Migration Toolkit for Applications 7.0

CLI 指南

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

Last Updated: 2024-05-25

Migration Toolkit for Applications 7.0 CLI 指南

了解如何使用 Migration Toolkit for Applications 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 Applications CLI 来简化 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 Applications (MTA) 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 APPLICATIONS

什么是 Migration Toolkit for Applications?

Migration Toolkit for Applications (MTA) 在 Red Hat OpenShift 的混合云环境中加速大规模应用程序现代化的过程。此解决方案会考虑整个迁移过程的详细情况，包括产品组合和应用程序级别的信息：库存、评估、分析和管理应用程序，以便更快地通过用户界面迁移到 OpenShift。

MTA 使用大量默认问题作为评估应用程序的基础，或者您可以创建自己的自定义问题，允许您估算为容器化准备应用程序所需的难度、时间和其他资源。您可以使用评估结果作为与利益相关者进行的讨论基础，以确定哪些应用程序可以被容器化，哪些需要大量的准备工作，哪些不适用于容器化。

MTA 会根据一个或多个规则集来对应用程序进行分析，并找出应用程序的哪些部分需要进行修改才可以对其进行现代化。

MTA 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。

Migration Toolkit for Applications 如何简化迁移？

Migration Toolkit for Applications 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。

MTA 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。

1.2.1. 支持的 Migration Toolkit for Applications 迁移路径

Migration Toolkit for Applications (MTA)支持以下迁移：

- 从第三方企业应用服务器（如 Oracle WebLogic Server）迁移到 JBoss 企业应用平台(JBoss EAP)。
- 升级到最新版本的 JBoss EAP。

MTA 提供了一组全面的规则，用于评估应用程序以在 Red Hat OpenShift Container Platform (RHOCP) 上进行容器化和部署。您可以运行 MTA 分析来评估您的应用程序是否适合迁移到多个目标平台。

表 1.1. 支持的迁移路径： Source platform busybox Target platform

源平台 TOKEN	迁移到 JBoss EAP 7 和 8	OpenShift（云就绪）	OpenJDK 11、17 和 21	Jakarta EE 9	Camel 3 和 4	Red Hat Runtimes 中的 Spring Boot	Quarkus	Open Liberty
Oracle WebLogic Server	✔	✔	✔	-	-	-	-	-
IBM WebSphere Application Server	✔	✔	✔	-	-	-	-	✔
JBoss EAP 4	✘ [a]	✔	✔	-	-	-	-	-
JBoss EAP 5	✔	✔	✔	-	-	-	-	-
JBoss EAP 6	✔	✔	✔	-	-	-	-	-
JBoss EAP 7	✔	✔	✔	-	-	-	✔	-
Thorntail	✔ [b]	-	-	-	-	-	-	-
Oracle JDK	-	✔	✔	-	-	-	-	-
Camel 2	-	✔	✔	-	✔	-	-	-
Spring Boot	-	✔	✔	✔	-	✔	✔	-
任何 Java 应用程序	-	✔	✔	-	-	-	-	-
任何 Java EE 应用程序	-	-	-	✔	-	-	-	-

[a] 虽然 MTA 目前不提供此迁移路径的规则，但红帽咨询可帮助从任何源平台迁移到 JBoss EAP 7。

[b] 需要 JBoss Enterprise Application Platform expansion pack 2 (EAP XP 2)

有关用例和迁移路径的更多信息，请参阅适用于开发人员网页的 MTA。

1.3. MTA CLI

CLI 是 Migration Toolkit for Applications 的 Migration Toolkit for Applications 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 MTA 分析选项或与外部自动化工具集成。

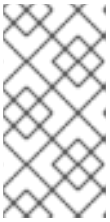
第 2 章 安装和运行 CLI

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

- **registry.redhat.io** 的 Red Hat Container Registry 身份验证。红帽从需要身份验证的 registry.redhat.io 中分发容器镜像。如需了解更多详细信息，请参阅 Red Hat Container Registry 身份验证。
- 必须安装 podman



PODMAN

Podman 是一个无守护进程的开源 Linux 原生工具，旨在方便地使用开放容器项目 (OCI)容器和容器镜像查找、运行、构建、共享和部署应用程序。Podman 提供了对使用 Docker Container Engine 的任何人熟悉的命令行界面(CLI)。有关安装和使用 Podman 的更多信息，请参阅 Podman 安装说明。

2.1.1. 安装 CLI .zip 文件

流程

使用可下载的 **.zip** 文件安装：

1. 导航到 MTA Download 页面 并下载特定于操作系统的 CLI 文件或 **src** 文件：
   - mta-7.0.3-cli-linux.zip
   - mta-7.0.3-cli-macos.zip
   - mta-7.0.3-cli-windows.zip
   - mta-7.0.3-cli-src.zip
2. 将 **.zip** 文件提取到您选择的目录。**.zip** 文件提取一个二进制文件，名为 **mta-cli**。
   在本指南中遇到 **<MTA_HOME>** 时，将其替换为 MTA 安装的实际路径。

