Red Hat build of Eclipse Vert.x 4.3

Eclipse Vert.x 4.3 发行注记

用于 Eclipse Vert.x 4.3.7

Last Updated: 2023-04-07

Red Hat build of Eclipse Vert.x 4.3 Eclipse Vert.x 4.3 发行注记

用于 Eclipse Vert.x 4.3.7

法律通告

摘要

本发行注记包含与 Eclipse Vert.x 4.3.7 相关的重要信息

目录

前言

发行日期： 2023 年 2 月 13 日

RED HAT BUILD OF ECLIPSE VERT.X 4.3.7 - PLANNED END OF LIFE

Red Hat build of Eclipse Vert.x 4.3.7 是红帽计划提供的最后一个受支持版本。全面支持于 2023 年 5 月 31 日结束。详情请查看产品生命周期页面。在产品结束前，红帽将继续为使用 4.3.x 版本的 Eclipse Vert.x 版本提供安全和程序错误修复。

您可以将 Eclipse Vert.x 应用程序迁移到红帽构建的 Quarkus。

Red Hat build of Quarkus

Quarkus 是一个针对 JVM 和原生编译量身定制的 Kubernetes 原生 Java 框架，使用最佳 Java 库和标准创建。它为在无服务器、微服务、容器、Kubernetes、FaaS 或云等环境中运行 Java 应用程序提供了有效的解决方案。
Quarkus 的被动功能在内部使用 Eclipse Vert.x，您可以在 Quarkus 中重复使用 Eclipse Vert.x 应用程序。因此，建议使用 Eclipse Vert.x 应用程序迁移到 Quarkus。

如需更多信息，请参阅 Quarkus 产品文档。https://access.redhat.com/documentation/zh-cn/red_hat_build_of_quarkus

我们将创建资源来帮助您完成迁移过程。

对红帽文档提供反馈

我们感谢您对我们文档的反馈。要提供反馈，您可以突出显示文档中的文本并添加注释。

本节介绍如何提交反馈。

前提条件

- 登录到红帽客户门户网站。
- 在红帽客户门户网站中，以 **多页 HTML** 格式查看文档。

流程

要提供反馈，请执行以下步骤：

1. 点击文档右上角的 **反馈** 按钮查看现有反馈。

注意

反馈功能仅在**多页 HTML** 格式中启用。

2. 高亮标记您要提供反馈的文档中的部分。
3. 点击在高亮文本旁边弹出的 **添加反馈**。
   文本框会出现在页面右侧的 feedback部分中。
4. 在文本框中输入您的反馈，然后点 **Submit**。
   已创建一个文档问题。
5. 要查看问题，请单击反馈视图中的问题跟踪器链接。

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 所需的基础架构组件版本

当您使用红帽构建的 Eclipse Vert.x 时，您可以使用以下组件：但是，除了 Red Hat OpenShift 集群和 Red Hat OpenJDK 之外，红帽不对下面列出的组件提供支持。

所需的组件

使用 Eclipse Vert.x 构建和开发应用程序需要以下组件。

组件名称	版本
maven	3.6.0 或更高版本
JDK[a]	8、11 或 17

[a] 需要完整的 JDK 安装，因为 JRE 不提供从源代码编译 Java 应用程序的工具。

可选组件

红帽建议根据您的开发和生产环境使用以下组件。

组件名称	版本
git	2.0 或更高版本
OpenShift Maven Plugin	1.1.1
oc 命令行工具	3.11 或更高版本[a]
访问 Red Hat OpenShift 集群[b]	3.11 或更高版本

[a] **oc** CLI 工具的版本应与您使用的 OCP 版本对应。

[b] OpenShiftCluster 受红帽支持

第 2 章 支持的 ECLIPSE VERT.X RUNTIME 组件配置和集成

以下资源定义了红帽产品与 Eclipse Vert.x 支持的配置和集成：

- 有关在生产环境中与 Eclipse Vert.x 集成的技术列表，请查看 支持的 Eclipse Vert.x 配置和集成。
- 如需 Eclipse Vert.x 运行时工件及其版本列表，请参阅 组件详情页面。

第 3 章 功能

3.1. 新增和改变的功能

本节论述了本发行版本中引入的新功能。它还包含有关现有功能更改的信息。

3.1.1. 4.3 版本中引入的新功能

Eclipse Vert.x 4.3 提供以下新功能或更改的功能。

3.1.1.1. Micrometer 将指标类型添加到 JMX 对象名称

从 Eclipse Vert.x 4.3.4 开始，因为升级到 Micrometer 1.9.3 时，对象名称现在在使用带 Java 管理扩展 (JMX)的 Eclipse Vert.x Micrometer Metrics 时包括指标类型。

这个增强只与 **micrometer-registry-jmx** 模块的用户相关。

3.1.1.2. 带有 GraphQL Java 19 的 Eclipse Vert.x 默认使用平台区域设置

从 Eclipse Vert.x 4.3.3 开始，Eclipse Vert.x 支持版本 19 of GraphQL Java，这是 GraphQL 查询语言的 Java 服务器实现。当使用 GraphQL Java 19 时，如果您未在 JVM 中设置区域设置，则 GraphQL 引擎现在使用 JVM 默认区域设置，这是安装 JVM 的平台的区域设置。另外，您可以将 JVM 默认 **Locale** 配置为使用不同的值，也可以使用 Eclipse Vert.x Web GraphQL 处理程序来设置自定义区域设置。

注意

Eclipse Vert.x 4.3.3 或更高版本也支持 GraphQL Java 版本 18。

3.1.1.3. 使用 **jackson-databind** 功能的用户必须将这个依赖项包含在其项目中

在 Eclipse Vert.x 4.3.2 onward 中，如果使用 Jackson Databind 库以及 **vertx-web-openapi**,**vertx-auth-webauthn** 或 **vertx-config-yaml** 模块，您必须将以下依赖项添加到项目描述符中：

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
</dependency>
```

由于 Jackson Databind 库已导致一些安全漏洞和其他模块通常只使用 Jackson Databind 来执行某些内部操作，所以使用 Eclipse Vert.x 解析器代替了任何需要使用 **vertx-web-openapi**、**vertx-auth-webauthn** 或 **vertx-config-yaml** 模块及 Jackson Databind。但是，如果您要继续使用 Jackson Databind 中的任何这些模块，则必须在项目中明确包含这个依赖项，如上例中所示。

3.1.1.4. 使用 Eclipse Vert.x OpenAPI 的正文处理器设置的更改

在 Eclipse Vert.x 4.3.1 onward 中，Eclipse Vert.x OpenAPI 需要使用 **routerBuilder.rootHandler （）** 方法，以确保在任何 PLATFORM 或 SECURITY_POLICY 处理程序 *后* 正确设置了正文处理器。

例如：

```
BodyHandler bodyHandler = BodyHandler.create("my-uploads");
routerBuilder.rootHandler(bodyHandler);
```

在较早版本的 Eclipse Vert.x 中，Eclipse Vert.x OpenAPI 支持 **routerBuild.bodyHandler （）** 方法来添加正文处理程序。但是，**bodyHandler （）** 方法有以下缺点：

- Eclipse Vert.x 没有执行任何验证，以确保设置正确顺序。
- Eclipse Vert.x OpenAPI 存储为特殊处理程序，以确保它始终是路由上的第一个处理器，但并不总是保证。

bodyHandler （） 方法在 Eclipse Vert.x 4.3.1 中弃用。前面的 **rootHandler** 调用现在会取代之前版本中提供的以下 **bodyHandler** 调用：

```
BodyHandler bodyHandler = BodyHandler.create("my-uploads");
routerBuilder.bodyHandler(bodyHandler);
```

3.1.1.5. 用于 BLOB 和 RAW 数据值的 Eclipse Vert.x 重新主动 Oracle 客户端增强

从 Eclipse Vert.x 4.3.1 开始，Eclipse Vert.x reactive Oracle 客户端包括以下对 **BLOB** 和 **RAW** 数据的改进：

- 当读取 **BLOB** 或 **RAW** 数据时，客户端现在返回一个 **io.vertx.core.buffer.Buffer** 值。
  例如：

