Red Hat CodeReady Studio 12.16

CodeReady Studio ツールのスタートガイド

Red Hat CodeReady Studio ツールの使用方法

Last Updated: 2021-03-23

Red Hat CodeReady Studio 12.16 CodeReady Studio ツールのスタートガイド

Red Hat CodeReady Studio ツールの使用方法

Levi Valeeva
levi@redhat.com

Supriya Takkhi
sbharadw@redhat.com

Yana Hontyk
yhontyk@redhat.com

法律上の通知

概要

本書のトピックでは、開発を効率的に進めるために Red Hat CodeReady Studio ツールの使用する方法を説明します。

目次

第1章 CODEREADY STUDIO の GIT の基本 4
1.1. GIT PERSPECTIVE の設定 4
1.2. GIT PERSPECTIVE でのリポジトリーの管理 6
1.2.1. 新規 Git リポジトリーの作成 6
1.2.2. 既存のローカル Git リポジトリーの追加 7
1.2.3. 既存の Git リポジトリーのクローン作成 8
1.2.4. リポジトリーのリモートの追加 10
1.3. GIT PERSPECTIVE でのブランチの管理 15
1.3.1. 新規ブランチの作成 15
1.3.2. ブランチの使用 18
1.3.3. プルリクエストの送信 19
1.4. 変更のコミットおよびプッシュ 20

第2章 CODEREADY STUDIO における MAVEN の基本 23
2.1. 新規 MAVEN プロジェクトの作成 23
2.2. 既存の MAVEN プロジェクトのインポート 26
2.2.1. ローカルに保存された既存の Maven プロジェクトのインポート 27
2.2.2. リモートで保存された 既存の Maven プロジェクトのインポート 29
2.3. 新しい MAVEN モジュールの作成 31
2.4. MAVEN 依存関係の MAVEN プロジェクトへの追加 34
2.5. MAVEN サポートを MAVEN 以外の既存プロジェクトへ追加 36
2.6. その他のリソース 38

第3章 CODEREADY STUDIO でのアプリケーションデプロイメント 39
3.1. ローカルサーバーの設定 39
3.2. リモートサーバーの設定 42
3.3. アプリケーションのデプロイ 46

第4章 CODEREADY STUDIO の JBOSS EAP および JBOSS WFK の基本 49
4.1. MAVEN リポジトリーの設定 49
4.2. JBOSS EAP の設定 52

第5章 CODEREADY STUDIO での OPENSHIFT の基本 60
5.1. OPENSHIFT APPLICATION EXPLORER ビューの設定 60
5.2. OPENSHIFT APPLICATION EXPLORER を使用した OPENSHIFT クラスターへの接続 62
5.3. 新しいランチャープロジェクトの作成 63
5.4. OPENSHIFT APPLICATION EXPLORER を使用した新規プロジェクトの作成 66
5.5. OPENSHIFT APPLICATION EXPLORER を使用した新規コンポーネントの作成 67
5.6. OPENSHIFT APPLICATION EXPLORER を使用したクラスターへのコンポーネントのデプロイ 68
5.7. OPENSHIFT APPLICATION EXPLORER を使用した外部アクセス URL の定義 69
5.8. OPENSHIFT APPLICATION EXPLORER を使用したクラスターでのアプリケーションのデバッグ 71

第6章 CODEREADY STUDIO の QUARKUS ツールの基本 73
6.1. 新しい QUARKUS プロジェクトの作成 73
6.2. QUARKUS アプリケーションの実行 76
6.3. QUARKUS アプリケーションのデバッグ 78
6.4. CODEREADY STUDIO での言語サポートの使用 79
6.4.1. Quarkus コード補完の使用 79
6.4.2. MicroProfile の言語サポートの有効化 81

第7章 CODEREADY STUDIO の HIBERNATE TOOLS の基本 85
7.1. 新規 JPA プロジェクトの作成 85
7.2. ライブラリーの追加 94

7.3. エンティティーの生成 97
7.4. HIBERNATE マッピングファイルの作成 99
7.5. HIBERNATE 設定ファイルの作成 102
7.6. HIBERNATE コンソール設定ファイルの作成 105
7.7. HIBERNATE プロジェクト設定の編集 111

第8章 CODEREADY STUDIO での MOBILE WEB TOOLS の基本 117
8.1. HTML5 プロジェクトの作成 117
8.2. 新しい HTML5 JQUERY MOBILE ファイルの追加 119
8.3. 新しいモバイルページの追加 123

第1章 CODEREADY STUDIO の GIT の基本

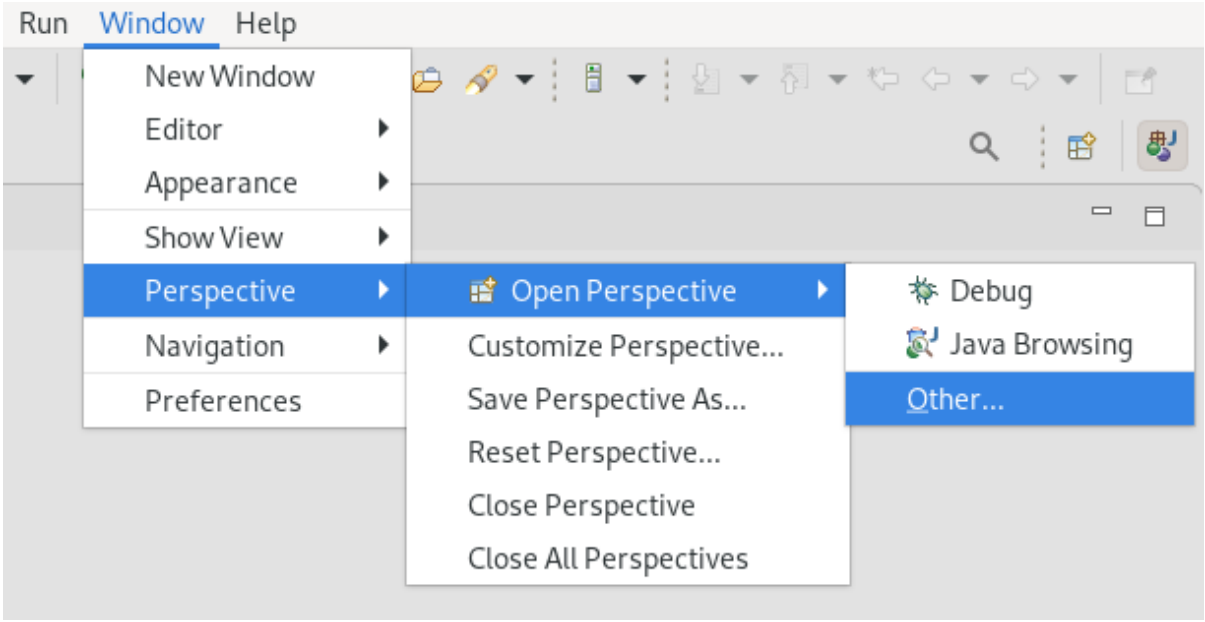
CodeReady Studio には、開発者がグラフィカルインターフェースから Git リポジトリーを管理できる Git Perspective が含まれています。ここでは、Git Perspective における Git プロジェクトの基本ワークフローの概要と、最も一般的な Git 関連のタスクを実行する方法を説明します。

1.1. GIT PERSPECTIVE の設定

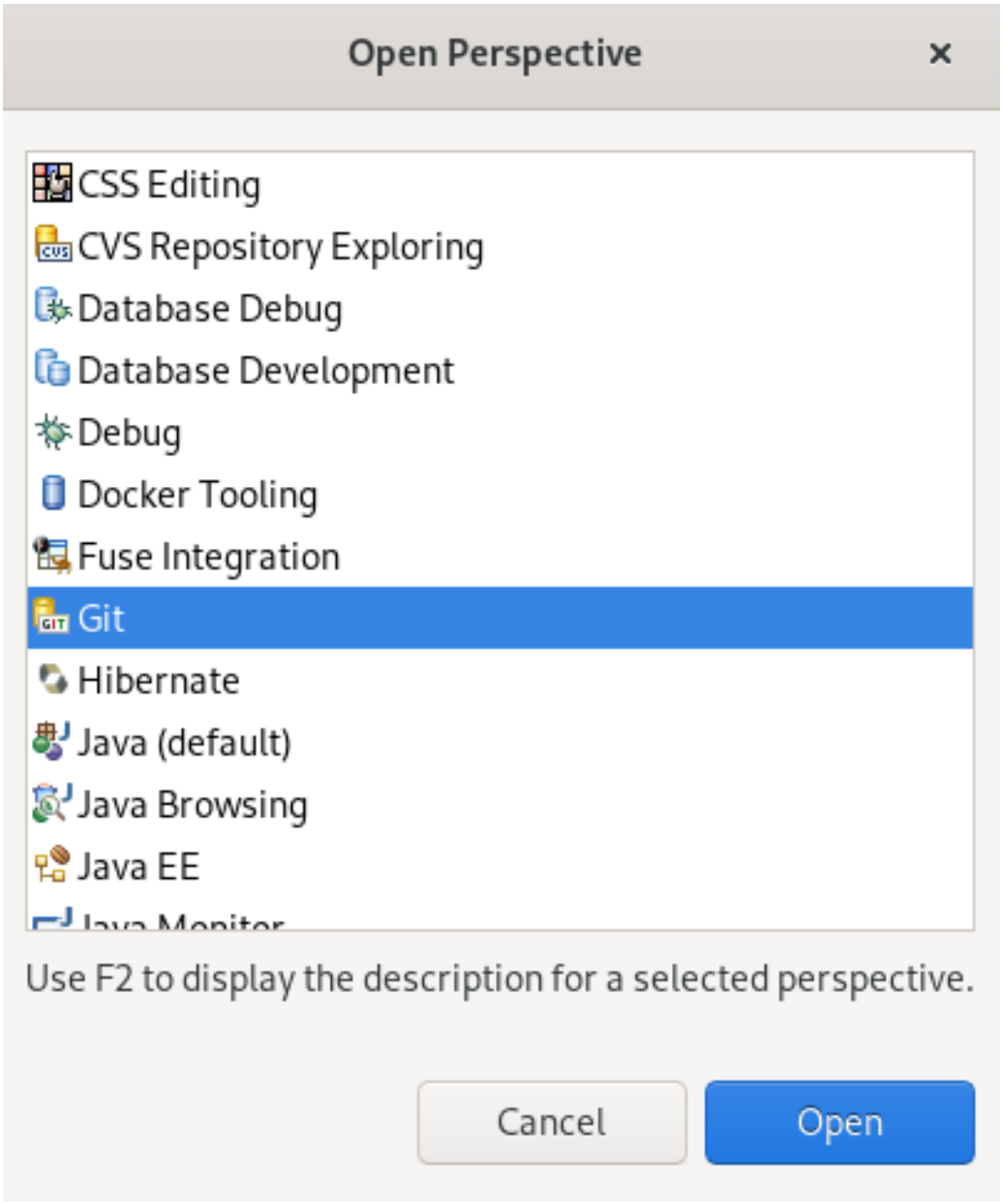
CodeReady Studio で Git Perspective を開く方法を説明します。

手順

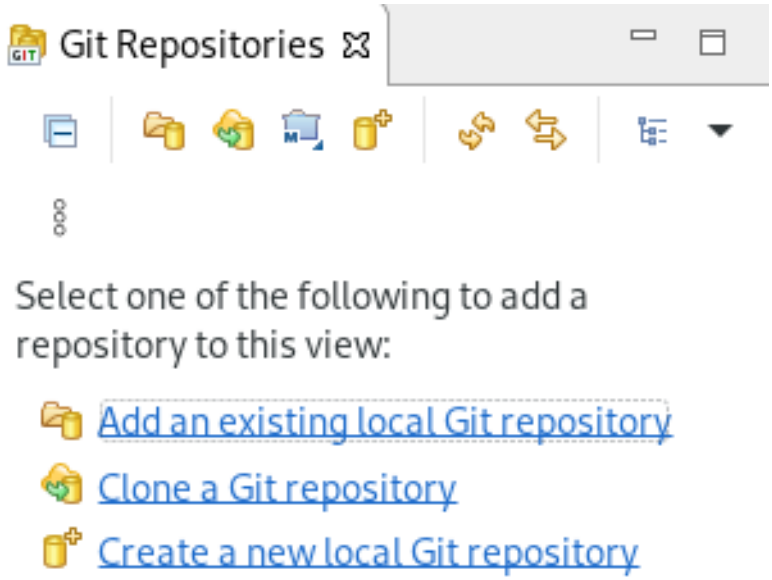
1. CodeReady Studio を起動します。
2. **Window → Perspective → Open Perspective → Other** とクリックします。



Open Perspective ウィンドウが表示されます。



3. **Git** を選択します。
4. **Open** をクリックします。
Git Repositories ビューが表示されます。



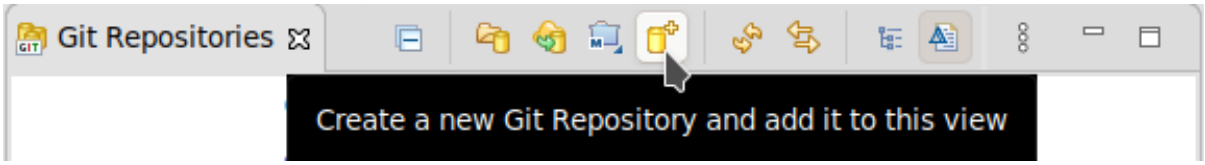
1.2. GIT PERSPECTIVE でのリポジトリーの管理

1.2.1. 新規 Git リポジトリーの作成

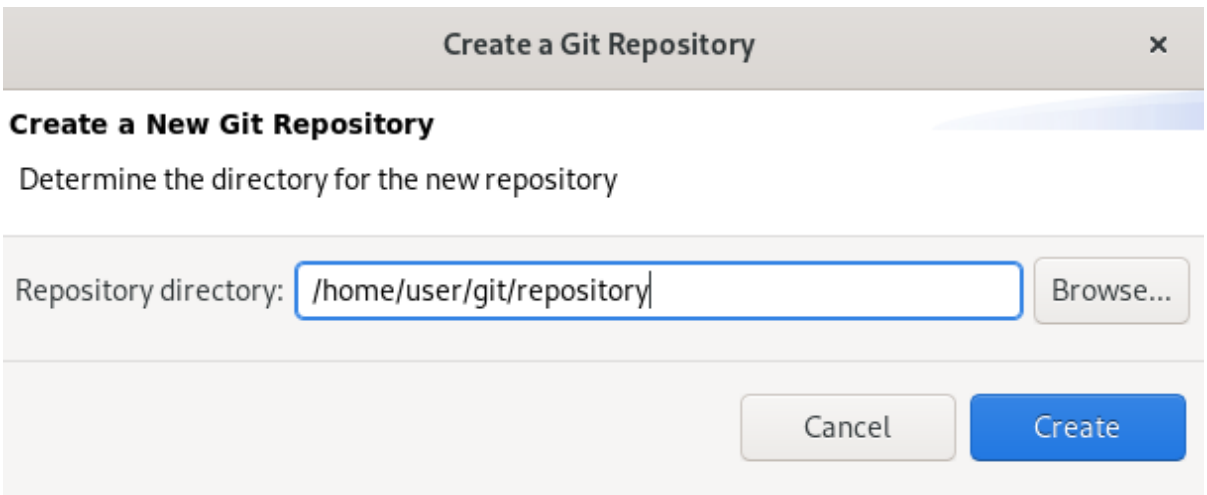
Git Perspective を使用して新規 Git リポジトリーを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Git Perspective** を開きます。
3. **Create a new Git Repository and add it to this view** アイコンをクリックします。



Create a Git Repository ウィンドウが表示されます。



デフォルトの **Repository ディレクトリー** へのパスが自動的に生成されます。デフォルトのパスが適切であれば、リポジトリーの作成を続行します。

必要に応じて、**Create as bare repository**チェックボックスを選択します。

注記

ベアリポジトリーは中央リポジトリーに推奨されますが、開発環境には推奨されません。ベアリポジトリーには、ソースファイルの作業用のコピーや、チェックアウトされたコピーが含まれません。そのため、ファイルの編集や変更のコミットができません。さらに、リポジトリーの Git リビジョンの履歴は、**.git** サブフォルダーではなく、ルートのフォルダーに保存されます。

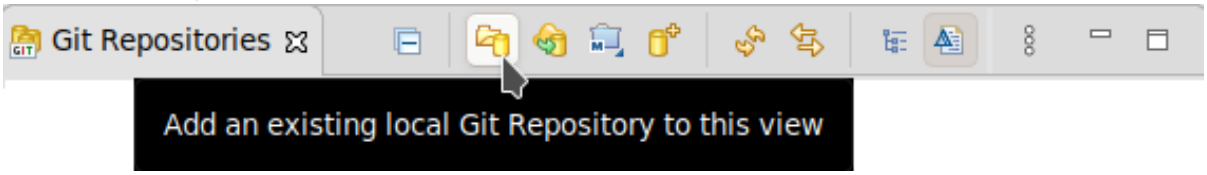
4. **Create** をクリックします。
新しい Git リポジトリーがローカルマシンに作成され、**Git Repositories** ビューに表示されます。

1.2.2. 既存のローカル Git リポジトリーの追加

Git Perspective を使用して、ローカル Git リポジトリーを IDE に追加する方法を説明します。

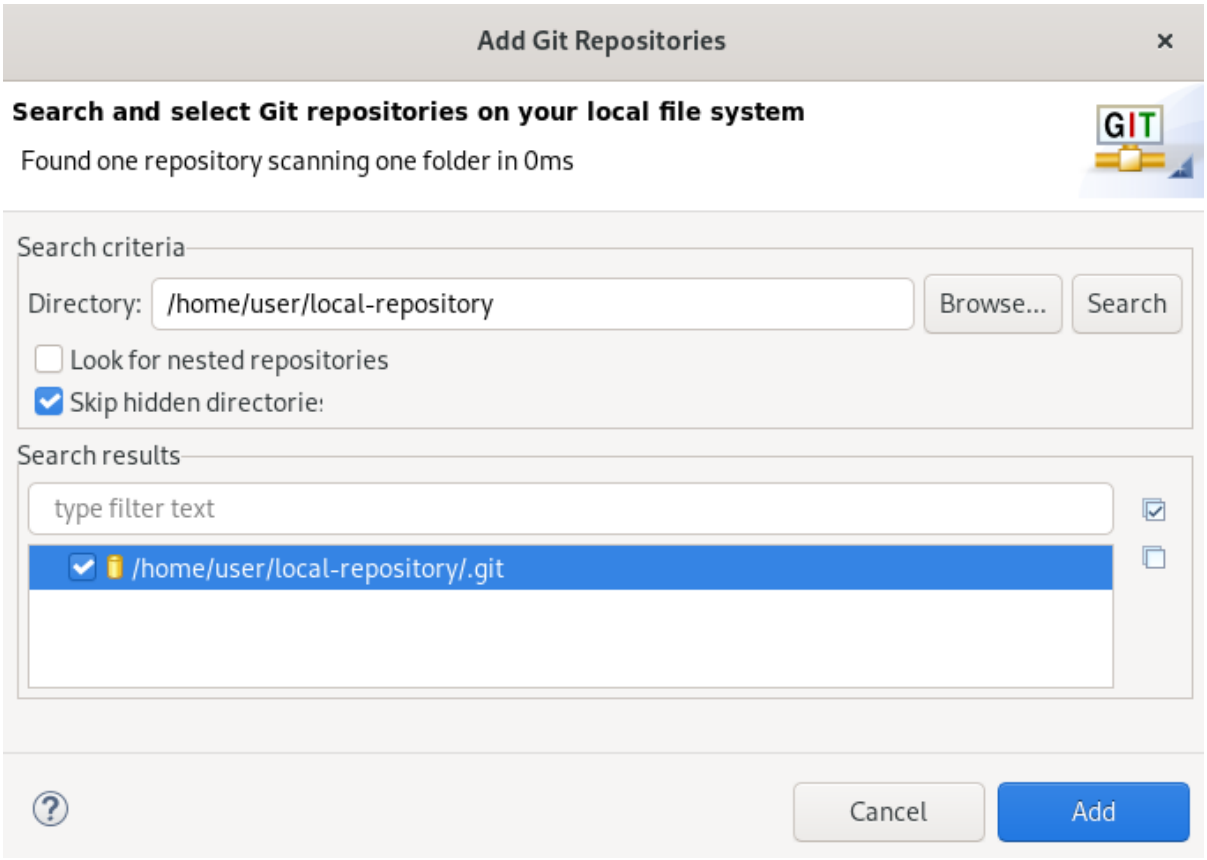
手順

1. CodeReady Studio を起動します。
2. **Git Perspective** を開きます。
3. **Add an existing local Git Repository to this view**アイコンをクリックします。



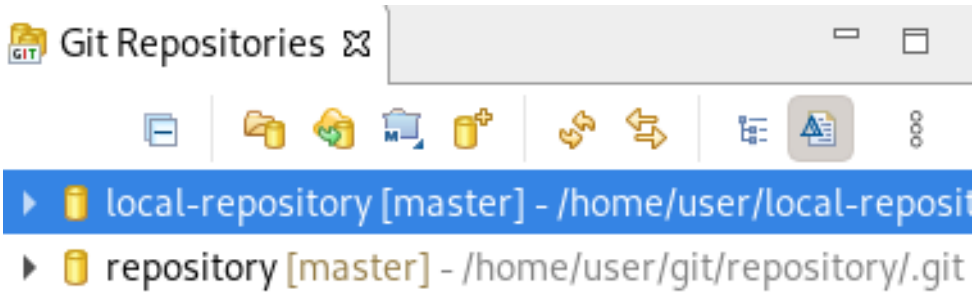
Add Git Repositories ウィンドウが表示されます。

4. **Browse** をクリックして、ローカルの Git リポジトリーを見つけます。



5. **Search results** フィールドで、**.git** ファイルへのパスを示すチェックボックスを選択します。
6. **Add** をクリックします。

ローカルリポジトリーが **Git Repositories** ビューに表示されます。

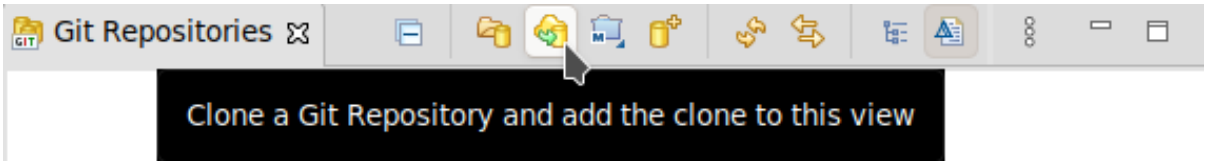


1.2.3. 既存の Git リポジトリーのクローン作成

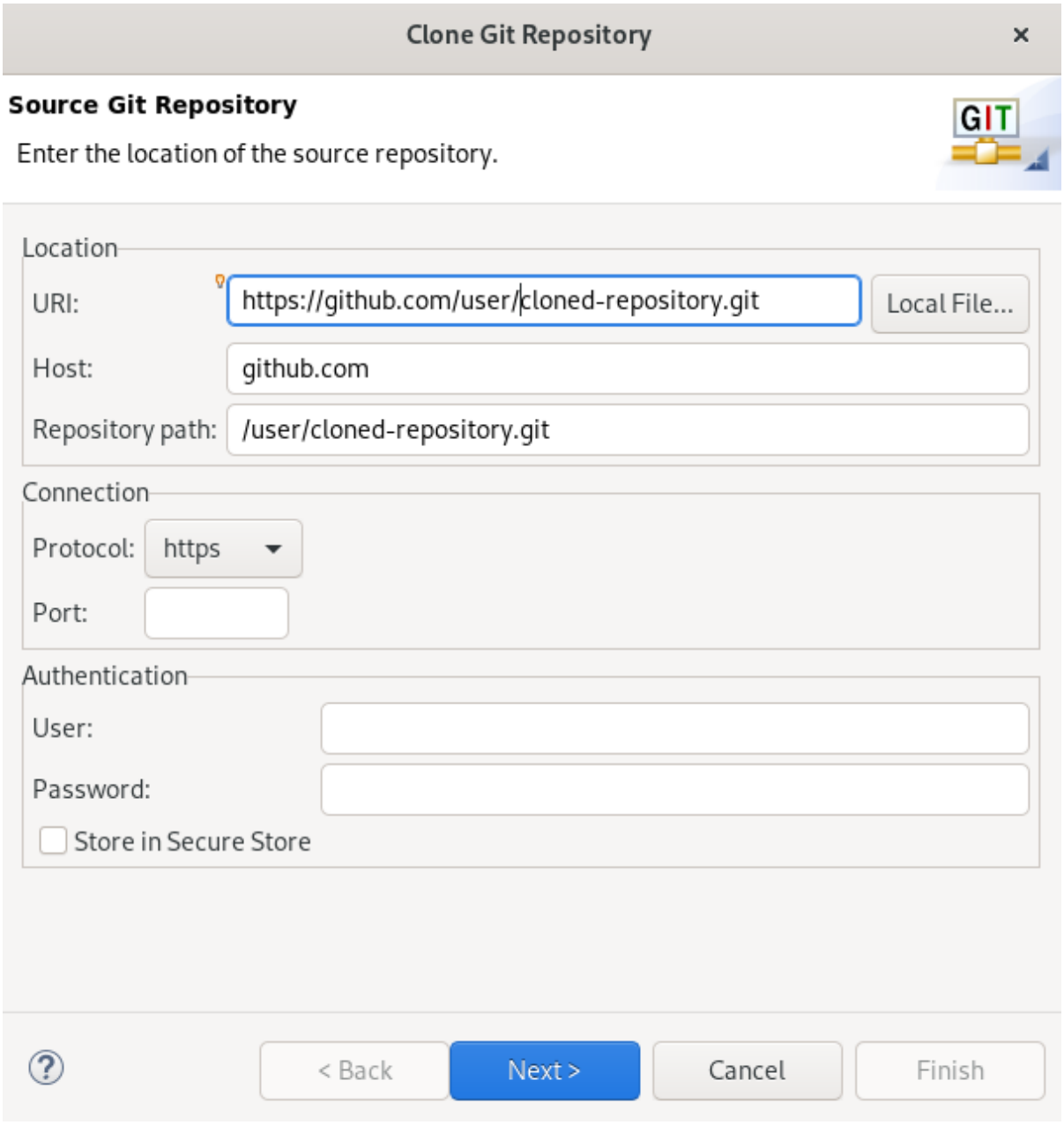
Git Perspective を使用して、オンラインにすでに存在するリポジトリー (GitHub、GitLab) のローカルクローンを作成する方法を説明します。

手順

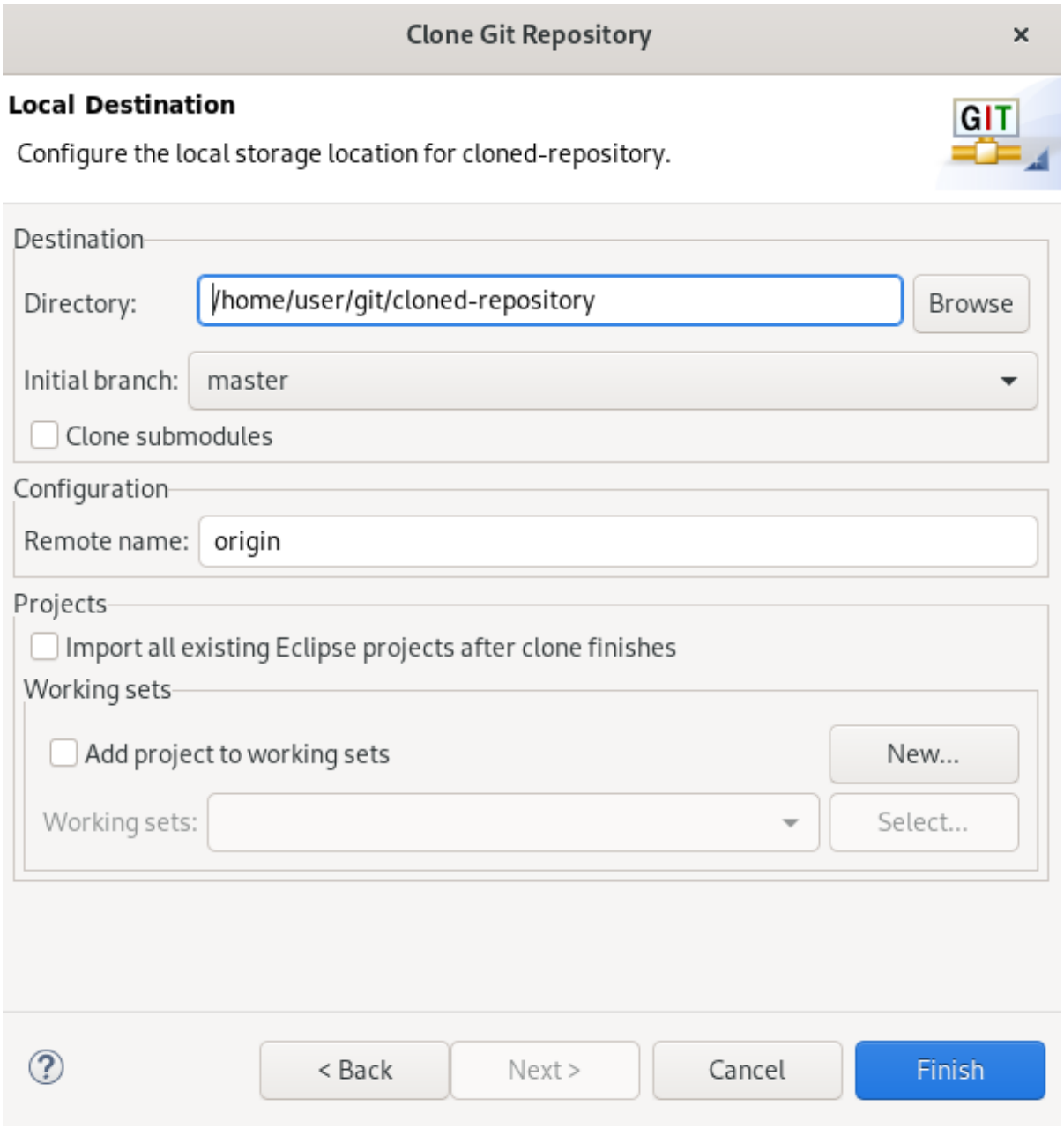
1. CodeReady Studio を起動します。
2. **Git Perspective** を開きます。
3. **Clone a Git Repository and add the clone to this view**アイコンをクリックします。



Clone Git Repository ウィンドウが表示されます。

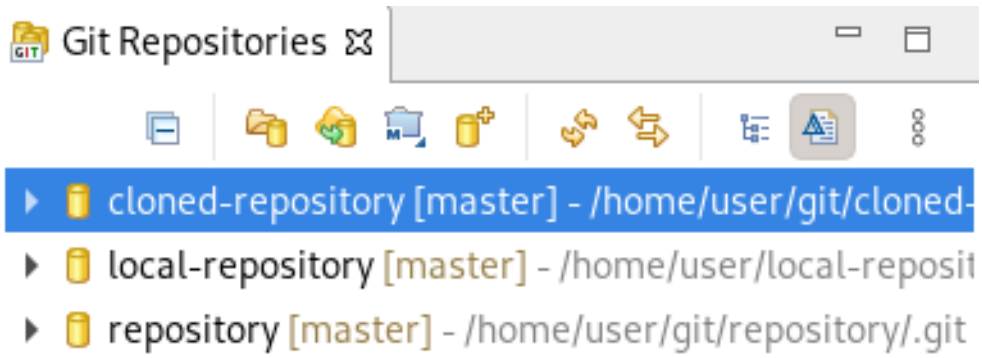


4. ソースリポジトリーのアドレスを **URI** フィールドに追加します。
Host および **Repository path** フィールドが自動的に入力されます。
5. **Next** をクリックします。
6. クローンを作成するブランチを選択します。
7. **Next** をクリックします。
8. **Directory** パスと **Initial branch** が正しく設定されていることを確認してください。



9. **Finish** をクリックします。

クローンしたリポジトリーが、CodeReady Studio の **Git Repository** ビューに表示されます。



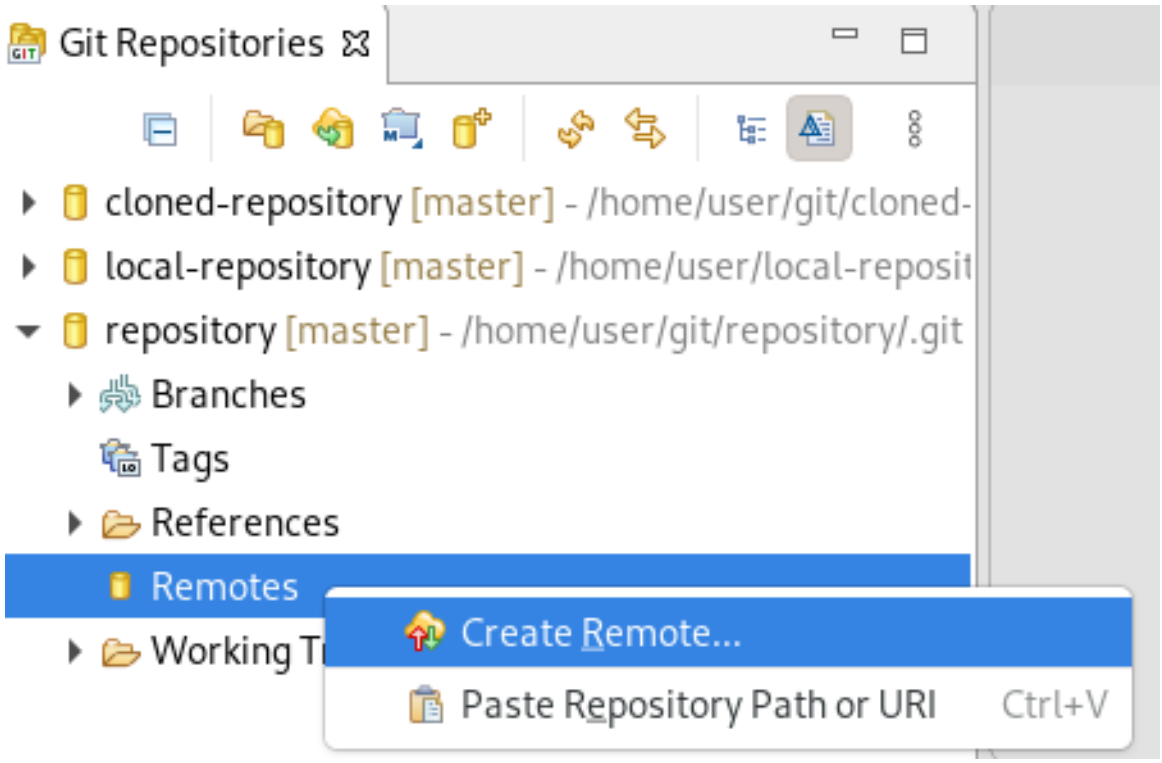
1.2.4. リポジトリーのリモートの追加

Git Perspective にリポジトリーを設定したら、リポジトリーにリモートを追加します。これは、新たに作成または追加されたリポジトリーに 1 度だけ必要な設定ステップです。

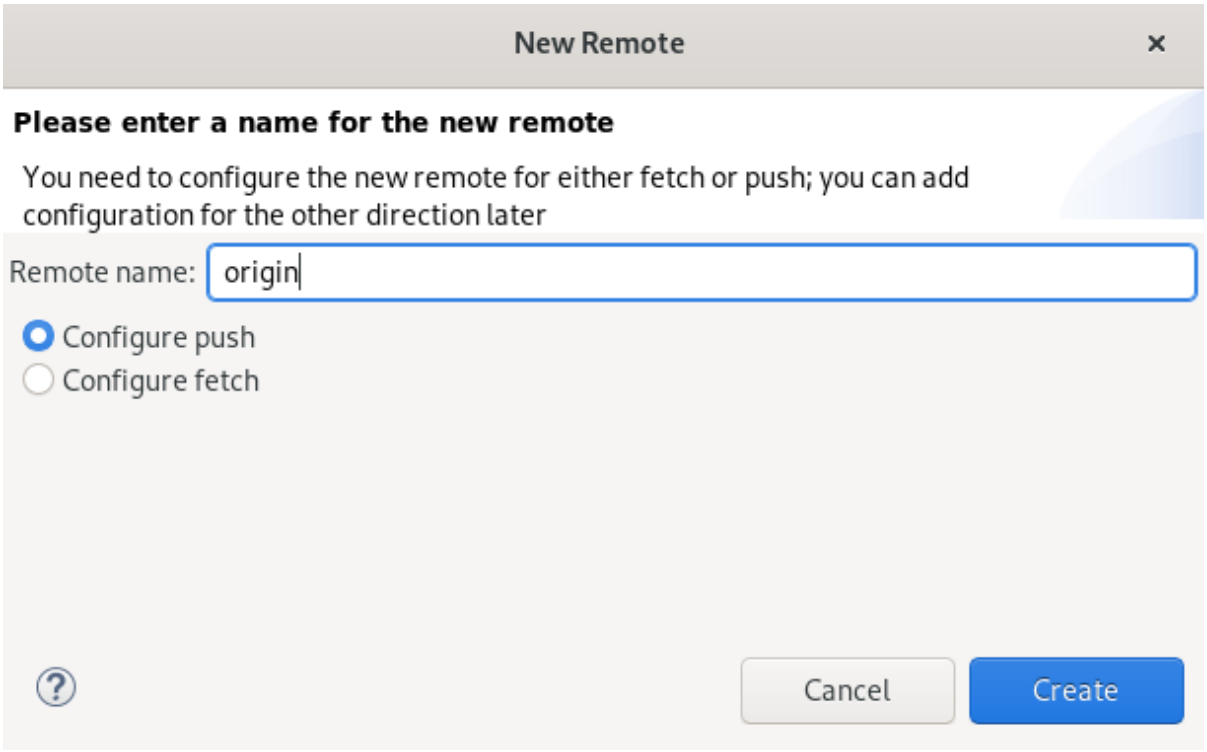
Git Perspective を使用してリポジトリーにリモートを設定する方法を説明します。

手順

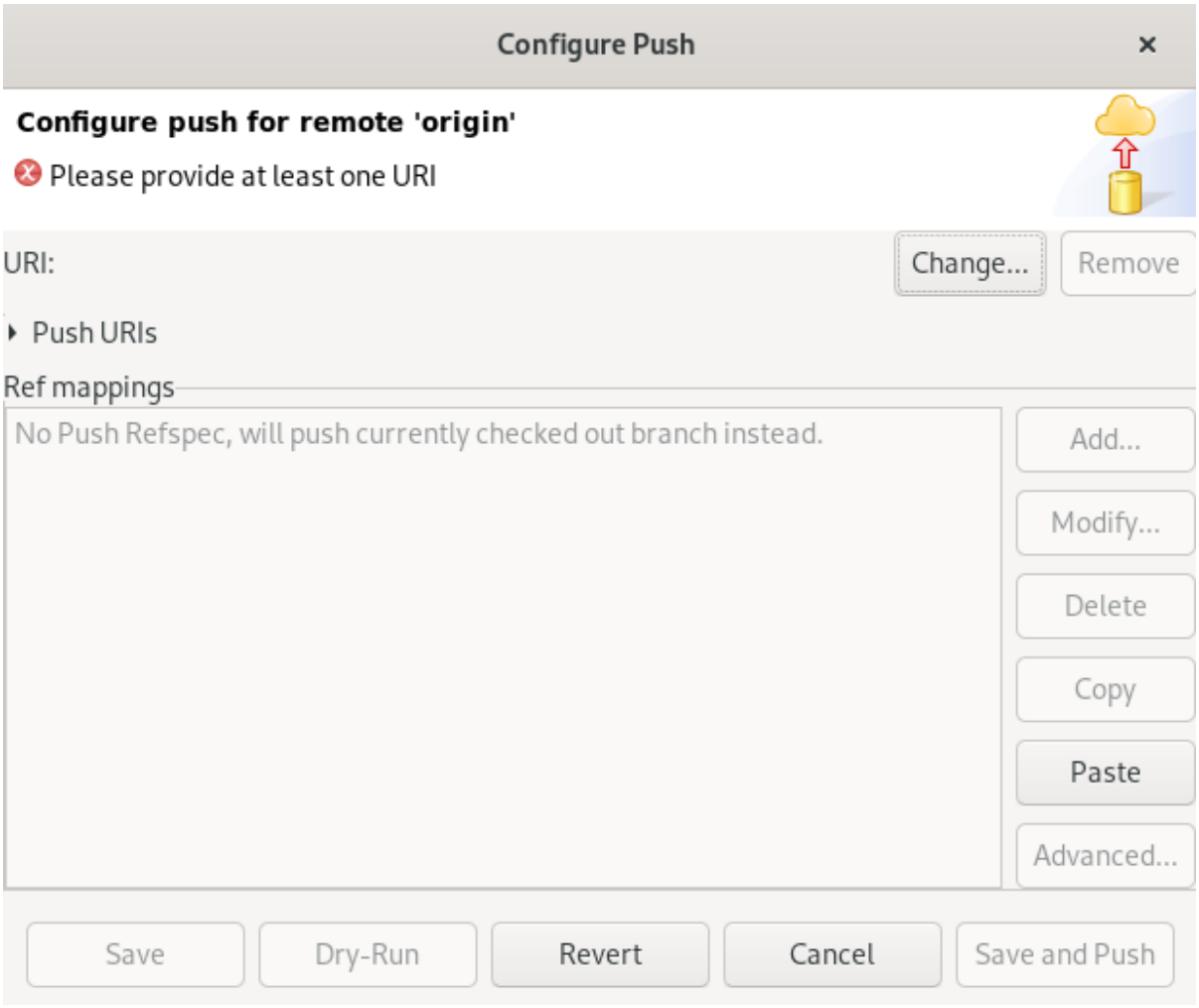
1. CodeReady Studio を起動します。
2. **Git Perspective** を開きます。
3. リポジトリーを展開します。