2.1.2. 使用 Podman 安装 CLI

先决条件

- **registry.redhat.io** 的 Red Hat Container Registry 身份验证。红帽从需要身份验证的 registry.redhat.io 中分发容器镜像。如需了解更多详细信息，请参阅 Red Hat Container Registry 身份验证。

流程

使用 **podman pull** 进行安装：

1. 使用 Podman 向 registry.redhat.io 进行身份验证：

```
podman login registry.redhat.io
Username: <username>
Password: <***********>
```

2. 问题：

```
podman cp $(podman create registry.redhat.com/mta-toolkit/mta-mta-cli-rhel9:
{ProductVersion}):/usr/local/bin/mta-cli ./
```

此命令将复制用于系统范围使用的二进制 **PATH**。

警告

虽然可以使用 Podman 安装，但下载并安装 **.zip** 文件是首选的安装。

2.1.3. 已知的 CLI 问题

Microsoft Windows 上的 Podman 的限制

CLI 构建并分发，支持 Microsoft Windows。

但是，当运行基于 Red Hat Enterprise Linux 9 (RHEL9)或通用基础镜像 9 (UBI9)的任何容器镜像时，启动容器时可能会返回以下错误：

```
Fatal glibc error: CPU does not support x86-64-v2
```

造成此错误的原因是，Red Hat Enterprise Linux 9 或通用基础镜像 9 容器镜像必须在支持 **x86-64-v2** 的 CPU 架构上运行。

如需了解更多详细信息，请参阅 (Running Red Hat Enterprise Linux 9 (RHEL)或 Universal Base Image (UBI) 9 容器镜像失败，并显示 "Fatal glibc error: CPU does not support x86-64-v2")。

CLI 可以正确运行容器运行时。但是，不支持不同的容器运行时配置。

虽然不支持，但您可以使用 **Docker** 而不是 **Podman** 运行 CLI，这将解决这个问题。

要达到此目的，您需要将 **PODMAN_BIN** 路径替换为 Docker 的路径。

例如，如果您遇到这个问题，而不是发出：

```
PODMAN_BIN=/usr/local/bin/docker mta-cli analyze
```

将 **PODMAN_BIN** 替换为 Docker 的路径：

```
<Docker Root Dir>=/usr/local/bin/docker mta-cli analyze
```

虽然不支持此功能，但您可以浏览 CLI，同时您负责升级硬件或迁移到支持 **x86_64-v2** 的硬件。

2.2. 运行 CLI

您可以针对您的应用程序运行 MTA。

流程

1. 打开一个终端，再进入 <**MTA_HOME**>/ 目录。
2. 为 Windows 执行 **mta-cli** 脚本或 **mta-cli.exe**，并指定适当的参数：

```
$ ./mta-cli analyze --input /path/to/jee-example-app-1.0.0.ear \
    --output /path/to/output --source weblogic --target eap6 \
```

   - **--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。

3. 访问报告。

2.2.1. MTA 命令示例

在应用程序存档中运行 MTA
以下命令分析 jee-example-app-1.0.0.ear 示例 EAR 存档，用于从 JBoss EAP 5 迁移到 JBoss EAP 7：

```
$ <MTA_HOME>/mta-cli analyze \
    --input /path/to/jee-example-app-1.0.0.ear \
    --output /path/to/report-output/ --source eap5 --target eap7 \
```

在源代码上运行 MTA
以下命令分析 Migrate -booking-5.2 示例源代码以迁移到 JBoss EAP 6。

```
$ <MTA_HOME>/mta-cli analyze --mode source-only --input /path/to/seam-booking-5.2/ \
    --output /path/to/report-output/ --target eap6 --packages org.jboss.seam
```

运行云就绪规则
以下命令分析用于迁移到 JBoss EAP 7 的 jee-example-app-1.0.0.ear 示例 EAR 存档。它还评估了云就绪情况：

```
$ <MTA_HOME>/mta-cli analyze --input /path/to/jee-example-app-1.0.0.ear \
    --output /path/to/report-output/ \
    --target eap7
```

2.2.2. 使用命令行执行分析

分析 允许使用 **分析器-lsp** 运行源代码和二进制分析。

要对应用程序源代码运行分析，请运行以下命令：

```
mta-cli analyze --input=<path/to/source/code> --output=<path/to/output/dir>
```