```
client.preparedQuery("SELECT data FROM images WHERE id = ?")
    .execute(Tuple.of(id))
    .onComplete(ar -> {
        if (ar.succeeded()) {
            Row row = ar.result().iterator().next();

            // Use io.vertx.core.buffer.Buffer when reading
            Buffer data = row.getBuffer("data");
        }
    });
```

注意

这个更改是作为查询参数修复 **RAW** 值的问题的一部分而造成的一致性。在较早版本的 Eclipse Vert.x 中，**B** LOB 或 **RAW** 数据作为字节阵列返回。

- 在编写或过滤 **BLOB** 数据时，数据现在由新的 **io.vertx.oracleclient.data.Blob** 类型表示。
  例如：

```
client.preparedQuery("INSERT INTO images (name, data) VALUES (?, ?)")
    // Use io.vertx.oracleclient.data.Blob when inserting
    .execute(Tuple.of("beautiful-sunset.jpg", Blob.copy(imageBuffer)))
    .onComplete(ar -> {
        // Do something
    });
```

3.1.1.6. 默认禁用自动生成的密钥的检索

在 Eclipse Vert.x 4.3.1 onward 中，在 Eclipse Vert.x Oracle reactive 客户端中使用 Eclipse Vert.x 4.3.1，默认禁用了自动生成的密钥的检索。Eclipse Vert.x Oracle reactive 客户端通常不需要检索自动生成的密钥，因为大多数应用程序都不依赖于 **ROWID**。

此功能增强还促进 **INSERT...SELECT** 等查询，在启用自动检索后无法成功运行。

3.1.1.7. 使用 io.vertx.core.shareddata.ClusterSerialble 接口

在 Eclipse Vert.x 4.3.0 onward 中，Eclipse Vert **.x 支持 io.vertx.core.shareddata.ClusterSerial** ble 接口，用于读取对象并在缓冲区中回写出。

较早版本的 Eclipse Vert.x 支持 **io.vertx.core.shareddata.impl.ClusterSerialble** 接口。但是，由于该界面在实施软件包中提供，因此被视为不太可靠。**io.vertx.core.shareddata.impl.ClusterSerialble** 接口现在在 Eclipse Vert.x 4.3.0 中被弃用，并使它成为公共。

3.1.1.8. 为 Micrometer 请求指标重命名 requestsTagsProvider 选项

从 Eclipse Vert.x 4.3.0 onward，在 **MicrometerMetricsOptions** 类中，服务器请求指标的 **requestsTagsProvider** 选项被重命名为 **serverRequestTagsProvider**。此功能增强是必要的，因为现在还针对客户端请求指标提供了类似的 **clientRequestTagsProvider** 选项。

在早期的 Eclipse Vert.x 版本中，**requestTagsProvider** 选项使用 getter 和 a setter （名为 **getRequestsTagsProvider** 和 **setRequestsTagsProvider** ）。在 Eclipse Vert.x 4.3.0 及更高版本中，**serverRequestTagsProvider** 选项的 getter 和 setter 重命名为 **getServerRequestTagsProvider**，**setServerRequestTagsProvider**。

3.1.1.9. OAuth2 OBO 调用预期明确的 OAuth2Credentials 而不是 TokenCredentials

在 Eclipse Vert.x 4.3.0 onward 中，当 OAuth2 授权配置为 on-behalf-of (OBO)模式时，OAuth2 要求明确指定 **OAuth2Credentials** 对象来授权请求。

例如：

```
oauth2.authenticate(
  new Oauth2Credentials().setAssertion("head.body.signature").addScope("a").addScope("b"))
```

在较早版本的 Eclipse Vert.x 中，OBO 模式中的 OAuth2 授权允许使用 **TokenCredentials**。但是，因为网络流是可选的，以允许重复使用同一 **OAuth2Credentials** 对象，前面的 **Oauth2Credentials** 调用现在会取代之前版本中可用的以下 **TokenCredentials** 调用：

```
oauth2.authenticate(
  new TokenCredentials("head.body.signature").addScope("a").addScope("b"));
```

3.1.1.10. RoutingContext.fileUploads （） 方法返回一个列表

在 Eclipse Vert.x 4.3.0 onward 中，**RoutingContext.fileUploads （）** 方法返回一个 **List<FileUpload>** 值。在列表中存储文件上传有助于保留上传顺序。

例如：

```
List<FileUpload> uploads = ctx.fileUploads();
```

在较早版本的 Eclipse Vert.x 中，**RoutingContext.fileUploads （）** 方法返回 **Set<FileUpload>** 值。但是，在集合中存储文件上传与表单内容类型的 World Wide Web Consortium (W3C)规格不一致，因为它没有保留正确的顺序，用户无法依赖上传名称作为唯一键。现在，前面的示例替换了之前版本中提供的以下方法声明：

```
Set<FileUpload> uploads = ctx.fileUploads();
```

3.1.1.11. 实施子路由器的单方法

从 Eclipse Vert.x 4.3.0 onward， **Route.subRouter (Router)** 方法是实施子路由器唯一支持的方法。

早期版本的 Eclipse Vert.x 支持两种不同的方法来实现子路由器：

- **Route.subRouter(Router)**
- **router.mountSubRouter (String, Router)**

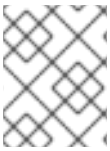
但是，这两种方法之间的行为不一致，因为 **Router.mountSubRouter** 方法允许任何路径，而 **Route.subRouter** 方法明确需要一个通配符星号(*)来代表子路由路径。 **Router.mountSubRouter** 方法也通过附加缺少的通配符来委派给 **Route.subRouter** 方法。

在 Eclipse Vert.x 4.3.0 及更高版本中，**Router.mountSubRouter** 方法已弃用。路由器对象现在还必须使用 **Route.subRouter** 方法来实施子路由器。例如：

```
router.route("/eventbus/*").subRouter(otherRouter);
```

前面的 **router.route （）.subRouter （）** 调用现在取代之前版本中可用的以下类型的 **router.mountSubRouter （）** 调用：

```
router.mountSubRouter("/eventbus", otherRouter);
```



注意

在以前的版本中，路由器对象也可以使用 **router.route （）.subRouter （）** 调用作为使用 **router.mountSubRouter （）** 的替代选择。

3.1.1.12. 在多个处理程序调用之间解析请求正文的缓存

从 Eclipse Vert.x 4.3.0 onward，在正文处理程序解析 web 请求的正文后，正文处理程序会将正文缓冲区提供给请求的路由上下文。 在路由上下文中缓存正文缓冲区意味着需要请求正文正文的多个不同处理程序可以获取缓存的结果，而无需再次解析正文。 此增强还支持正文内容为 **application/json** 类型的情况。

RoutingContext 类提供了一个新的 **body （）** 方法，用于将请求正文作为指定类型获取。

例如：

```
RoutingContext.body().asString()
RoutingContext.body().asString(String encoding)
RoutingContext.body().asJsonObject()
RoutingContext.body().asJsonArray()
RoutingContext.body().asJsonObject(int maxLength)
RoutingContext.body().asJsonArray(int maxLength)
RoutingContext.body().buffer()
```

新的 **body （）** getter 还提供以下额外功能：

```
// the length of the buffer (-1) for null buffers
RoutingContext.body().length()
```

```
// Converting to POJO
RoutingContext.body().asPOJO(Class<T> clazz)
RoutingContext.body().asPOJO(Class<T> clazz, int maxLength)
```

此功能增强提供以下优点：

- **body （）** getter 不会为空，这有助于避免任何需要执行 null 检查。
- 请求正文只需要解析一次，除非基本缓冲区变化。对基本缓冲区的任何更改都会触发另一个解析，缓存的值会在此时被覆盖。

在早期的 Eclipse Vert.x 版本中，**RoutingContext** 类提供了以下使用 **body （）** 方法被弃用的方法：