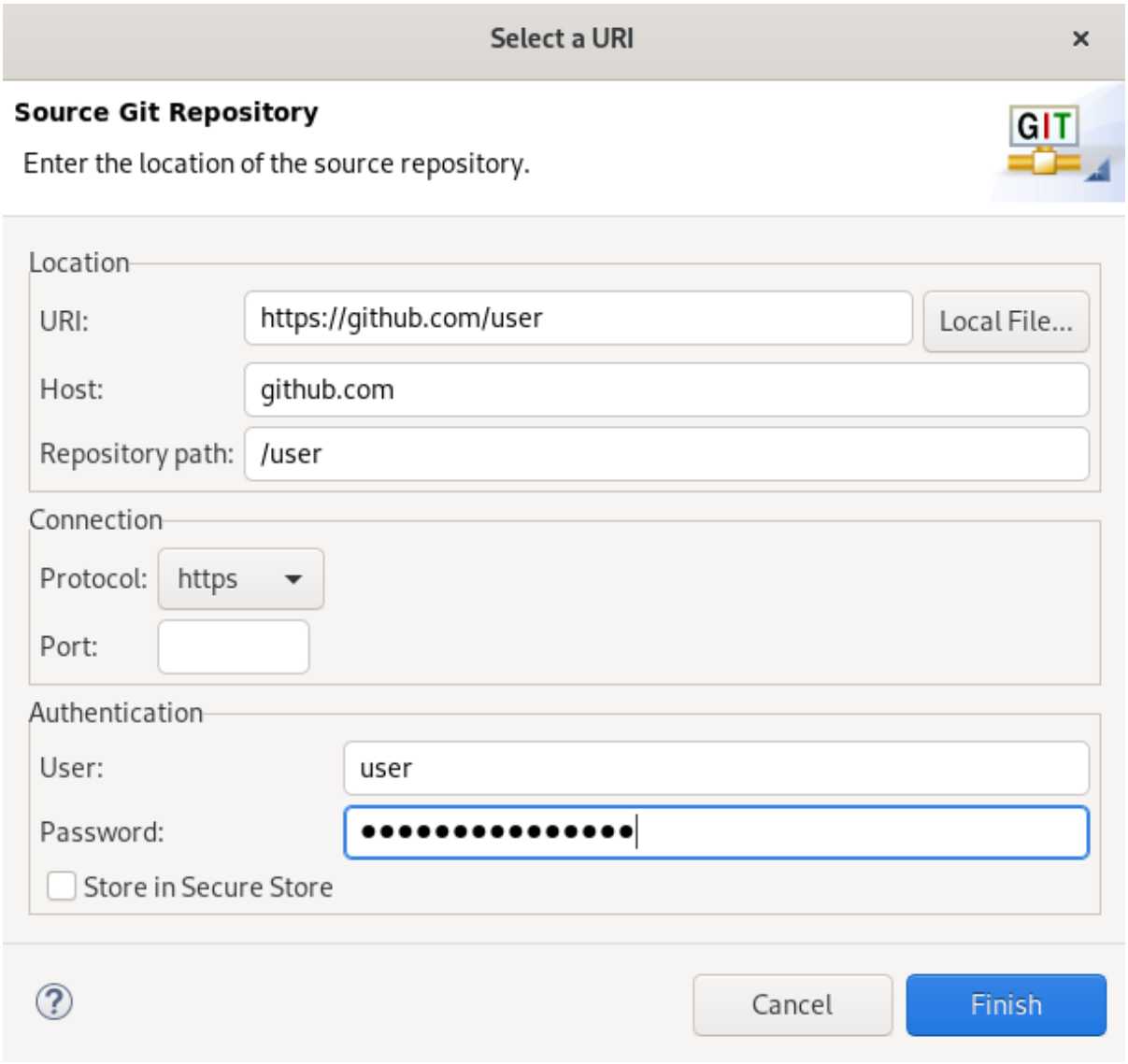
4. **Remotes → Create Remote** と右クリックします。
New Remote ウインドウが表示されます。



5. リモートに名前を付けます。
6. **Configure push** が選択されていることを確認します。
7. **Create** をクリックします。
   Configure Push ウィンドウが表示されます。

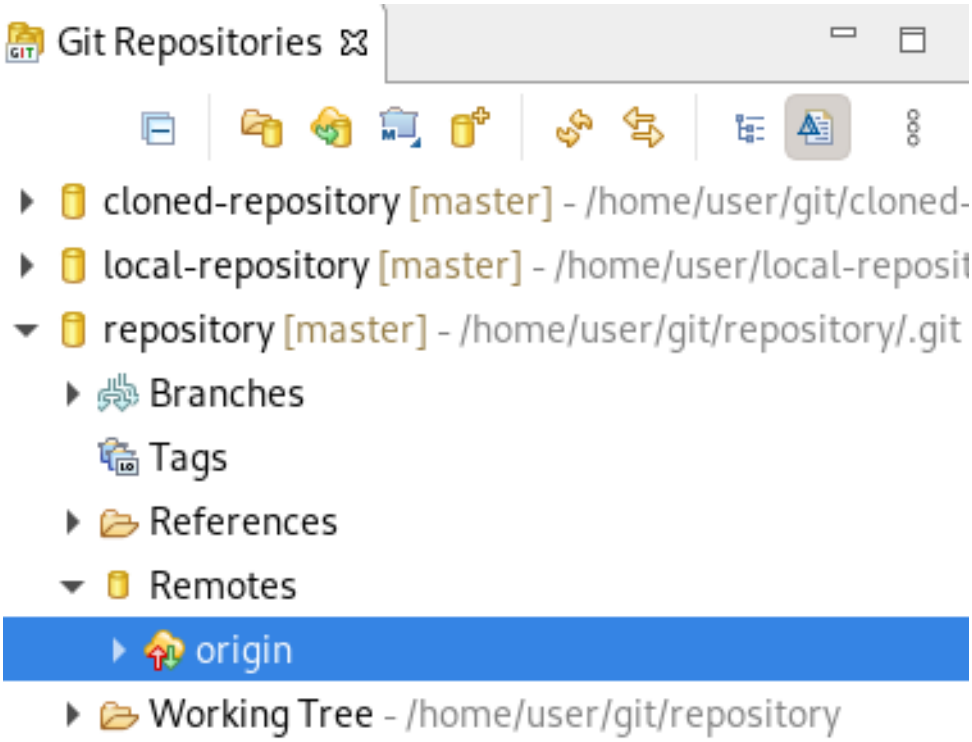


8. **Change** をクリックします。
Select a URI ウィンドウが表示されます。



9. ソースリポジトリーの URI、ユーザー名、およびパスワードを追加します。
Host および **Repository** パスフィールドが自動的に入力されます。
10. **Finish** をクリックします。
11. **Save** をクリックします。

新たに追加されたリモートが、CodeReady Studio の **Git Repositories** ビューに表示されます。



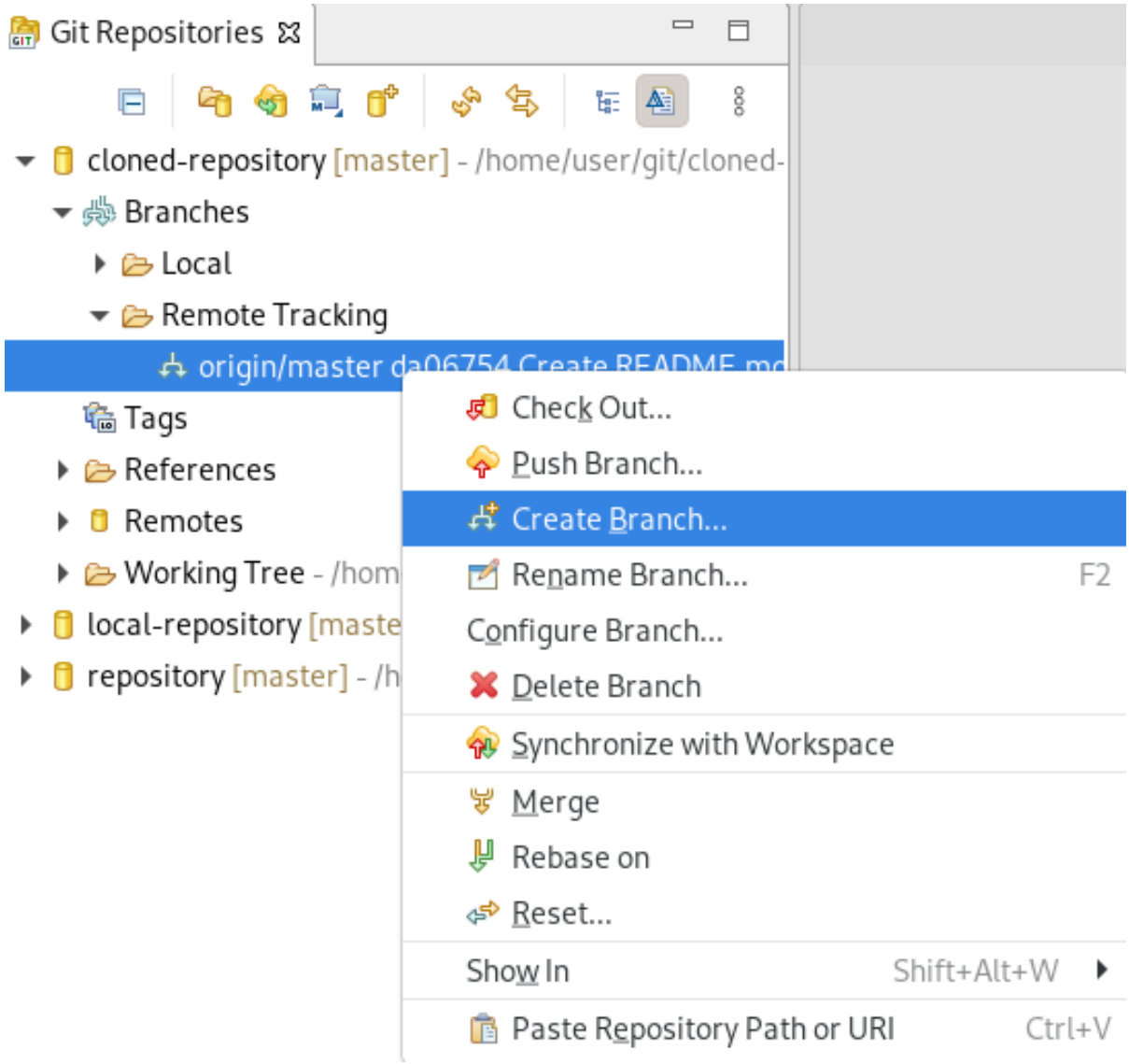
1.3. GIT PERSPECTIVE でのブランチの管理

1.3.1. 新規ブランチの作成

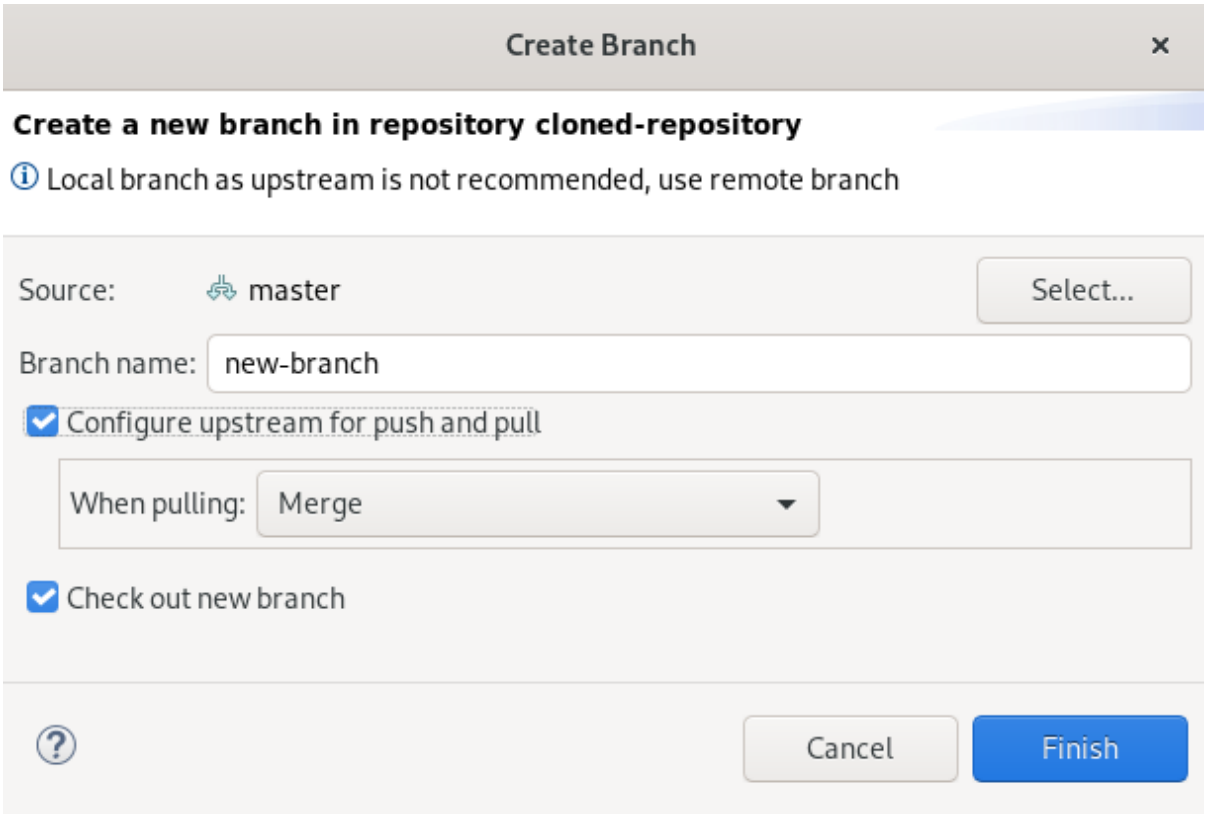
Git パースペクティブを使用して新規ブランチを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Git Perspective** を開きます。
3. リポジトリーを展開して、すべてのリモートブランチを表示します。
4. **master → Create Branch** と右クリックします。

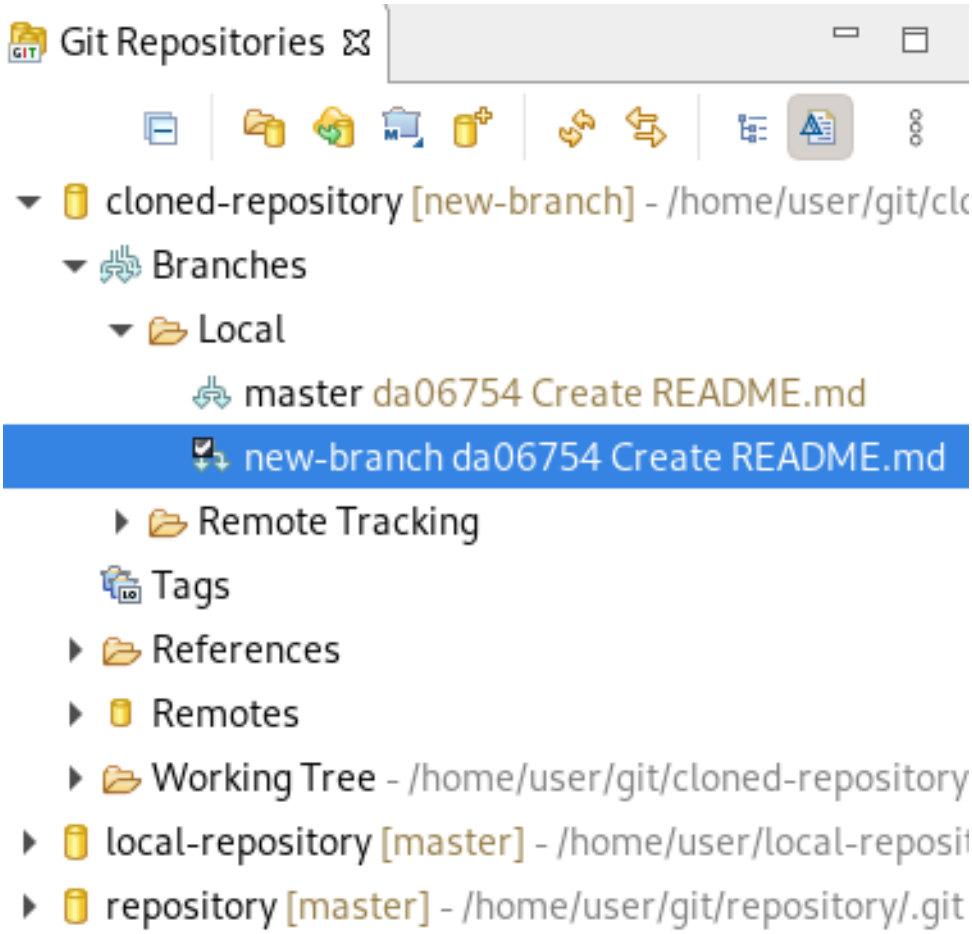


Create Branch ウインドウが表示されます。



5. **Select** をクリックして、新規ブランチのソースを選択します。
6. ブランチに名前を付けます。
7. **Configure upstream for push and pull**および **Checkout new branch** チェックボックスを選択します。
8. **When pulling** フィールドで、適切なオプションを選択します。
9. **Finish** をクリックします。

新たに追加されたブランチが、CodeReady Studio の **Git Repositories** ビューに表示されます。

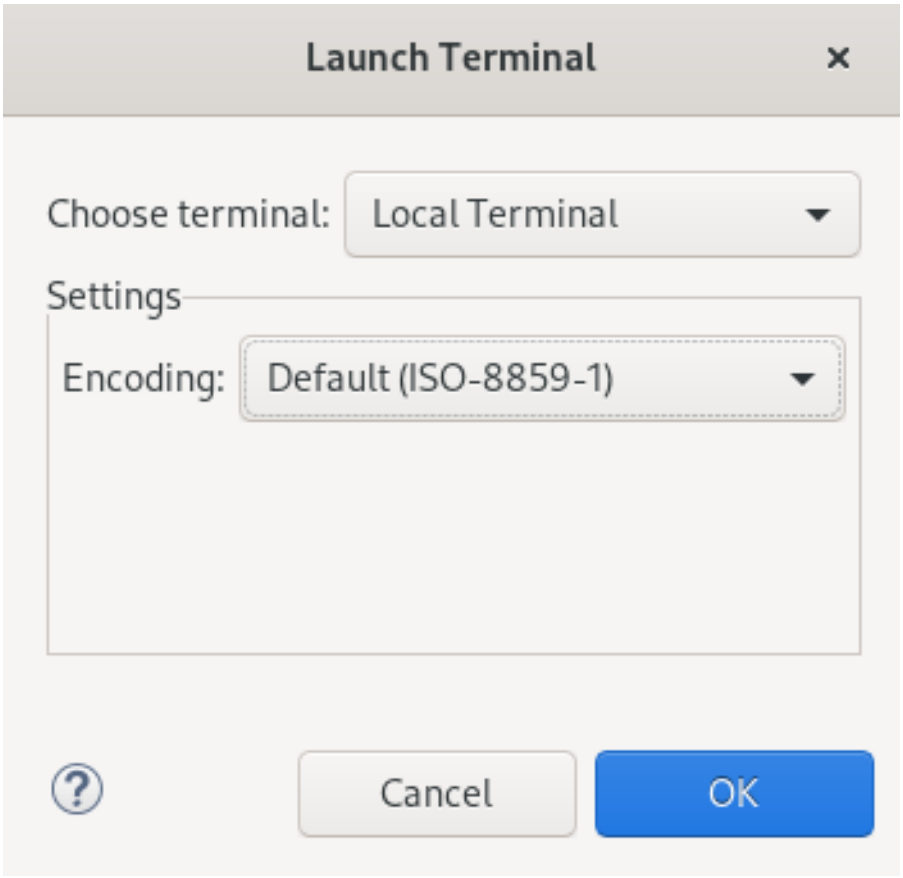


1.3.2. ブランチの使用

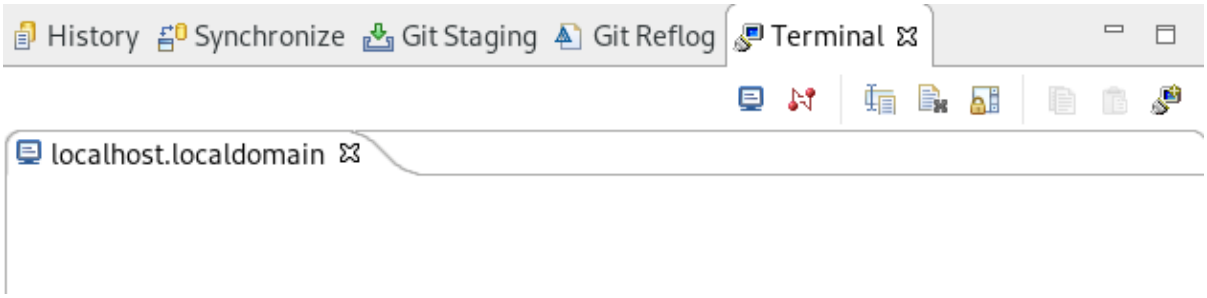
Git Perspective で組み込みターミナルを開く方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Git Perspective** を開きます。
3. **Shift**+**Ctrl**+**Alt**+**T** キーを押します。
   Launch Terminal ウインドウが表示されます。



4. **Local Terminal** を選択します。
5. **Encoding** を **Default (ISO-8859-1)** に設定します。
6. **OK** をクリックします。
Terminal ウインドウにコマンドラインターミナルが表示されます。



デフォルトでは、現在の作業用ディレクトリーは、現在のユーザーのホームディレクトリーであることに注意してください。

1.3.3. プルリクエストの送信

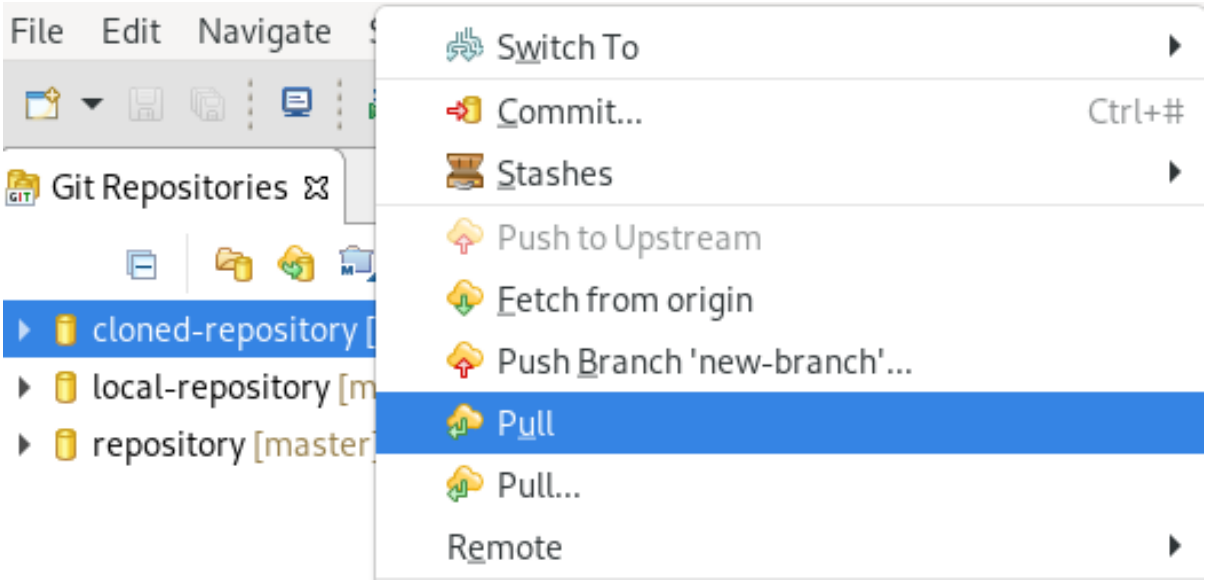
特に共有リポジトリーで作業を行う場合、変更をマージする前に、ローカルリポジトリーを更新することが強く推奨されます。

Git Perspective を使用してプルリクエスト (PR) を送信する方法を説明します。

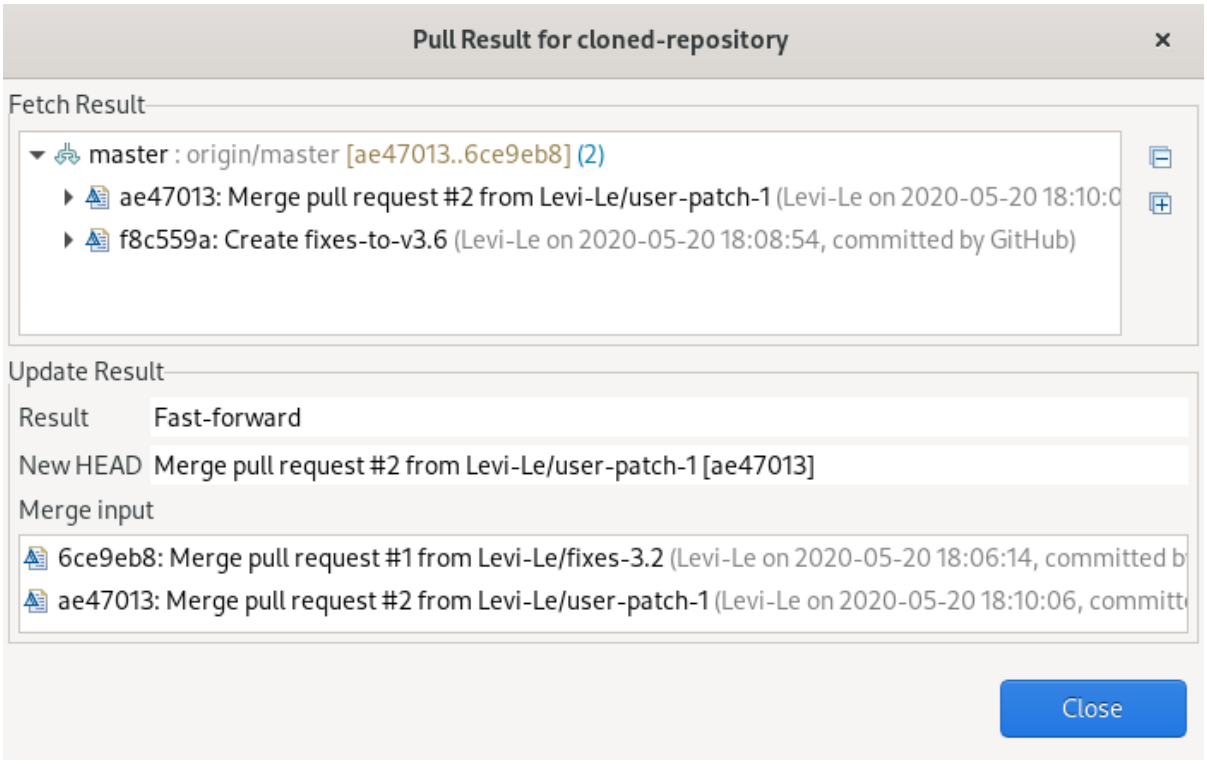
手順

1. CodeReady Studio を起動します。

2. **Git Perspective** を開きます。
3. **repository → Pull** と右クリックします。



Pull Results ウインドウが表示されます。



4. **Close** をクリックします。

これで、リモートリポジトリーからの変更がローカルリポジトリーにマージされます。

1.4. 変更のコミットおよびプッシュ

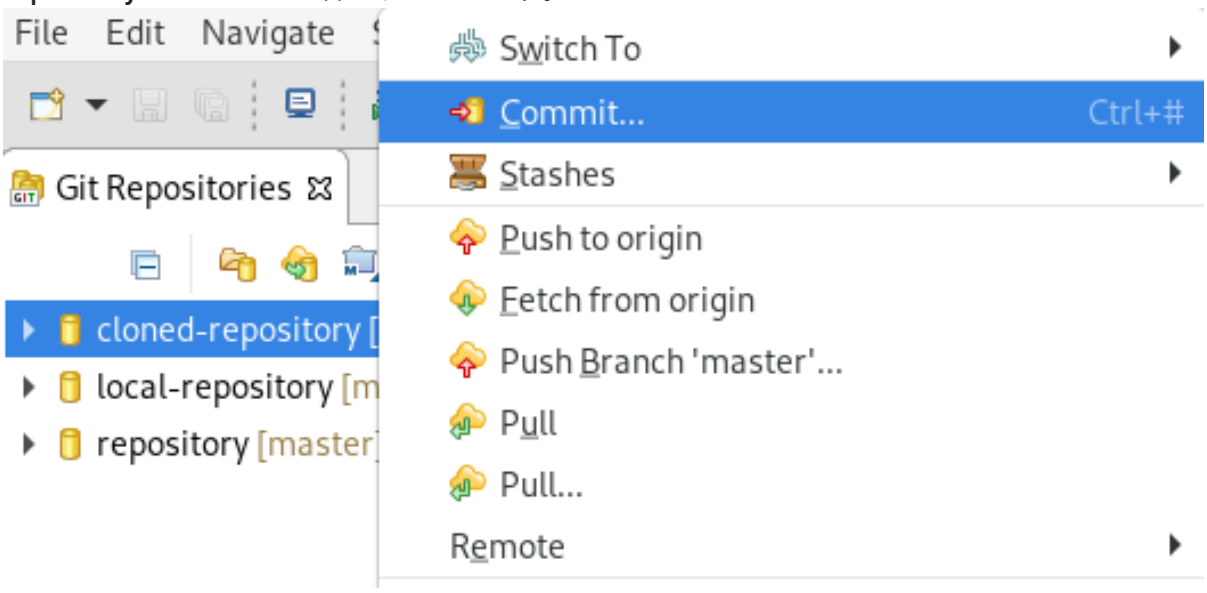
CodeReady Studio で変更をコミットおよびプッシュする方法を説明します。

手順

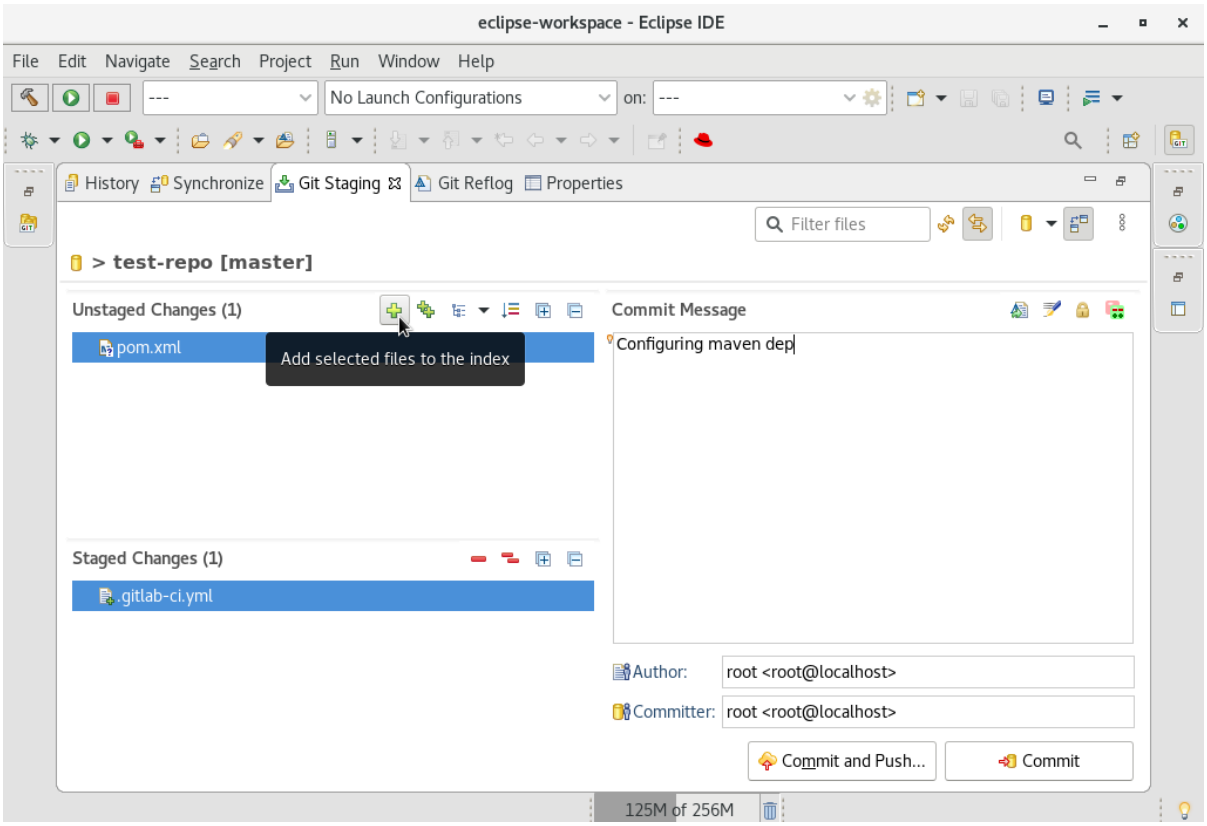
1. CodeReady Studio を起動します。

2. Git Perspective を開きます。

3. repository → Commit と右クリックします。



Git Staging ビューが表示されます。



4. ステージする変更を選択します。

5. Add selected files to the indexアイコンをクリックし、変更をステージします。

6. Commit Message フィールドにコミットメッセージを追加します。
Author および Committer フィールドが自動的に入力されます。

7. **Commit** をクリックして変更をコミットするか、**Commit and Push** をクリックして変更をコミットし、リモートリポジトリーにプッシュします。

Commit and Push オプションを選択すると、リポジトリーアドレスと、リポジトリーのアクセスユーザー名およびパスワードの入力を要求されることに注意してください。

第2章 CODEREADY STUDIO における MAVEN の基本

Maven はアプリケーション開発の標準化されたビルドシステムを提供し、1 つ以上のリポジトリーから依存関係のフェッチを容易にします。

ルート Maven プロジェクトは、複数の Maven モジュール (サブプロジェクト) のアグリゲーターとして提供できます。Maven プロジェクトの一部である各モジュールには、<module> エントリーがプロジェクトの **pom.xml** ファイルに追加されます。**pom.xml** には <module> エントリーが含まれ、**アグリゲーター pom** とも呼ばれます。

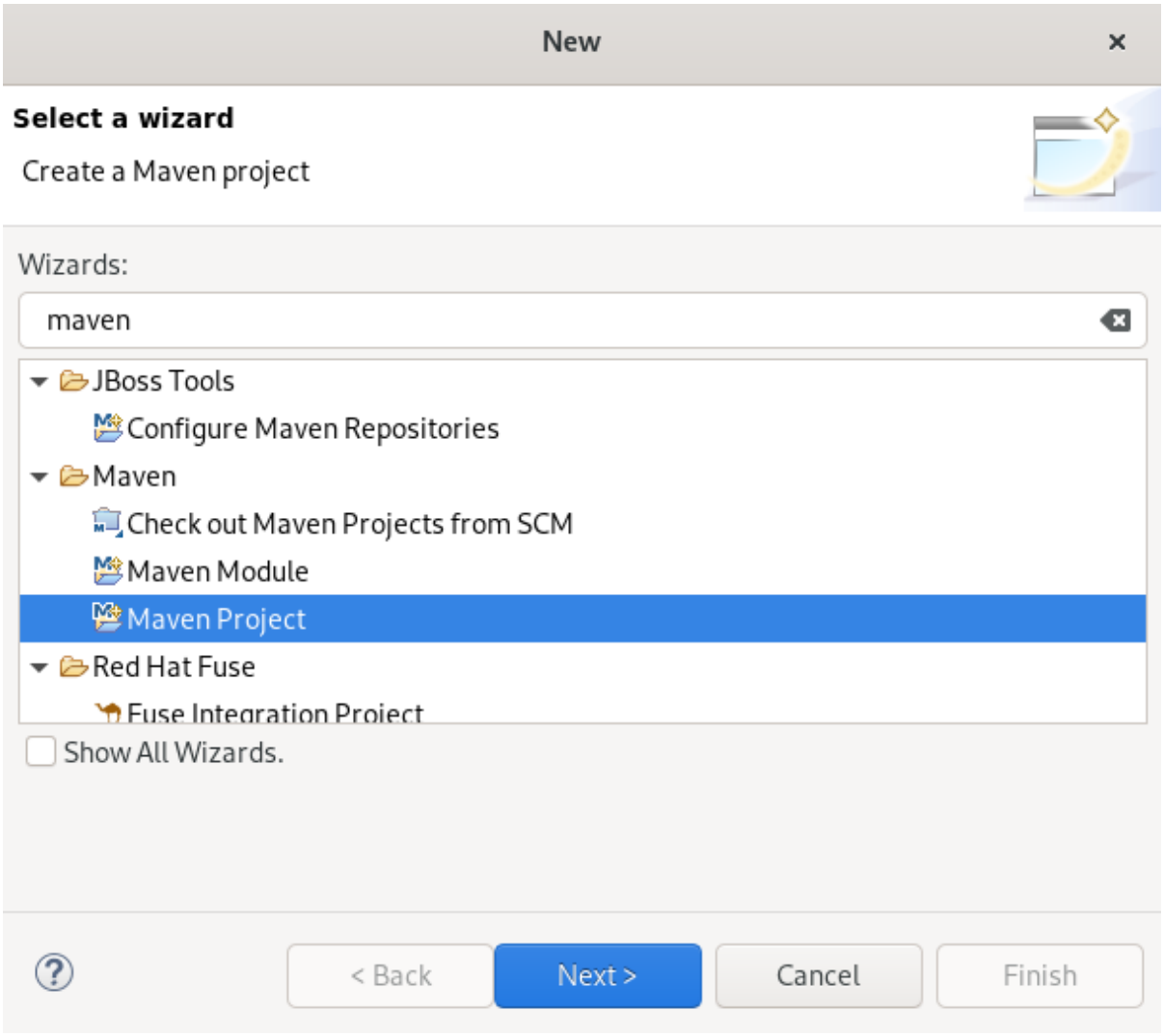
モジュールがプロジェクトに含まれる場合は、親プロジェクトディレクトリーから実行された 1 つコマンドにて、すべてのモジュールで Maven ゴールを実行できます。