所有标记：

```
Analyze application source code
```

```
Usage:
  mta-cli analyze [flags]

Flags:
      --analyze-known-libraries   analyze known open-source libraries
  -h, --help                      help for analyze
  -i, --input string              path to application source code or a binary
      --json-output               create analysis and dependency output as json
      --list-sources              list rules for available migration sources
      --list-targets              list rules for available migration targets
  -l, --label-selector string     run rules based on specified label selector expression
      --maven-settings string     path to a custom maven settings file to use
      --overwrite                 overwrite output directory
      --skip-static-report        do not generate the static report
  -m, --mode string               analysis mode. Must be one of 'full' or 'source-only' (default "full")
  -o, --output string             path to the directory for analysis output
      --rules stringArray         filename or directory containing rule files
      --skip-static-report        do not generate the static report
  -s, --source string             source technology to consider for analysis. To specify multiple sources,
repeat the parameter: --source <source_1> --source <source_2> etc.
  -t, --target string             target technology to consider for analysis. To specify multiple targets,
repeat the parameter: --target <target_1> --target <target_2> etc.

Global Flags:
      --log-level uint32   log level (default 4)
      --no-cleanup         do not cleanup temporary resources
```

使用示例

1. 获取要运行分析的示例应用。
2. 列出可用的目标技术。

```
mta-cli analyze --list-targets
```

3. 使用指定的目标技术运行分析，如 **cloud-readiness**。

```
mta-cli analyze --input=<path-to/example-applications/example-1> --output=<path-to-output-dir> --target=cloud-readiness
```

4. 在指定的输出路径中创建了几个分析报告：

```
$ ls ./output/ -1
analysis.log
dependencies.yaml
dependency.log
output.yaml
static-report
```

output.yaml 是包含问题报告的文件。

static-report 包含静态 HTML 报告。

dependencies.yaml 包含依赖项报告。

2.2.3. 使用命令行执行转换

转换有两个子命令 - **openrewrite** 和 **rules**。

```
Transform application source code or mta XML rules

Usage:
  mta-cli transform [flags]
  mta-cli transform [command]

Available Commands:
  openrewrite Transform application source code using OpenRewrite recipes
  rules       Convert XML rules to YAML

Flags:
  -h, --help   help for transform

Global Flags:
      --log-level uint32   log level (default 4)
      --no-cleanup         do not cleanup temporary resources

Use "mta-cli transform [command] --help" for more information about a command.
```

2.2.3.1. OpenRewrite

openrewrite sub 命令允许在源代码上运行 **OpenRewrite** recipes。

```
Transform application source code using OpenRewrite recipes

Usage:
  mta-cli transform openrewrite [flags]

Flags:
  -g, --goal string     target goal (default "dryRun")
  -h, --help            help for openrewrite
  -i, --input string    path to application source code directory
  -l, --list-targets    list all available OpenRewrite recipes
  -s, --maven-settings string   path to a custom maven settings file to use
  -t, --target string   target openrewrite recipe to use. Run --list-targets to get a list of packaged
recipes.

Global Flags:
      --log-level uint32   log level (default 4)
      --no-cleanup         do not cleanup temporary resources
```

要在应用程序源代码上运行 转换 **openrewrite，请运行以下命令：**

```
mta-cli transform openrewrite --input=<path/to/source/code> --target=
<exactly_one_target_from_the_list>
```

注意

您只能使用单个目标来运行 转换**覆盖** 命令。

2.2.3.2. 规则

rules 子命令允许使用 **windup-shim** 将 mta XML 规则转换为分析器-lsp YAML 规则。

```
Convert XML rules to YAML

Usage:
  mta-cli transform rules [flags]

Flags:
  -h, --help                help for rules
  -i, --input stringArray   path to XML rule file(s) or directory
  -o, --output string       path to output directory

Global Flags:
      --log-level int   log level (default 5)
```

要在应用程序源代码上运行 转换规则，**请运行以下命令：**

```
mta-cli transform rules --input=<path/to/xmlrules> --output=<path/to/output/dir>
```

使用示例

1. 获取示例应用程序来转换源代码。
2. 查看可用的 OpenRewrite recipes。

   ```
   mta-cli transform openrewrite --list-targets
   ```

3. 在示例应用程序上运行方法。

   ```
   mta-cli transform openrewrite --input=<path-to/jakartaee-duke> --target=jakarta-imports
   ```

   检查 **jakartaee-duke** 应用程序源代码 diff 以查看转换

2.2.3.3. 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**	**<MTA_HOME>/rules/openrewrite/jakarta \ /javax/imports/rewrite.yml**	**org.jboss.windup.JavaxToJakarta**
Java EE 到 Jakarta EE	重命名 bootstrap 文件	**<MTA_HOME>/rules/openrewrite/jakarta \ /javax/bootstrapping/rewrite.yml**	**org.jboss.windup.jakarta.javax. \ BootstrappingFiles**