```
RoutingContext.getBodyAsString()
RoutingContext.getBodyAsString(String encoding)
RoutingContext.getBodyAsJson()
RoutingContext.getBodyAsJsonArray()
RoutingContext.getBodyAsJson(int maxLength)
RoutingContext.getBodyAsJsonArray(int maxLength)
RoutingContext.getBody()
```

3.1.1.13. EventBus 通知默认值的变化

从 Eclipse Vert.x 4.3.0 onward 中，为了避免不必要的流量，Eclipse Vert.x 断路器状态会禁用有关 Eclipse Vert.x 断路器状态的通知。要启用这些通知，请使用没有 null 的参数调用 **CircuitBreakerOptions** 对象的 **setNotificationAddress** 方法。

例如：

```
CircuitBreakerOptions options = new CircuitBreakerOptions()
  .setNotificationAddress(CircuitBreakerOptions.DEFAULT_NOTIFICATION_ADDRESS);
```

当您按照上例中所示启用通知时，默认行为是仅将通知发送到本地使用者。要在集群范围内发送通知，请使用参数 **false** 调用 **setNotificationLocalOnly** 方法。

例如：

```
CircuitBreakerOptions options = new CircuitBreakerOptions()
  .setNotificationAddress(CircuitBreakerOptions.DEFAULT_NOTIFICATION_ADDRESS)
  .setNotificationLocalOnly(false);
```

3.1.1.14. MySQL 客户端批处理执行的变化

在 Eclipse Vert.x 4.3.0 onward 中，Eclipse Vert.x reactive SQL 客户端支持管道查询，并在 pipelining 模式中运行批处理查询。pipelining 表示请求会在同一连接上发送，而不等待对之前的请求的响应。

在较早版本的 Eclipse Vert.x 中，由于 MySQL 不支持批处理，因此 SQL 客户端通过按顺序运行准备的查询来运行批量查询，因此用户可以直接通过 API 调用操作。

3.1.1.15. 用于提示和提示字符串的 MongoDB 增强

在 Eclipse Vert.x 4.3.0 onward 中，Eclipse Vert.x 包括以下用于提示和提示字符串的 MongoDB 增强：

- **FindOptions** 对象现在支持类型为 **JSONObject** 的 hints。这会取代之前版本中的行为，其中 **FindOptions** 对象支持提示类型 **String**。
- **BulkOperations** 和 **UpdateOptions** 对象现在支持类型为 **JSONObject** 的提示。**BulkOperations** 和 **UpdateOptions** 类各自提供 **getHint （）** 和 **setHint （）** 方法。
- **BulkOperations**、**UpdateOptions** 和 **FindOptions** 对象现在也支持类型为 **String** 的提示字符串。**BulkOperations**、**UpdateOptions** 和 **FindOptions** 类各自提供 **getHintString （）** 和 **setHintString （）** 方法。

3.1.1.16. 构建模式的更改

在构建 schema 时，从 Eclipse Vert.x 4.3.0 onward 中，使用 Eclipse Vert.x JSON 模式提供的 JSON 表示。JSON 表示允许使用任何验证器。

例如：

```
JsonSchema schema = JsonSchema.of(dsl.toJson());
```

在早期的 Eclipse Vert.x 版本中，**SchemaBuilder** 类提供了 **build （）** 方法，它使用特定验证器实施。**build （）** 方法在 Eclipse Vert.x 4.3.0 中弃用。前面的 JsonSchema 示例现在取代之前版本中提供的以下 **build （）** 方法调用：

```
Schema schema = dsl.build(parser);
```

3.1.2. 之前 4.x 版本中引入的新功能

在之前的 4.x 版本中引入了以下新功能。

3.1.2.1. Java 17 支持

在 Eclipse Vert.x 4.2.7 onward 中，Eclipse Vert.x 已通过 Red Hat OpenJDK 17 认证。

3.1.2.2. RequestOptions中的 HTTP 标头验证

在 Eclipse Vert.x 4.2.4 onward 中，**RequestOptions** 方法验证 HTTP 标头，如果标头名称无效，请求会失败。

在较早版本的 Eclipse Vert.x 中，**HTTPClientRequest** 验证 HTTP 标头，因为 **RequestOptions** 方法使用了没有验证标头名称的 Multimap 实施。

3.1.2.3. 使用 simple 作为本地化的默认区域设置

在 Eclipse Vert.x 4.2.4 onward 中，简单 区域被用作 MongoDB 冲突的默认区域设置。

Eclipse Vert.x 4.2.3 引入了对 collation 选项的支持，用于支持比较字符串的特定语言规则。在 Eclipse Vert.x 4.2.3 中，平台 default 被用作默认的区域设置。但是，因为平台默认设置不是恒定值，因此当使用 MongoDB 不支持的区域的系统中，可能会导致失败。例如，**Locale.FR** 可以成功运行，但不支持 **Locale.FR_FR**

3.1.2.4. StaticHandler 文件系统配置更改

从 Eclipse Vert.x 4.2.4 onward 中，webroot 目录和文件系统访问权限的 **StaticHandler** 配置属性在 **StaticHandler** 工厂构造器调用中定义。

例如，以下构造器调用定义了一个 webroot 目录、**静态/资源** 以及相对文件系统访问：

```
StaticHandler.create(FileSystemAccess.RELATIVE, "static/resources");
```

例如，以下构造器调用定义了一个 webroot 目录、**/home/paulo/Public**、root 文件系统访问：

```
StaticHandler.create(FileSystemAccess.ROOT, "/home/paulo/Public");
```

在较早版本的 Eclipse Vert.x 中，允许使用 setters 定义 **allowRootFileSystemAccess** 和 **webroot** 属性。但是，这些值不是最终的，这可能会导致无效的静态配置。在 Eclipse Vert.x 4.2.4 中，上述构造器调用现在取代以下 setter 声明：

```
StaticHandler.create()
  .setAllowRootFileSystemAccess(true)
  .setWebRoot("/home/paulo/Public");
```

注意

StaticHandler.create （） 方法仍然使用默认值 **RELATIVE** 和 **webroot**，如之前版本中一样。

3.1.2.5. verticle 中的随机服务器端口共享

从 Eclipse Vert.x 4.2.0 开始，与负端口号绑定的两个不同的 HTTP 服务器（如 **-1**） 在特定验证部署的实例中共享相同的随机端口。这意味着多个带有端口 **-1** 的 HTTP 服务器将共享相同的随机端口。同样，使用端口 **-2** 绑定的多个 HTTP 服务器将共享相同的随机端口，以此类推。此端口共享基于负端口号的行为独立于白色，因为它允许不同的 HTTP 服务器具有不同的随机端口。

在较早版本的 Eclipse Vert.x 中，随机服务器端口共享基于两个 HTTP 服务器，与端口 **0** 绑定。但是，这可以防止同一聚合将两个带有不同随机端口的 HTTP 服务器绑定在同一顶端的实例中。

3.1.2.6. HTTP 服务器 cookie 更改

Eclipse Vert.x 4.2.0 包含一个新方法 **Set<Cookie> cookies （）**，它允许获取所有 Cookie。

在较早版本的 Vert.x 中，H **ttpServerRequest** 和 **HttpServerResponse** 接口使用在 Eclipse Vert.x 4.2.0 中弃用的以下方法：

```
Map<String, Cookie> cookieMap()
```

RFC 6265 - HTTP State Management Mechanism 规格指出每个 Cookie 都是根据 tuple < **name, domain, path** > 的唯一标识。但是，在以前的 Eclipse Vert.x 版本中使用 **Map<String, Cookie> cookieMap （）** 方法错误地假定 cookies 只能根据其名称来标识。这意味着，当多个 Cookie 共享同一名称时，要解析最后一个 Cookie 的映射都会被静默覆盖。

3.1.2.7. 使用 GraphQLContext 对象进行上下文管理

Eclipse Vert.x 4.2.0 支持 GraphQL Java 版本 17，这是 GraphQL 查询语言的 Java 服务器实施。使用 GraphQL Java 17 时，**GraphQLContext** 对象现在是在 GraphQL Java 应用程序组件间共享上下文数据的标准。

Eclipse Vert.x 4.2.0 引入了以下新机制来配置 GraphQL 执行：