2.1. 新規 MAVEN プロジェクトの作成

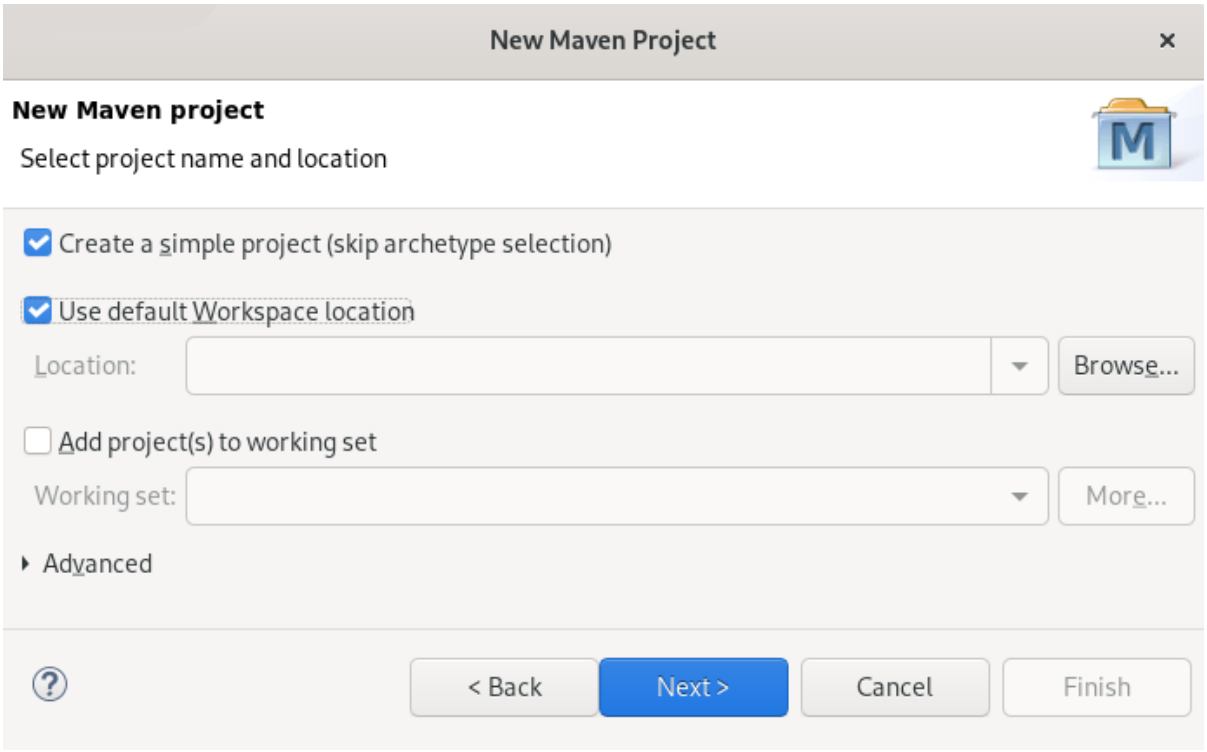
CodeReady Studio で新しい Maven プロジェクトを作成する方法を説明します。この手順では、パッケージオプションが、マルチモジュールの Maven プロジェクトの要件である **pom** に設定されていることを確認します。代わりにスタンドアロンの Maven プロジェクトを作成する場合は、パッケージオプションを **jar** または **war** に設定します。

手順

1. CodeReady Studio を起動します。
2. **Ctrl**+**N** キーを押します。
   Select a wizard ウインドウが表示されます。



3. **Wizards** フィールドに **Maven** と入力します。
4. **Maven Projects** を選択します。
5. **Next** をクリックします。
New Maven Project ウインドウが表示されます。

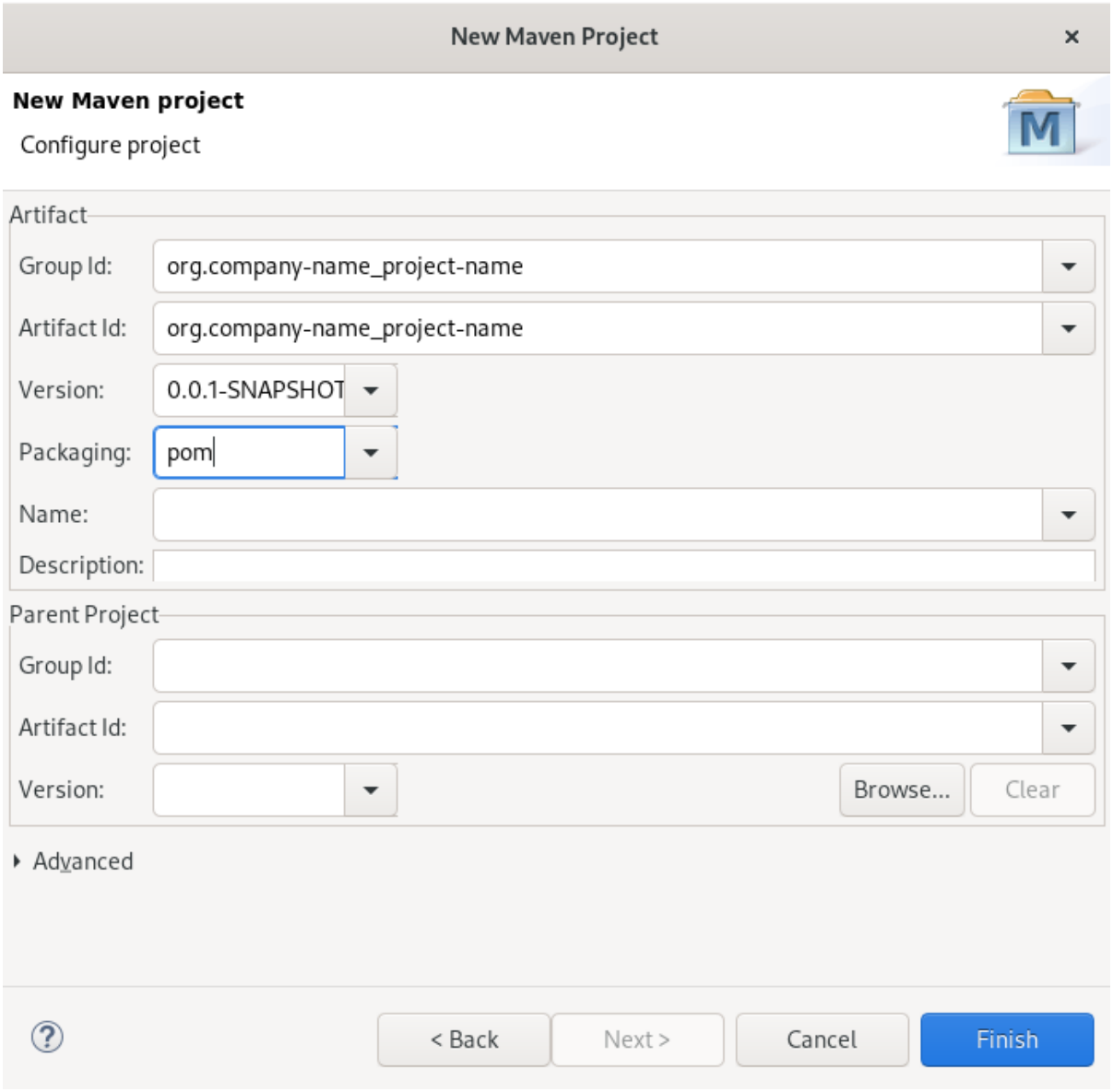


6. **Create a simple project**チェックボックスを選択します。

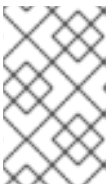
注記

Create a simple projectチェックボックスを選択すると、archetype の選択が省略され、プロジェクトタイプは POM (Project Object Model) に自動設定されます。スタンドアロンのプロジェクトを作成するには、**Create a simple project** チェックボックスを未選択にし、画面の指示に従います。

7. **Browse** をクリックしてワークスペースの場所を選択します。
8. **Next** をクリックします。



9. グループ ID とアーティファクト ID を入力します。



注記

値には、スペースや特殊文字を使用できません。使用できる特別文字は、ピリオド (.)、アンダースコア (_)、およびダッシュ (-) のみです。一般的なグループ ID またはアーティファクト ID の例は **org.company-name_project-name** です。

必要に応じて、プロジェクトに名前を付け、説明を追加できます。

10. **Packaging** を **pom** に設定します。
11. **Finish** をクリックします。

新たに作成された Maven プロジェクトが CodeReady Studio ビューに表示されます。

2.2. 既存の MAVEN プロジェクトのインポート

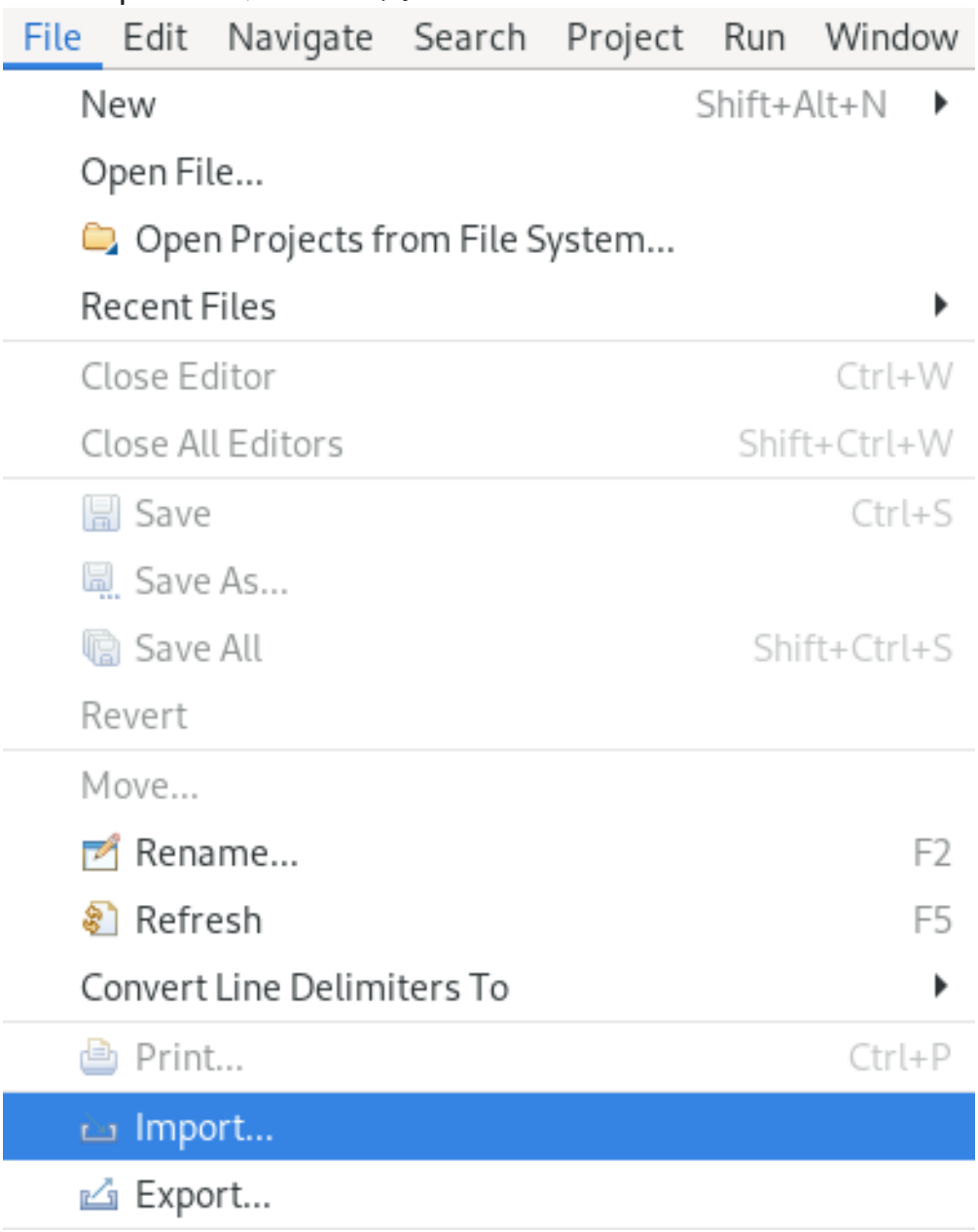
既存の Maven プロジェクトを CodeReady Studio にインポートする方法を説明します。

2.2.1. ローカルに保存された既存の Maven プロジェクトのインポート

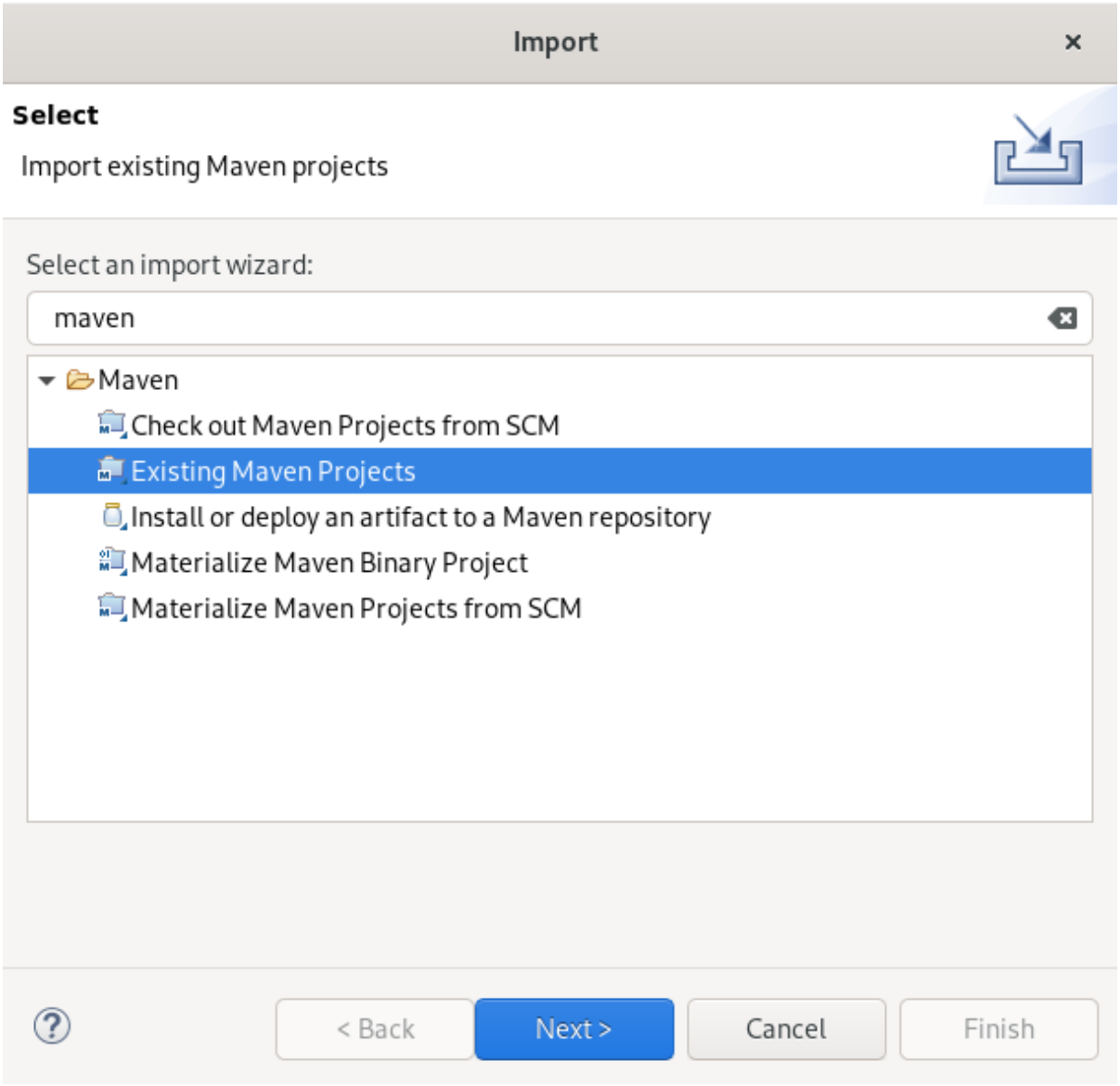
ローカルに保存された既存の Maven プロジェクトを CodeReady Studio にインポートする方法を説明します。

手順

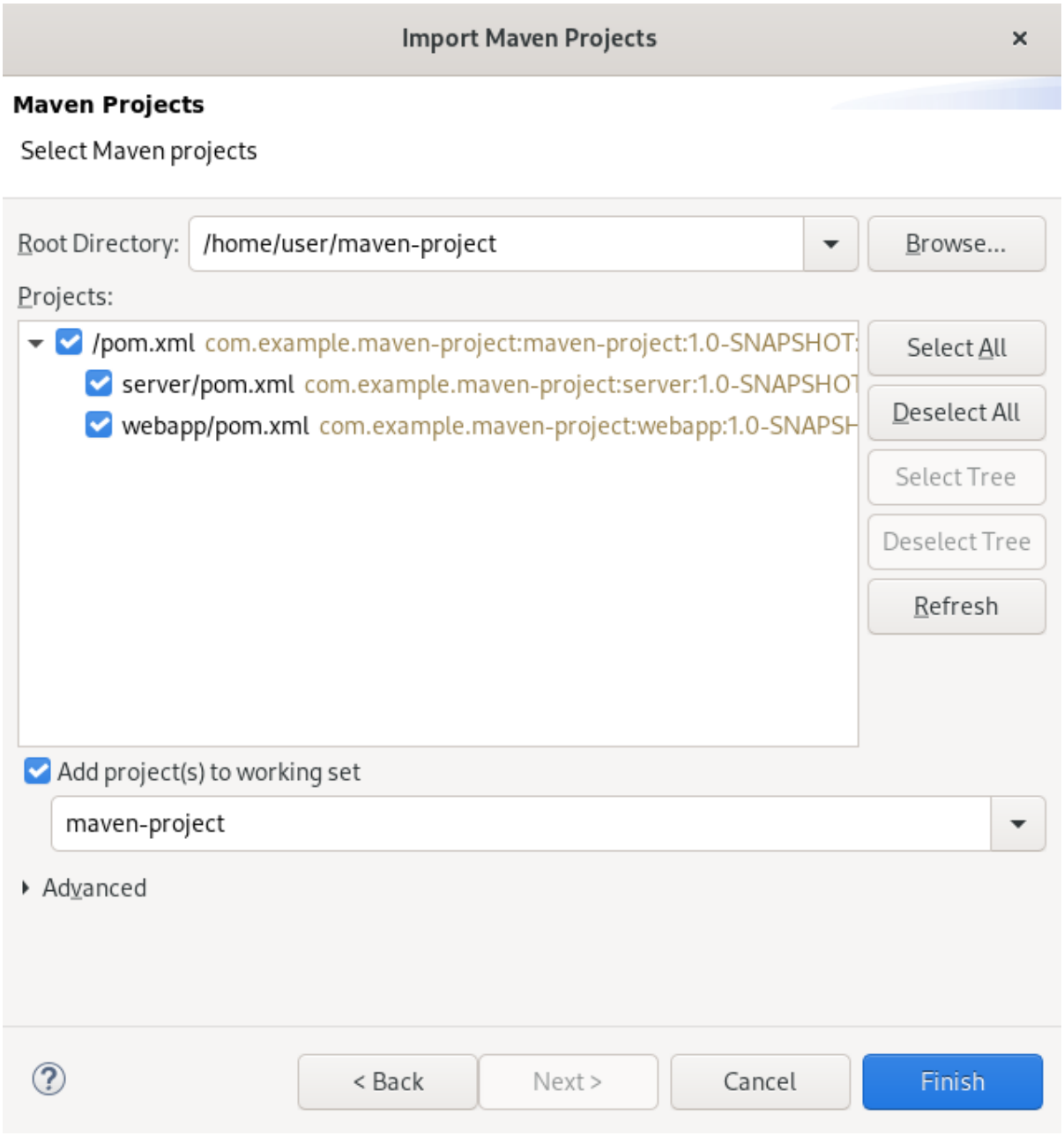
1. CodeReady Studio を起動します。
2. **File → Import** とクリックします。



Import ウインドウが表示されます。



3. **Select an import wizard** フィールドに **Maven** と入力します。
4. **Existing Maven Projects** を選択します。
5. **Next** をクリックします。
   Import Maven Project ウインドウが表示されます。



6. **Browse** をクリックして、Maven プロジェクトを見つけます。
7. **Finish** をクリックします。

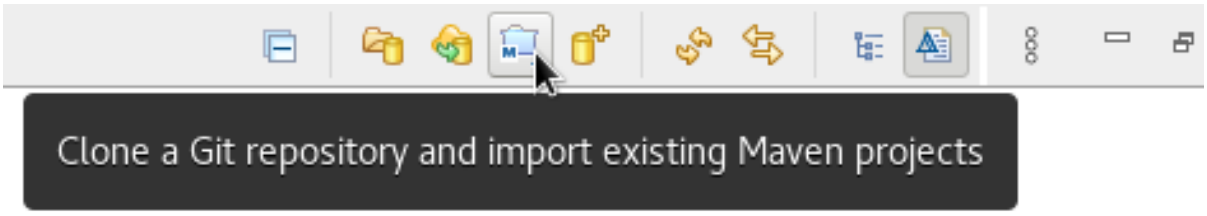
ローカルの Maven プロジェクトが CodeReady Studio ビューに表示されます。

2.2.2. リモートで保存された 既存の Maven プロジェクトのインポート

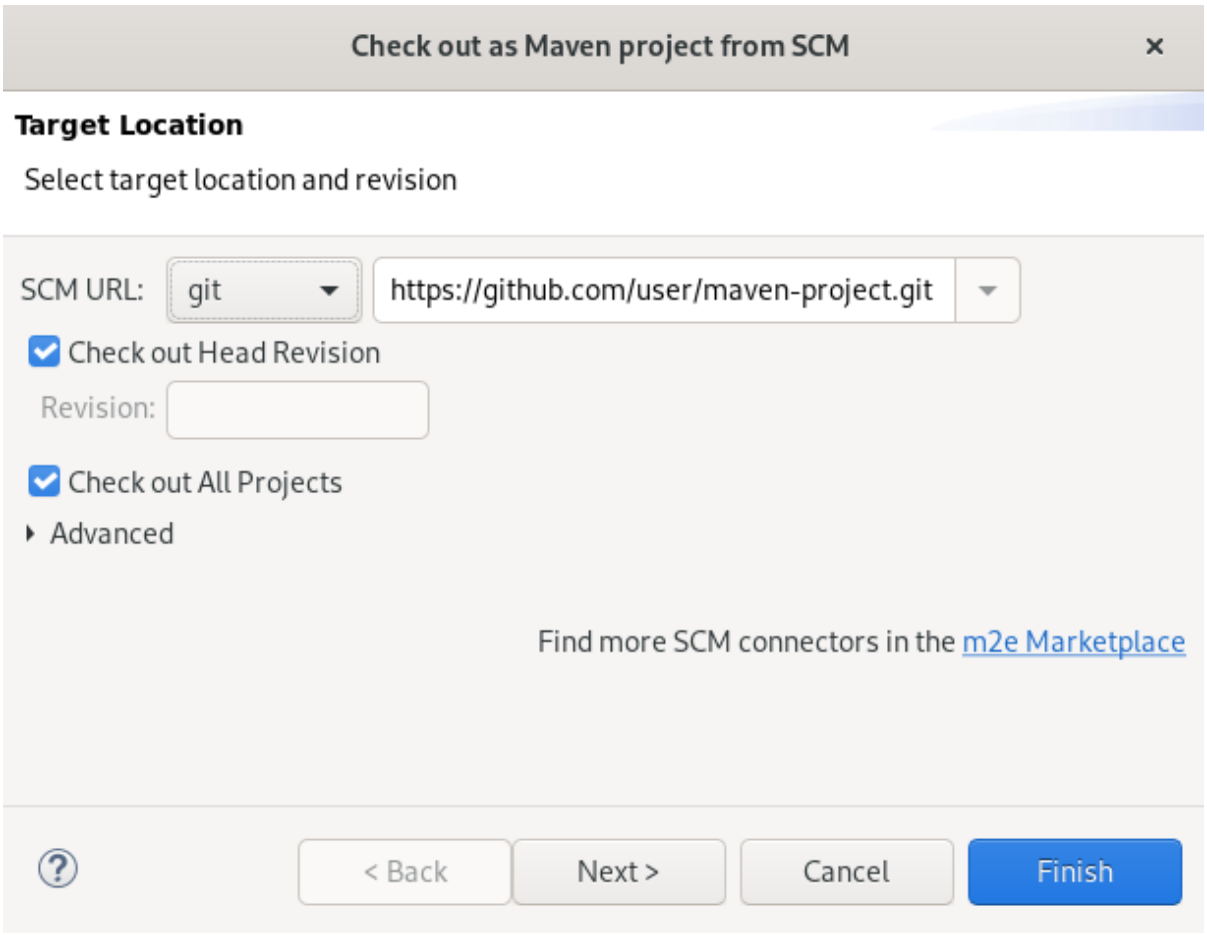
リモートで保存された既存の Maven プロジェクトを CodeReady Studio にインポートする方法を説明します。

手順

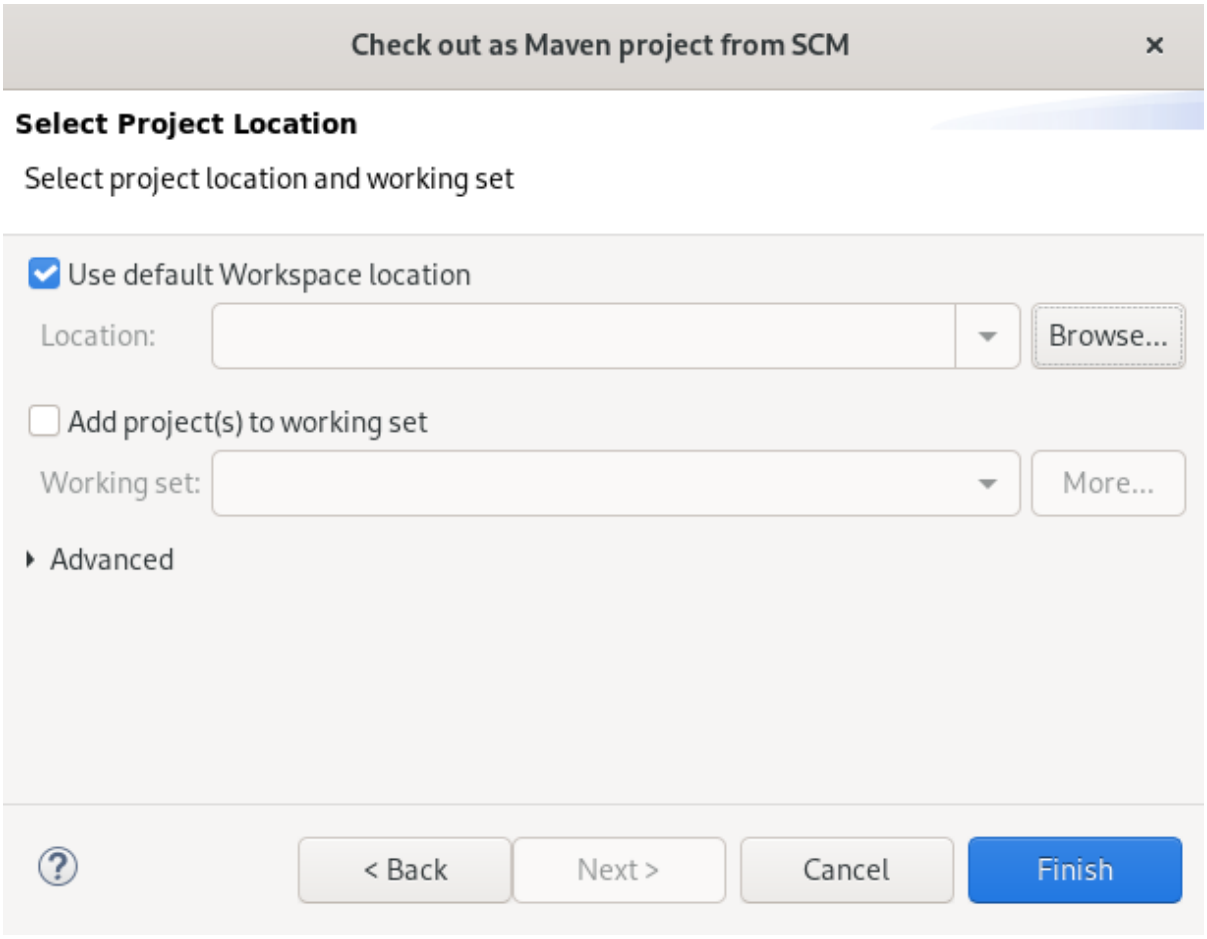
1. CodeReady Studio を起動します。
2. Git Perspective を開きます。
3. **Clone a Git repository and import existing Maven projects**アイコンをクリックします。



Check out as Maven project from SCMウィンドウが表示されます。



4. ソースリポジトリーのアドレスを **SCM URL** フィールドに追加します。
5. **Next** をクリックします。
   Select Project Location ウインドウが表示されます。



6. **Browse** をクリックしてワークスペースの場所を選択します。
7. **Finish** をクリックします。

リモート Maven プロジェクトが **Git Perspective** ビューに追加されます。

2.3. 新しい MAVEN モジュールの作成

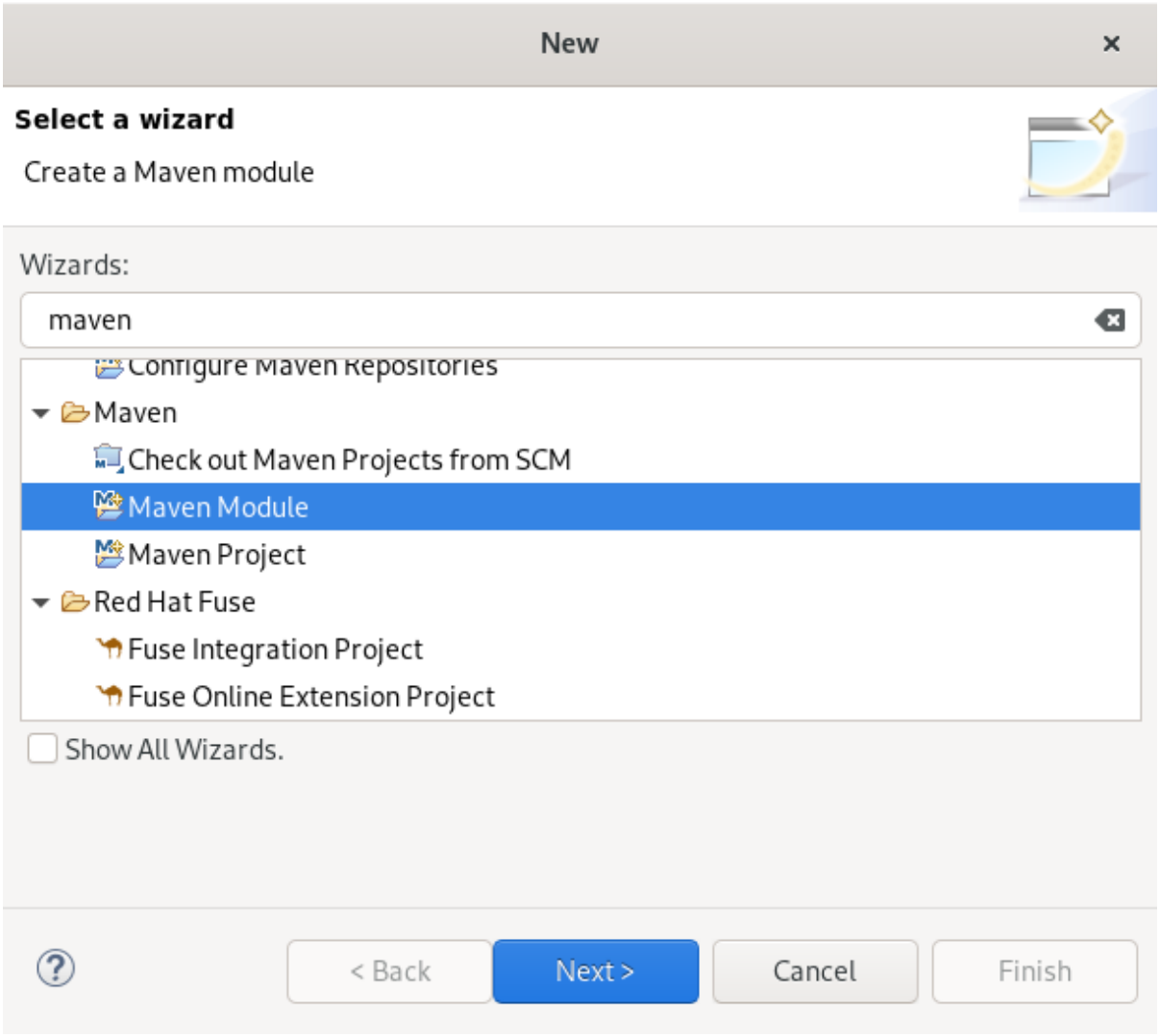
新しい Maven モジュールの作成方法を説明します。

前提条件

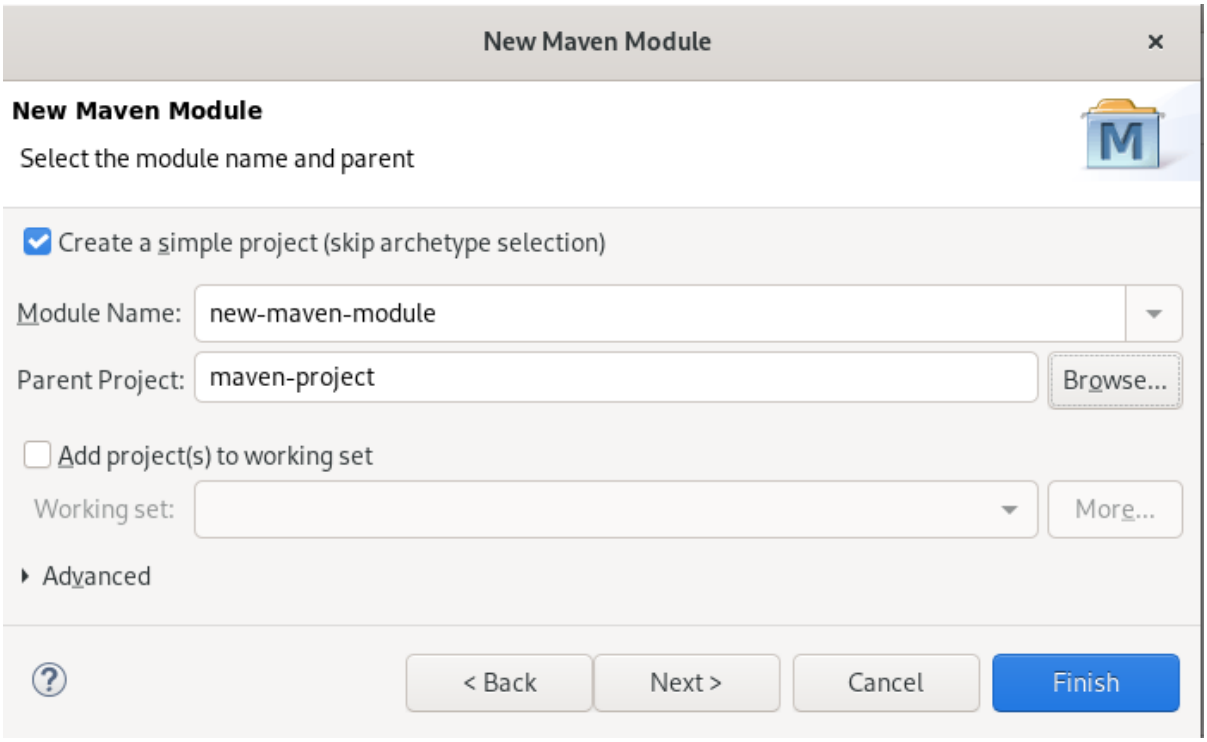
- 既存の Maven プロジェクト。
  Maven プロジェクトの作成方法に関する詳細は、「新規 Maven プロジェクトの作成」 を参照してください。

手順

1. CodeReady Studio を起動します。
2. **Ctrl**+**N** キーを押します。
   Select a wizard ウインドウが表示されます。



3. **Wizards** フィールドに **Maven** と入力します。
4. **Maven Module** を選択します。
5. **Next** ボタンをクリックします。
   New Maven Module ウインドウが表示されます。

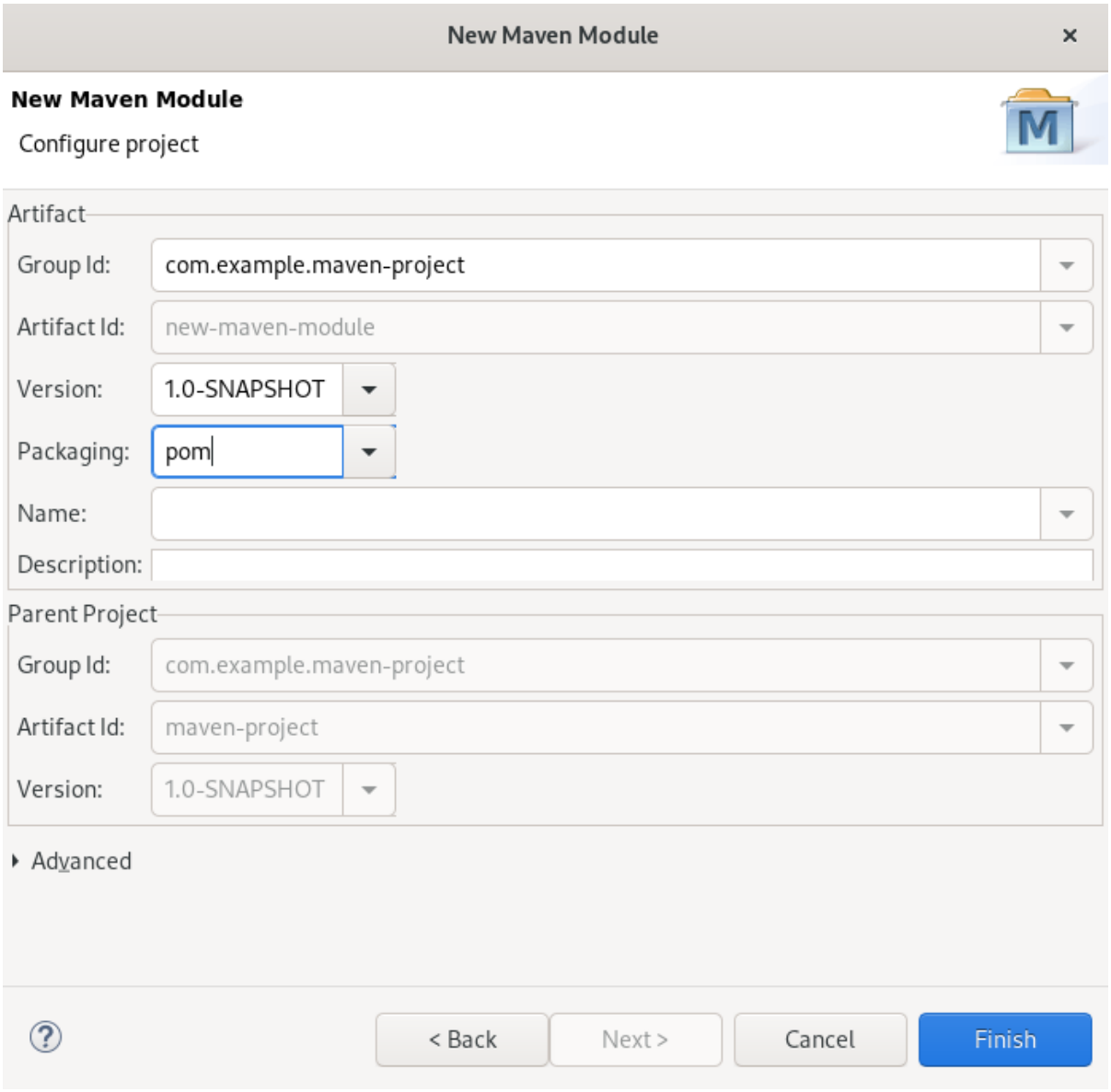


6. **Create a simple project**チェックボックスを選択します。

注記

Create a simple projectチェックボックスを選択すると、archetype の選択が省略され、プロジェクトタイプは POM (Project Object Model) に自動設定されます。スタンドアロンのモジュールを作成するには、**Create a simple project** チェックボックスを未選択にし、画面の指示に従います。

7. モジュールに名前を付けます。
8. **Browse** をクリックして、親プロジェクトを選択します。
9. **Next** をクリックします。
Configure Project ウインドウが表示されます。