迁移路径	用途	rewrite.configLocation	activeRecipes
Java EE 到 Jakarta EE	转换 **persistence.xml** 配置	**<MTA_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**	**<MTA_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 Applications 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 MTA 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

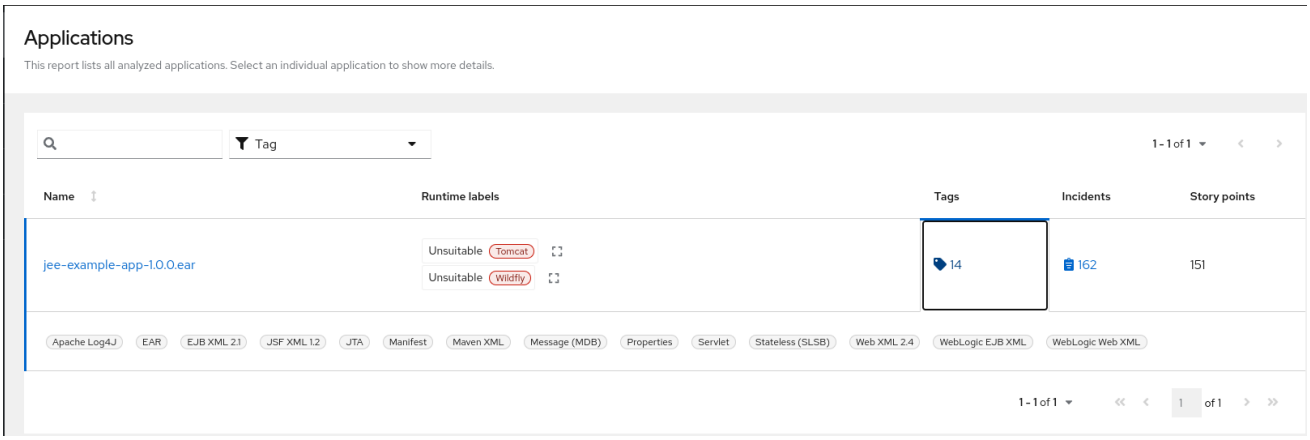
以下部分中显示的报告示例是分析 jee-example-app-1.0.0.ear 示例应用程序中的 **com.acme** 和 **org.apache** 软件包，它们位于 MTA GitHub 源存储库中。