```
GraphQLHandler handler = GraphQLHandler.create(graphQL).beforeExecute(builderWithContext ->
{
  DataLoader<String, Link> linkDataLoader = DataLoaderFactory.newDataLoader(linksBatchLoader);
  DataLoaderRegistry dataLoaderRegistry = new DataLoaderRegistry().register("link",
linkDataLoader);
  builderWithContext.builder().dataLoaderRegistry(dataLoaderRegistry);
});
```

在早期的 Eclipse Vert.x 版本中，以下 hook 用于 Vert.x Web GraphQL 处理程序来配置数据加载程序。以下 hook 现在在 Eclipse Vert.x 4.2.0 中弃用。

```
GraphQLHandler handler = GraphQLHandler.create(graphQL).dataLoaderRegistry(rc -> {
  DataLoader<String, Link> linkDataLoader = DataLoader.newDataLoader(linksBatchLoader);
  return new DataLoaderRegistry().register("link", linkDataLoader);
});
```

3.1.2.8. OpenJDK11 OpenShift 镜像支持多个构架

用于 IBM Z 和 IBM Power 系统的 OpenJ9 镜像已弃用。以下 OpenJDK11 镜像已更新，以支持多个架构：

- **ubi8/openjdk-11**

您可以将 OpenJDK11 镜像用于以下架构：

- x86 (x86_64)
- s390x (IBM Z)
- ppc64le (IBM Power Systems)

3.1.2.9. 支持在启用了 FIPS 的 Red Hat Enterprise Linux (RHEL)系统中的 Eclipse Vert.x 运行时

Red Hat build of Eclipse Vert.x 在启用了 FIPS 的 RHEL 系统上运行，并使用 RHEL 提供的 FIPS 认证库。

3.1.2.10. HTTP 客户端重定向处理程序传播标头

从 Eclipse Vert.x 4.1.0 onward 中，如果 HTTP 重定向中存在标头，则 HTTP 客户端重定向处理程序会将标头传播到下一个请求。这个更改可让重定向处理程序对整个重定向请求具有更多的控制。

在早期的 Eclipse Vert.x 版本中，使用标头重定向请求，HTTP 客户端会在重定向后处理标头。

以下示例演示了如何在 Eclipse Vert.x 4.1.0 中处理重定向：

```
RequestOptions options = new RequestOptions();
options.setMethod(HttpMethod.GET);
options.setHost(uri.getHost());
options.setPort(port);
options.setSsl(ssl);
options.setURI(requestURI);
```

```
// From 4.1.0 propagate headers
options.setHeaders(resp.request().headers());
options.removeHeader(CONTENT_LENGTH);
```

3.1.2.11. 升级到 Infinispan 12

在 Eclipse Vert.x 4.1.0 中，Infinispan 集群管理器已更新，并基于 Infinispan 12。

Infinispan 11 存在一个程序漏洞，它允许在多映射缓存中存储字节阵列。作为临时解决方案，Eclipse Vert.x 集群管理器必须使用内部 Infinispan 类 **WrappedBytes** 来存储事件总线订阅数据。这个问题已在 Infinispan 12 中解决。

3.1.2.12. JSON 配置优先于 MongoDB 客户端中的连接字符串选项

在 Eclipse Vert.x 4.1.0 中，即使有 **connection_string** 选项，也会应用 JSON 配置选项。

现在应用以下配置选项：

```
{
   mongo:{
      db_name: "mydb"
      connection_string: "mongodb://localhost:27017"
      maxPoolSize: 10
      minPoolSize: 3
   }
}
```

在较早版本的 Eclipse Vert.x 中，连接字符串可用时会忽略 JSON 配置选项。例如，考虑前面的示例：在较早版本的 Eclipse Vert.x、**db_name**、**maxPoolSize** 和 **minPoolSize** 选项将被忽略。

3.1.2.13. 删除了已弃用的 JWT 选项方法

在 Eclipse Vert.x 4.0 上，使用 JWT 和 OAuth2 处理程序来处理范围。

从 Eclipse Vert.x 4.1.0 onward, **JWTOptions.setScopes (List<String>)**, **JWTOptions.addScope (String)** 和 **JWTOptions.withScopeDelimiter (String)** 方法已被删除。这些方法不符合规格。

以下示例演示了如何处理 Eclipse Vert.x 4.1.0 中的范围。

```
// before 4.1.0
JWTAuthOptions authConfig = new JWTAuthOptions()
  .setJWTOptions(new JWTOptions()
    .addScope("a")
    .addScope("b")
    .withScopeDelimiter(" ")));

JWTAuth authProvider = JWTAuth.create(vertx, authConfig);

router.route("/protected/*").handler(JWTAuthHandler.create(authProvider));

// in 4.1.0
JWTAuth authProvider = JWTAuth.create(vertx, new JWTAuthOptions());

router.route("/protected/*").handler(
  JWTAuthHandler.create(authProvider)
```

```
    .addScope("a")
    .addScope("b")
    .withScopeDelimiter(" "));
```

3.1.2.14. 弃用了接受函数的自定义格式器方法

从 Eclipse Vert.x 4.1.0，**LoggerHandler.customFormatter (Function)** 方法已弃用。此函数取为 **HttpServerRequest** 的输入并返回格式化的日志字符串。由于输出是一个字符串，因此无法访问上下文。

使用新方法 Logger **Handler customFormatter (LoggerFormatter)** 替代。该方法取为自定义格式，提供对上下文的访问权限。

3.1.2.15. 处理 HTTP 失败的新例外

从 Eclipse Vert.x 4.1.0，提供了一个新的异常类 **io.vertx.ext.web.handler.HttpException**，可用于处理 HTTP 故障。您可以使用例外指定 500 以外的自定义状态代码。例如，新的 **HttpException (401, "Forbidden")** 表示禁止的请求应返回 status code 401。

3.1.2.16. 支持 RxJava 3

在 Eclipse Vert.x 4.1.0 中，支持 RxJava 3。

- 一个新的 rxified API 位于 **io.vertx.rxjava3** 软件包中。
- 与 Eclipse Vert.x JUnit5 集成由 **vertx-junit5-rx-java3** 绑定提供。

3.1.2.17. 上下文服务器拦截器绑定所有类型的数据，并更安全

从 Eclipse Vert.x 4.0.3，**ContextServerInterceptor.bind （）** 方法会将所有类型的数据绑定到上下文。现在，该方法更为安全，因为它不会公开存储详情。

在 Eclipse Vert.x 4.0.3 之前的发行版本中，仅用于将 'String' 数据类型绑定到上下文的方法。它还公开了存储详情。

要使用更新的 **ContextServerInterceptor.bind （）** 方法，您必须更新您的应用程序。

以下示例显示了 Eclipse Vert.x 4.0.3 之前的发行版本中的代码。

```
// Example code from previous releases

class X extends ContextServerInterceptor {
  @Override
  public void bind(Metadata metadata, ConcurrentMap<String, String> context) {
```

以下示例显示了替换代码 fpr Eclipse Vert.x 4.0.3 发行版本。