10. **Packaging** を **pom** に設定します。
必要に応じて、モジュールに名前を付け、説明を追加できます。

11. **Finish** をクリックします。

新たに作成された Maven モジュールが CodeReady Studio ビューに表示されます。

2.4. MAVEN 依存関係の MAVEN プロジェクトへの追加

CodeReady Studio で Maven プロジェクトに Maven 依存関係を追加する方法を説明します。

前提条件

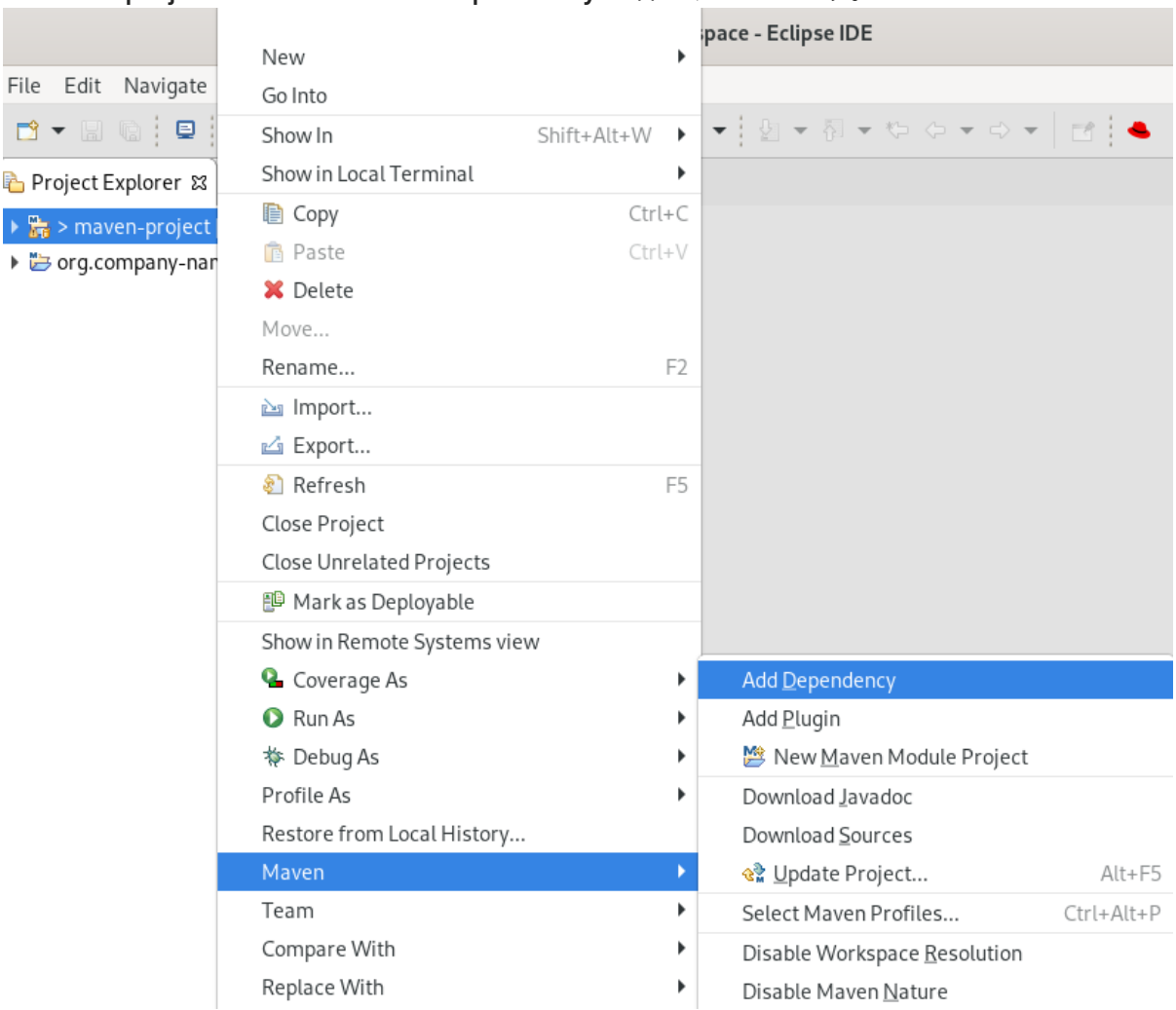
- 既存の Maven プロジェクト。
Maven プロジェクトの作成方法に関する詳細は、「新規 Maven プロジェクトの作成」 を参照してください。

手順

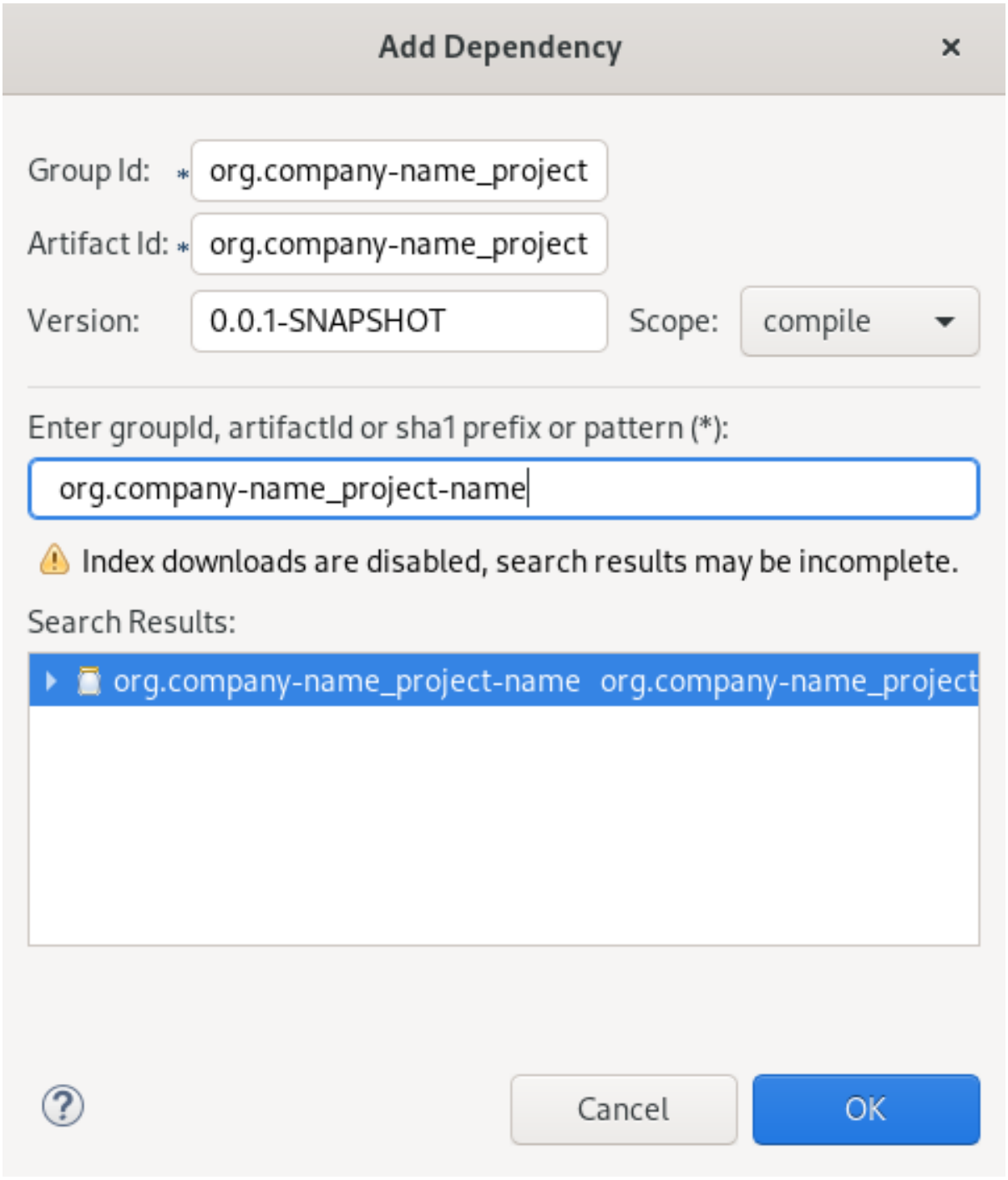
1. CodeReady Studio を起動します。

2. Project Explorer を開きます。

3. Maven の **project → Maven → Add Dependency** を右クリックします。



Add Dependency ウインドウが表示されます。



4. **Enter groupId, artifactId or sha1 prefix or pattern**フィールドに、グループ ID またはアーティファクト ID を入力します。
   上記のフィールドは自動的に入力されます。

5. **OK** をクリックします。

依存関係がプロジェクトの **pom.xml** ファイルに追加されます。

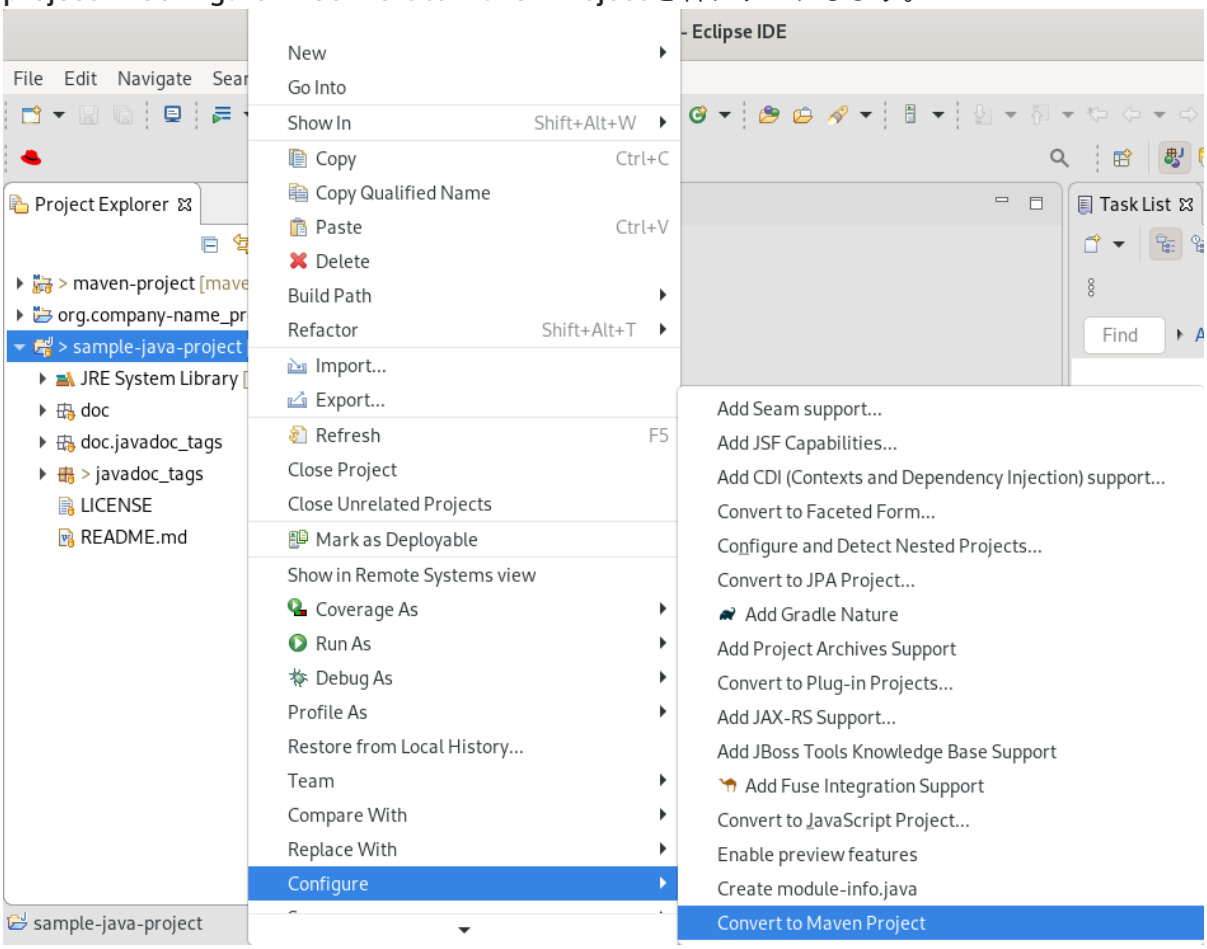
2.5. MAVEN サポートを MAVEN 以外の既存プロジェクトへ追加

Maven サポートを使用せずに作成されたアプリケーションに Maven サポートを追加する方法を説明します。

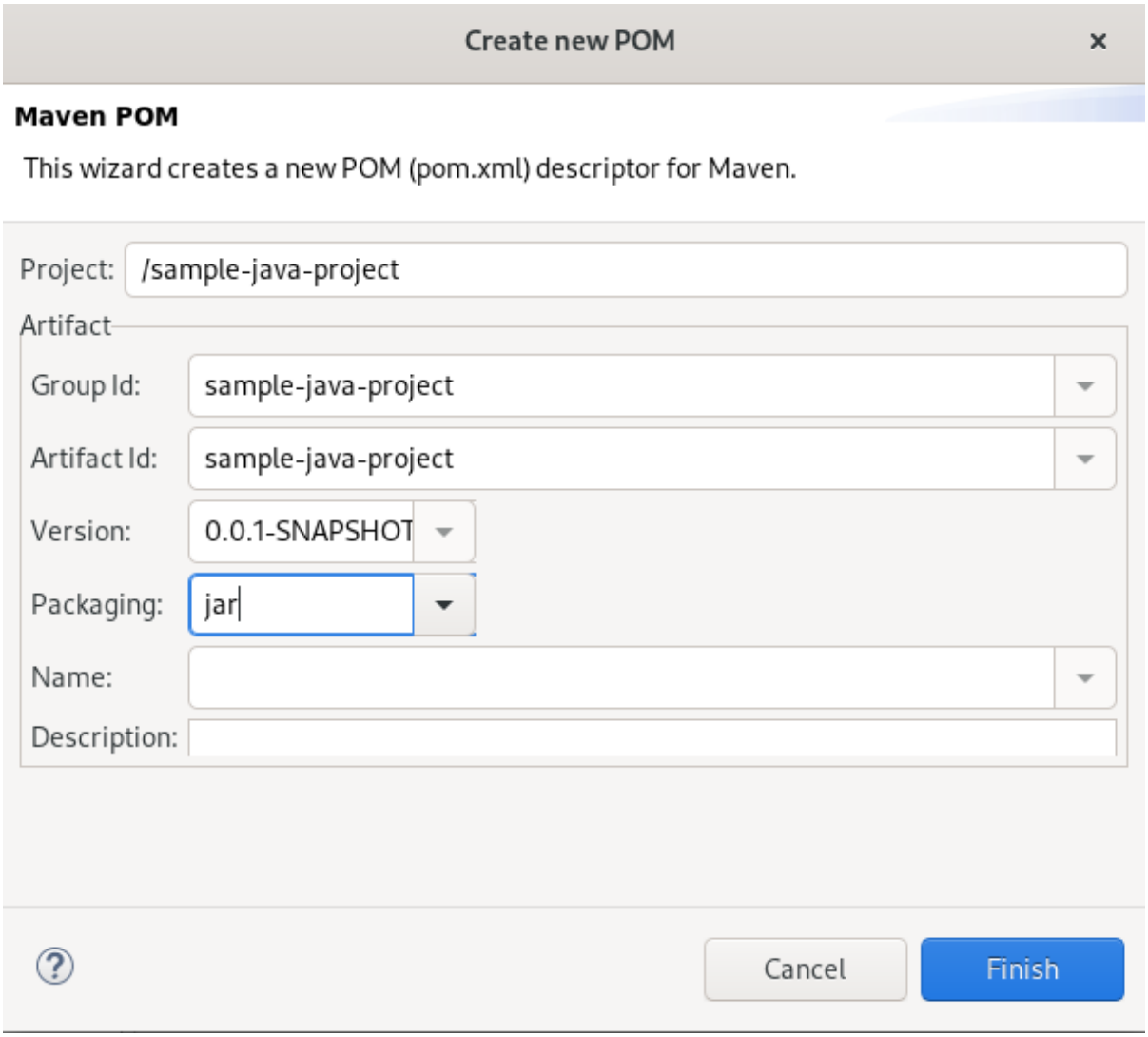
1. CodeReady Studio を起動します。

2. Project Explorer を開きます。

3. **project → Configure → Convert to Maven Project**を右クリックします。



Create a new POMウインドウが表示されます。



すべてのフィールドは自動的に入力されます。グループ ID またはアーティファクト ID を変更する場合は、値にスペースまたは特殊文字を使用できません。使用できる特別文字は、ピリオド (.)、アンダースコア (_)、およびダッシュ (-) のみです。

4. **Finish** をクリックします。
新たに生成された **pom.xml** ファイルが CodeReady Studio ビューに表示されます。

2.6. その他のリソース

- Maven の使用方法については、「JBoss Community Archive」を参照してください。

第3章 CODEREADY STUDIO でのアプリケーションデプロイメント

IDE 内からサーバーにアプリケーションをデプロイするには、サーバーに関する情報で IDE を設定する必要があります。ローカルサーバーの場合、この情報には以下が含まれます。

- サーバーランタイム環境 (サーバーの場所、ランタイム JRE、設定ファイルの詳細など)
- サーバーアダプターとサーバーランタイム環境の管理設定 (アクセスパラメーター、起動引数、パブリッシュオプションなど)

JBoss Server Tools を使用すると、Runtime Detection (ランタイム検出) を使用して、ローカルサーバーを効率的に設定し、IDE と使用できるようにすることができます。この機能はアプリケーションのデプロイとテストを行うサーバーを迅速に設定するのに役立ちます。

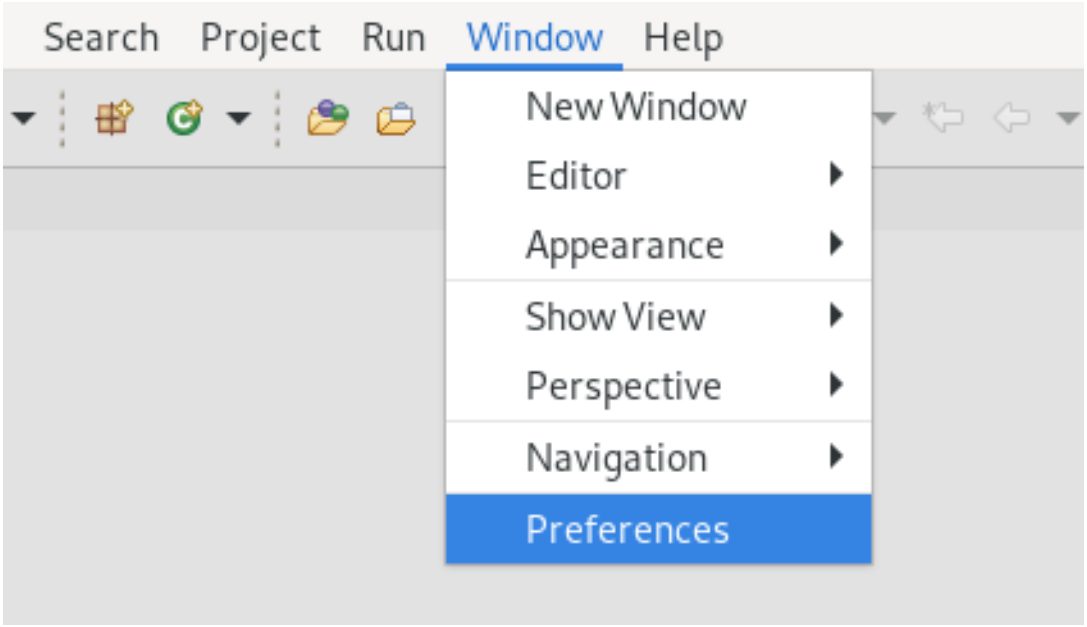
3.1. ローカルサーバーの設定

Runtime Detection (ランタイム検出) は、指定のローカルシステムパスを検索し、特定タイプのランタイムサーバーを見つけます。Runtime Detection は、見つかったサーバーに対してデフォルトのサーバーランタイム環境とデフォルトのサーバーアダプターの両方を自動的に生成します。これらは、アプリケーションを即座にデプロイするためにそのまま使用したり、要件に合わせてカスタマイズしたりできます。

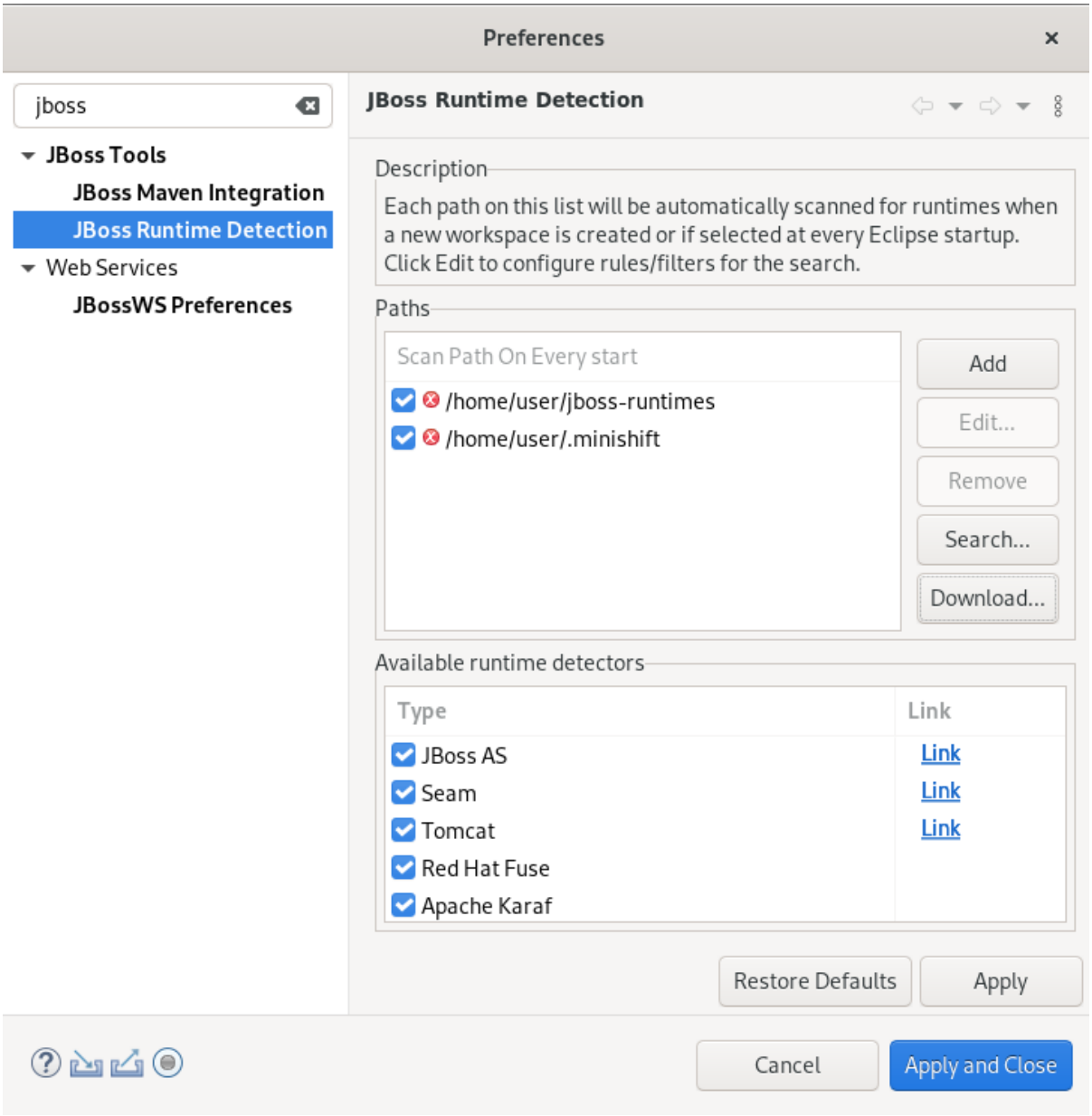
CodeReady Studio でローカルサーバーを設定する方法を説明します。

手順

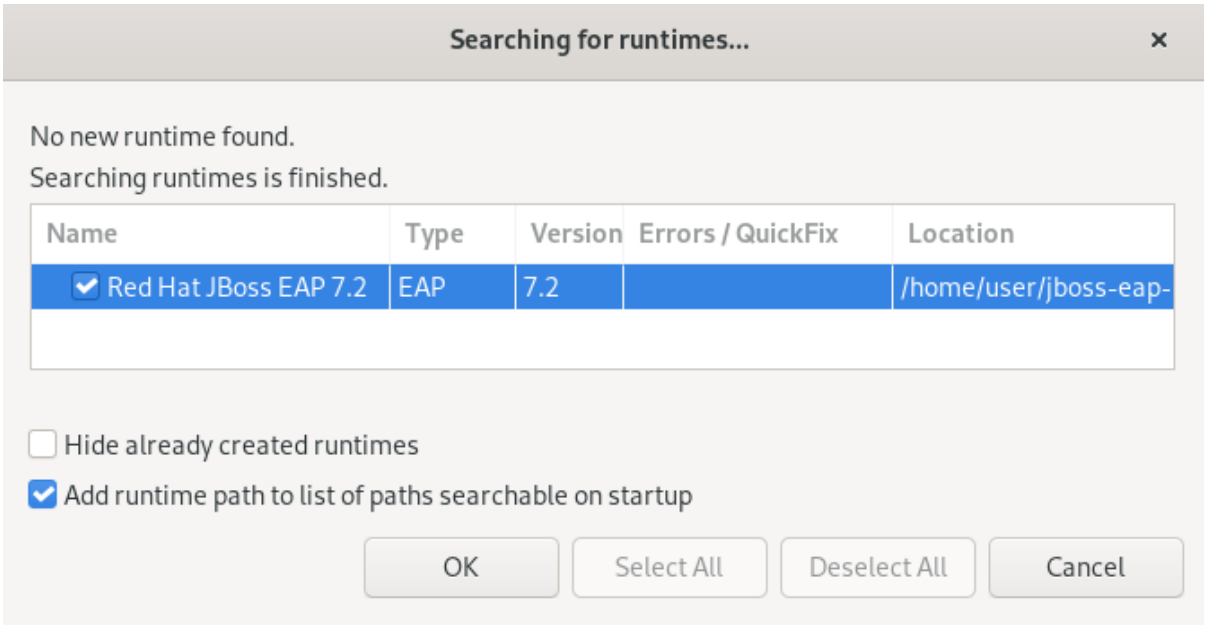
1. CodeReady Studio を起動します。
2. **Window → Preferences** をクリックします。



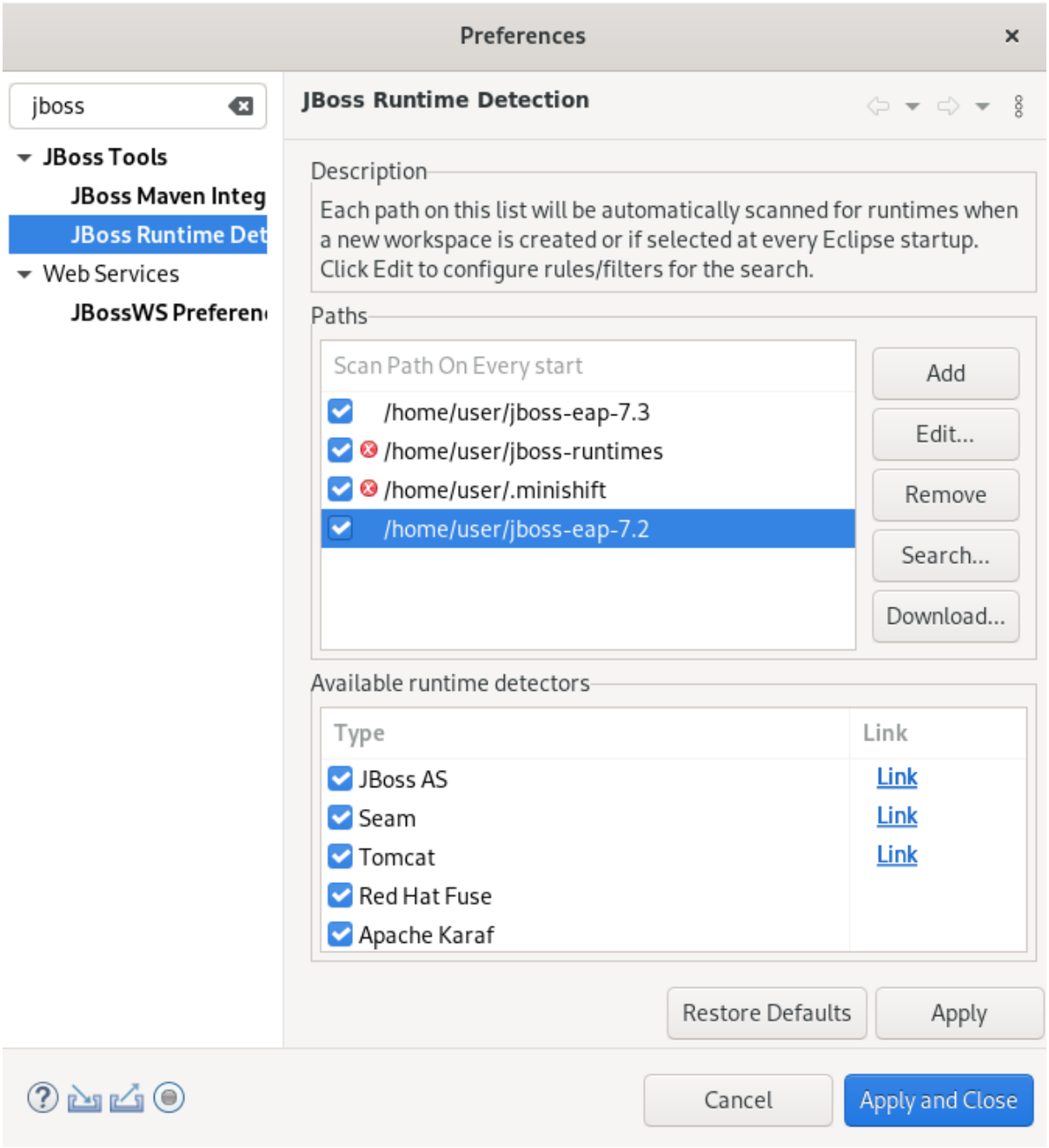
Preferences ウィンドウが表示されます。



3. 検索フィールドに **JBoss** と入力します。
4. **JBoss Runtime Detection** を選択します。
5. **Add** ボタンをクリックします。
6. ランタイムサーバーが含まれるディレクトリーを見つけます。
7. **Open** をクリックします。
   Searching for runtimes ウィンドウが表示されます。



8. **OK** をクリックします。
9. ランタイムサーバーのディレクトリーへのパスを選択します。



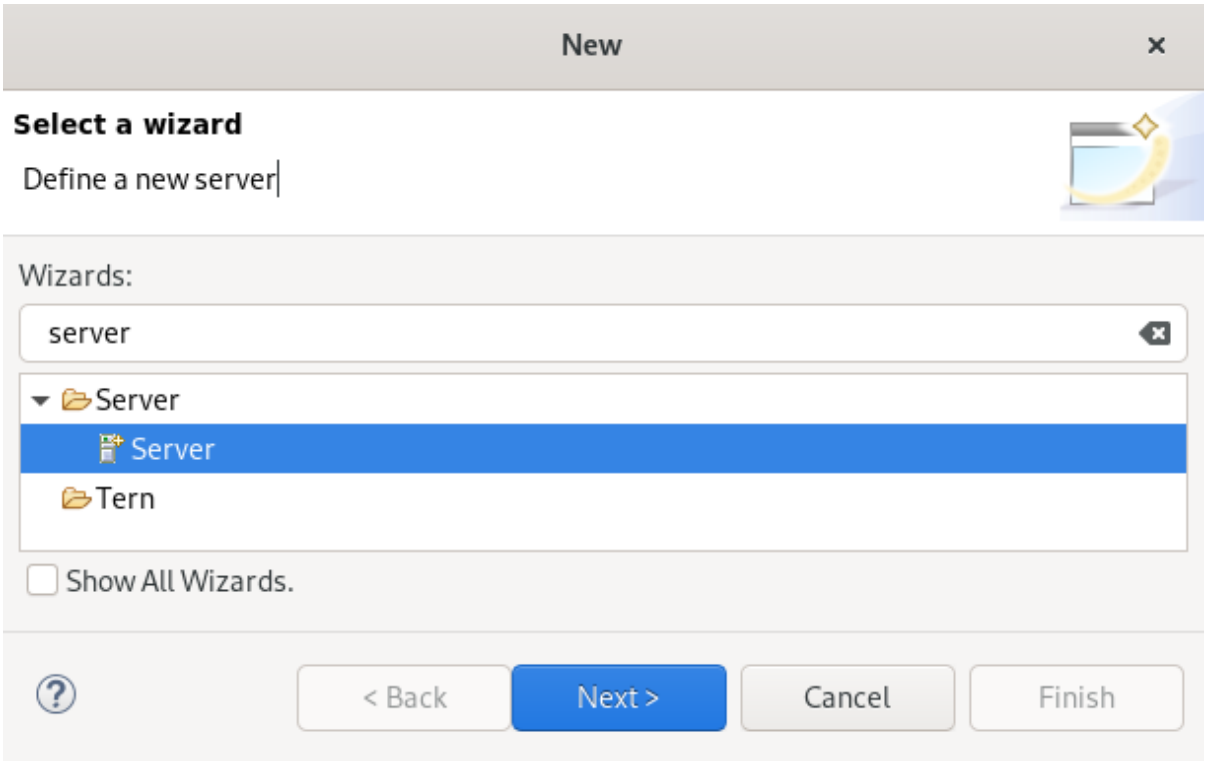
10. **Apply and Close** をクリックします。

3.2. リモートサーバーの設定

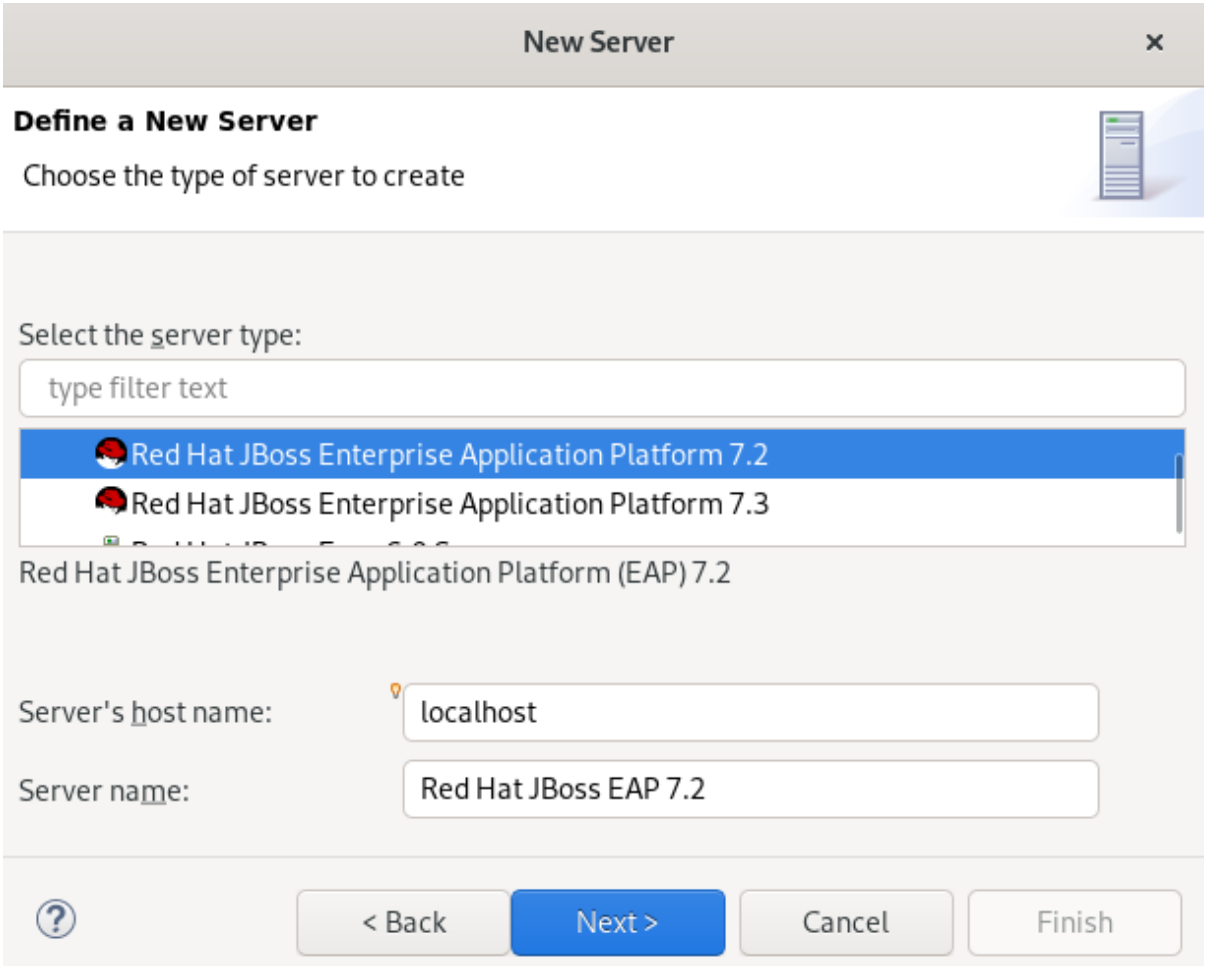
CodeReady Studio でリモートサーバーを設定する方法を説明します。

手順

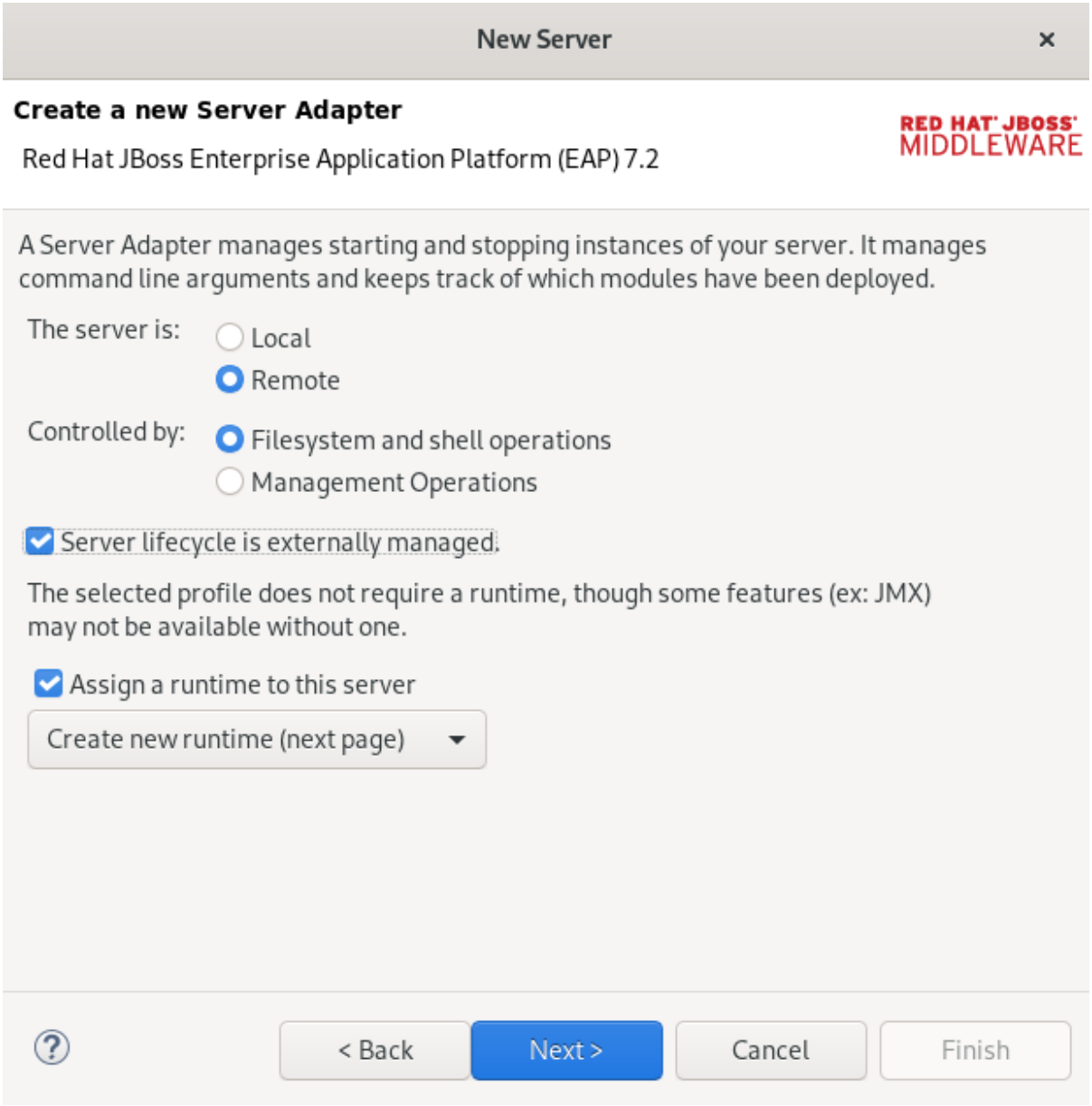
1. CodeReady Studio を起動します。
2. **Ctrl**+**N** キーを押します。
   Select a wizard ウインドウが表示されます。