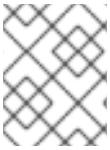
报告是通过以下命令生成的。

```
$ <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 eap6 --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

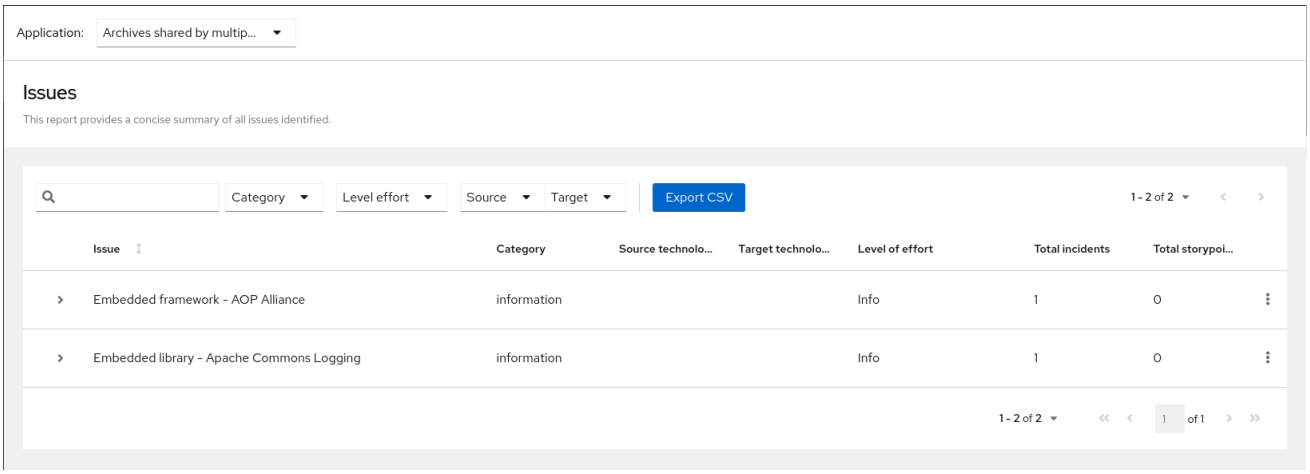
随着新规则添加到 MTA 时，事件和预计的故事点改变。测试此应用程序时，这里的值可能与您看到的值不同。

下表列出了可以从此主 MTA 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

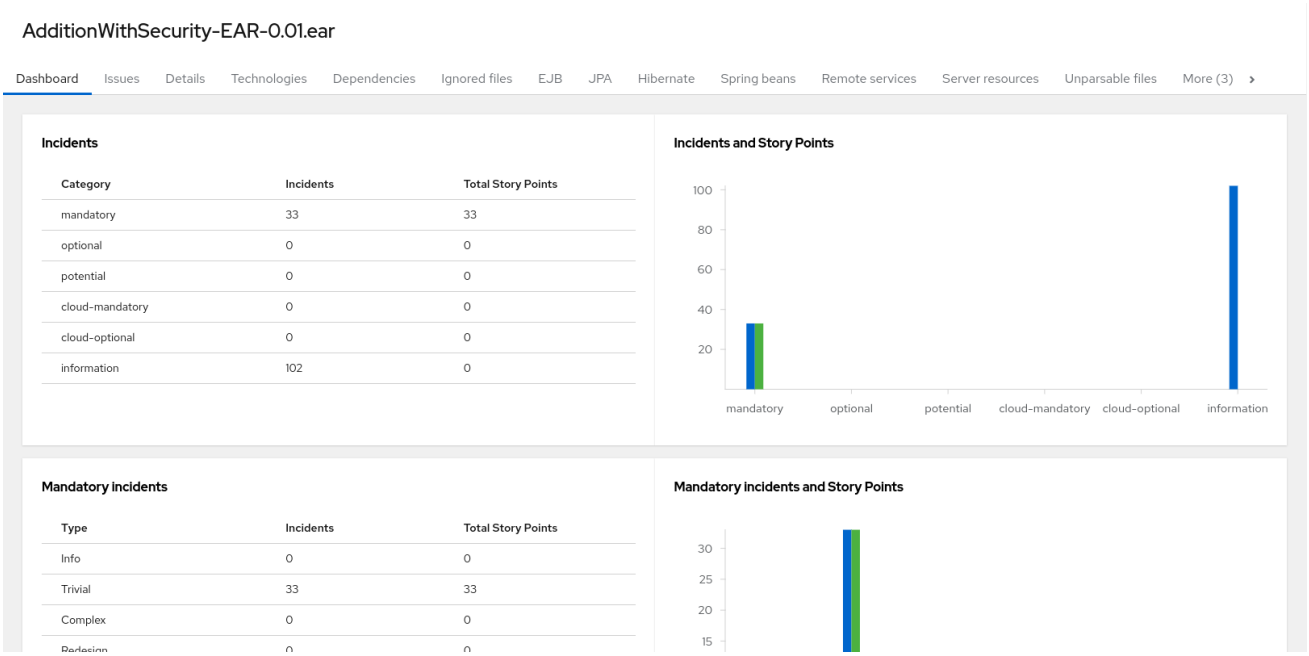
3.1.1. Dashboard