```
// Replacing code for Eclipse Vert.x 4.0.3 release

class X extends ContextServerInterceptor {
  @Override
  public void bind(Metadata metadata) {
```

3.1.2.18. 不再需要匹配以通配符结尾的路由路径(/)

在 Eclipse Vert.x 4.0.3 之前的发行版本中，如果路由以斜杠结尾，则路由仅包括结束斜杠 / 时，才会调用路由。当通配符为空时，这个规则会导致问题。

从 Eclipse Vert.x 4.0.3 onward 中，这个规则不再应用。您可以创建路径以斜杠(/)结尾的路由。但是，不需要指定请求 URL 中的斜杠。

另外，您可以创建并使用请求 URL 调用以通配符以其路径而不是斜杠(/)结尾的路由。例如，带有通配符的路由可以定义为 /foo/*。此处，该路由必须与路径末尾的打开通配符匹配。请求 URL 可以是 /foo。

在发送请求 URL /foo/* 时，表显示了 Eclipse Vert.x 4.0.3 和之前的版本中的行为。您可以看到，结尾斜杠在 Eclipse Vert.x 4.0.3 中是可选的，并且请求与路由匹配。

Route（路由）	Eclipse Vert.x 4.0.3	Eclipse Vert.x 4.0.3 之前的发行版本
/foo	匹配	没有 Match
/foofighters	没有 Match	没有 Match
/foo/	匹配	匹配
/foo/bar	匹配	匹配

3.1.2.19. 从服务发现选项中删除 autoRegistrationOfImporters 属性

autoRegistrationOfImporters 属性已从服务发现选项中删除。

3.1.2.20. 验证供应商类中的验证方法已更新，以支持 令牌 作为输入凭证

在 Eclipse Vert.x 4.0.3 之前的发行版本中，AuthenticationProvider.authenticate （）方法错误地采用 jwt: someValue 作为输入凭证。

从 Eclipse Vert.x 4.0.3，AuthenticationProvider.authenticate （）方法已更新，并取 token: someValue 作为输入凭证。此更改可确保 JSON 和键入的 API 都一致，并可互换使用。

以下代码显示了在 Eclipse Vert.x 4.0.3 之前对版本验证方法的实现。

```
new JsonObject().put("jwt", "token...");
```

-

以下代码显示了 Eclipse Vert.x 4.0.3 发行版本中对验证方法的实现。