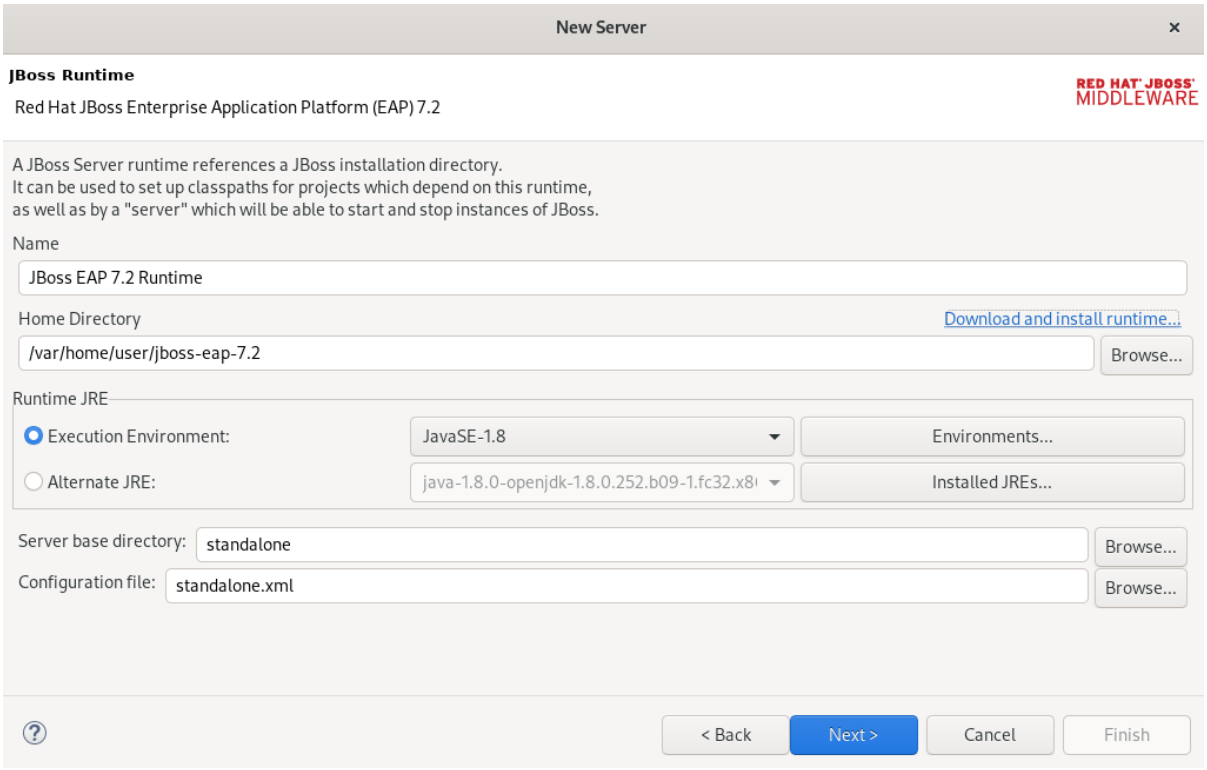
3. 検索フィールドに **Server** と入力します。
4. **Server** を選択します。
5. **Next** ボタンをクリックします。
Define a New Server ウィンドウが表示されます。



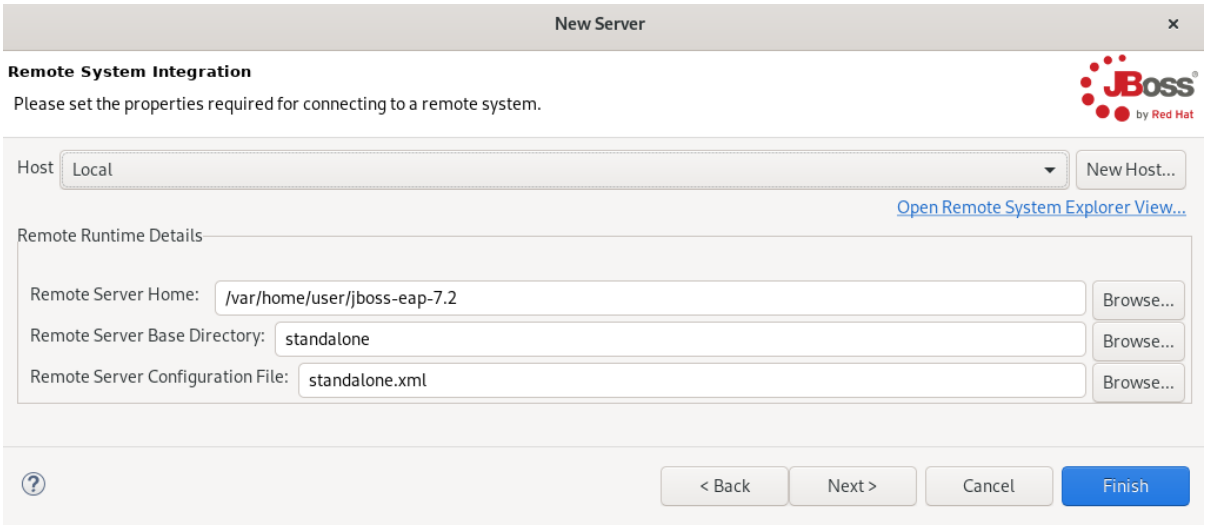
6. 適切なサーバータイプを選択します。
7. **Next** をクリックします。
   Create a new Server Adapterウィンドウが表示されます。



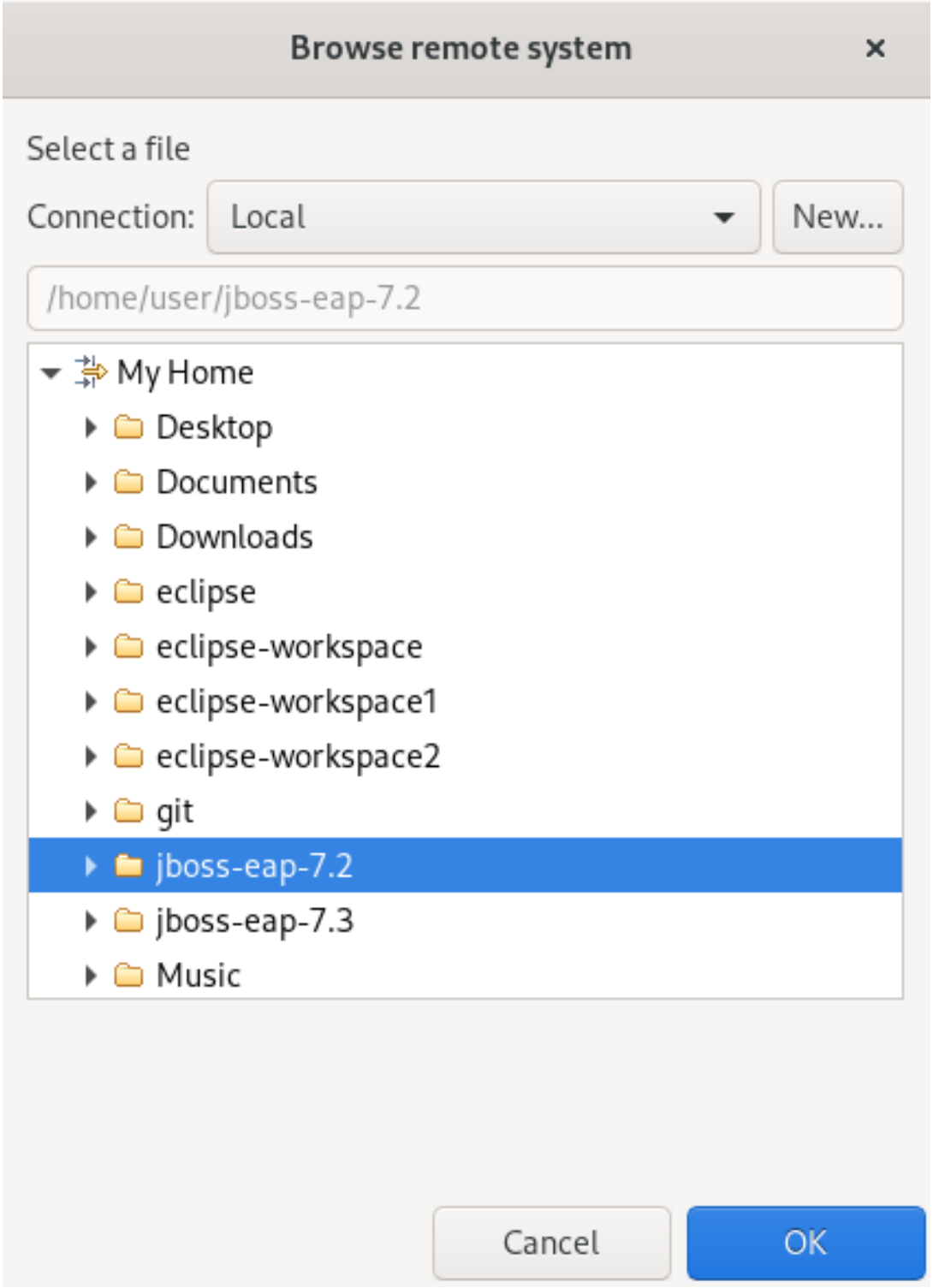
8. **Remote** チェックボックスを選択します。
9. 適切な **Controlled by** オプションを選択します。
10. **Server lifecycle external managed** チェックボックスを選択します。
11. **Assign a runtime to server**チェックボックスを選択します。
12. **Next** をクリックします。
    JBoss Runtime ウィンドウが表示されます。



13. **Home Directory** フィールドの **Browse** をクリックして、ランタイムサーバーを見つけます。
14. **Next** をクリックします。
Remote System Integration ウィンドウが表示されます。



15. **Remote Server Home** フィールドで **Browse** をクリックします。
Browse remote system ウインドウが表示されます。



16. リモートサーバーが含まれるディレクトリーへのパスを指定します。

17. **Finish** をクリックします。

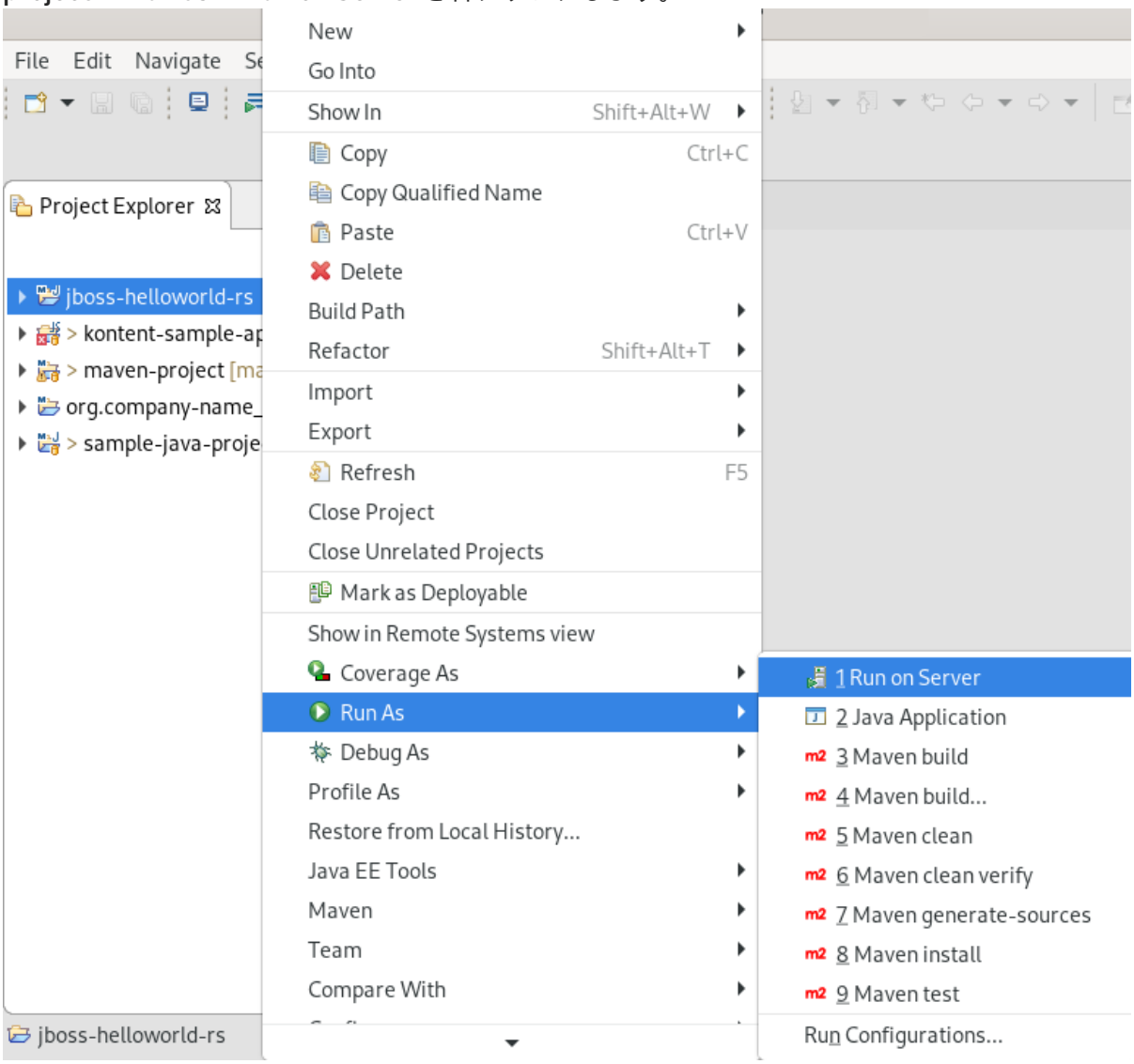
3.3. アプリケーションのデプロイ

ローカルサーバーの設定後に、サーバーアダプターを使用して IDE からサーバーにアプリケーションをデプロイできます。サーバーアダプターは、アプリケーションとサーバー管理を簡単にデプロイできるように、サーバーと IDE 間のランタイムの通信を可能にします。

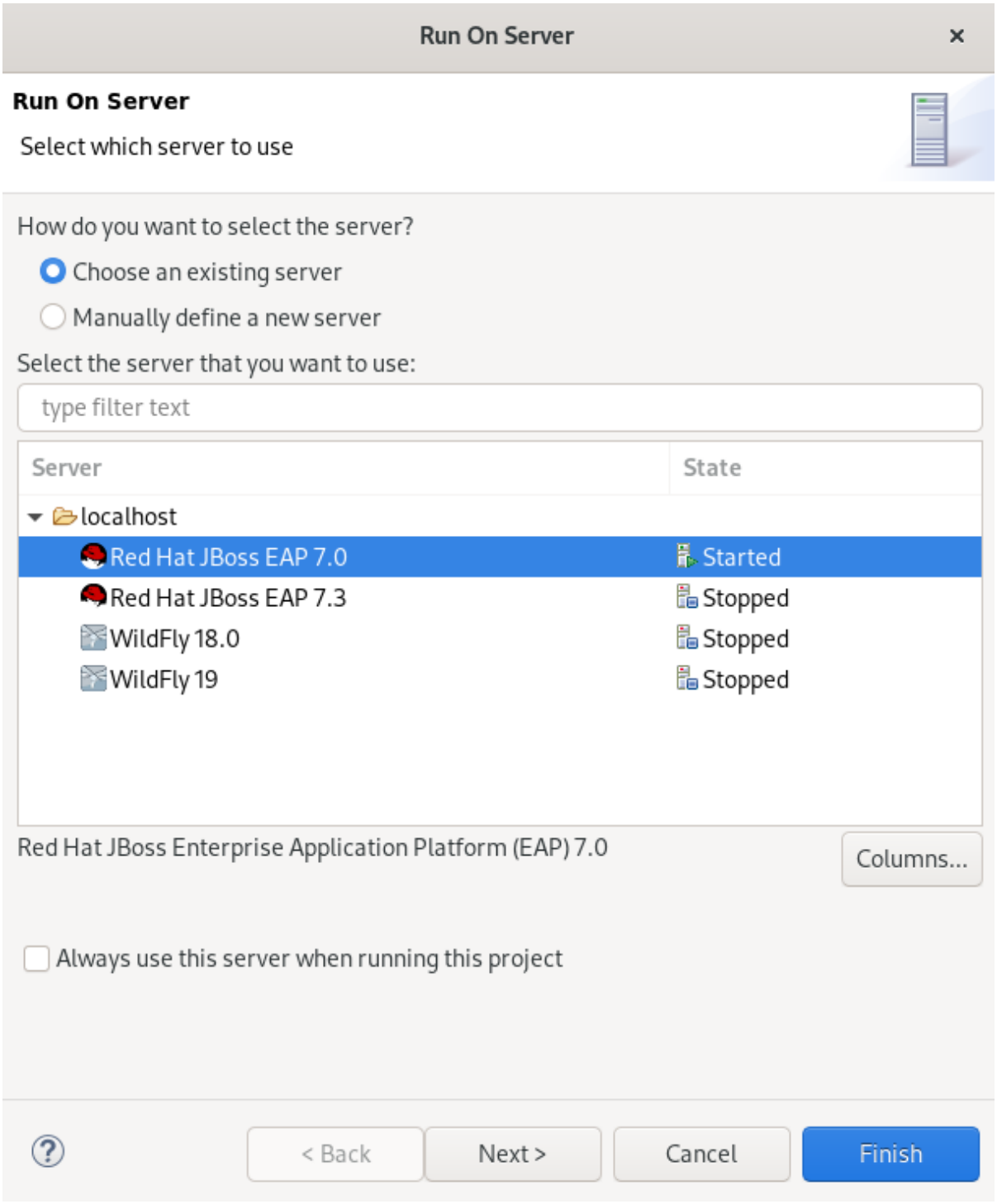
CodeReady Studio のサーバーにアプリケーションをデプロイする方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **project → Run as → Run on Server** を右クリックします。



Run on Server ウインドウが表示されます。



3. **Choose an existing server**チェックボックスを選択します。
4. デプロイメントのサーバーを選択します。
5. **Finish** をクリックします。

内部の CodeReady Studio Web ブラウザーでアプリケーションが開かれます。

第4章 CODEREADY STUDIO の JBOSS EAP および JBOSS WFK の基本

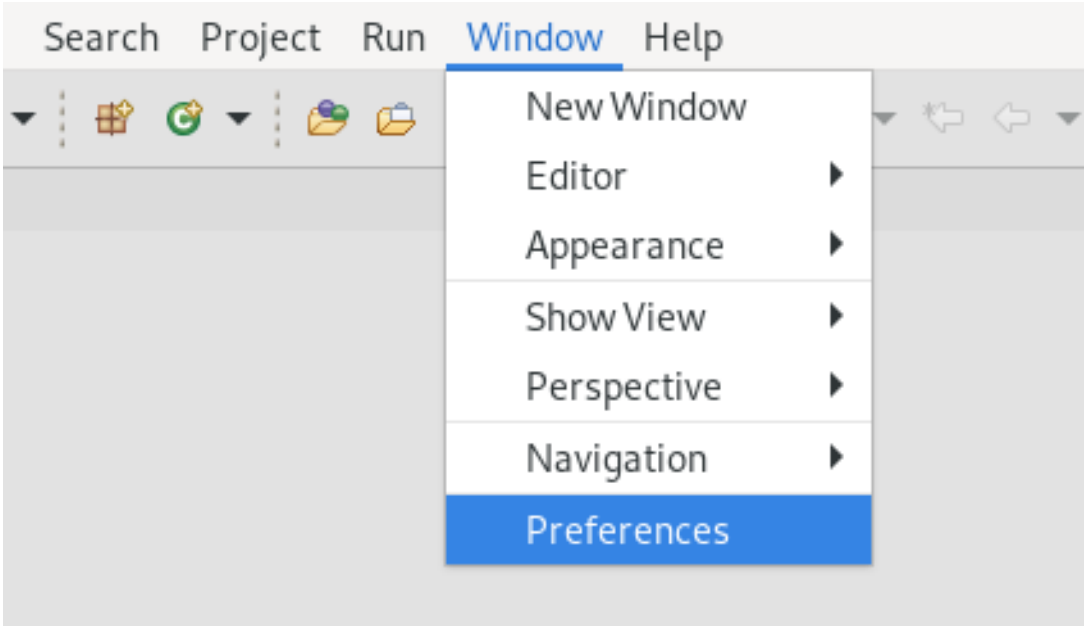
Eclipse IDE は、Red Hat JBoss Enterprise Application Platform (JBoss EAP) および Red Hat JBoss Web Framework Kit (JBoss WFK) でのアプリケーション開発およびデプロイメントをサポートします。ただし、最初に Maven リポジトリーを設定する必要があります。この設定は、Red Hat Central で提供される、エンタープライズバージョンの Maven プロジェクトサンプルの使用に不可欠です。これらのプロジェクトは JBoss EAP へのデプロイメントを目的としており、JBoss EAP リポジトリーおよび JBoss WFK リポジトリーへの IDE のアクセスが必要になります。

4.1. MAVEN リポジトリーの設定

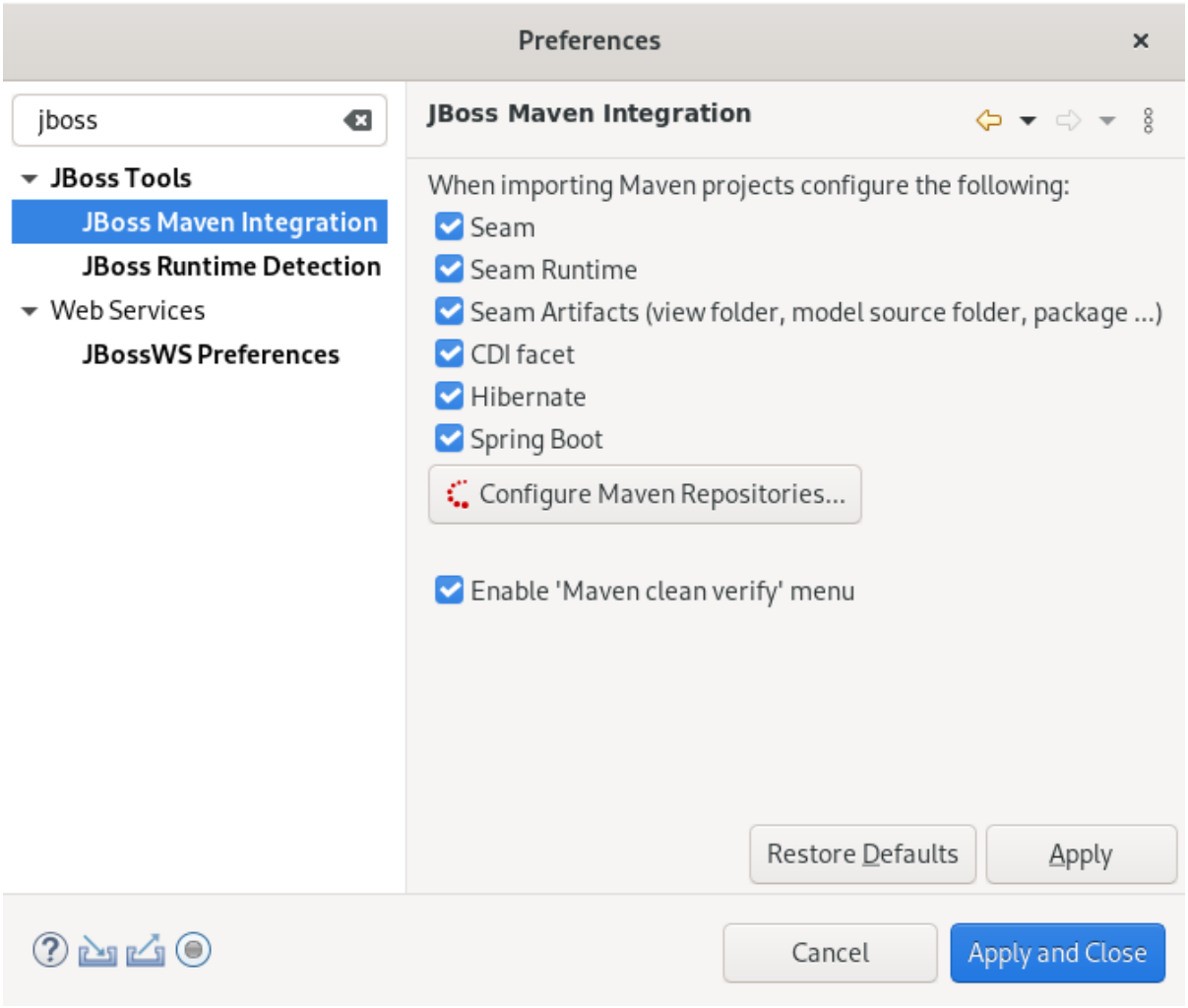
Maven リポジトリーを設定する方法を説明します。

手順

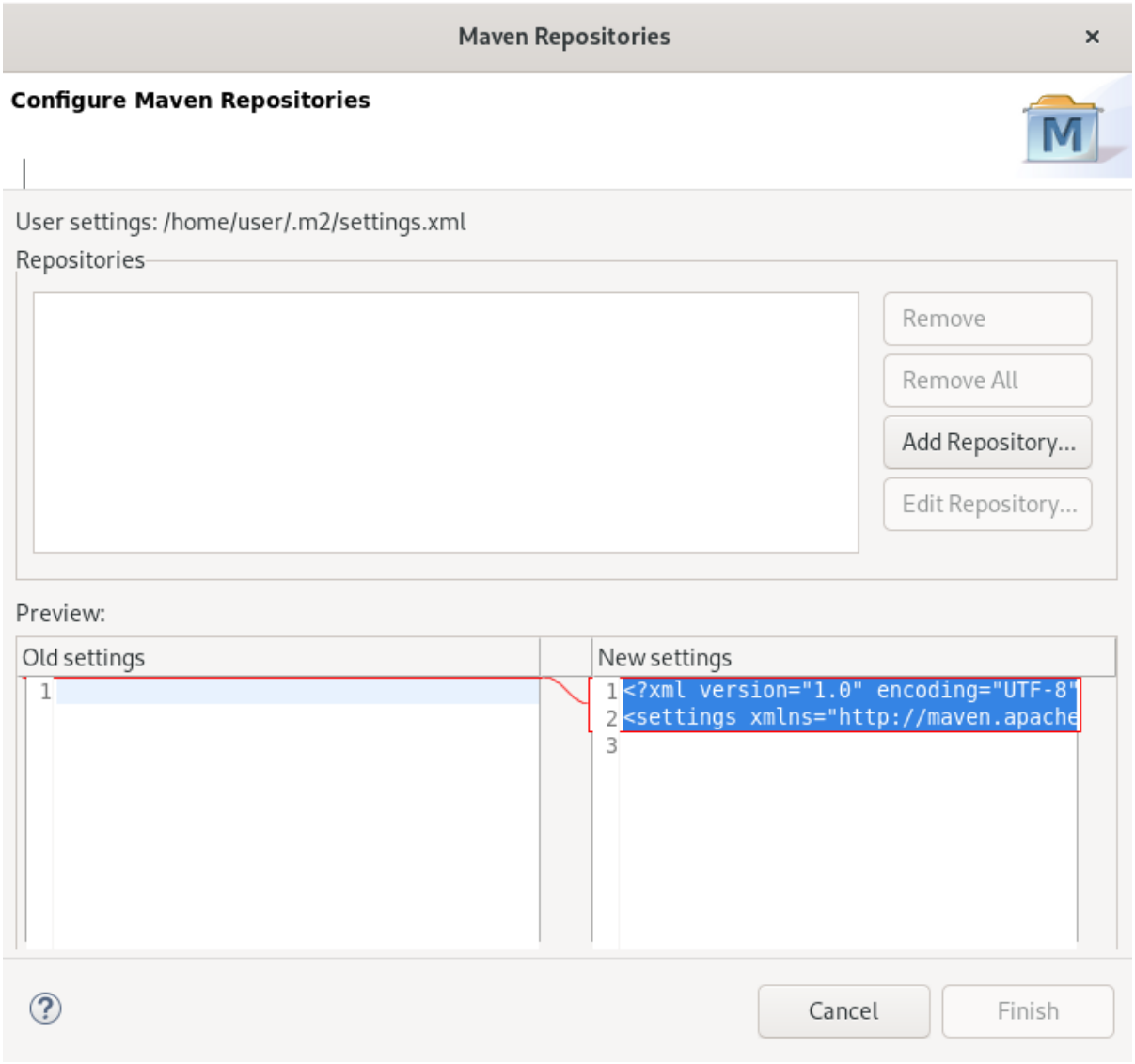
1. CodeReady Studio を起動します。
2. **Window → Preferences** をクリックします。



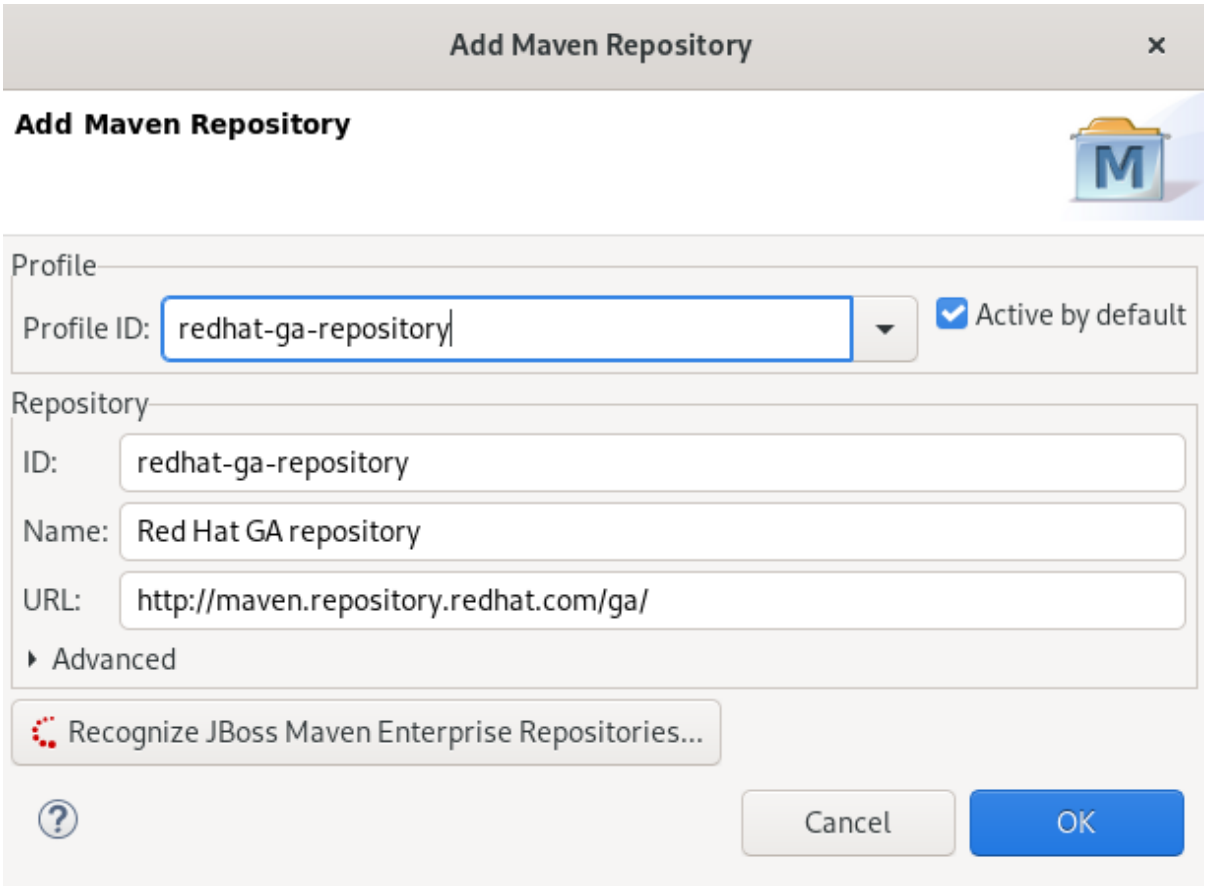
Preferences ウィンドウが表示されます。



3. 検索フィールドに **JBoss** と入力します。
4. **JBoss Maven Integration** を選択します。
5. **Configure Maven Repositories** をクリックします。
   Configure Maven Repositories ウインドウが表示されます。



6. **Add Repository** をクリックします。
   Add Maven Repository ウインドウが表示されます。



7. **Profile ID** フィールドの下向き矢印をクリックします。
8. **redhat-ga-repository** を選択します。
   他のフィールドは自動的に入力されます。
9. **OK** をクリックします。
10. **Finish** をクリックします。
    Confirm File Update ウインドウが表示されます。
11. **Yes** をクリックします。
12. **Apply and Close** をクリックします。

その他のリソース

- Maven リポジトリーの詳細は、「Maven: Getting Started - Developers」を参照してください。

4.2. JBOSS EAP の設定

Eclipse IDE で JBoss EAP を設定するには、IDE がローカルまたはリモートのランタイムサーバーを示すようにする必要があります。これにより、IDE と JBoss EAP サーバー間の通信チャネルが確立され、デプロイメントおよびサーバー管理のワークフローが効率的になります。

CodeReady Studio で JBoss EAP をインストールする方法を説明します。

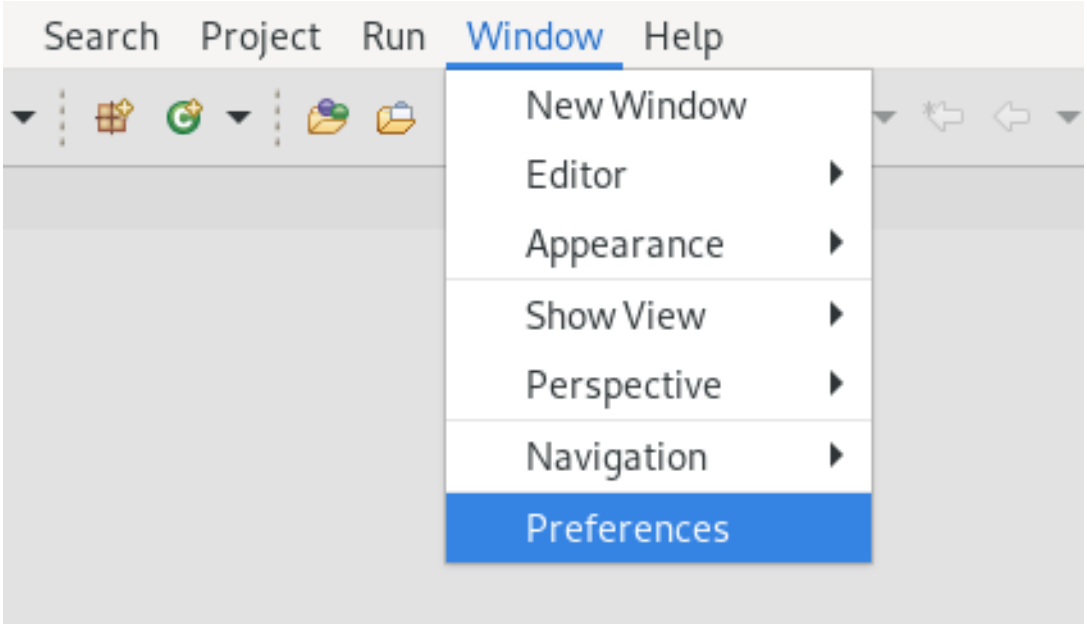
前提条件

- 設定済みの Maven リポジトリー。

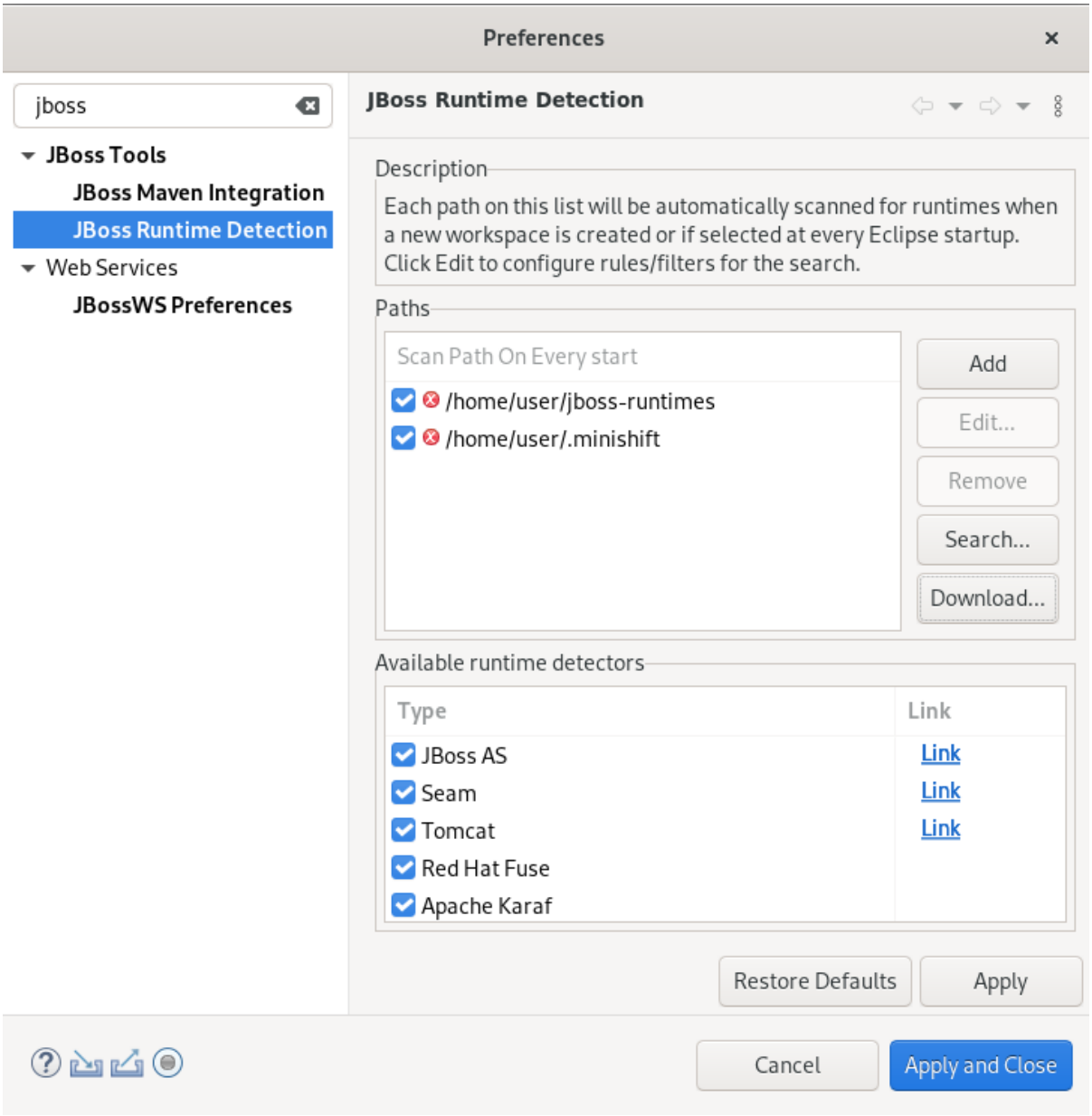
Maven リポジトリーの設定方法に関する詳細は、「Maven リポジトリーの設定」 を参照してください。