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 MTA 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 MTA 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 MTA，并为进一步帮助提供有用的链接。

3.1.2. 问题报告

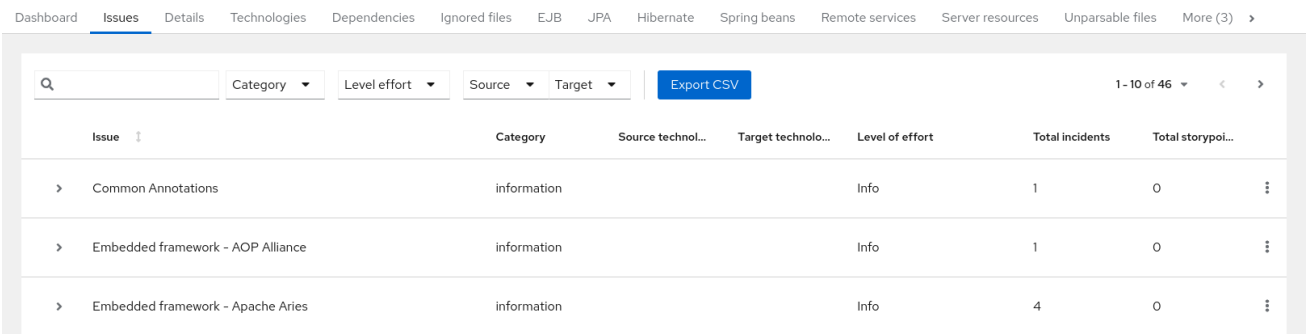
点 Problem 链接，从仪表板 访问 此报告。

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。

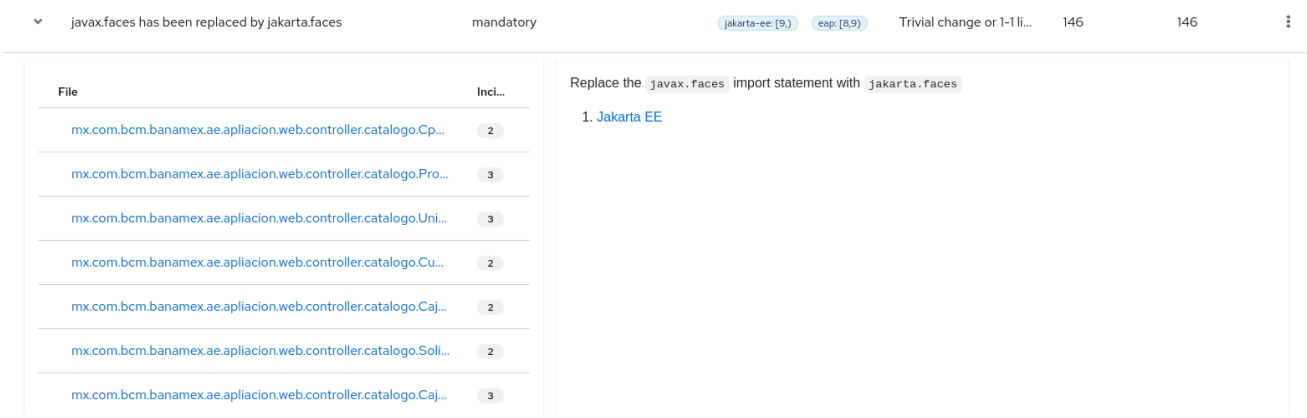
图 3.4. 问题报告



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