```
new JsonObject().put("token", "token...");
```

3.1.2.21. 获取 PEM 密钥的方法返回 缓冲 程序，而不是 String

PubSecKeyOptions.getBuffer （） 方法返回 PEM 或 secret 密钥缓冲。在 Eclipse Vert.x 4.0.2 之前的发行版本中，密钥缓冲区被存储并返回为 String。但是，建议将 secret 保存为 缓冲区。在 Eclipse Vert.x 4.0.2 onward 中，方法存储并将密钥缓冲区返回为 缓冲。这个更改可提高 secret 的安全性和处理。

PubSecKeyOptions.setBuffer （） 方法继续接受 String 参数。在 set 方法中，添加了对缓冲的超载，以安全地处理非 ASCII 机密资料。这个更改不需要对现有代码进行任何更改。

3.1.2.22. Kubernetes 服务导入程序不再自动注册

从 Eclipse Vert.x 4 中，KubernetesServiceImporter 发现网桥不再会自动注册。即使您已在 Maven 项目的类路径中添加了网桥，它也不会自动注册。

在创建 ServiceDiscovery 实例后，您必须手动注册该网桥。

3.1.2.23. 对异步操作使用将来的方法

Eclipse Vert.x 4 使用将来的异步操作。每个回调方法都有对应的未来方法。

将来可用于编写异步操作。当您使用将来时，错误处理会更好。因此，建议将回调和将来在您的应用程序中结合使用。

3.1.2.24. 不依赖于 Jackson Databind 库

在 Eclipse Vert.x 4 中，Jackson Databind 是一个可选 Maven 依赖项。如果要使用这个依赖项，您必须在 classpath 中明确添加它。例如，如果您要对象映射 JSON，则必须明确添加依赖项。

3.1.2.25. 处理弃用和删除

在 Eclipse Vert.x 4 中，提供了新的增强功能。在 Eclipse Vert.x 4 中弃用或删除旧的功能。在将应

用程序迁移到 Eclipse Vert.x 4 之前，请检查 弃用和删除。

使用已弃用的 API 时，Java 编译器会生成警告。在将应用迁移到 Eclipse Vert.x 4 时，您可以使用编译器检查已弃用的方法。

3.1.2.26. 支持分布式追踪

Eclipse Vert.x 4 支持分布式追踪。您可以使用追踪来监控微服务并识别性能问题。

Eclipse Vert.x 4 与 OpenTracing 系统集成。

以下 Eclipse Vert.x 组件可以记录追踪：

- HTTP 服务器和 HTTP 客户端
- Eclipse Vert.x SQL 客户端
- Eclipse Vert.x Kafka 客户端

重要

跟踪作为技术预览提供。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。

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。

3.1.2.27. EventBus JavaScript Client 的新发布位置

在 Eclipse Vert.x 4 中，EventBus JavaScript 客户端 vertx-web-client.js 没有在 Maven 存储库中作为红帽工件发布。

客户端在 npm 存储库中发布。您可以从以下位置访问客户端 ：@vertx/eventbus-bridge-client.js

3.1.2.28. 使用 OpenShift Maven 插件部署 Eclipse Vert.x 应用

使用 OpenShift Maven 插件在 OpenShift 上部署 Eclipse Vert.x 应用。Fabric8 Maven 插件不再被支持。如需更多信息，请参阅从 Fabric8 Maven 插件迁移到 Eclipse JKube 的部分。

3.1.2.29. 用于 OpenShift 的 Eclipse Vert.x metering 标签

您可以在 Eclipse Vert.x pod 中添加 metering 标签，并使用 OpenShift Metering Operator 检查红帽订阅详情。

注意

- 不要将 metering 标签添加到 Operator 或模板部署和管理的任何 pod 中。
- 您可以使用 OpenShift Container Platform 版本 4.8 及更早版本上的 Metering Operator 将标签应用到 pod。从 4.9 版本中，在没有直接替换的情况下，Metering Operator 不再可用。

Eclipse Vert.x 应使用以下 metering 标签：

- com.company: Red_Hat
- rht.prod_name: Red_Hat_Runtimes
- rht.prod_ver: 2023-Q1
- rht.comp: Vert.x
- rht.comp_ver: 4.3.7

- rht.subcomp: <leave_blank>
- rht.subcomp_t: application

其他资源

- 在 OpenShift Container Platform 中配置和使用 Metering

3.1.2.30. 支持 OpenJDK 8 和 OpenJDK 11 RHEL 8 通用基础镜像(UBI8)

Eclipse Vert.x 引进了对使用 Red Hat OpenJDK 8 和 Red Hat OpenJDK 11 的 OCI 兼容 通用基础镜像 在 OpenShift 中构建和部署 Eclipse Vert.x 应用程序的支持。https://access.redhat.com/documentation/zh-cn/red_hat_enterprise_linux/8/

RHEL 8 OpenJDK 通用基础镜像替换 RHEL 8 OpenJDK 构建器镜像。RHEL 8 OpenJDK 基础镜像不再支持与 Eclipse Vert.x 搭配使用。

3.2. 已弃用的功能

本节列出了本发行版本中已弃用或删除的功能。

3.2.1. 4.3 发行版本中已弃用的功能

4.3 版本中已弃用了以下功能。

- Eclipse Vert.x Core

删除的元素	替换元素
io.vertx.core.shareddata.impl.ClusterSerializable	io.vertx.core.shareddata.ClusterSerializable

- Eclipse Vert.x Micrometer Metrics

弃用的方法	替换方法
io.vertx.micrometer.MicrometerMetricsOptions.getRequestsTagsProvider()	**io.vertx.micrometer.MicrometerMetricsOptions.getServerRequestsTagsProvider()**
io.vertx.micrometer.MicrometerMetricsOptions.setRequestsTagsProvider()	**io.vertx.micrometer.MicrometerMetricsOptions.setServerRequestsTagsProvider()**
io.vertx.micrometer.VertxInfluxDbOptions.getNumThreads()	没有替换方法
io.vertx.micrometer.VertxInfluxDbOptions.setNumThreads()	没有替换方法

- **Eclipse Vert.x Web**

弃用的方法	替换方法
router.mountSubRouter (String, Router)	**Router.route(String).subRouter(Router)**
RoutingContext.getBodyAsString()	**RoutingContext.body().asString()**
RoutingContext.getBodyAsString(String encoding)	**RoutingContext.body().asString(String encoding)**
RoutingContext.getBodyAsJson()	**RoutingContext.body().asJsonObject()**
RoutingContext.getBodyAsJsonArray()	**RoutingContext.body().asJsonArray()**
RoutingContext.getBodyAsJson(int maxLength)	**RoutingContext.body().asJsonObject(int maxLength)**
RoutingContext.getBodyAsJsonArray(int maxLength)	**RoutingContext.body().asJsonArray(int maxLength)**
RoutingContext.getBody()	**RoutingContext.body().buffer()**
RouterBuilder.bodyHandler()	**RouterBuilder.rootHandler()**

- **SchemaBuilder**

删除的方法	替换方法

删除的方法	替换方法
build()	使用 Eclipse Vert.x Json Schema 提供的 JSON 表示。例如： **JsonSchema schema = JsonSchema.of(dsl.toJson());**

3.2.2. 之前 4.x 版本中已弃用的功能

在之前的 4.x 版本中，以下功能已弃用或删除。

- **HttpServerOptions**

删除的方法	替换方法
getMaxWebsocketFrameSize()	**getMaxWebSocketFrameSize()**
setMaxWebsocketFrameSize()	**setMaxWebSocketFrameSize()**
getMaxWebsocketMessageSize()	**getMaxWebSocketMessageSize()**
setMaxWebsocketMessageSize()	**setMaxWebSocketMessageSize()**
getPerFrameWebsocketCompressionSupported()	**getPerFrameWebSocketCompressionSupported()**
setPerFrameWebsocketCompressionSupported()	**setPerFrameWebSocketCompressionSupported()**
getPerMessageWebsocketCompressionSupported()	**getPerMessageWebSocketCompressionSupported()**
setPerMessageWebsocketCompressionSupported()	**setPerMessageWebSocketCompressionSupported()**
getWebsocketAllowServerNoContext()	**getWebSocketAllowServerNoContext()**
setWebsocketAllowServerNoContext()	**setWebSocketAllowServerNoContext()**
getWebsocketCompressionLevel()	**getWebSocketCompressionLevel()**
setWebsocketCompressionLevel()	**setWebSocketCompressionLevel()**

删除的方法	替换方法
getWebsocketPreferredClientNoContext()	getWebSocketPreferredClientNoContext()
setWebsocketPreferredClientNoContext()	setWebSocketPreferredClientNoContext()
getWebsocketSubProtocols()	getWebSocketSubProtocols()
setWebsocketSubProtocols()	setWebSocketSubProtocols()

- Eclipse Vert.x Web

删除的元素	替换元素
io.vertx.ext.web.Cookie	io.vertx.core.http.Cookie
io.vertx.ext.web.handler.CookieHandler	io.vertx.core.http.Cookie
io.vertx.ext.web.Locale	io.vertx.ext.web.LanguageHeader
RoutingContext.acceptableLocales()	RoutingContext.acceptableLanguages()
StaticHandler.create (String, ClassLoader)	---
SessionHandler.setAuthProvider(AuthProvider)	SessionHandler.addAuthProvider()
HandlebarsTemplateEngine.getHandlebars （）HandlebarsTemplateEngine.getResolvers （）HandlebarsTemplateEngine.setResolvers （）JadeTemplateEngine.getJadeConfiguration （）ThymeleafTemplateEngine.getThymeleafTemplateEngine （）ThymeleafTemplateEngine.setMode （）	TemplateEngine.unwrap()

- 消息传递

删除的方法	替换方法
MessageProducer<T>.send(T)	MessageProducer<T>.write(T)
MessageProducer.send(T,Handler)	EventBus.request(String,Object,Handler)

- EventBus

删除的方法	替换方法
EventBus.send (…, Handler<AsyncResult<Message<T>>)Message.reply (…, Handler<AsyncResult<Message<T>>)	replyAndRequest

- 处理程序（handler）

删除的方法	替换方法
Future<T>.setHandler()	Future<T>.onComplete()Future<T>.onSuccess()Future<T>.onFailure()
HttpClientRequest.connectionHandler()	HttpClient.connectionHandler()

- JSON

删除的字段/Methods	新方法
JSON.mapper （） 字段	DatabindCodec.mapper()
Json.prettyMapper() field	DatabindCodec.prettyMapper()
Json.decodeValue(Buffer, TypeReference<T>)	JacksonCodec.decodeValue(Buffer, TypeReference)
Json.decodeValue(String, TypeReference<T>)	JacksonCodec.decodeValue(String, TypeReference)

- JUnit5

弃用的方法	新方法
VertxTestContext.succeeding()	VertxTestContext.succeedingThenComplete()
VertxTestContext.failing()	VertxTestContext.failingThenComplete()

- 被动扩展(Rx)

弃用的方法	新方法
WriteStreamSubscriber.onComplete()	WriteStreamSubscriber.onWriteStreamEnd()WriteStreamSubscriber.onWriteStreamError()

- 断路器

删除的方法	替换方法
CircuitBreaker.executeCommand()	CircuitBreaker.execute()
CircuitBreaker.executeCommandWithFallback()	CircuitBreaker.executeWithFallback()

- MQTT

删除的方法	替换方法
MqttWill.willMessage()	MqttWill.getWillMessage()
MqttWill.willTopic()	MqttWill.getWillTopic()
MqttWill.willQos()	MqttWill.getWillQos()
MqttAuth.username()	MqttAuth.getUsername()
MqttAuth.password()	MqttAuth.getPassword()
MqttClientOptions.setKeepAliveTimeSeconds()	MqttClientOptions.setKeepAliveInterval()

- **AMQP 客户端**

删除的方法	替换方法
AmqpClient.createReceiver (String address, Handler<AmqpMessage> messageHandler, …)	**AmqpClient createReceiver (String address, Handler<AsyncResult<AmqpReceiver> completionHandler)**
AmqpConnection createReceiver(…, Handler<AsyncResult<AmqpReceiver>> completionHandler)	**AmqpConnection createReceiver (String address, Handler<AsyncResult<AmqpReceiver>> completionHandler)**
