmCache
- Swing
- SwitchYard
- Syringe
- Talend ESB
- Teiid
- TensorFlow
- Test Interface
- TestNG
- Thymeleaf
- TieFaces
- tinylog
- Tomcat
- Tornado Inject

- Trimou
- Trund JGuard
- Twirl
- Twitter Util Logging
- UberFire
- Unirest
- Unitils
- Vaadin
- Velocity
- Vlad
- Water Template Engine
- Web Services Metadata
- Web Session
- Web XML File
- WebLogic Web XML
- Webmacro
- WebSocket
- WebSphere EJB
- WebSphere EJB Ext
- WebSphere Web XML
- WebSphere WS Binding
- WebSphere WS Extension
- Weka
- Weld
- WF Core JTA
- Wicket
- Winter
- WSDL
- WSO2

- WSS4J
- XACML
- XFire
- XMLUnit
- Zbus Client
- Zipkin

A.3. 关于规则故事点

A.3.1. 什么是故事点？

*故事点*是敏捷软件开发中常用的抽象指标，用于估算实施功能或更改所需的*工作量水平*。

应用的 Migration Toolkit for Runtimes 使用故事点来代表迁移特定应用程序构造所需的工作程度，以及整个应用程序。它不一定转换为“人-小时”，但该值在不同的任务间应保持一致。

A.3.2. 故事点如何在规则中估计

估算一个规则的故事点的工作量水平可能很棘手。以下是估算规则所需工作量时的一般准则 MTR。

努力级别	故事点	描述
信息	0	迁移过程中具有非常低或没有优先级的信息。
微小	1	迁移是一个微小的变化或一个简单的库交换，没有或有最小的 API 更改。
复杂	3	迁移任务所需的更改比较复杂，但有一个已包括在文档中的解决方案。
重新设计	5	迁移任务需要重新设计或完整的库更改，并有显著的 API 更改。
架构重组	7	迁移会需要组件或子系统的完全的架构重组。
Unknown	13	迁移解决方案并非已知的，可能需要进行彻底的重写。

A.3.3. 任务类别

除了工作程度外，您还可以对迁移任务进行分类，以指明任务的严重性。下列类别用于对问题进行分组，以帮助确定迁移的工作量。

Mandatory（必需）

必须成功完成该任务才能成功迁移。如果没有进行任何更改，则生成的应用不会成功构建或运行。例如，替换在目标平台中不支持的专有 API。

选填

如果没有完成迁移任务，应用程序应该可以正常工作，但结果可能不是最佳。如果迁移时没有进行任何更改，建议在迁移完成后尽快按计划设置。

Potential

应在迁移过程中检查该任务，但没有足够的详细信息来确定任务是否成功完成。当没有直接兼容类型时，这将迁移第三方专有类型。

信息

该任务会包括告知您存在某些文件。可能需要将它们检查或修改为现代化工作的一部分，但通常不需要进行更改。

有关分类任务的更多信息，请参阅使用自定义规则类别。

A.4. 其它资源

A.4.1. 对项目贡献

为了帮助 Migration Toolkit for Runtimes 包括大多数应用程序结构和服务器配置，您可以使用以下项目帮助：

- 发送电子邮件到 jboss-migration-feedback@redhat.com，并告知我们必须覆盖哪些 MTR 迁移规则。
- 提供示例应用程序来测试迁移规则。
- 识别迁移可能很难迁移的应用程序组件和问题区域：
  - 编写问题迁移区域的简短描述。
  - 编写简短概述，了解如何解决问题迁移区域。
- 在应用程序上尝试 Migration Toolkit for Runtimes。确保报告您满足的任何问题。
- 贡献 Migration Toolkit for Runtimes 规则存储库：
  - 编写 Migration Toolkit for Runtimes 规则以识别或自动迁移过程。
  - 为新规则创建一个测试。

  如需更多信息，*请参阅规则开发指南*。
- 贡献项目源代码：
  - 创建核心规则。
  - 提高 MTR 性能或效率。

任何级别的参与都非常感谢！

A.4.2. Migration Toolkit for Runtimes 开发资源

使用以下资源来学习并贡献 Migration Toolkit for Runtimes 开发：

- MTR论坛 ：https://developer.jboss.org/en/windup
- JIRA issue tracker: https://issues.redhat.com/projects/WINDUP

- MTR 邮件列表： jboss-migration-feedback@redhat.com

A.4.3. 报告问题

MTR 使用 JIRA 作为其问题跟踪系统。如果您在执行 MTR 时遇到问题，请提交 JIRA 问题。

更新于 2024-06-12