手順

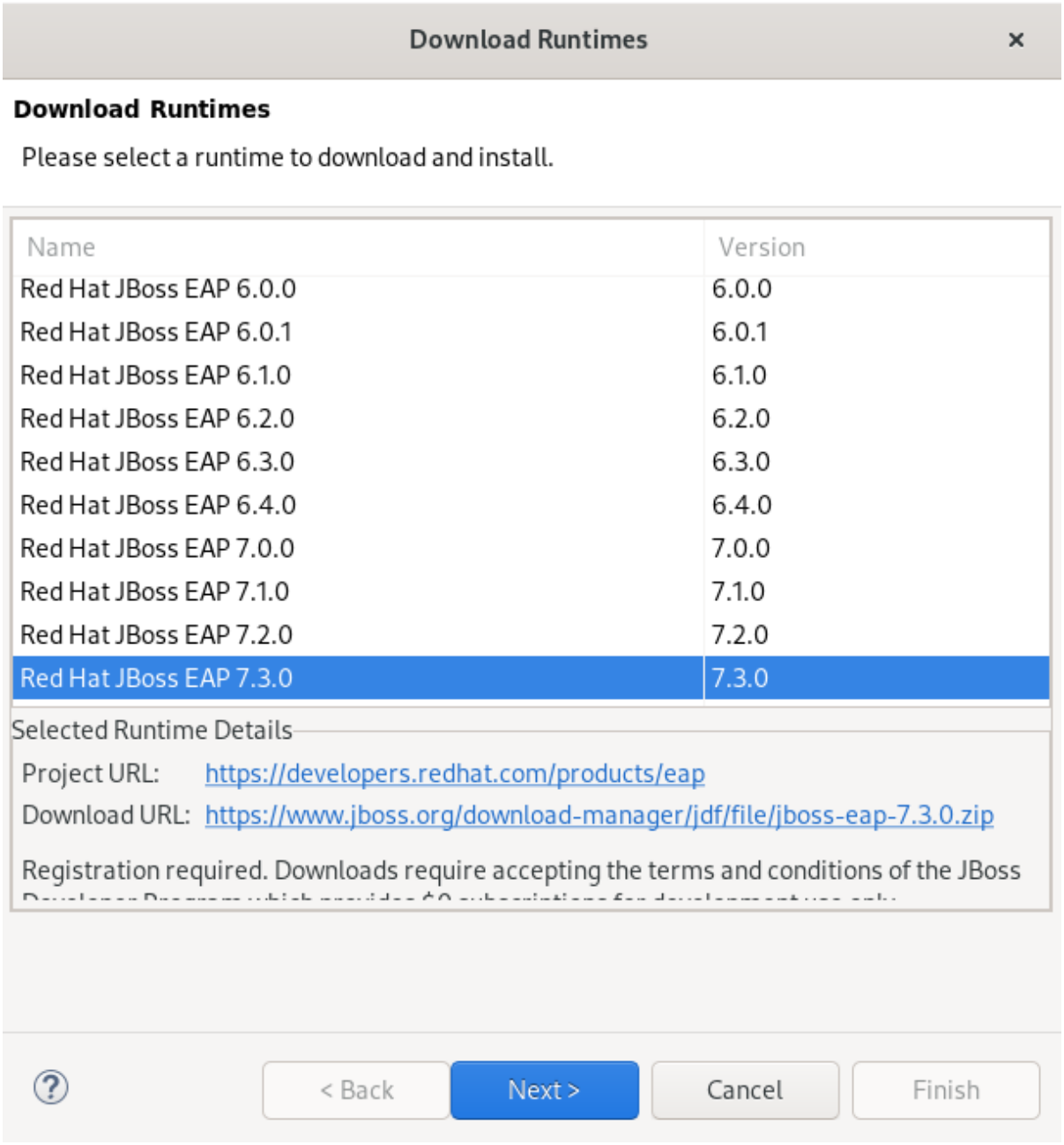
1. CodeReady Studio を起動します。
2. **Window → Preferences** をクリックします。



Preferences ウィンドウが表示されます。



3. 検索フィールドに **JBoss** と入力します。
4. **JBoss Runtime Detection** を選択します。
5. **Download** をクリックします。
   Download Runtimes ウィンドウが表示されます。

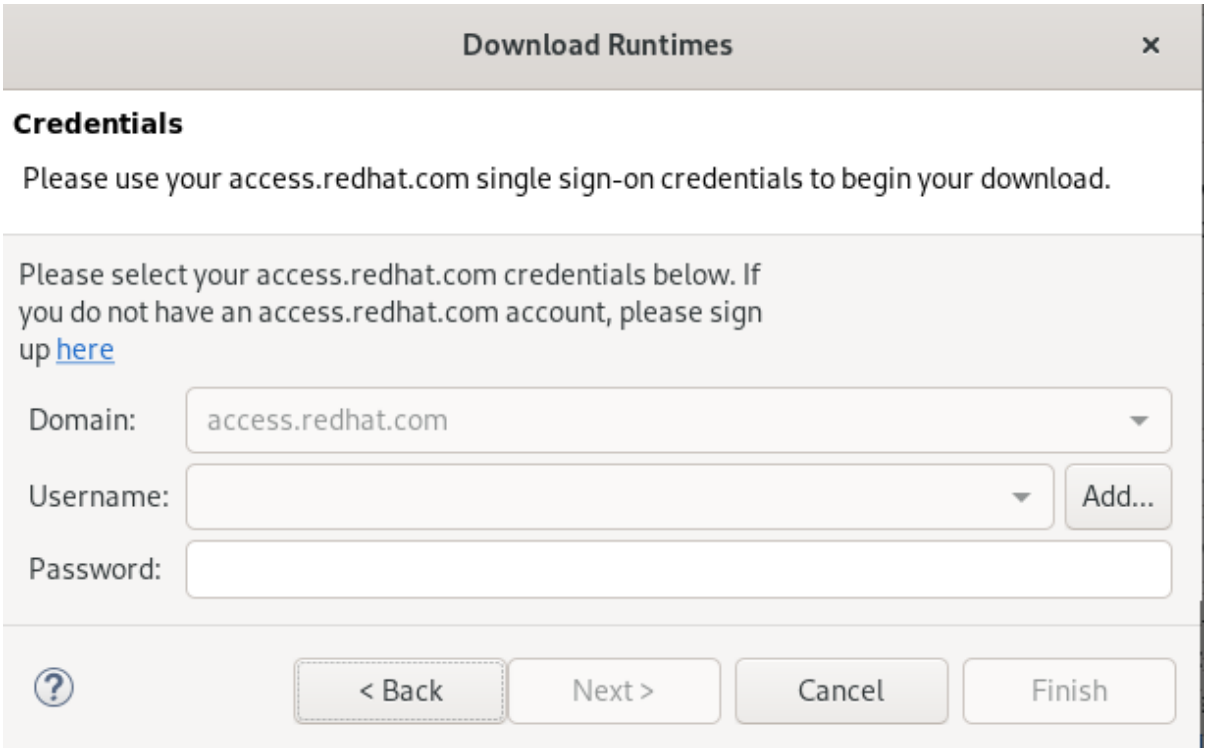


6. 適切な JBoss EAP のバージョンを選択します。

注記

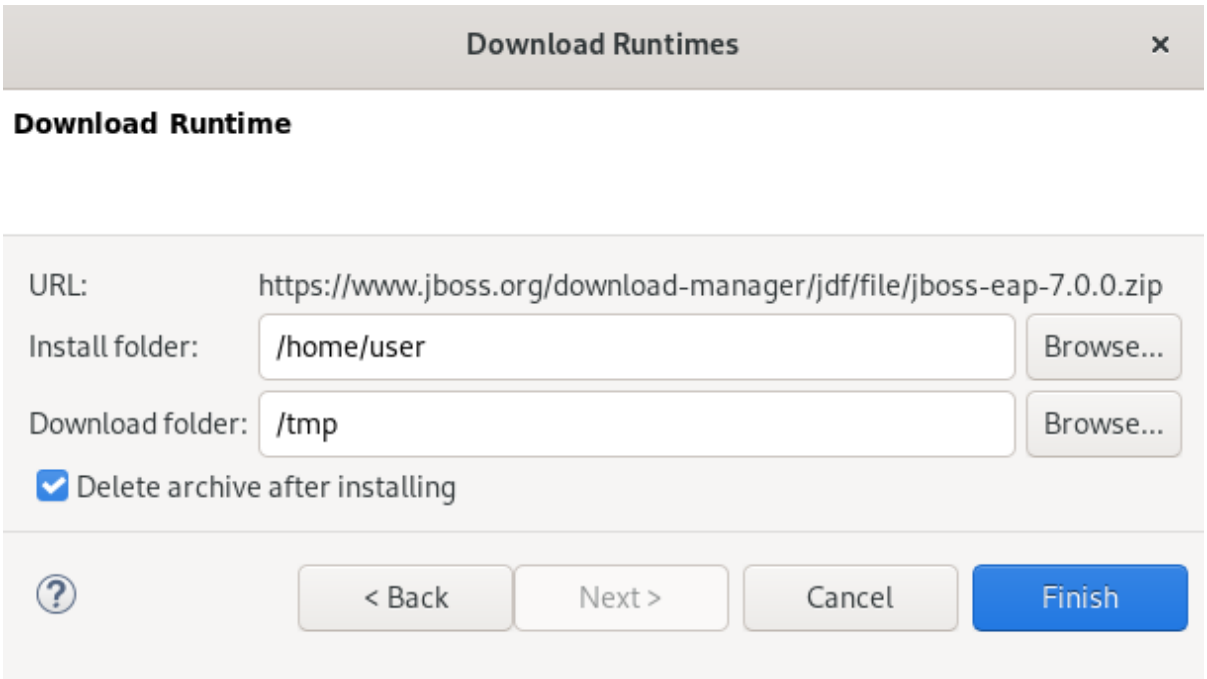
JBoss EAP バージョン 6.0.x 以前を選択した場合は、画面の指示に従います。
6.0.x よりも新しいバージョンを選択した場合は、以下の手順に従います。

7. **Next** をクリックします。
Credentials ウインドウが表示されます。



8. **Add** をクリックします。
9. **access.redhat.com** のユーザー名とパスワードを入力します。
10. **OK** をクリックします。
11. **Next** をクリックします。
ライセンス契約の内容を確認します。ライセンス契約に同意し、**Next** をクリックしてインストールを続行します。

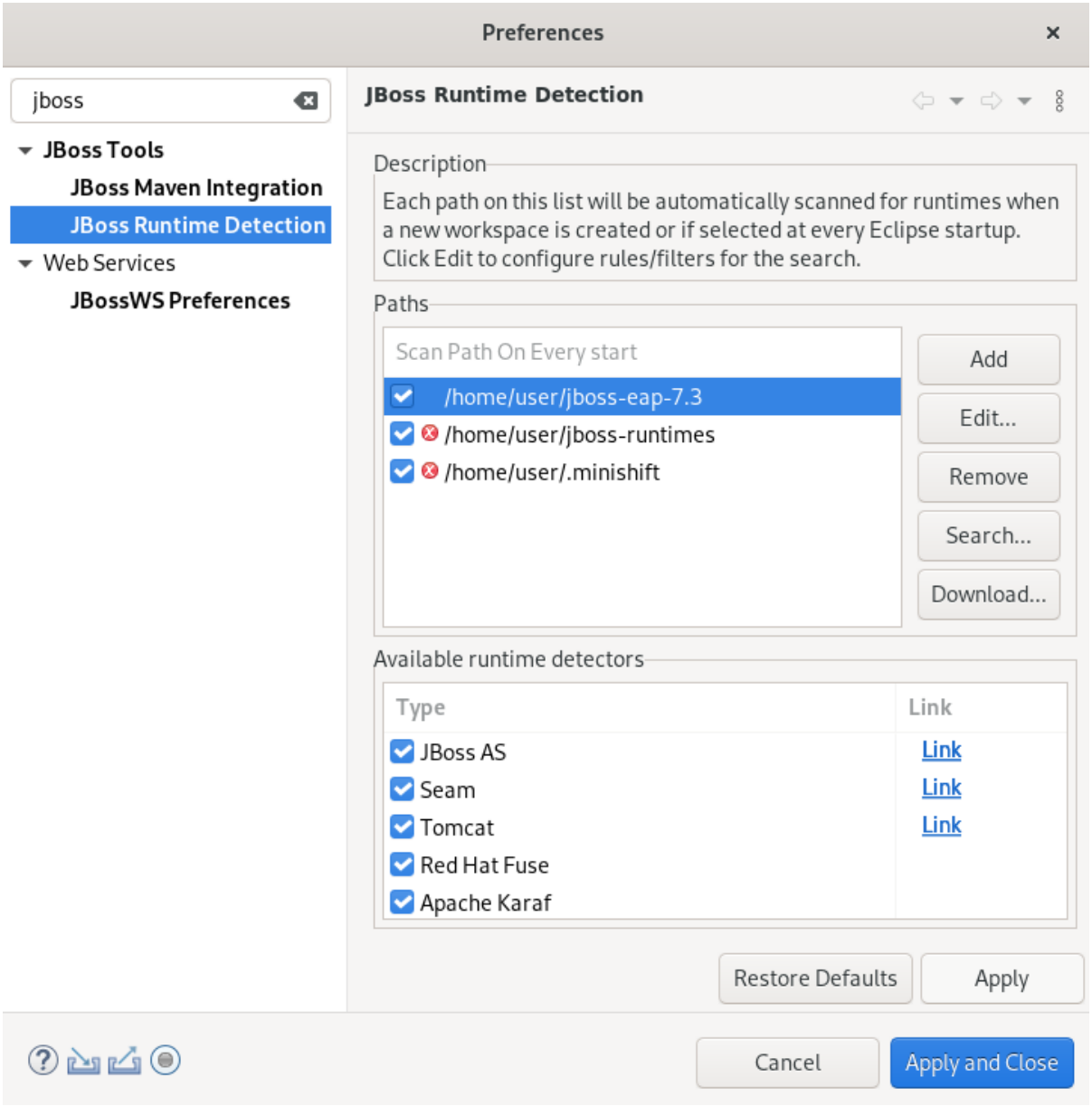
    Download Runtimes ウィンドウが表示されます。



12. **Browse** をクリックして、**Install folder** を選択します。
13. **Browse** をクリックして、**Download folder** を選択します。

14. **Finish** をクリックします。
Runtime のダウンロードとインストールには時間がかかる場合があるため注意してください。

JBoss Runtime Detection ウインドウが表示されます。

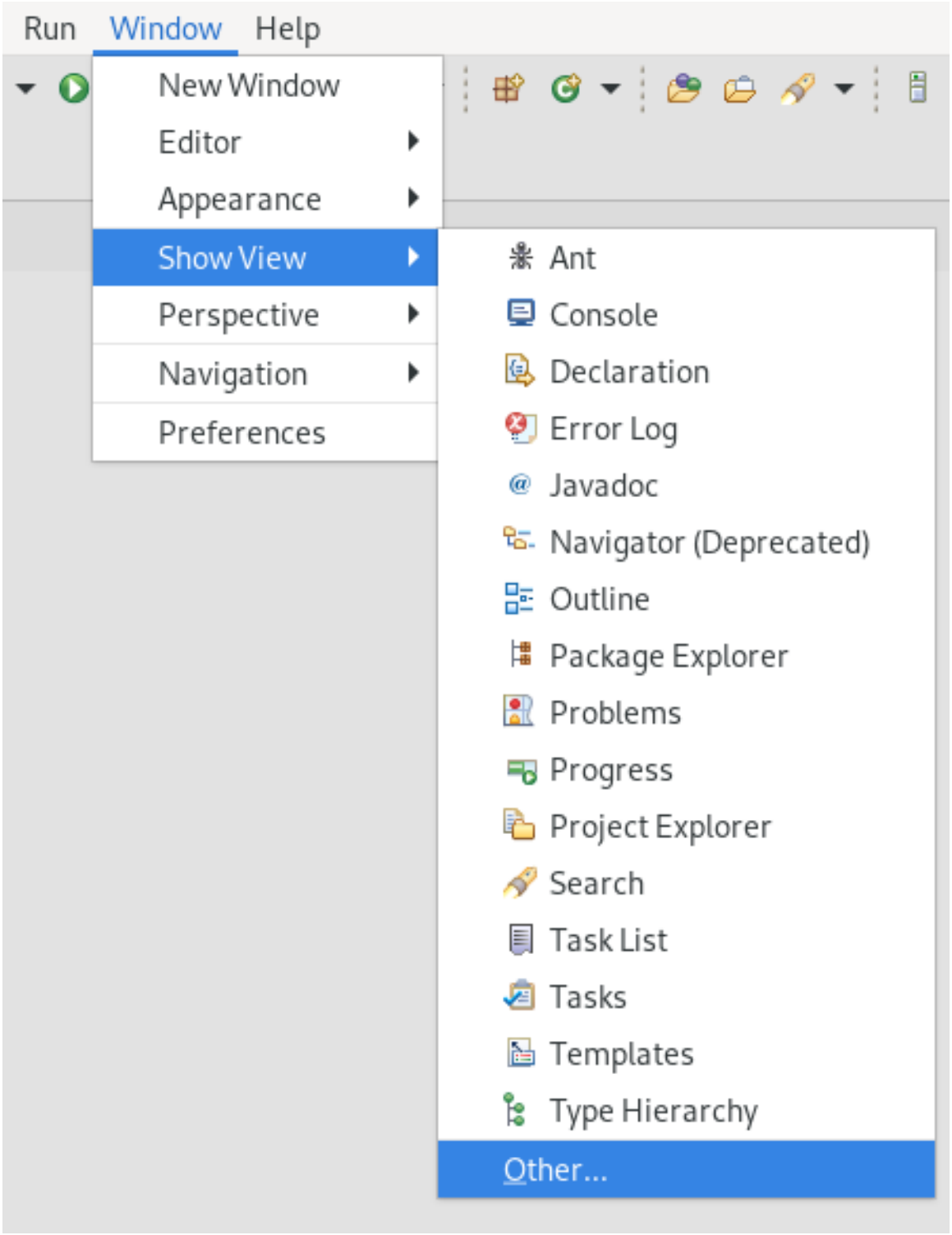


15. JBoss EAP インストールファイルへのパスを選択します。

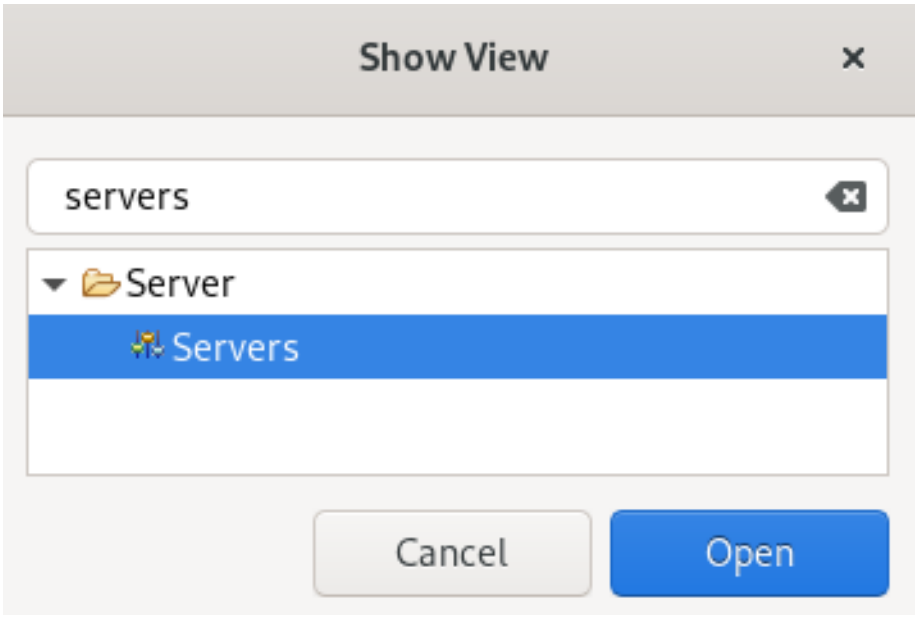
16. **Apply and Close** をクリックします。

検証手順

1. **Window → Show View → Other** とクリックします。

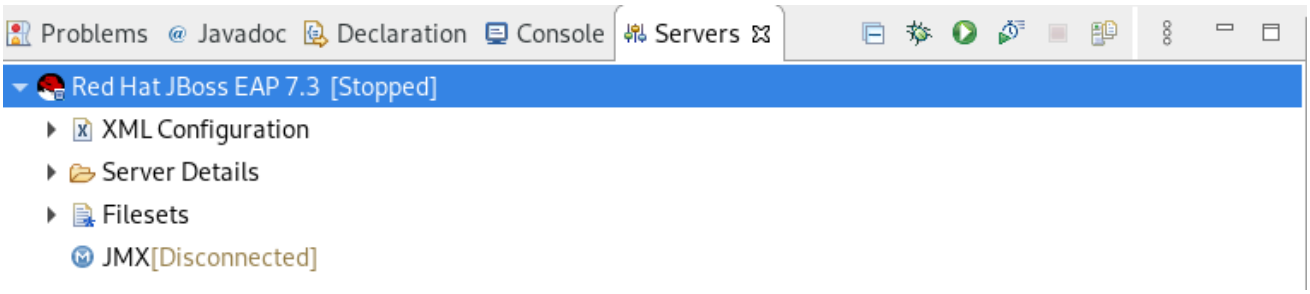


Show View ウインドウが表示されます。



2. 検索フィールドに **Servers** と入力します。
3. **Servers** を選択します。
4. **Open** をクリックします。
   Servers ビューが表示されます。

新たに追加された JBoss EAP が **Servers** ビューに表示されます。



第5章 CODEREADY STUDIO での OPENSHIFT の基本

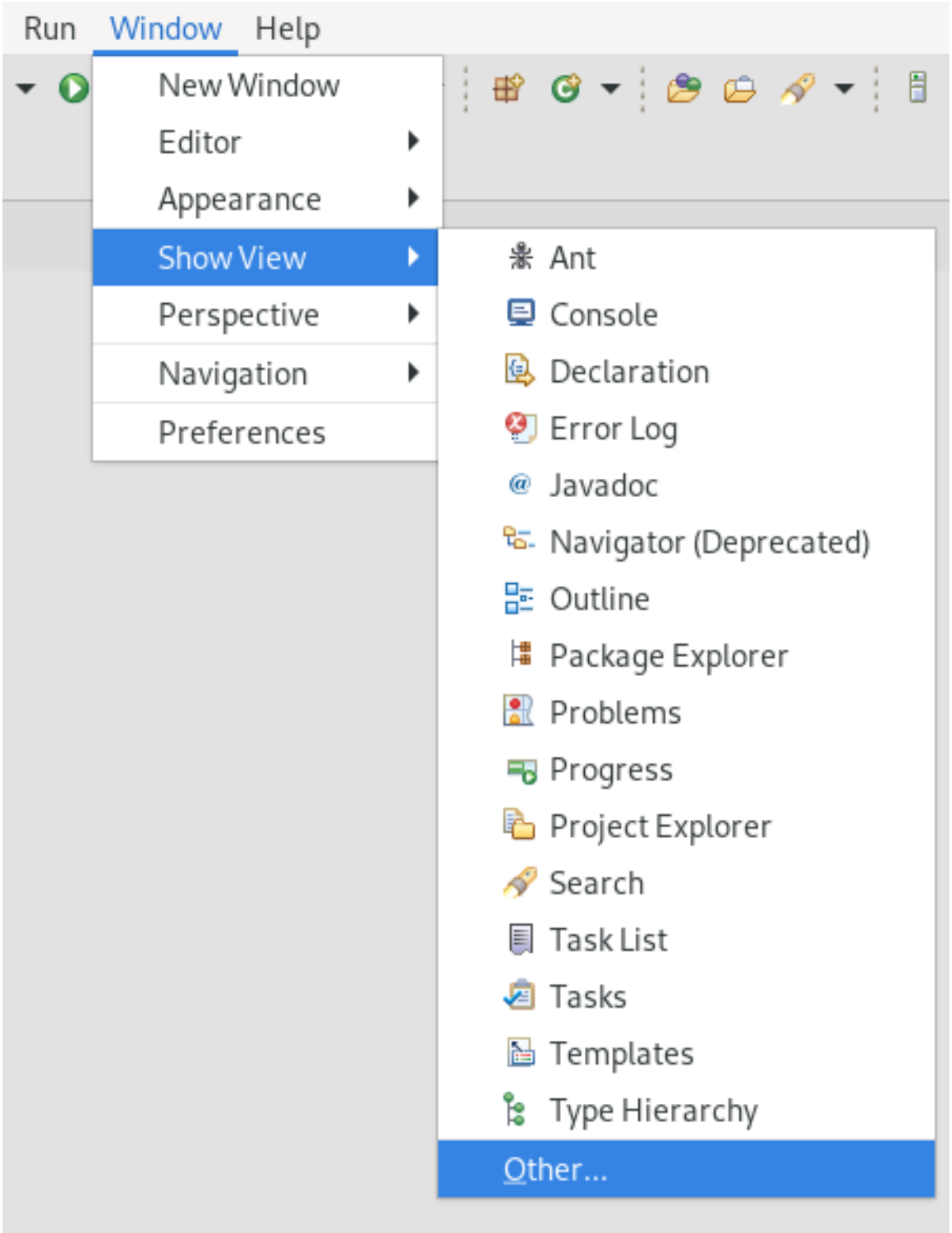
CodeReady Studio には OpenShift Application Explorer ビューが含まれています。これにより、ユーザーエクスペリエンスが簡素化され、内部ループで簡単および迅速にフィードバックすることができ、デバックも容易および迅速に行うことができます。

5.1. OPENSHIFT APPLICATION EXPLORER ビューの設定

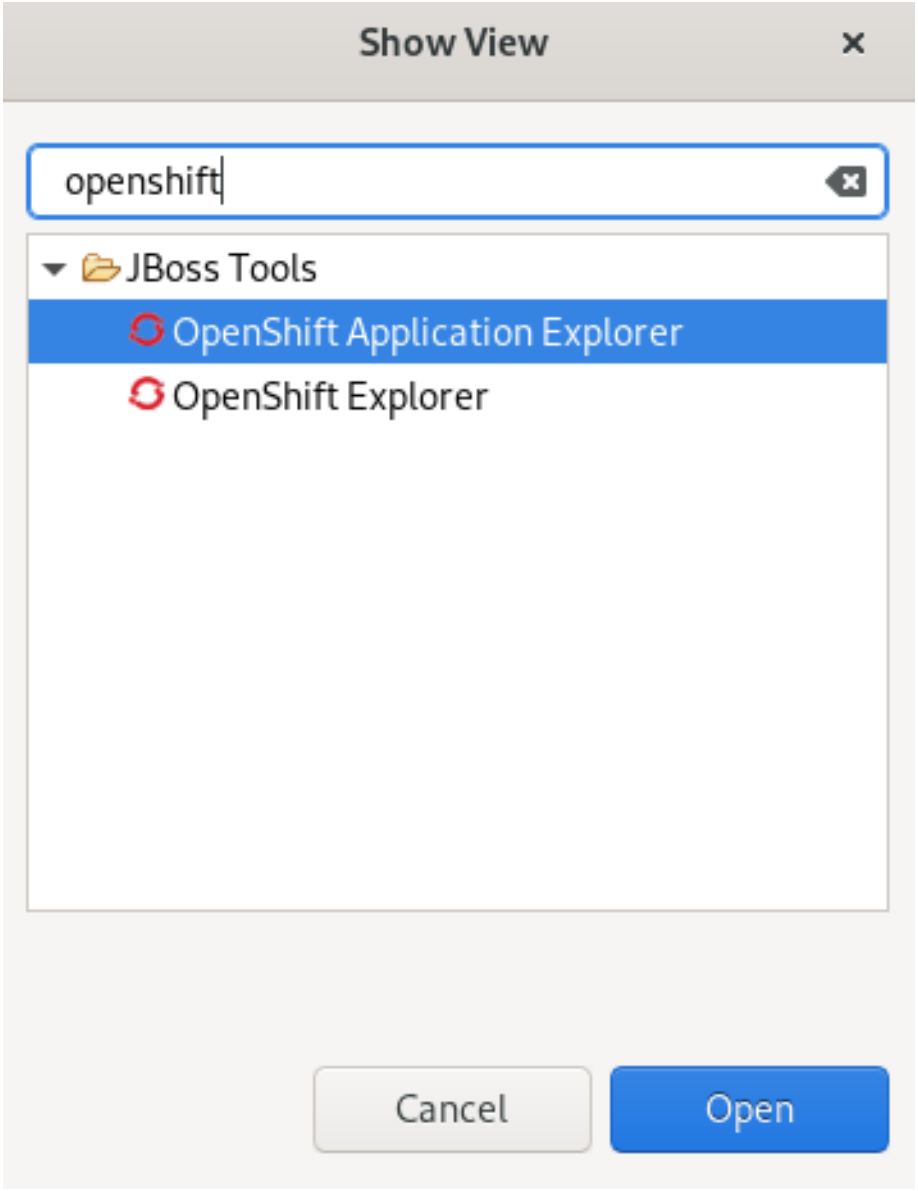
CodeReady Studio で OpenShift Application Explorer を開く方法を説明します。

手順

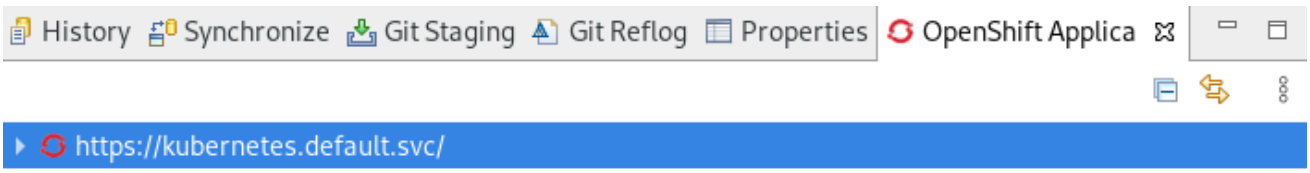
1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。



Show View ウインドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Application Explorer** を選択します。
5. **Open** をクリックします。
   OpenShift Application Explorer ビューが表示されます。

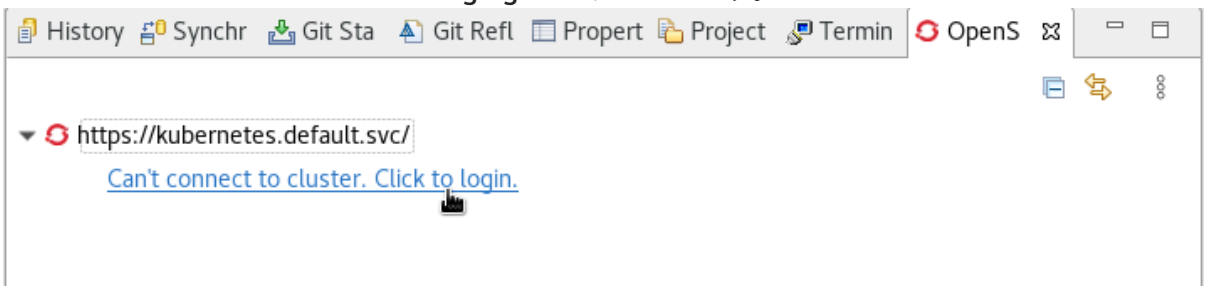


5.2. OPENSHIFT APPLICATION EXPLORER を使用した OPENSHIFT クラスターへの接続

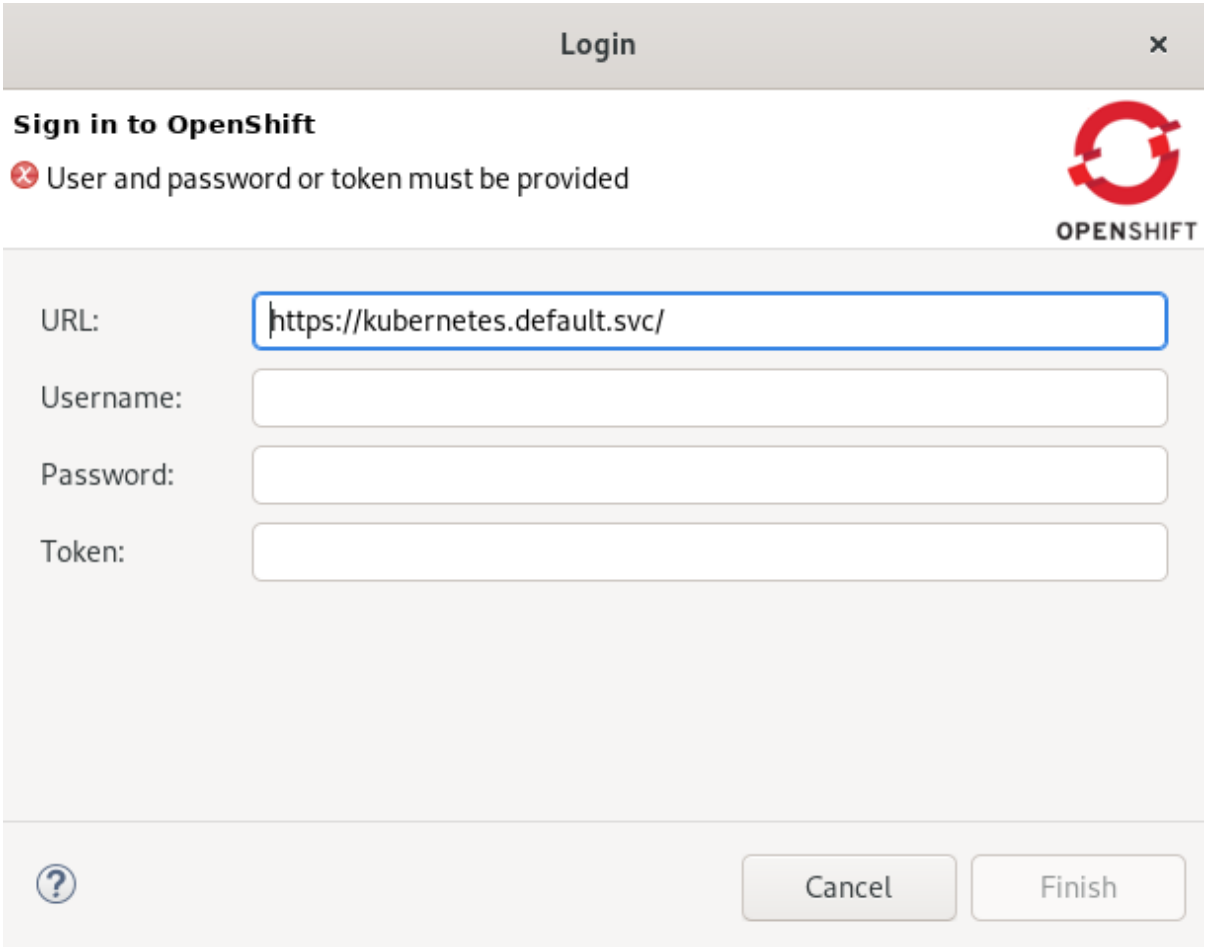
OpenShift Application Explorer を使用して CodeReady Studio で OpenShift クラスターにログインする方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **OpenShift Application Explorer** を開きます。
3. **Can't connect to cluster. Click to loging.**をクリックします。



Login ウインドウが表示されます。



4. ログインのクレデンシャルを入力します。
5. **Finish** ボタンをクリックします。

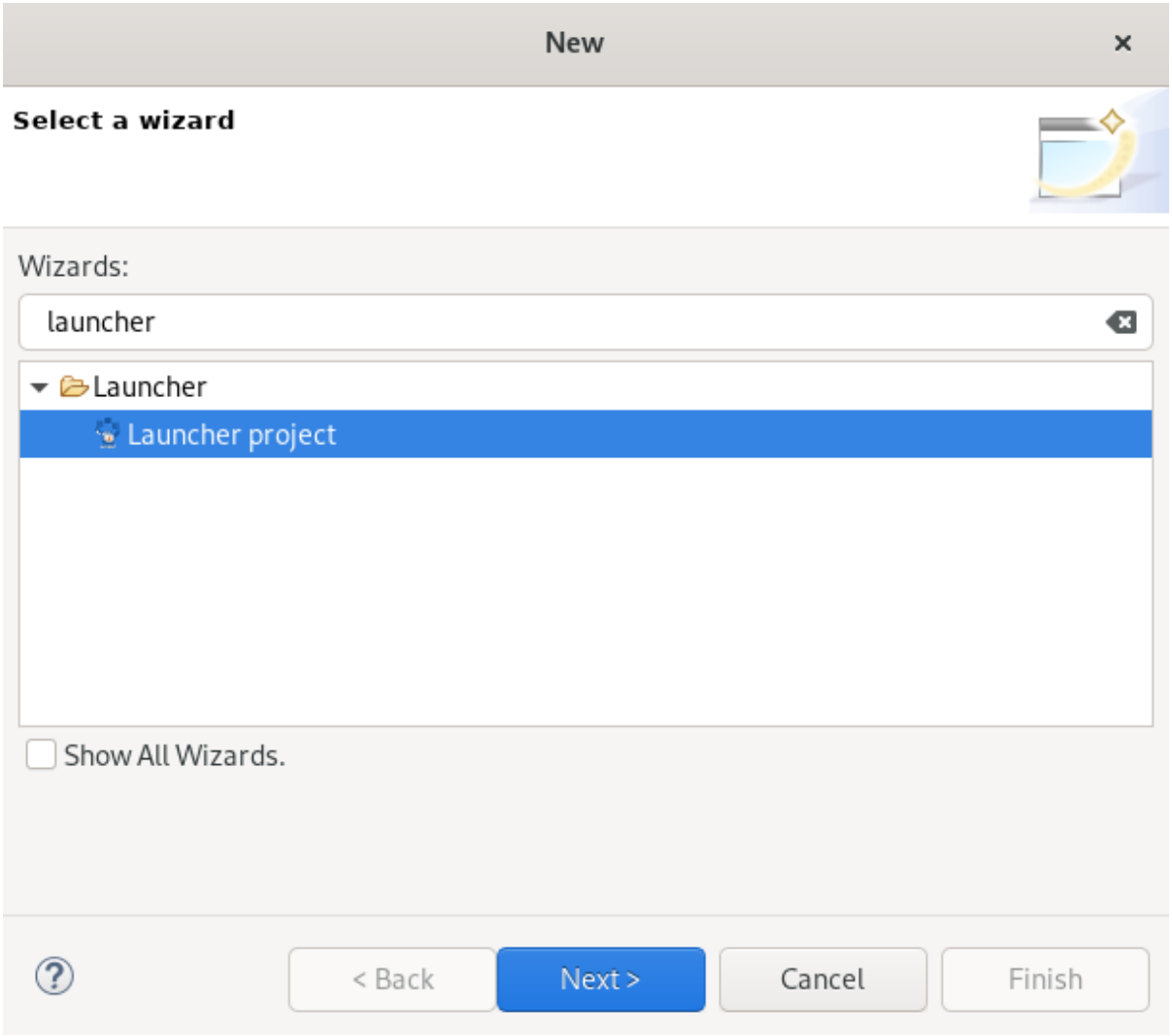
プロジェクトが **OpenShift Application Explorer** ビューに表示されます。

5.3. 新しいランチャープロジェクトの作成

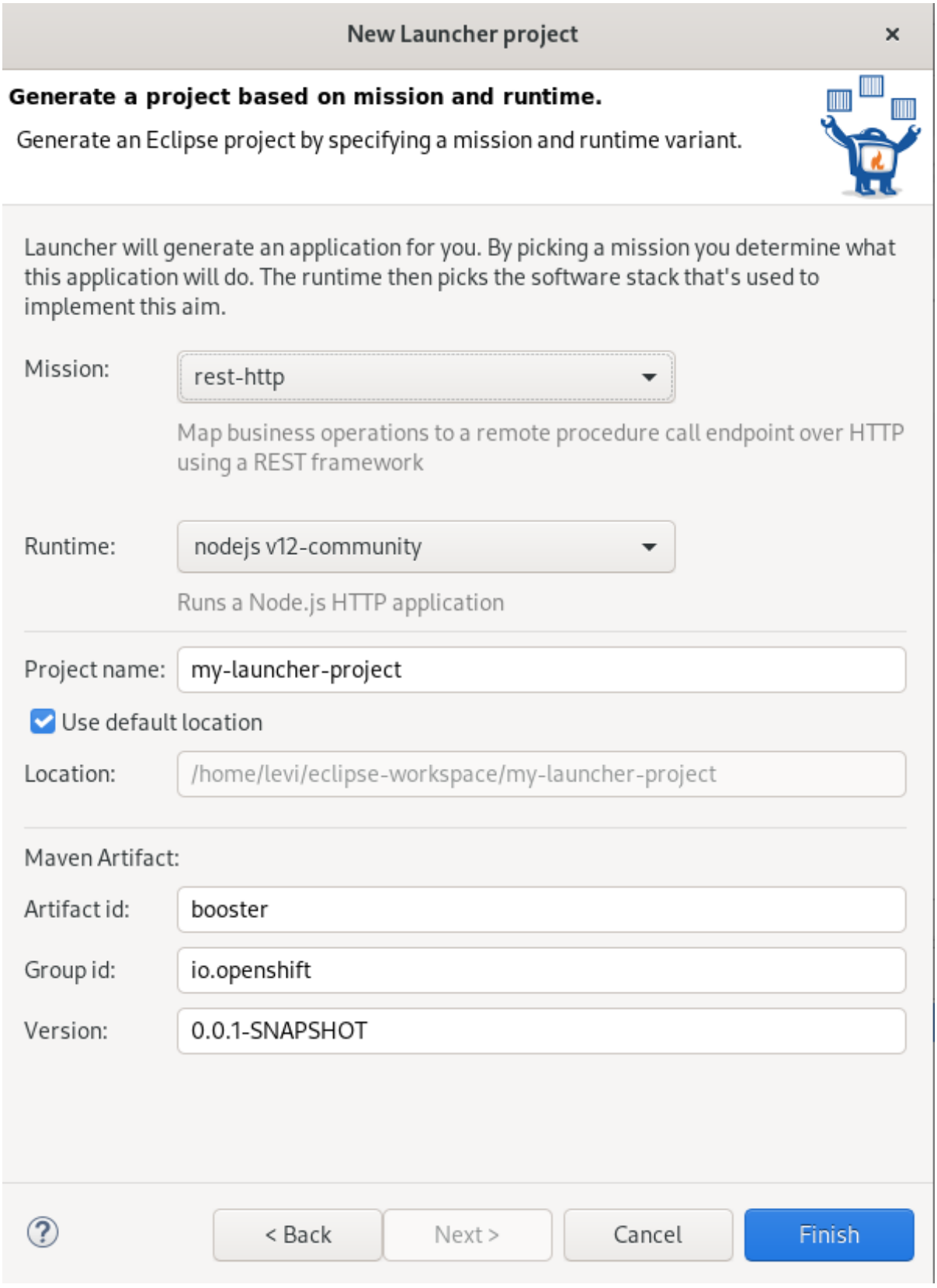
CodeReady Studio で新しいランチャープロジェクトを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Ctrl**+**N** キーを押します。
   Select a wizard ウインドウが表示されます。