AmqpConnection createReceiver (.., Handler<AmqpMessage> messageHandler, Handler<AsyncResult<AmqpReceiver>> completionHandler)	**AmqpConnection createReceiver (String address, Handler<AsyncResult<AmqpReceiver>> completionHandler)**

- 认证**和授权**

删除的元素	替换元素
OAuth2Options.isUseBasicAuthorizationHeader()	没有替换方法
OAuth2Options.setUseBasicAuthorizationHeader()	没有替换方法
OAuth2Options.getClientSecretParameterName()	没有替换方法
OAuth2Options.setClientSecretParameterName()	没有替换方法
OAuth2Auth.createKeycloak()	**KeycloakAuth.create(vertx, JsonObject)()**
OAuth2Auth.create (Vertx, OAuth2FlowType, OAuth2ClientOptions)()	**OAuth2Auth.create(vertx, new OAuth2ClientOptions().setFlow(YOUR_DESIRED_FLOW))**
OAuth2Auth.create (Vertx, OAuth2FlowType)	**OAuth2Auth.create(vertx, new OAuth2ClientOptions().setFlow(YOUR_DESIRED_FLOW))**

删除的元素	替换元素
User.isAuthorised()	**User.isAuthorized()**
AccessToken.refreshToken()	**AccessToken.opaqueRefreshToken()**
io.vertx.ext.auth.jwt.JWTOptions data object	**io.vertx.ext.jwt.JWTOptions** data object
SecretOptions 类	**PubSecKeyOptions** 类

弃用的方法	替换方法
OAuth2Auth.decodeToken()	**AuthProvider.authenticate()**
OAuth2Auth.introspectToken()	**AuthProvider.authenticate()**
OAuth2Auth.getFlowType()	没有替换方法
OAuth2Auth.loadJWK()	**OAuth2Auth.jwkSet()**
Oauth2ClientOptions.isUseAuthorizationHeader()	没有替换方法

弃用的类	替换类
AbstractUser	使用 ' User.create (JsonObject)' 方法创建用户对象。
AuthOptions	没有替换类
JDBCAuthOptions	用于身份验证的 **JDBCAuthenticationOptions**，以及用于授权的 **JDBCAuthorizationOptions**
JDBCHashStrategy	没有替换类
OAuth2RBAC	**AuthorizationProvider**
Oauth2Response	建议使用 **WebClient** 类
KeycloakHelper	没有替换类

- **服务发现**

删除的方法	替换方法
ServiceDiscovery.create (…, Handler<ServiceDiscovery> completionHandler)	ServiceDiscovery.create(Vertx)
ServiceDiscovery.create (…, Handler<ServiceDiscovery> completionHandler)	ServiceDiscovery.create (Vertx, ServiceDiscoveryOptions)

- Eclipse Vert.x 配置

删除的方法	替换方法
ConfigRetriever.getConfigAsFuture()	retriever.getConfig()

- MongoDB 客户端

删除的方法	替换方法
MongoClient.update()	MongoClient.updateCollection()
MongoClient.updateWithOptions()	MongoClient.updateCollectionWithOptions()
MongoClient.replace()	MongoClient.replaceDocuments()
MongoClient.replaceWithOptions()	MongoClient.replaceDocumentsWithOptions()
MongoClient.remove()	MongoClient.removeDocuments()
MongoClient.removeWithOptions()	MongoClient.removeDocumentsWithOptions()
MongoClient.removeOne()	MongoClient.removeDocument()
MongoClient.removeOneWithOptions	MongoClient.removeDocumentsWithOptions()

- 没有共享数据源的客户端

弃用的方法	新方法
MongoClient.createNonShared()	MongoClient.create()
JDBCClient.createNonShared()	wJDBCClient.create()
CassandraClient.createNonShared()	CassandraClient.create()
MailClient.createNonShared()	MailClient.create()

- **hook 方法**

删除的方法	新方法
Context.addCloseHook()	没有替换方法
Context.removeCloseHook()	没有替换方法

- **克隆方法**

删除的方法	新方法
KeyCertOptions.clone()	KeyCertOptions.copy()
TrustOptions.clone()	TrustOptions.copy()
SSLEngineOptions.clone()	SSLEngineOptions.copy()

- **VertxOptions**

删除的方法	新方法
VertxOptions.equals()	没有替换方法
VertxOptions.hashCode()	没有替换方法
VertxOptions.fileResolverCachingEnabled()	FileSystemOptions.isFileCachingEnabled()

- **池缓冲**

删除的方法	新方法
TCPSSLOptions.isUsePooledBuffers()	没有替换方法
TCPSSLOptions.setUsePooledBuffers()	没有替换方法

第 4 章 发行组件

4.1. 这个版本引入的 ARTIFACTS

这个版本还没有将工件从技术预览移到完全支持。

4.2. 这个版本引入的技术预览工件

本节论述了本发行版本中引入的技术预览工件。

4.2.1. 4.3 发行版本中引入的技术预览工件

4.3 版本中以技术预览形式提供以下工件。

- vertx-grpc-client

  Eclipse Vert.x gRPC 客户端是一个新的 Google Remote Procedure Call (gRPC)客户端，它依赖于 Eclipse Vert.x HTTP 客户端。Eclipse Vert.x gRPC 客户端提供了与服务器交互的两个替代方法：

  - 不需要生成的存根的 gRPC 请求和响应导向型 API
  - 使用 gRPC 频道生成存根

  注意

  Eclipse Vert.x gRPC 客户端会取代集成的 Netty 的 gRPC 客户端。

- vertx-grpc-server

  Eclipse Vert.x gRPC 服务器是一个新的 Google Remote Procedure Call (gRPC)服务器，它依赖于 Eclipse Vert.x HTTP 服务器。Eclipse Vert.x gRPC 服务器提供了与客户端交互的两个替代方法：

- 不需要生成的存根的 gRPC 请求和响应导向型 API
- 使用服务桥接生成的存根

注意

Eclipse Vert.x gRPC 服务器取代了集成的 Netty 的 gRPC 服务器。

- **vertx-grpc-common**

  Eclipse Vert.x gRPC 通用工件提供了 Eclipse Vert.x gRPC 客户端和 Eclipse Vert.x gRPC 服务器的通用功能。

- **vertx-grpc-aggregator**

  Eclipse Vert.x gRPC 聚合器由一个项目对象模型(POM)文件组成。Eclipse Vert.x gRPC 聚合器不提供任何其他功能。

4.2.2. 之前 4.x 版本中引入的技术预览工件

以下工件在之前的 4.x 版本中作为技术预览提供。

- **vertx-auth-otp**

  Eclipse Vert.x OTP Auth 供应商是 AuthenticationProvider 接口实现，它使用一次性密码来执行身份验证。Eclipse Vert.x OTP Auth 供应商支持 Google Authenticator。您可以使用任何方便的库创建带有密钥的快速响应(QR)。您还可以以 base32 格式传输密钥。

- **vertx-oracle-client**

  Eclipse Vert.x reactive Oracle 客户端是 Oracle 服务器的客户端。这是一个 API，有助于实现数据库的可扩展性并降低开销。由于 API 处于被动和非阻塞，所以您可以处理多个与单个线程的数据库连接。

注意

Eclipse Vert.x 重新主动 Oracle 客户端要求您使用 Oracle JDBC 驱动程序。红帽不提供对 Oracle JDBC 驱动程序的支持。

Eclipse Vert.x 重新主动 Oracle 客户端要求您使用 JDK 11 或 JDK 17。

- **vertx-http-proxy**

  Eclipse Vert.x HTTP 代理是一个反向代理。使用这个模块，您可以轻松地创建代理。代理服务器也可以彻底解决来自 或igin 服务器的 DNS 查询。

- **vertx-web-proxy**

  通过 Eclipse Vert.x web 代理，您可以在 Eclipse Vert.x web 路由器中挂载 Eclipse Vert.x HTTP 代理。

- **vertx-opentelemetry**

  支持打开 Telemetry tracing。您可以使用 Open Telemetry 进行 HTTP 和事件总线追踪。

4.3. 这个版本中删除的工件

这个版本没有删除工件。

4.4. 此发行版本中已弃用的工件

本发行版本中没有将工件标记为已弃用。

第 5 章 修复的问题

这个 Eclipse Vert.x 版本包含了来自社区版本 4.3.7 的所有程序错误修复。在社区版本中解决的问题在 Eclipse Vert.x 4.3.7 Wiki 页面中列出。

5.1. 修复了 4.3 发行版本中的问题

这部分论述了在 Eclipse Vert.x 4.3 中修复的问题。

5.1.1. 当使用 gRPC 进行客户端和服务器通信时，带有 io.vertx.coreVertxException 的线程被阻塞

在 Eclipse Vert.x 4.3.4 或更早版本中，Google 远程过程调用(gRPC)客户端和服务器间的通信会阻止线程，并导致 io.vertx.core.VertxException 错误。如果可用事件循环线程的数量不足，这会导致 Eclipse Vert.x gRPC Server (vertx-grpc-server)或 Eclipse Vert.x gRPC Client (vertx-grpc-client)改为 self-deadlock。

这个问题已在 Eclipse Vert.x 4.3.5 版本中解决。SSL 初始化的内部上下文现在使用 worker 上下文而不是 event-loop 上下文。

5.1.2. 按 ROWID搜索表数据时 JDBCClient 错误

在 Eclipse Vert.x 4.2 中，当尝试使用 ROWID 检索表数据时，vertx-jdbc-client 带有 Oracle JDBC 驱动程序 threw a java.sql.SQLException。出现这个问题的原因是，在 Eclipse Vert.x 4.2 中，ROWID 作为一个字节数组提供。在较早版本的 Eclipse Vert.x 中，ROWID 作为一个字符串类型提供。

这个问题已在 Eclipse Vert.x 4.3.1 发行版本中解决。Eclipse Vert.x 4.2 根据普通 Java 类型执行自定义类型。虽然此行为通常会生成正确的结果，但可能会错误地识别特殊数据库类型。由于自定义类型广播不是现代 JDBC 驱动程序的内置功能，所以 JDBC 客户端现在依赖于驱动程序执行特定于供应商的广播，这些广播更适合于旧的 Eclipse Vert.x heuristics。

5.1.3. 解析 ROWID时 JDBCPool 错误

在 Eclipse Vert.x 4.2 中，在尝试解析 ROWID 时，JDBC 池 threw 是一个 java.lang.UnsupportedOperationException。出现这个问题的原因是，在 Eclipse Vert.x 4.2 中，ROWID 无法直接解析为字节数组。在之前的 {VertX 版本中，ROWID 可以解析为字符串类型。

这个问题已在 Eclipse Vert.x 4.3.1 发行版中解决，它基于前面部分所述的解决方案。

5.1.4. 意外的结果是，在使用 PostgreSQL JDBC 驱动程序 9.0 或更高版本时存储的步骤调用

在 Eclipse Vert.x 4.2 中，vertx-jdbc-client 带有 PostgreSQL JDBC 驱动程序 9.0 或更高版本，从而导致存储过程调用。出现这个问题的原因是，在 Eclipse Vert.x 4.2 中，vertx-jdbc-client 不支持现代 PostgreSQL 数据库驱动程序和服务器在执行可调用语句时需要的显式 SQL 类型信息。

这个问题已在 Eclipse Vert.x 4.3.1 发行版本中解决。现在，可从查询涉及的所有源中提取可调用的 ResultSet 元数据，以及组成响应一部分的独立结果集。完整信息允许 JDBC 客户端正确识别列中的数据类型并执行正确的广播。

5.2. 修复了之前的 4.X 版本中的问题

这部分论述了在之前的 Eclipse Vert.x 4.x 版本中修复的问题。

5.2.1. GraphQL 构建中包含的 Google Guava 类

在 Eclipse Vert.x 4.0.0 和 4.0.2 版本中，vertx-web-graphql 依赖项不可用。这是因为使用版本 16.1.0.redhat-00001 的 GraphQL Java 构建不完整。在不完整的 GraphQL 构建中，缺少 Guava 类。

这个问题已在 Eclipse Vert.x 4.0.3 发行版本中解决。此发行版本包括 GraphQL Java 16.1.0.redhat-00002 版本，这个版本是一个带有 Guava 类的完整构建。这些 Guava 类被合并到 jar 中。

5.2.2. Eclipse Vert.x 构建中的 vertx-opentracing 可用

vertx-opentracing 依赖项作为 Eclipse Vert.x 4.0.0 中的技术预览功能提供。但是，其依赖项在 Eclipse Vert.x 4.0.0 和 4.0.2 版本中不可用。

这个问题已在 Eclipse Vert.x 4.0.3 发行版本中解决。发行版本包括 vertx-opentracing 依赖项。

第 6 章 已知问题

6.1. 4.3 发行版本中已知的问题

没有问题可用于影响这个版本。

6.2. 之前 4.X 版本中已知的问题

这部分论述了早期 Eclipse Vert.x 4.x 版本中的已知问题。

6.2.1. 在迁移到 Eclipse Vert.x 4.2 后，RH-SSO 发布的令牌会导致 OAuth2 验证失败

Description

如果您使用 Red Hat Single Sign-On (RH-SSO)作为身份提供程序，在以前的 Eclipse Vert.x 版本中有效令牌将会失败验证检查。

原因

在 Eclipse Vert.x 4.2 中，OAuth2 身份验证比之前的 Eclipse Vert.x 版本提供严格的安全验证。

临时解决方案

默认情况下，RH-SSO 使用 帐户 使用者而不是客户端 ID 签发令牌。如果您将 RH-SSO 用作身份提供程序，在迁移到 Eclipse Vert.x 4.2 后，您必须明确将 帐户 受众和客户端 ID 添加到 JWTOptions 配置中，以确保令牌成功验证。

6.2.2. 在使用带有 JDK 8 的 vertx-oracle-client 时编译错误

Description

在 Eclipse Vert.x 4.2 中，在使用 OpenJDK 8 时，vertx-oracle-client 会抛出一个错误 的类文件 编译错误。

原因

在 Eclipse Vert.x 4.2 中，vertx-oracle-client 的设计用于 OpenJDK 11 或更高版本。

临时解决方案

使用 OpenJDK 11 或 OpenJDK 17。

6.2.3. KubernetesServiceImporter （） 无法直接注册到 Eclipse Vert.x Reactive Extensions (Rx)

Description

您不能使用 Eclipse Vert.x 的 Reactive Extensions (Rx)直接注册 KubernetesServiceImporter （）。

原因

服务导入器没有生成的 RxJava 2 实现。

临时解决方案

您必须创建一个 KubernetesServiceImporter 实例，并使用 {@link io.vertx.reactivex.servicediscovery.spi.ServiceImporter} 封装它，如下例所示：