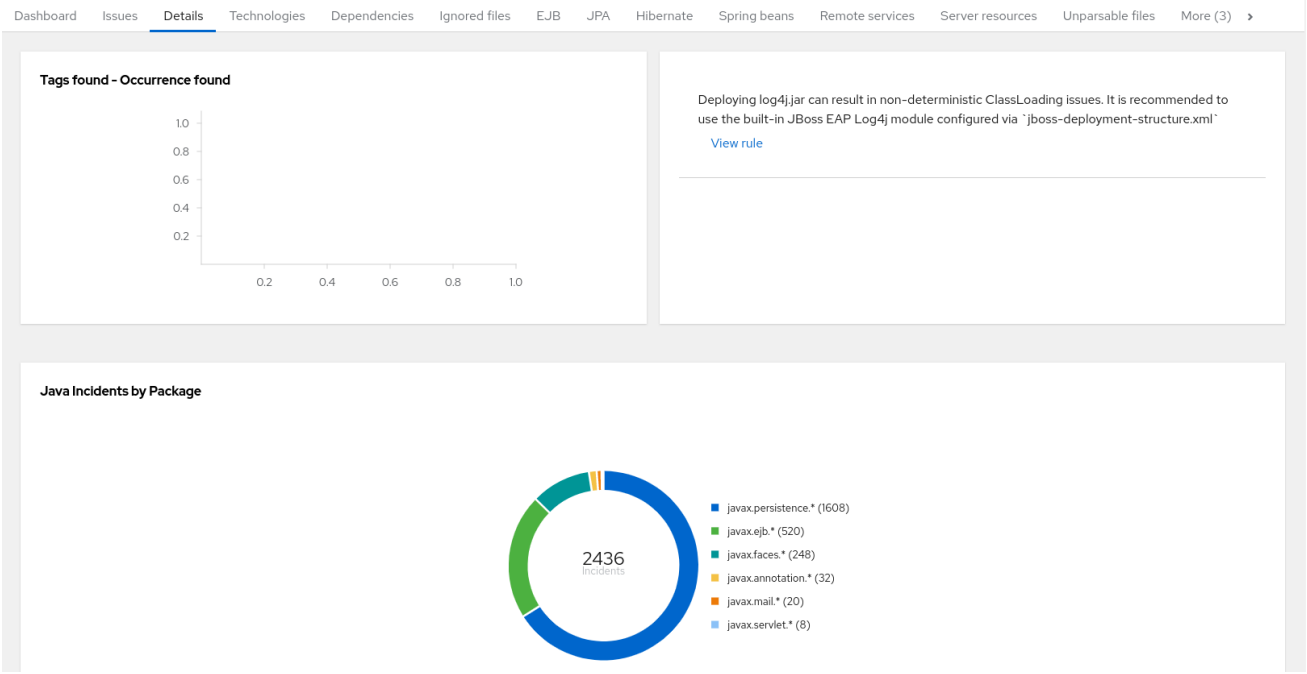
默认情况下，问题按照四个类别排序。有关这些类别的信息，请访问请求类别。

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

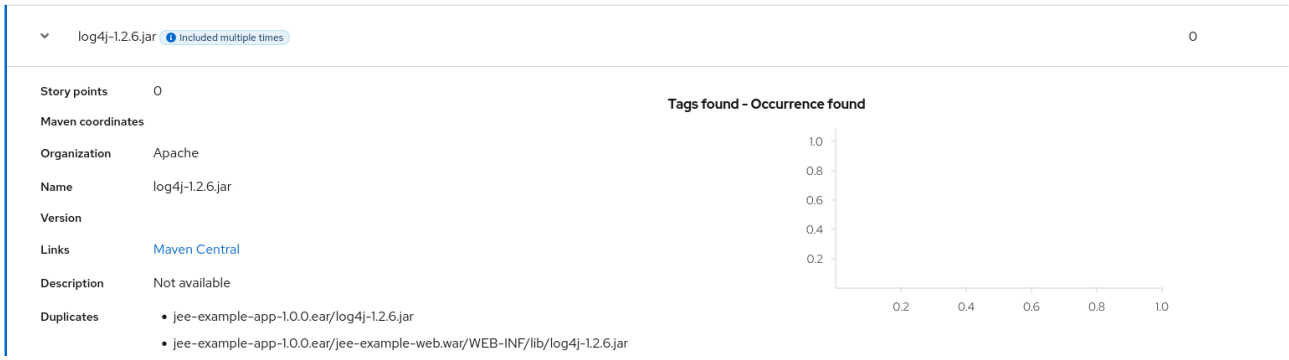


展开 **jee-example-app-1.0.0.ear/jee-example-services.jar** 以检查故事点、Java 事件软件包以及此存档中找到的技术计数。此摘要以分配给其迁移的故事总数开始，然后表详细说明存档中每个文件所需的更改。该报告包含以下几列。

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。

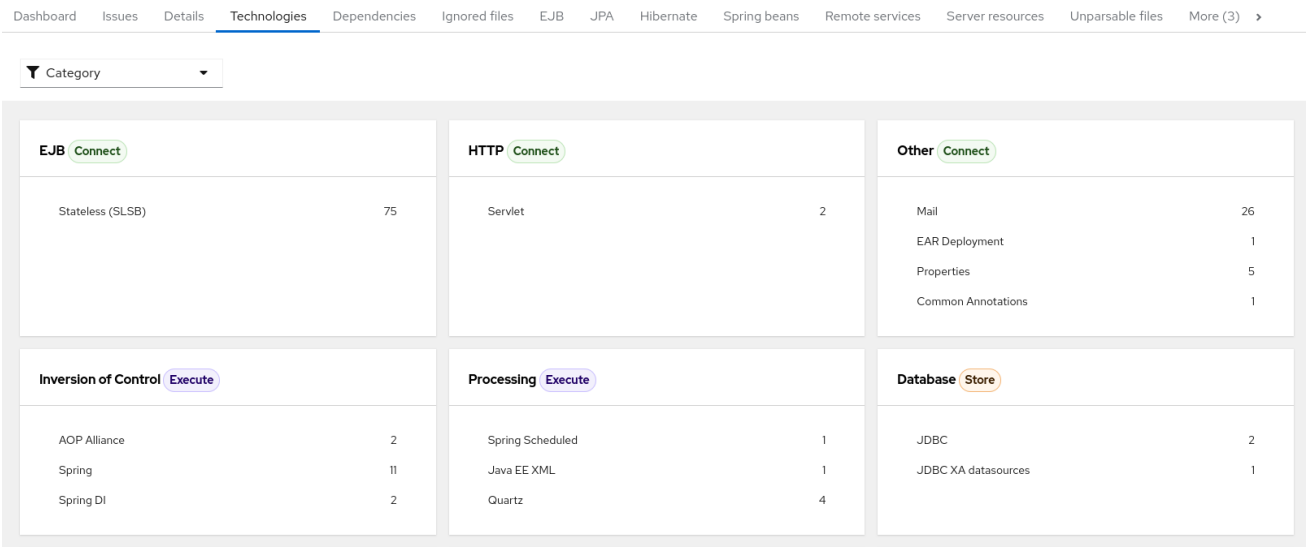
3.1.4. 技术报告