3. 検索フィールドに **Launcher** と入力します。
4. **Launcher project** を選択します。
5. **Next** をクリックします。
   New Launcher project ウインドウが表示されます。



6. 希望する **Mission** を設定します。
7. 希望する **Runtime** を設定します。
8. プロジェクトに名前を付けます。
9. プロジェクトの場所を選択します。
10. **Finish** ボタンをクリックします。

依存関係の解決プロセスが完了するまで時間がかかることがあるため注意してください。

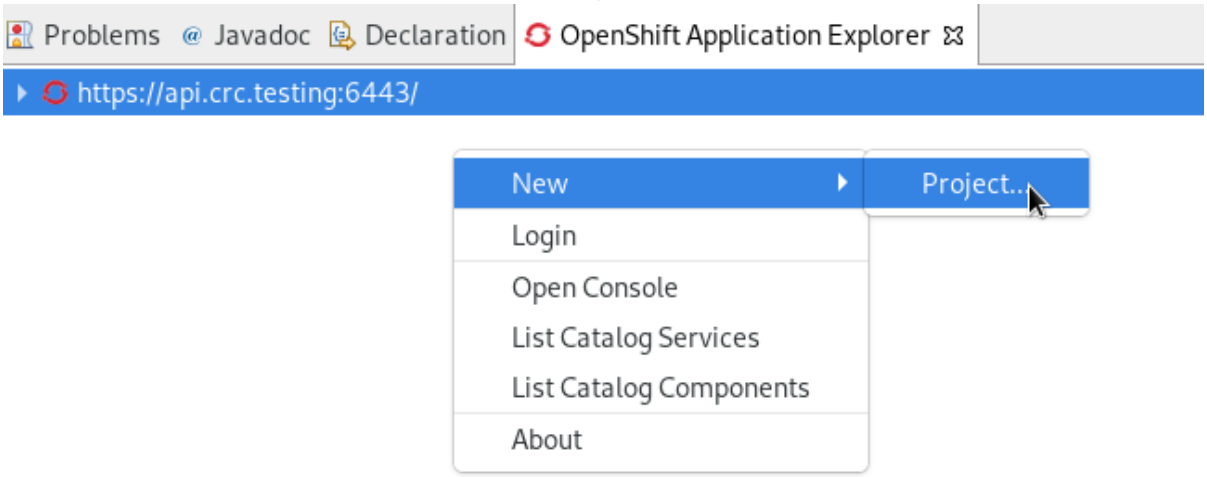
新たに作成されたランチャープロジェクトが **Project Explorer** ビューに表示されます。

5.4. OPENSHIFT APPLICATION EXPLORER を使用した新規プロジェクトの作成

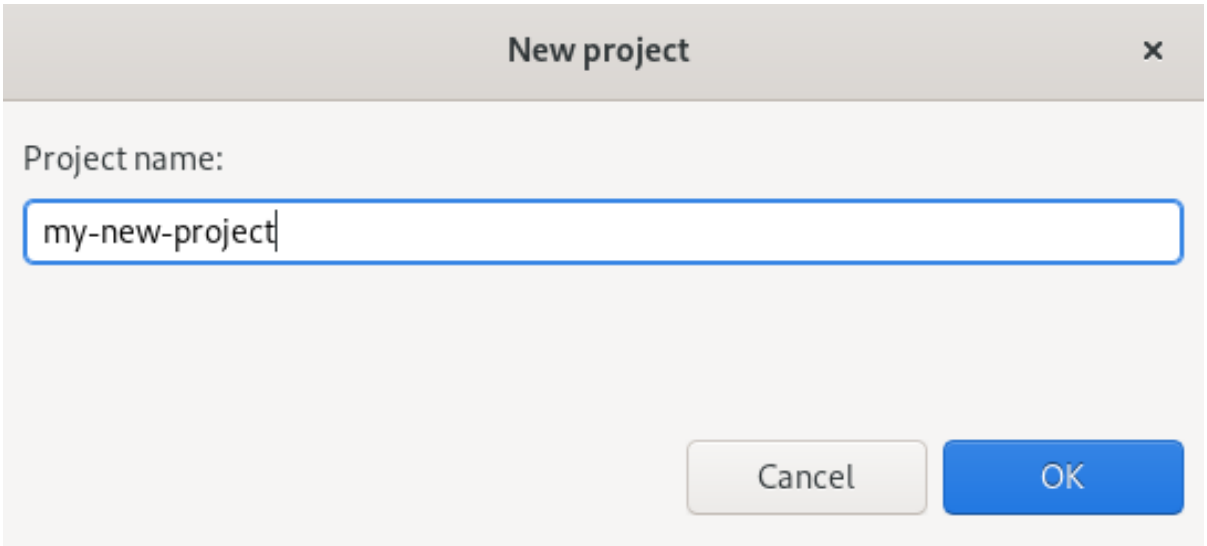
CodeReady Studio で OpenShift Application Explorer を使用して新しいプロジェクトを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **OpenShift Application Explorer** を起動します。
3. **OpenShift Application Explorer → New → Project** の任意の場所をクリックします。



New project ウインドウが表示されます。



4. プロジェクトに名前を付けます。
5. **OK** をクリックします。

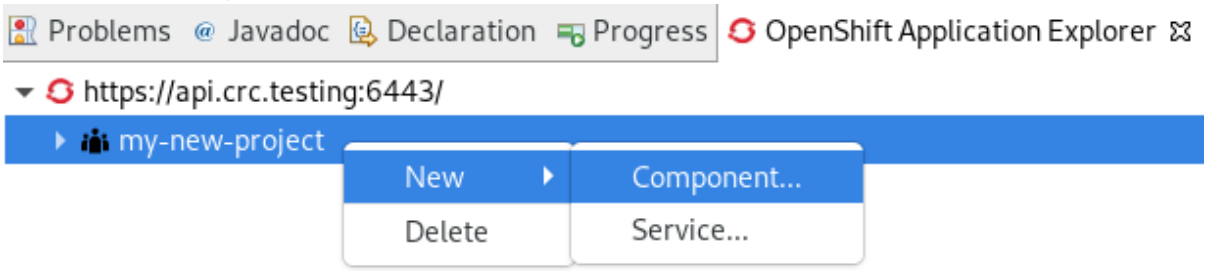
新たに作成したプロジェクトが **OpenShift Application Explorer** ビューに表示されます。

5.5. OPENSHIFT APPLICATION EXPLORER を使用した新規コンポーネントの作成

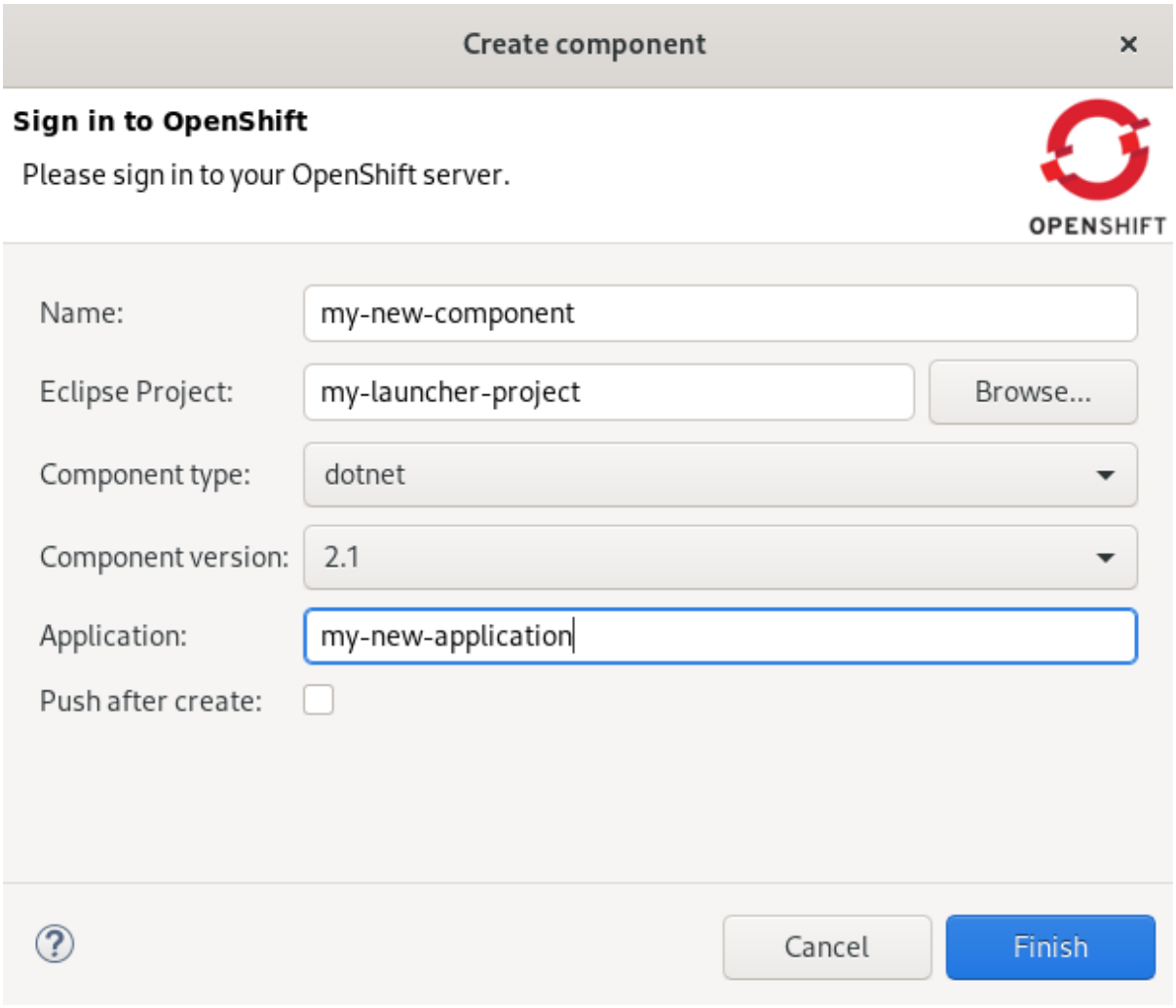
CodeReady Studio で OpenShift Application Explorer を使用して新しいコンポーネントを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **OpenShift Application Explorer** を起動します。
3. ターゲットの **Project → New → Component** を右クリックします。



Create component ウィンドウが表示されます。



4. プロジェクトに名前を付けます。
5. **Browse** をクリックして **Eclipse Project** を選択します。
6. 希望の **Component type** を設定します。
7. 希望の **Component version** を設定します。
8. アプリケーションに名前を付けます。
9. **Push after create** チェックボックスを未選択にします。
10. **Finish** をクリックします。
Console ビューが表示され、検証プロセスが表示されます。

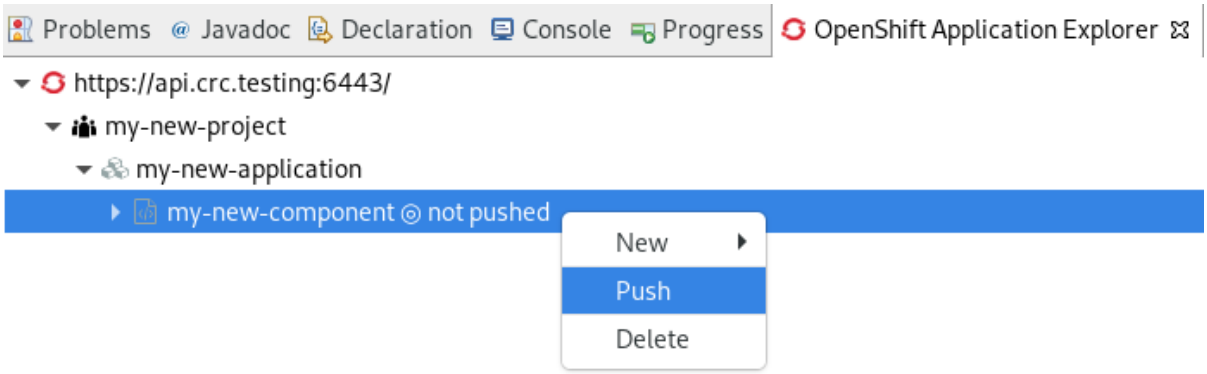
新たに作成されたコンポーネントが、プロジェクト下の **OpenShift Application Explorer** ビューに表示されます。

5.6. OPENSHIFT APPLICATION EXPLORER を使用したクラスターへのコンポーネントのデプロイ

CodeReady Studio で OpenShift Application Explorer を使用して、クラスターにコンポーネントをデプロイする方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **OpenShift Application Explorer** を起動します。
3. プロジェクトを展開します。
4. アプリケーションを展開します。
5. **component → Push** を右クリックします。



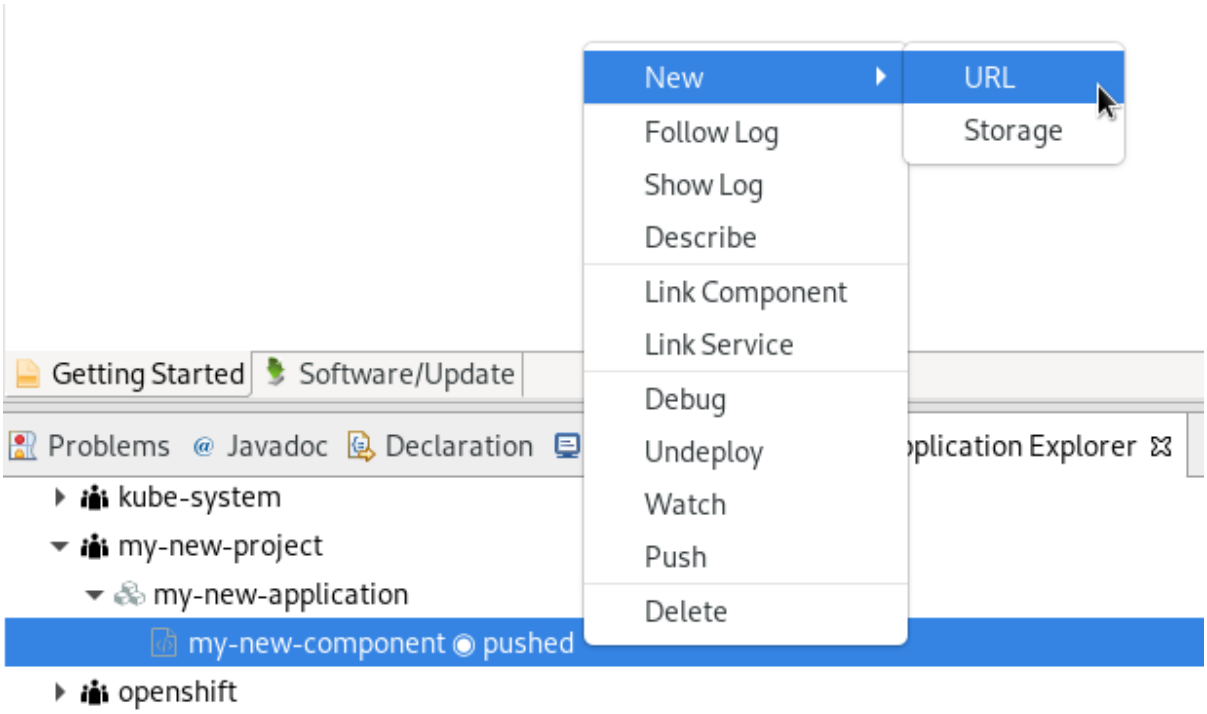
Console ビューが表示され、ファイル同期化のプロセスが表示されます。

5.7. OPENSHIFT APPLICATION EXPLORER を使用した外部アクセス URL の定義

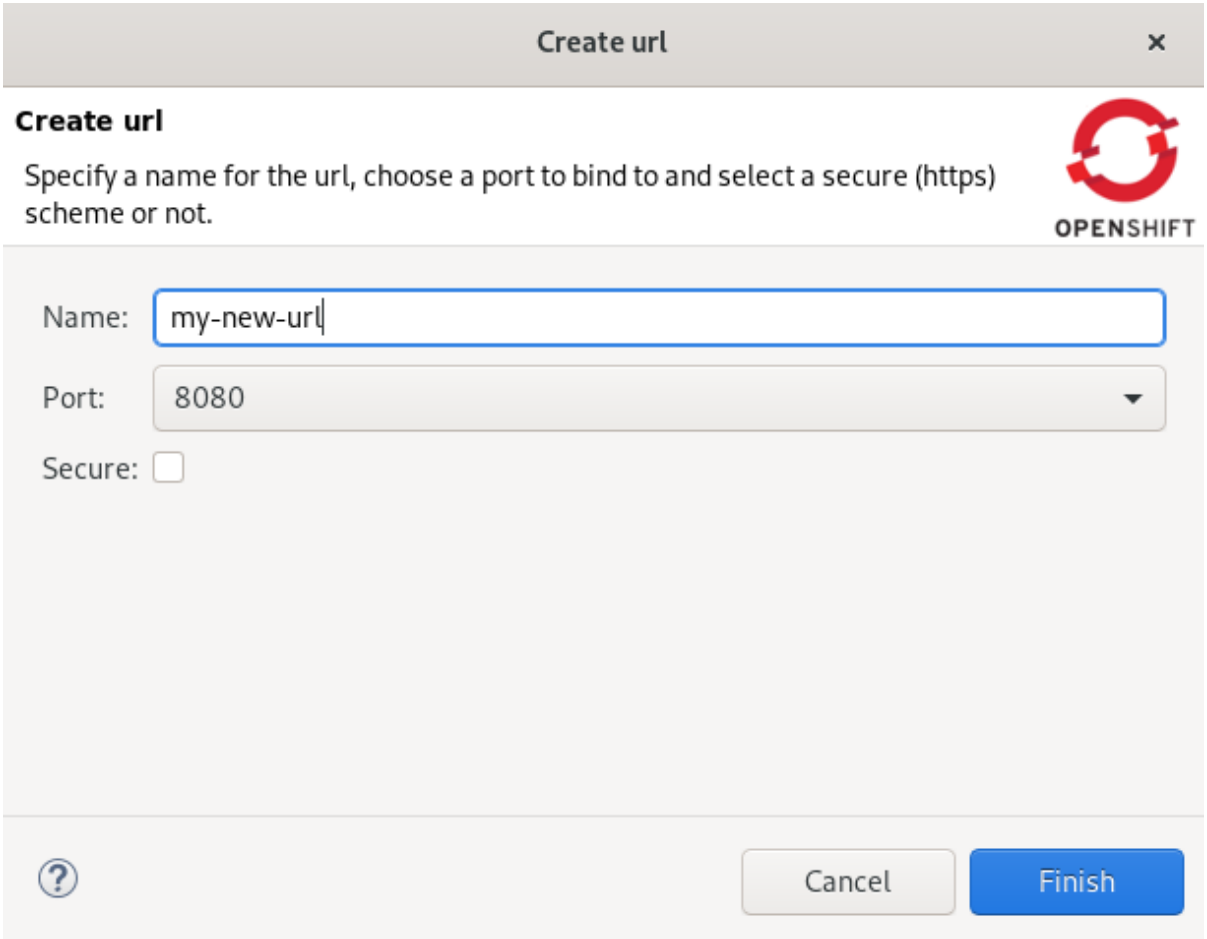
CodeReady Studio で OpenShift Application Explorer を使用して外部アクセス URL を定義する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **OpenShift Application Explorer** を起動します。
3. プロジェクトを展開します。
4. アプリケーションを展開します。
5. **component → New → URL** を右クリックします。

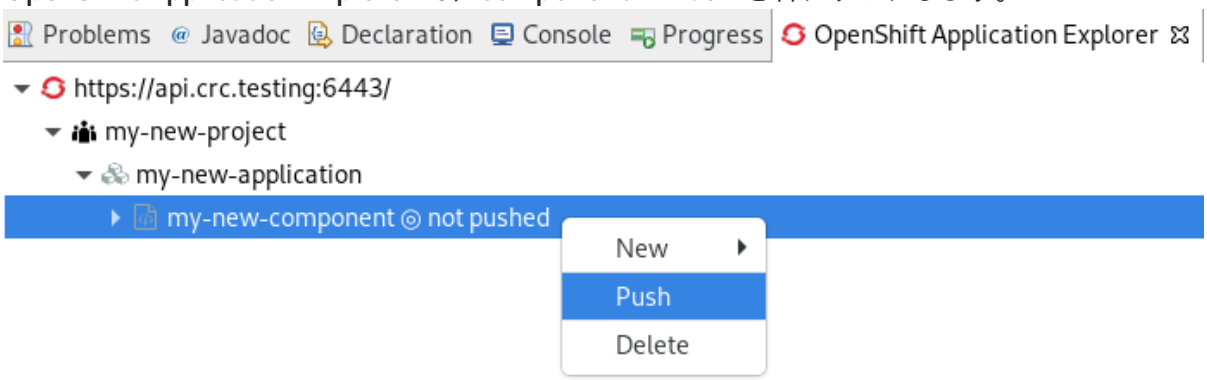


Create URL ウインドウが表示されます。



6. URL に名前を付けます。
7. **Port** の値を 8080 に設定します。
8. **Finish** をクリックします。
   Console ビューが表示され、URL 作成のプロセスが表示されます。

9. **OpenShift Application Explorer** で、**component → Push** を右クリックします。



Console ビューが表示され、ファイル同期化のプロセスが表示されます。

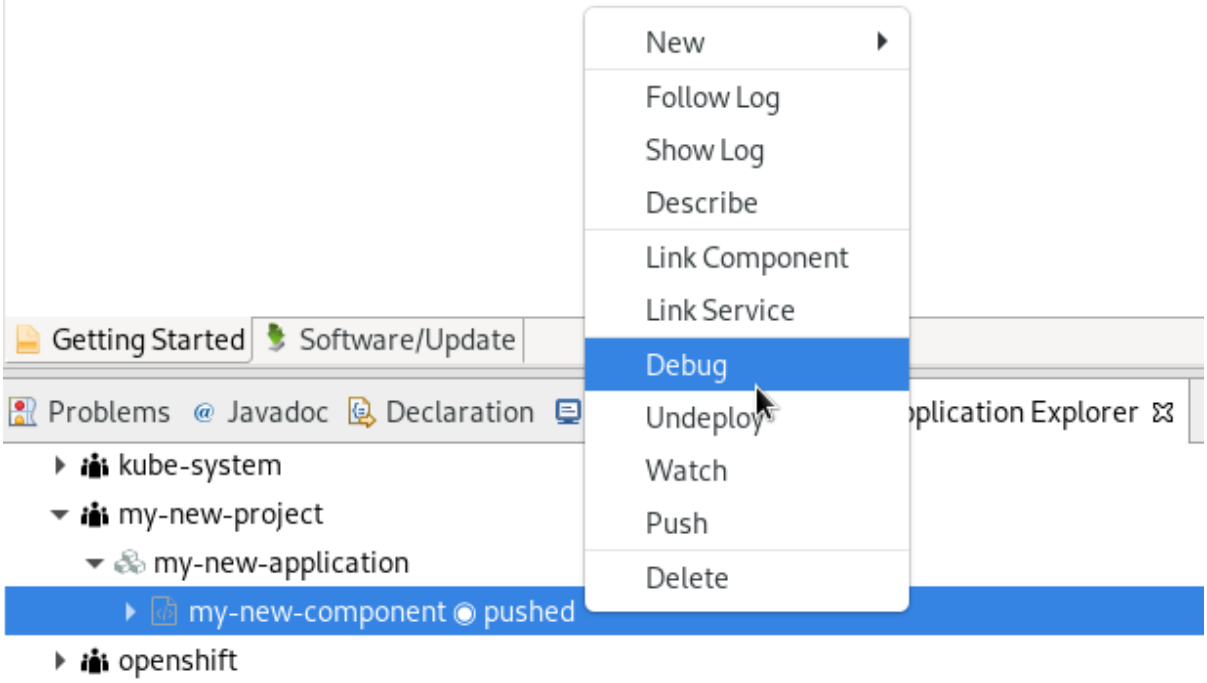
新たに作成された URL がコンポーネント下の **OpenShift Application Explorer** ビューに表示されます。

5.8. OPENSHIFT APPLICATION EXPLORER を使用したクラスターでのアプリケーションのデバッグ

CodeReady Studio で OpenShift Application Explorer を使用してコンポーネントをデバッグする方法を説明します。

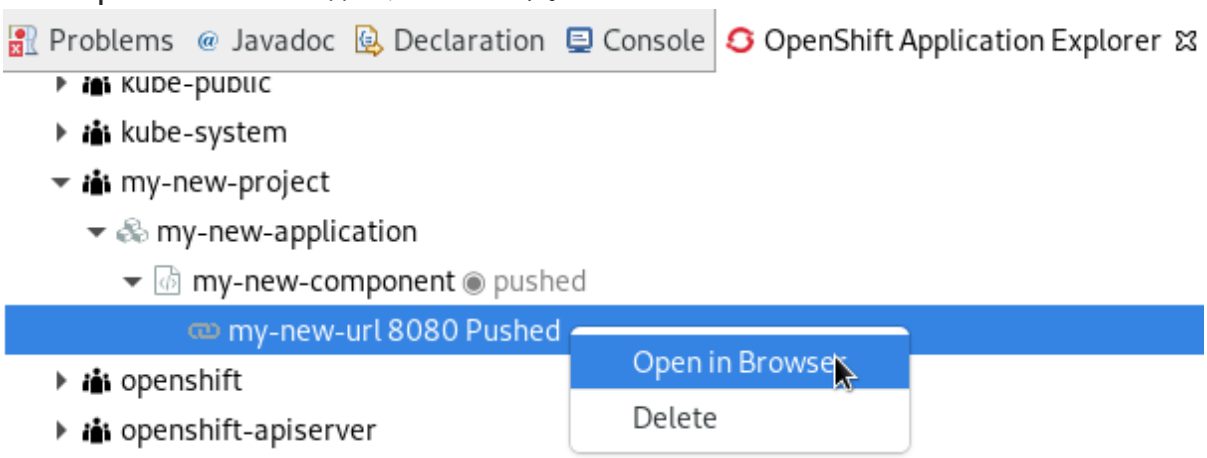
手順

1. CodeReady Studio を起動します。
2. **OpenShift Application Explorer** を起動します。
3. プロジェクトを展開します。
4. アプリケーションを展開します。
5. **component → Debug** を右クリックします。



Console ビューが表示されます。

6. OpenShift Application Explorer で、コンポーネントを展開します。
7. url → Open in Browserを右クリックします。



Confirm Perspective Switch ウインドウが表示されます。

8. Switch をクリックします。
Debug Perspective ウィンドウが表示され、デバッグプロセスが表示されます。

第6章 CODEREADY STUDIO の QUARKUS ツールの基本

Quarkus は Kubernetes ネイティブのフルスタック Java フレームワークで、Java 仮想マシンとの作業を最適化することを目的としています。Quarkus は、Quarkus アプリケーション開発者向けのツールを提供します。Java アプリケーションおよびコンテナーイメージフットプリントのサイズを削減し、プログラミングの負担をなくし、必要になるメモリー量を削減します。

前提条件

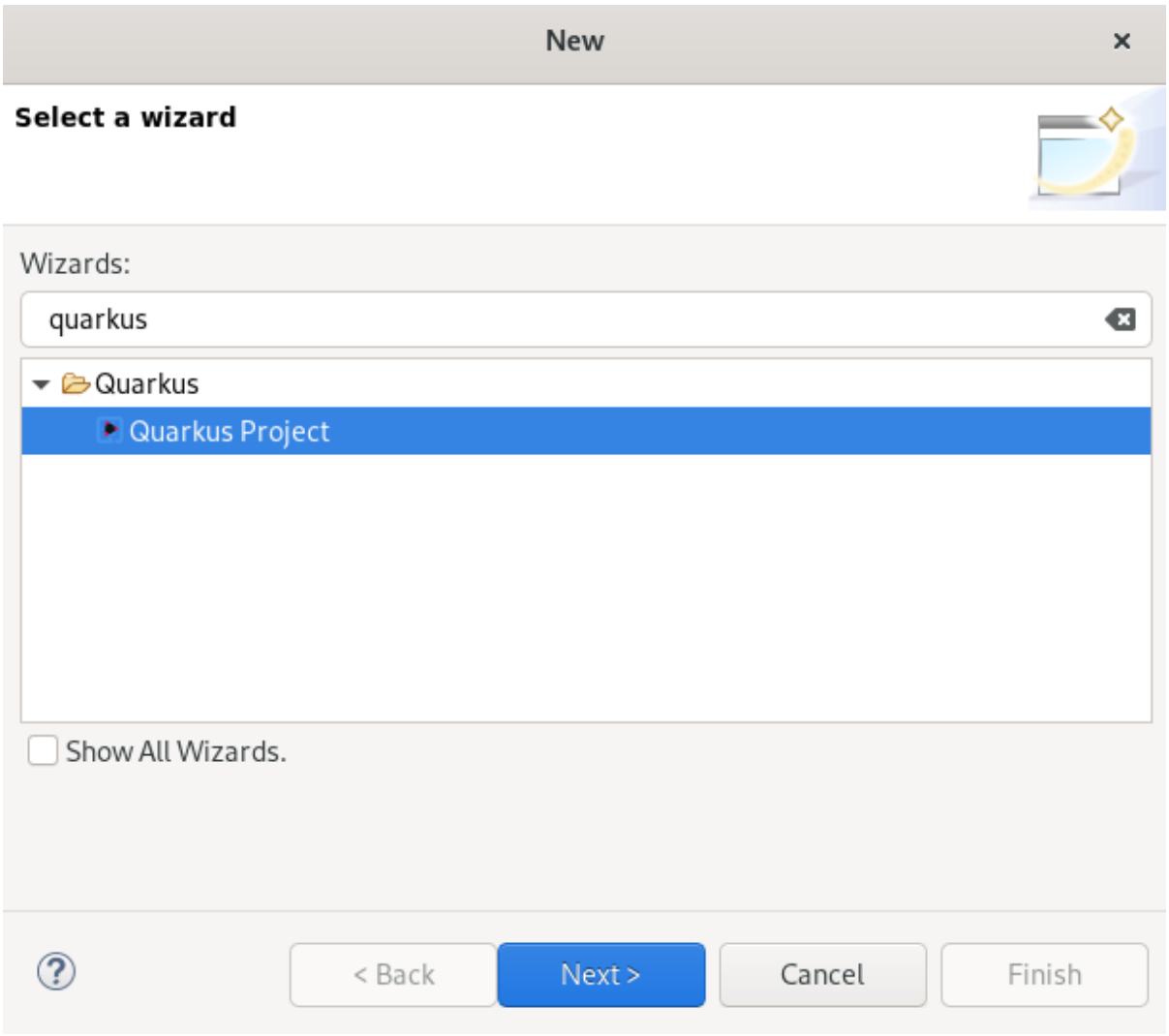
- 最新バージョンの JBoss Tools がインストールされていることを確認してください。詳細は、「JBoss Tools」を参照してください。

6.1. 新しい QUARKUS プロジェクトの作成

CodeReady Studio で新しい Quarkus プロジェクトを作成する方法を説明します。

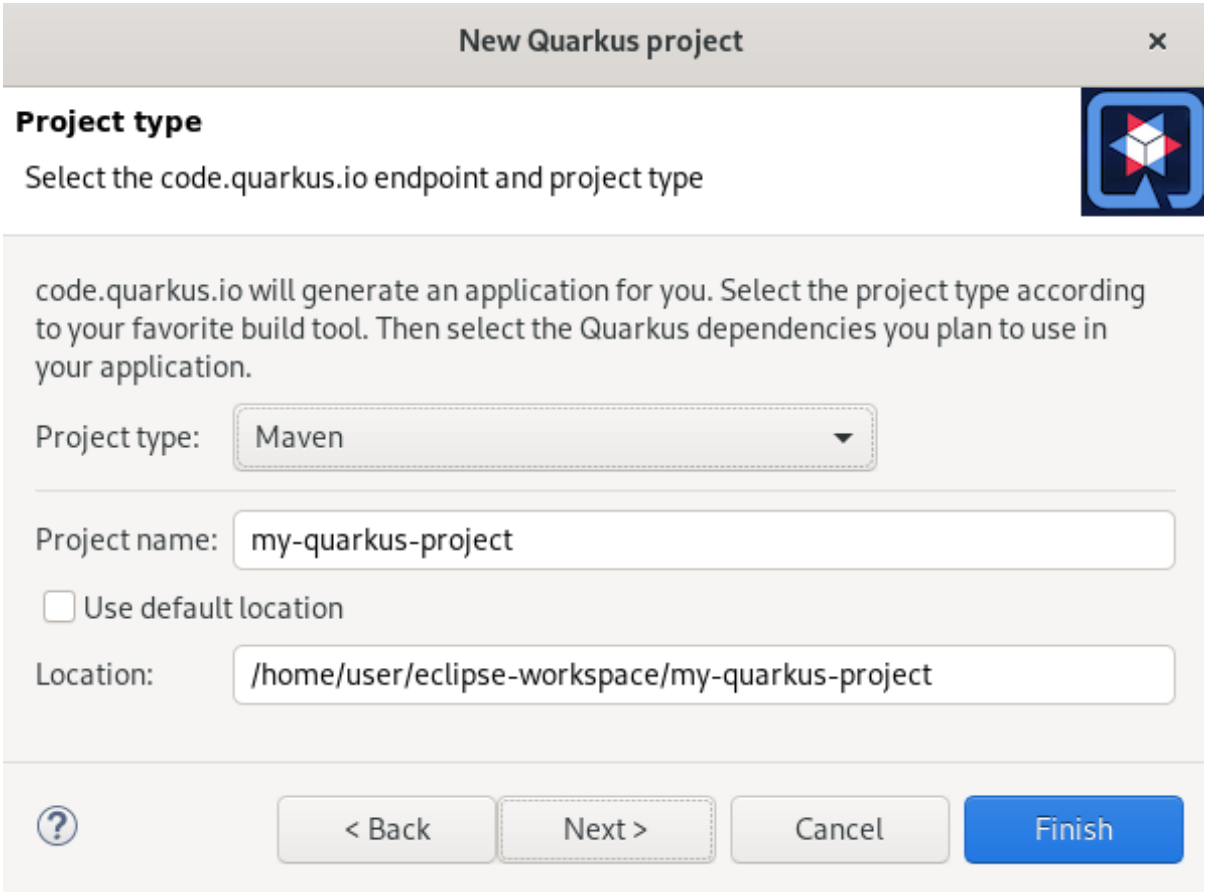
手順

1. CodeReady Studio を起動します。
2. **Ctrl**+**N** キーを押します。
   Select a wizard ウインドウが表示されます。

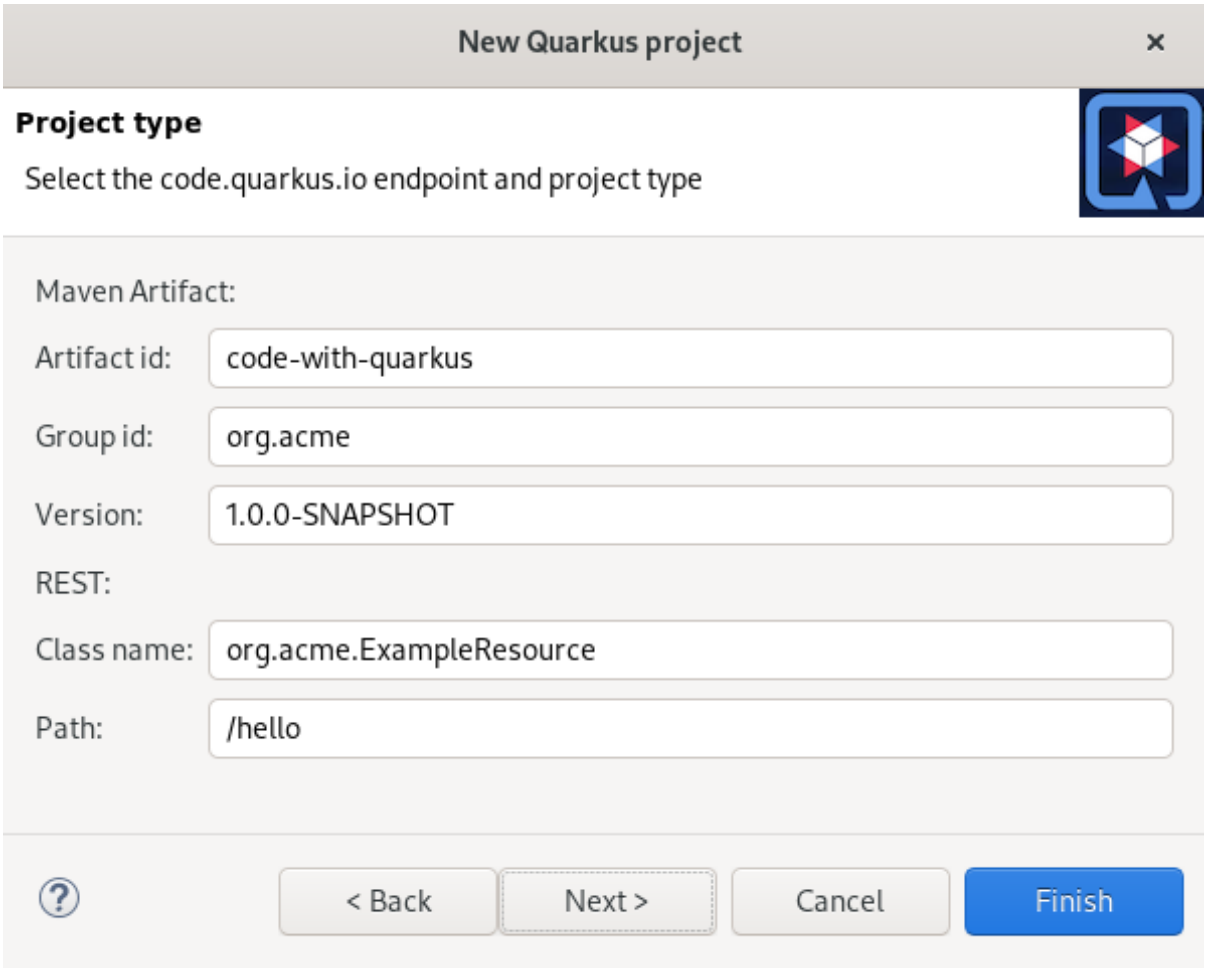


3. 検索フィールドに **Quarkus** と入力します。

4. **Quarkus Project** を選択します。
5. **Next** をクリックします。
   New Quarkus project ウインドウが表示されます。