```
{@link examples.RxServiceDiscoveryExamples#register(io.vertx.reactivex.servicediscovery.ServiceDiscovery)}
```

以下示例演示了如何在 Eclipse Vert.x Reactive Extensions (Rx)中注册 KubernetesServiceImporter （）。

```
ServiceDiscovery discovery = ServiceDiscovery.create(vertx);
discovery.getDelegate().registerServiceImporter(new KubernetesServiceImporter(), new JsonObject());
```

6.2.4. Red Hat AMQ Streams 镜像不适用于 IBM Z 和 IBM Power Systems

Red Hat AMQ Streams Operator 和 Kafka 镜像不适用于 IBM Z 和 IBM Power Systems。由于镜像不可用，vertx-kafka-client 模块没有可用于 IBM Z 和 IBM Power Systems 上的 AMQ Streams。

6.2.5. 基于 RHEL 8 的数据库应用程序和基于 RHEL 7 的 MySQL 5.7 数据库的连接会因为 TLS 协议版本不匹配而失败

Description

在基于 RHEL 8 的 OpenJDK 构建器镜像上构建的应用程序容器和基于 RHEL 7 的 MySQL 5.7 容器镜像上构建的数据库容器之间，尝试使用 OpenSSL 打开 TLS 保护的连接会导致连接失败，因为 javax.net.ssl.SSLHandshakeException 在运行时 会在以下位置查看问题 ：

```
...
Caused by: javax.net.ssl.SSLHandshakeException: No appropriate protocol (protocol is disabled or cipher suites are inappropriate)
...
```

-

原因

这个问题的原因是 RHEL 7 和 RHEL 8 之间最新支持的 TLS 协议版本有不同。RHEL 7 上的 TLS 实现支持 TLS 协议版本 1.0 （已弃用）、1.1 和 1.2。RHEL 8 上的 TLS 实现还支持 TLS 协议版本 1.3，这也是基于 RHEL 8 的构建器镜像中使用的默认 TLS 版本。这种差异可能会导致应用程序组件间的 TLS 协议版本不匹配，同时强制使用 TLS 握手，进而导致应用程序和数据库容器之间的连接失败。

临时解决方案

要防止上述问题，请手动指定数据库连接字符串中两个操作系统版本支持的 TLS 协议版本。例如：

```
jdbc:mysql://testdb-mysql:3306/testdb?enabledTLSProtocols=TLSv1.2
```

6.2.6. 调用应用程序端点时，由 peer 错误消息重置的 false 连接

使用 curl 工具或 Java HTTP 客户端在 Eclipse Vert.x 应用程序的端点上发出 HTTP 请求，在每个请求后在日志中生成以下错误：

```
io.vertx.core.net.impl.ConnectionBase
SEVERE: java.io.IOException: Connection reset by peer
```

此行为是由 Netty 应用框架和 OpenShift 使用的 HAProxy 负载平衡器交互造成的。此错误的原因是 HAProxy 在不关闭的情况下重新使用现有的 HTTP 连接。尽管错误消息已记录，但不会出现错误条件。HTTP 请求被正确处理，应用程序会如预期做出响应。

第 7 章 与此发行版本相关的公告

以下公告已发布，用于记录本发行版本中所含的增强、错误修复和 CVE 修复。

- RHSA-2023:0577