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。

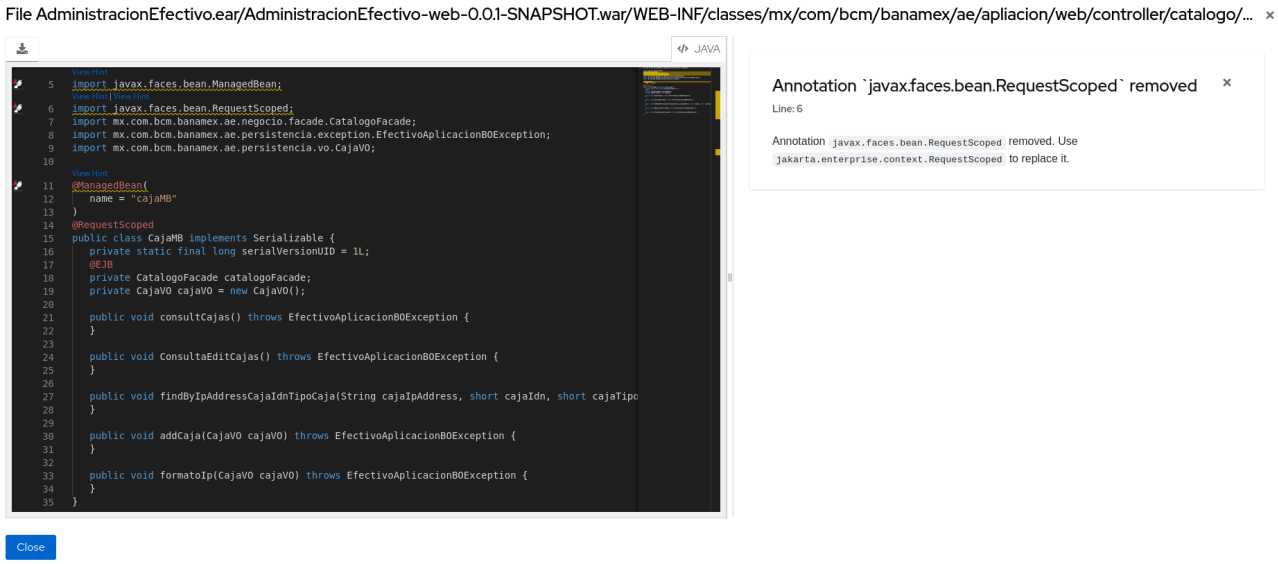
图 3.8. 应**用程序中的技**术



3.1.5. 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 3.9. **源**报**告**



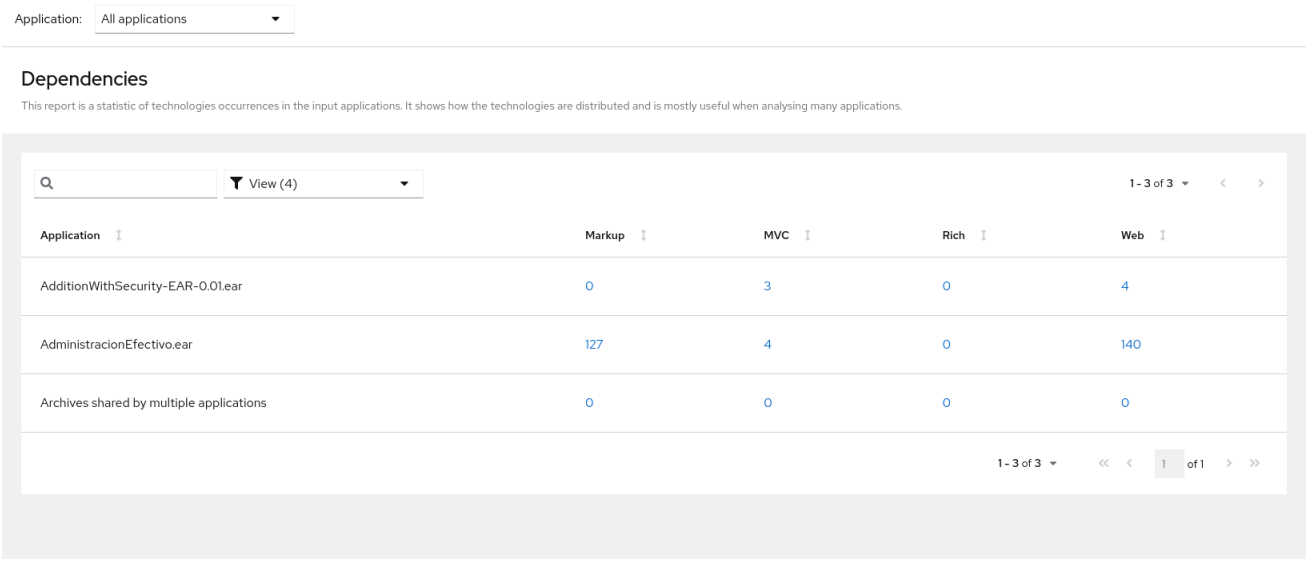
3.2. 技术报告

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 3.10. **跨多个应用程序使用的技术**



3.3. 选择软件包

MTA 要评估的软件包用空格分隔列表。强烈建议您使用此参数。

使用方法

- 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。