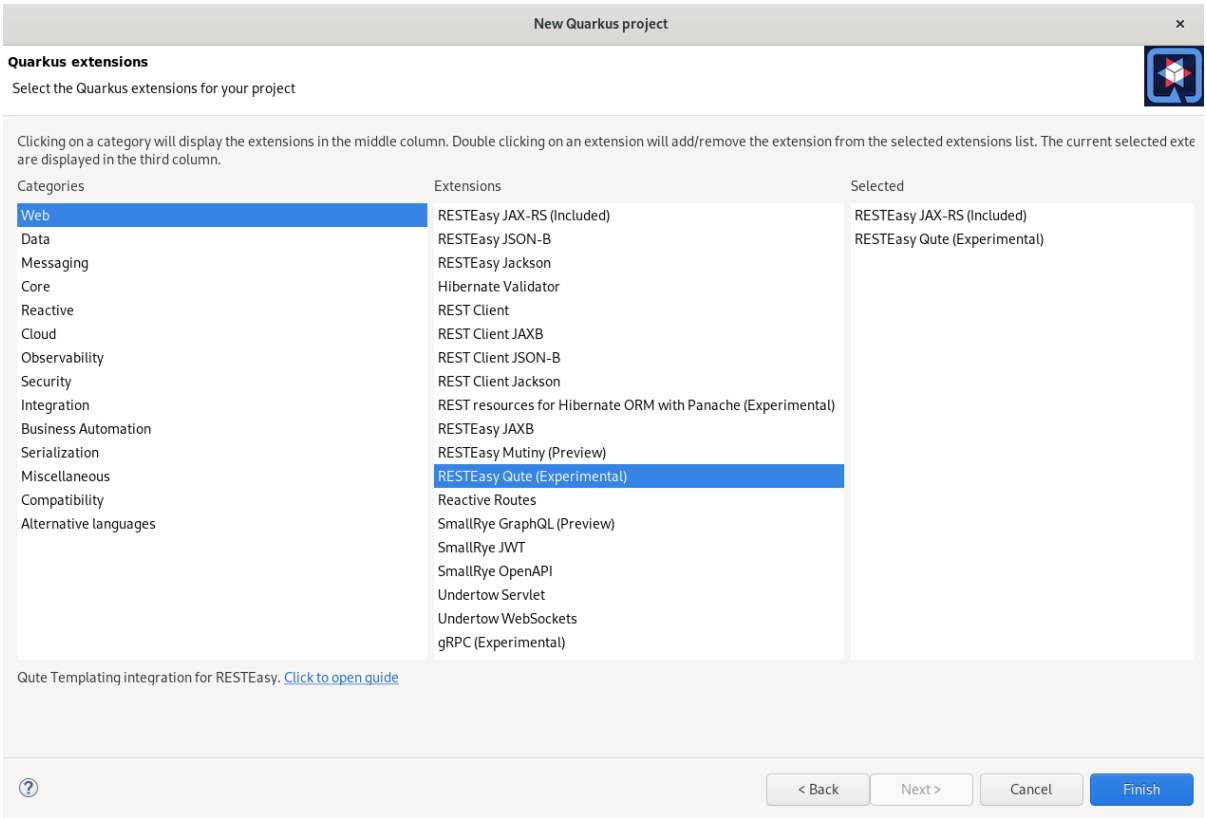


6. 適切なプロジェクトタイプを選択します。
7. プロジェクトに名前を付けます。
8. プロジェクトの場所を選択します。
9. **Next** をクリックします。
   Project type ウインドウが表示されます。



10. デフォルト値が正しいことを確認します。

11. **Next** をクリックします。
Quarkus extensions ウインドウが表示されます。



12. プロジェクトに適した **Categories** を選択します。
選択したカテゴリーに使用可能なエクステンションが **Extensions** 列に表示されます。

13. プロジェクトに適した **Extensions** を選択します。
エクステンションをダブルクリックして選択または選択解除します。選択したエクステンションが **Selected** 列に表示されます。

14. **Finish** をクリックします。

新たに作成された Quarkus プロジェクトが **Project Explorer** ビューに表示されます。

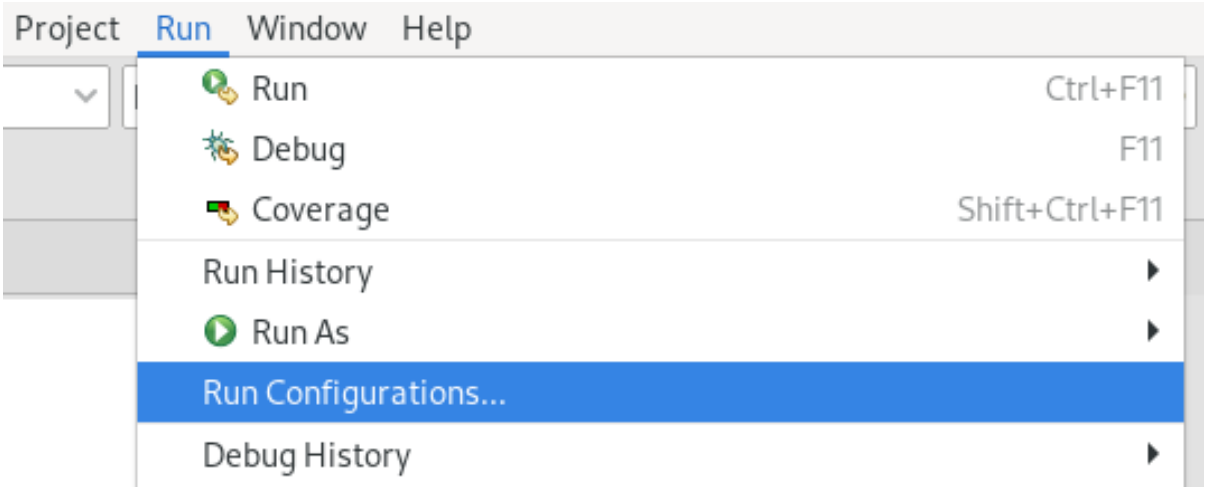
6.2. QUARKUS アプリケーションの実行

CodeReady Studio で Quarkus アプリケーションを実行する方法を説明します。

手順

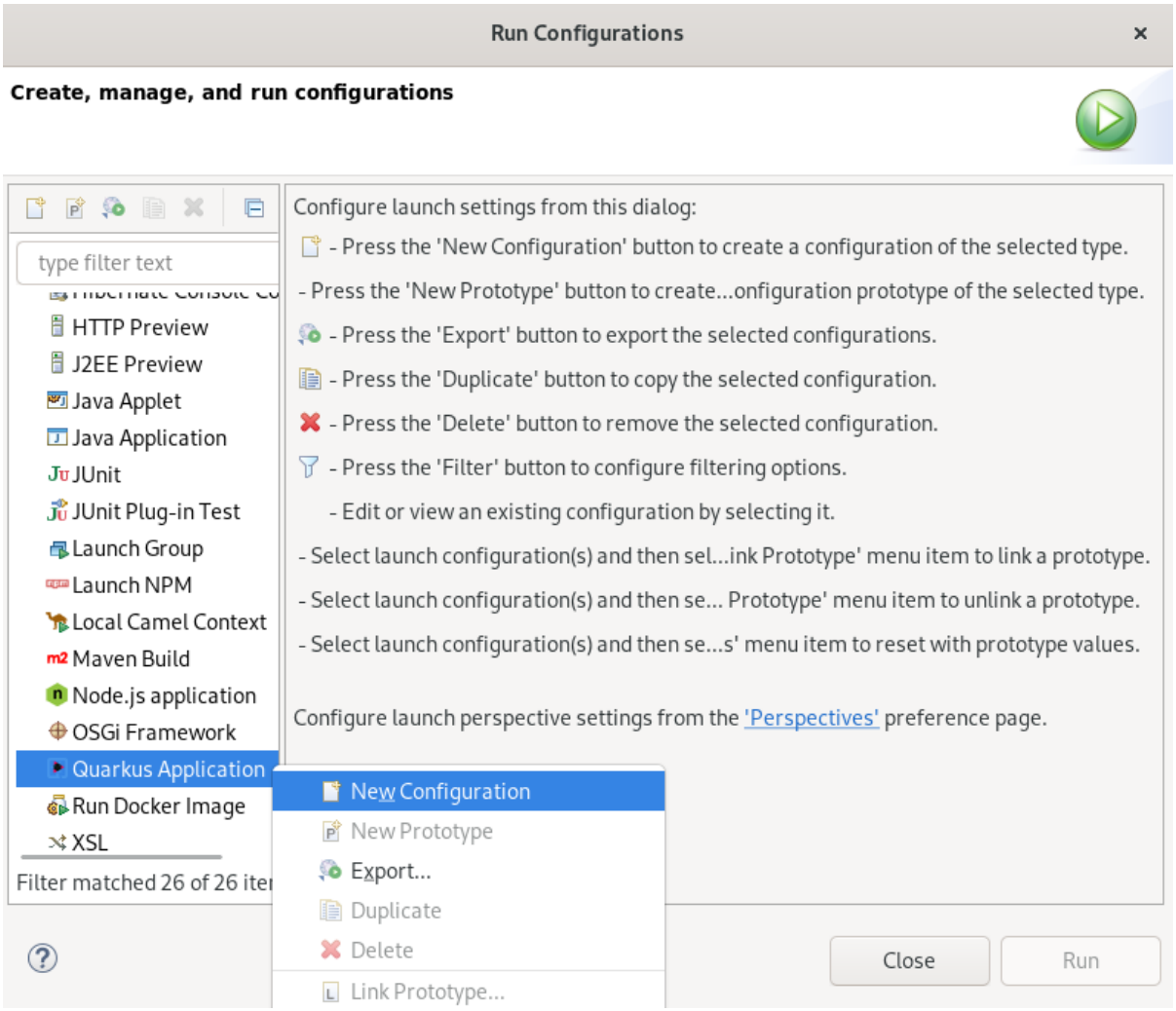
1. CodeReady Studio を起動します。

2. **Run → Run Configurations** とクリックします。

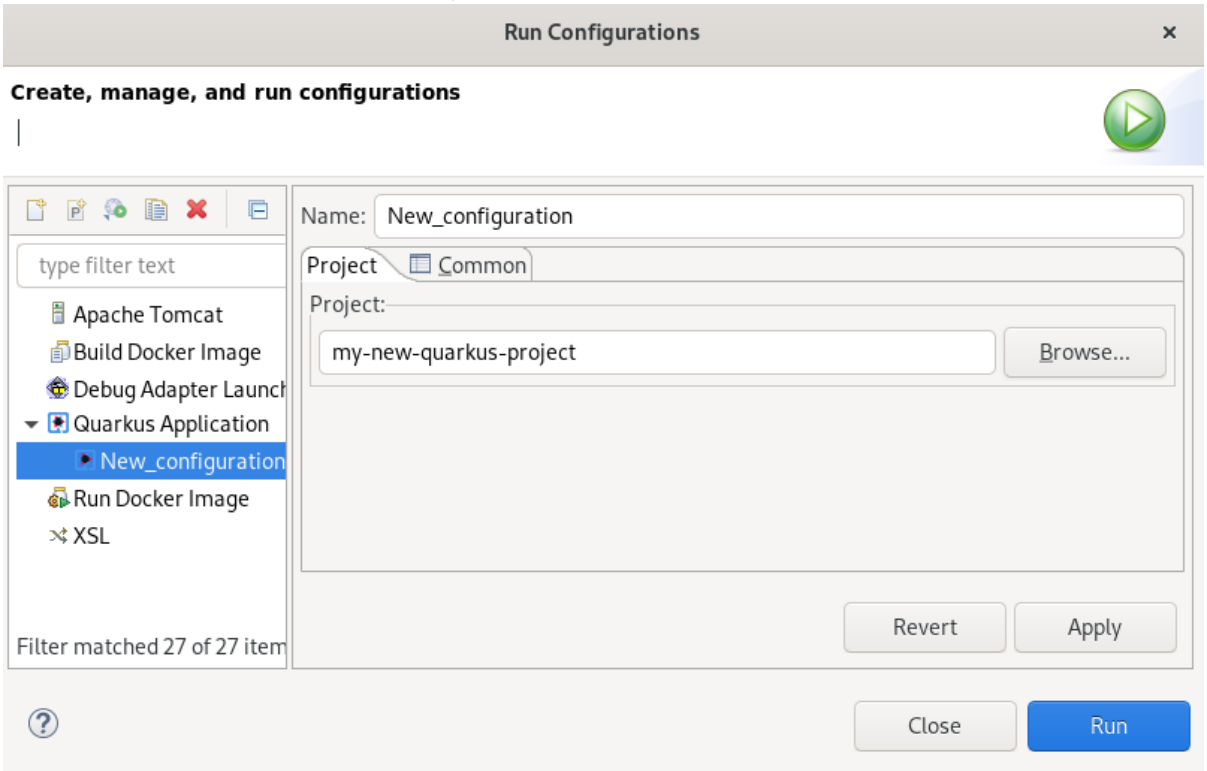


Run Configurations ウィンドウが表示されます。

3. **Quarkus Application** まで下へスクロールします。



4. Quarkus Application → New Configuration を右クリックします。



5. 設定に名前を付けます。

6. **Browse** をクリックしてプロジェクトを見つけます。

7. **Apply** をクリックします。
8. **Run** クリックします。
   Console ビューが表示されます。

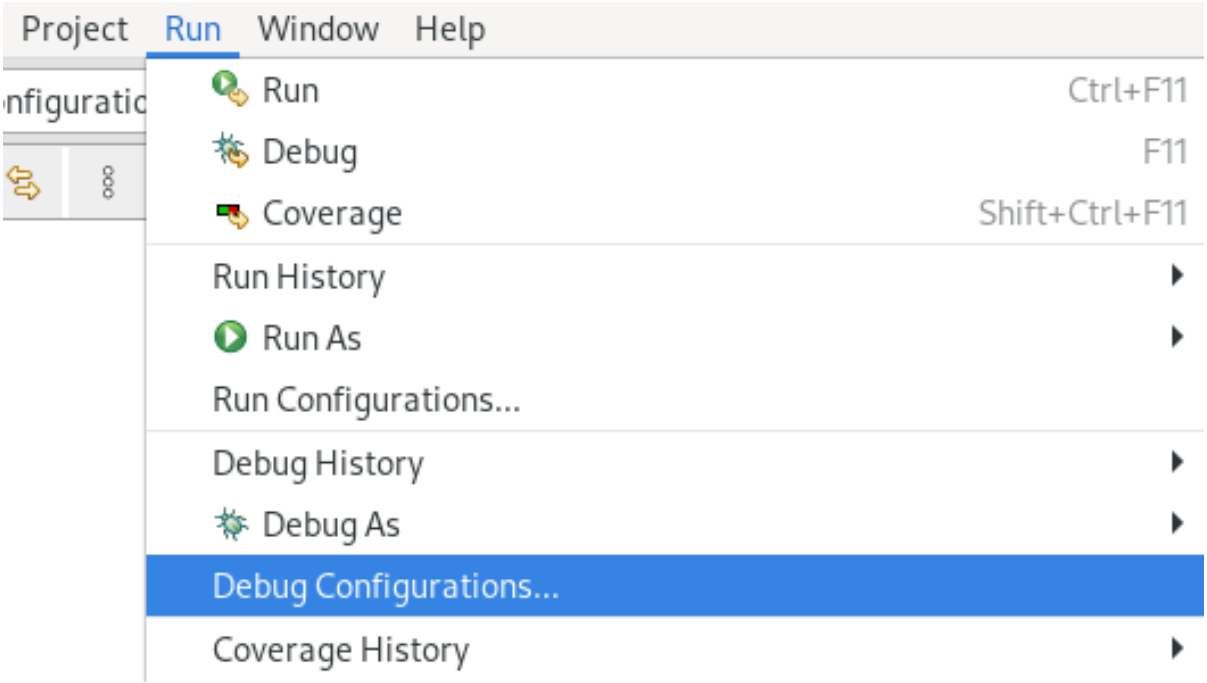
アプリケーションは、ビルドプロセスの後に起動します。

6.3. QUARKUS アプリケーションのデバッグ

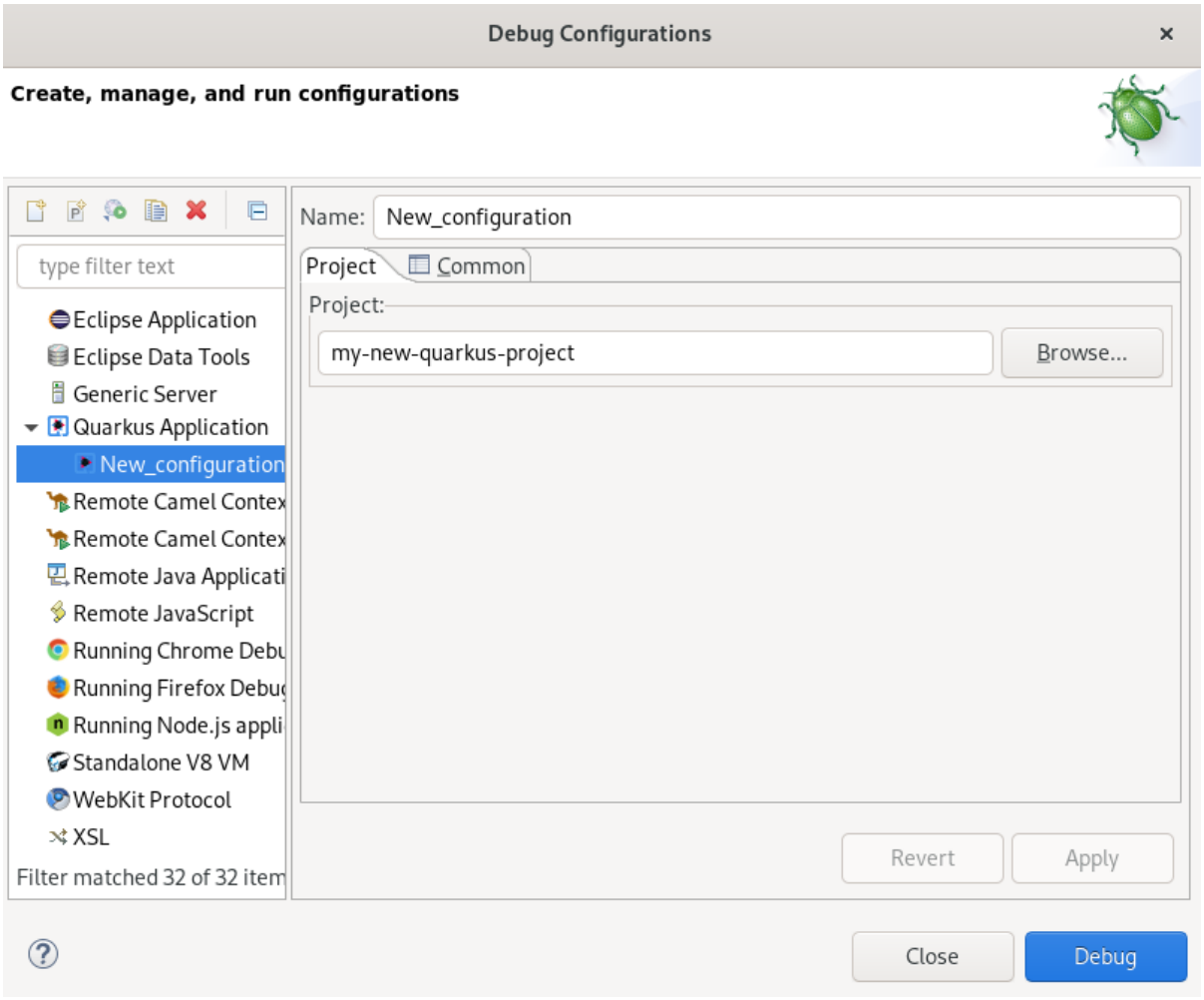
CodeReady Studio で Quarkus アプリケーションをデバッグする方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Run → Debug Configurations** とクリックします。



   Debug Configurations ウィンドウが表示されます。



3. **Quarkus Application** を展開します。
4. 設定を選択します。
5. **Debug** をクリックします。
   Console ビューが表示されます。

Quarkus アプリケーションが起動し、リモート JVM デバッグ設定に接続します。アプリケーションのソースファイルにブレークポイントを設定すると、ブレークポイントに到達した後に実行が自動的に停止します。

6.4. CODEREADY STUDIO での言語サポートの使用

すべての Quarkus アプリケーションは、**application.properties** 設定ファイルを使用して設定されます。この設定ファイルの内容は、アプリケーションが使用している Quarkus エクステンションのセットによって異なります。

Quarkus ツールには、コード補完、検証、およびドキュメントを提供するコンテンツアシストが含まれています。コード補完により、コードのステートメントを迅速に完了できます。ポップアップから複数の選択肢を利用することができます。

この言語サポートは、Kubernetes、OpenShift、S2i、Docker プロパティー、MicroProfile REST Client プロパティー、および MicroProfile Health アーティファクトで使用できるようになりました。

6.4.1. Quarkus コード補完の使用

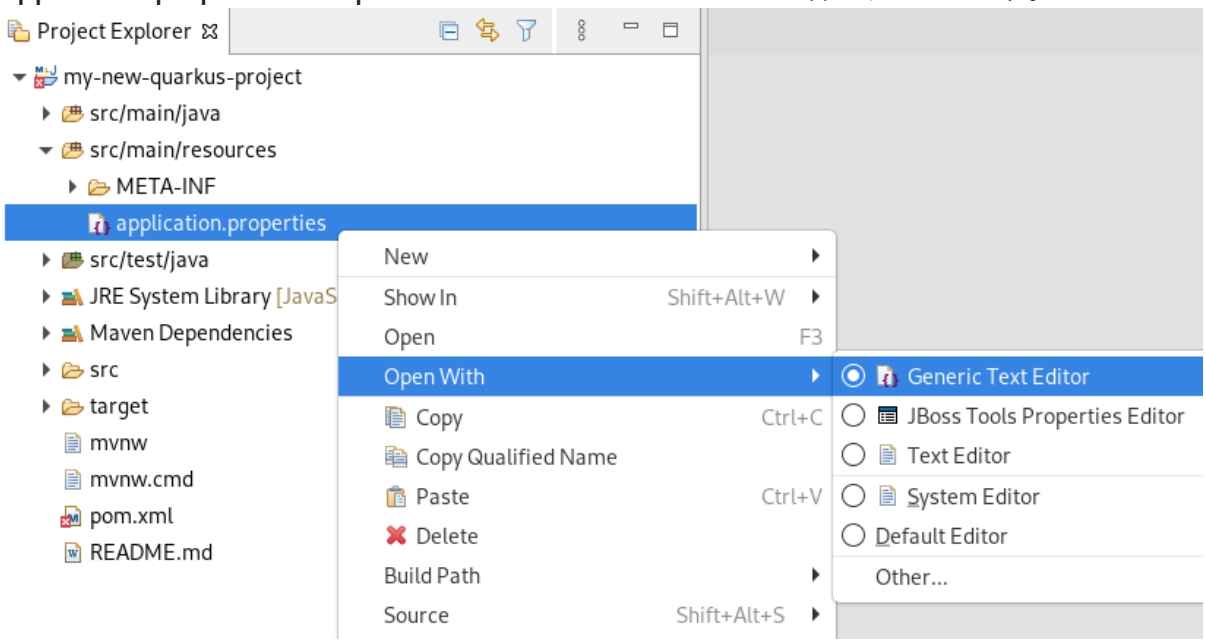
CodeReady Studio で Quarkus の **application.properties** コンテントアシストを使用する方法を説明します。

前提条件

- 既存の Quarkus プロジェクト。
  Quarkus プロジェクトの作成方法に関する詳細は、「新しい Quarkus プロジェクトの作成」 を参照してください。

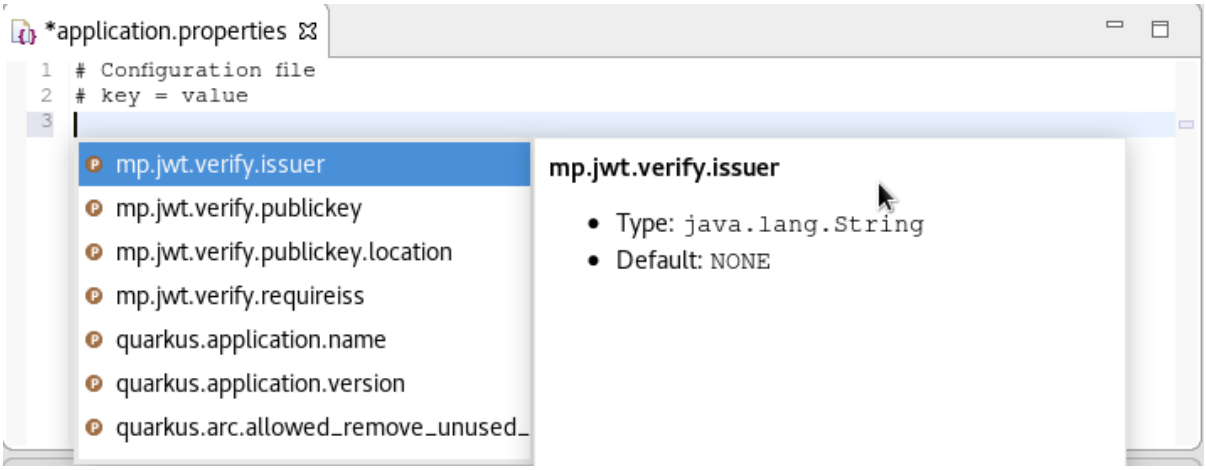
手順

1. CodeReady Studio を起動します。
2. **Project Explorer** を起動します。
3. **Quarkus project** → **src** → **main** → **resources** を展開します。
4. **application.properties** → **Open With** → **Generic Text Editor** を右クリックします。

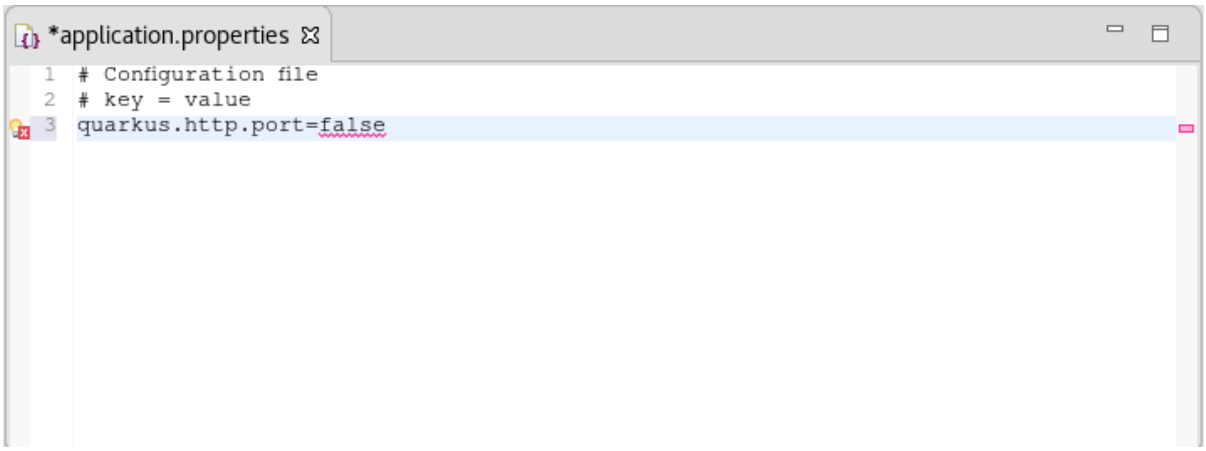


   Text Editor ウインドウが表示されます。

5. 空の行に移動します。
6. **Ctr+Space** キーを押して、コード補完を実行します。
   コード補完のサジェスチョンが表示されます。マウスカーソルをサジェスチョンの上に移動し、ドキュメントを表示します。



エディターで誤った値を入力すると、エラーの下に赤い波線が表示されます。



その他のリソース

- MicroProfile REST Client プロパティーと MicroProfile Health アーティファクトの言語サポートは、個別に有効にする必要があります。詳細は、「MicroProfile の言語サポートの有効化」 を参照してください。

6.4.2. MicroProfile の言語サポートの有効化

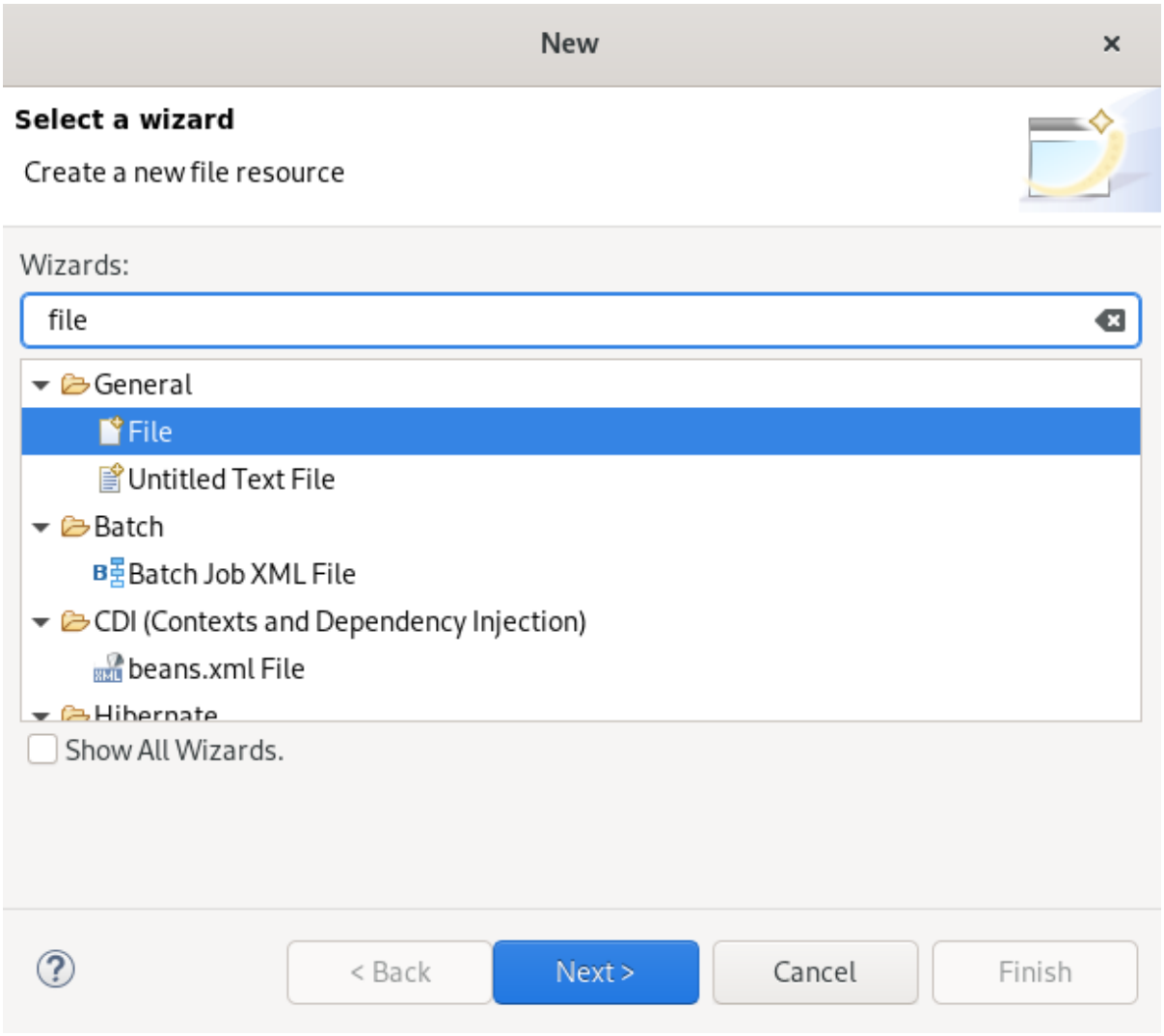
MicroProfile REST Client プロパティーの言語サポートを有効にする方法を説明します。

前提条件

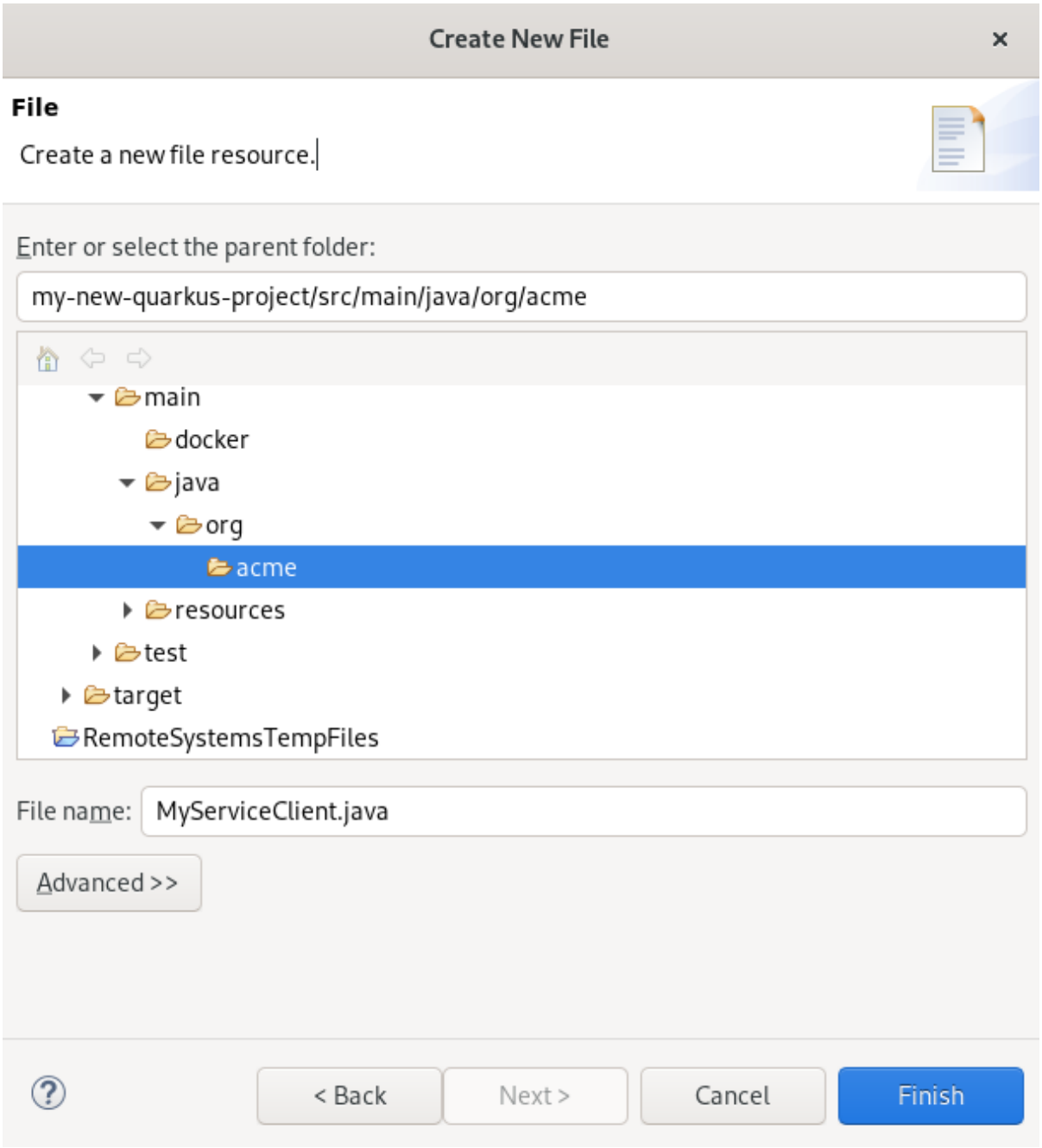
- 既存の Quarkus プロジェクト。
  Quarkus プロジェクトの作成方法に関する詳細は、「新しい Quarkus プロジェクトの作成」 を参照してください。

手順

1. CodeReady Studio を起動します。
2. **Project Explorer** を起動します。
3. **Quarkus project → src/main/java** を展開します。
4. **org.acme → New → Other** を右クリックします。
   Select wizard ウインドウが表示されます。



5. 検索フィールドに **file** と入力します。
6. **File** を選択します。
7. **Next** ボタンをクリックします。
Create a new file resourceウインドウが表示されます。



8. 新しいファイルに名前を付けます。
9. **Finish** をクリックします。
10. 新たに作成されたファイルに以下の内容を貼り付けます。

```
package org.acme;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;

@RegisterRestClient
public interface MyServiceClient {
 @GET
```

```
    @Path("/greet")
    Response greet();
}
```

11. **Ctrl**+**S** キーを押して変更を保存します。

クライアントの設定キーを **@RegisterRestClient** から **@RegisterRestClient(configKey = "myclient")** に変更すると、言語サポートを調整することができます。言語サポートは状況に応じて調整されます。

その他のリソース

- 言語サポートの使用方法の詳細は、「Quarkus コード補完の使用」 を参照してください。

第7章 CODEREADY STUDIO の HIBERNATE TOOLS の基本

Hibernate Tools は、Hibernate バージョン 5 以前に関連するプロジェクトのツールのコレクションです。このツールは、Hibernate とのリバースエンジニアリング、コード生成、可視化、および対話を行う Eclipse プラグインを提供します。

前提条件

1. h2 バージョンの Sakila データベース をダウンロードします。
2. **runh2.sh** ファイルが含まれるディレクトリーに移動します。
3. **runh2.sh** ファイルを実行します。

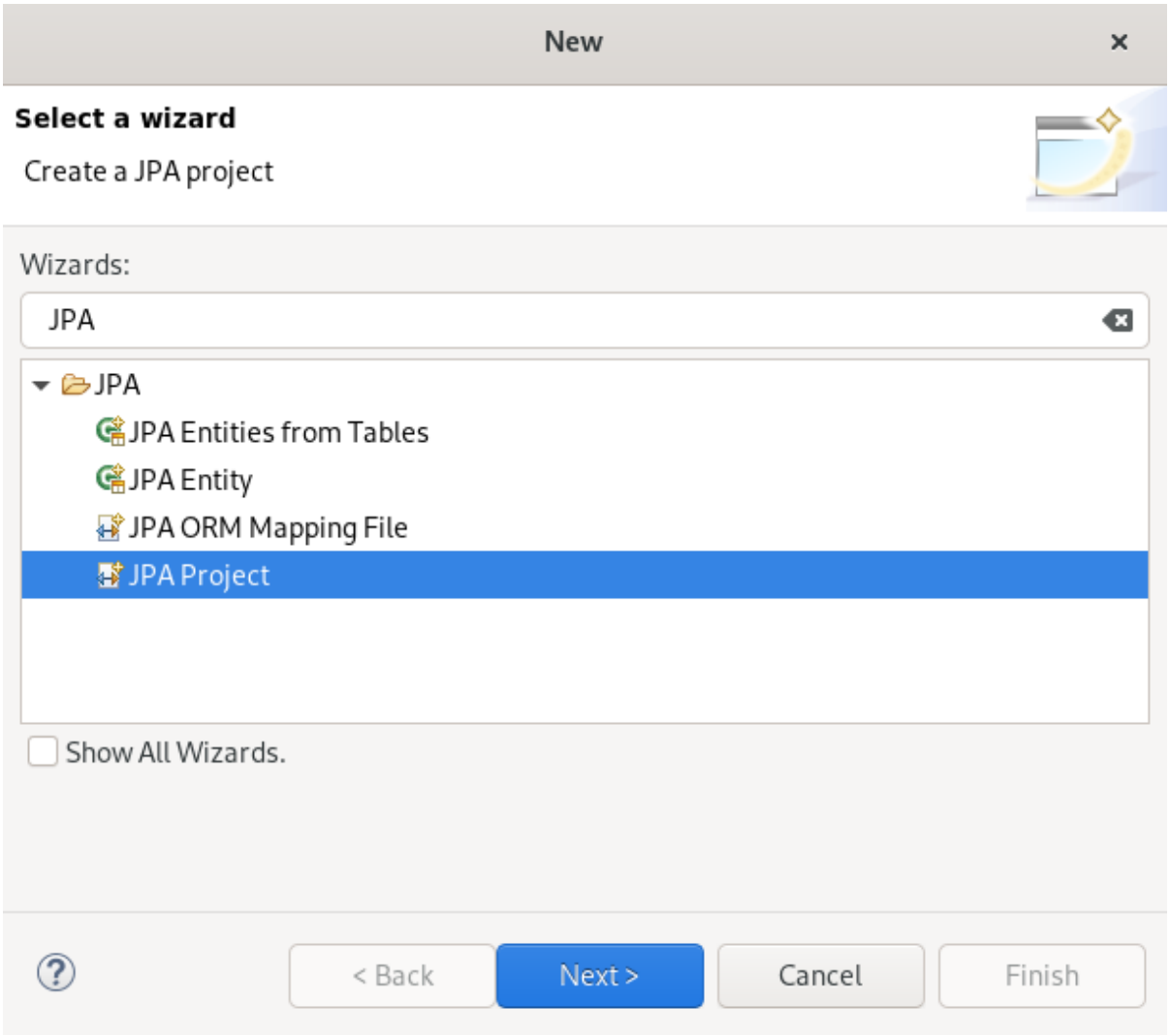
```
$ ./runh2.sh
```

7.1. 新規 JPA プロジェクトの作成

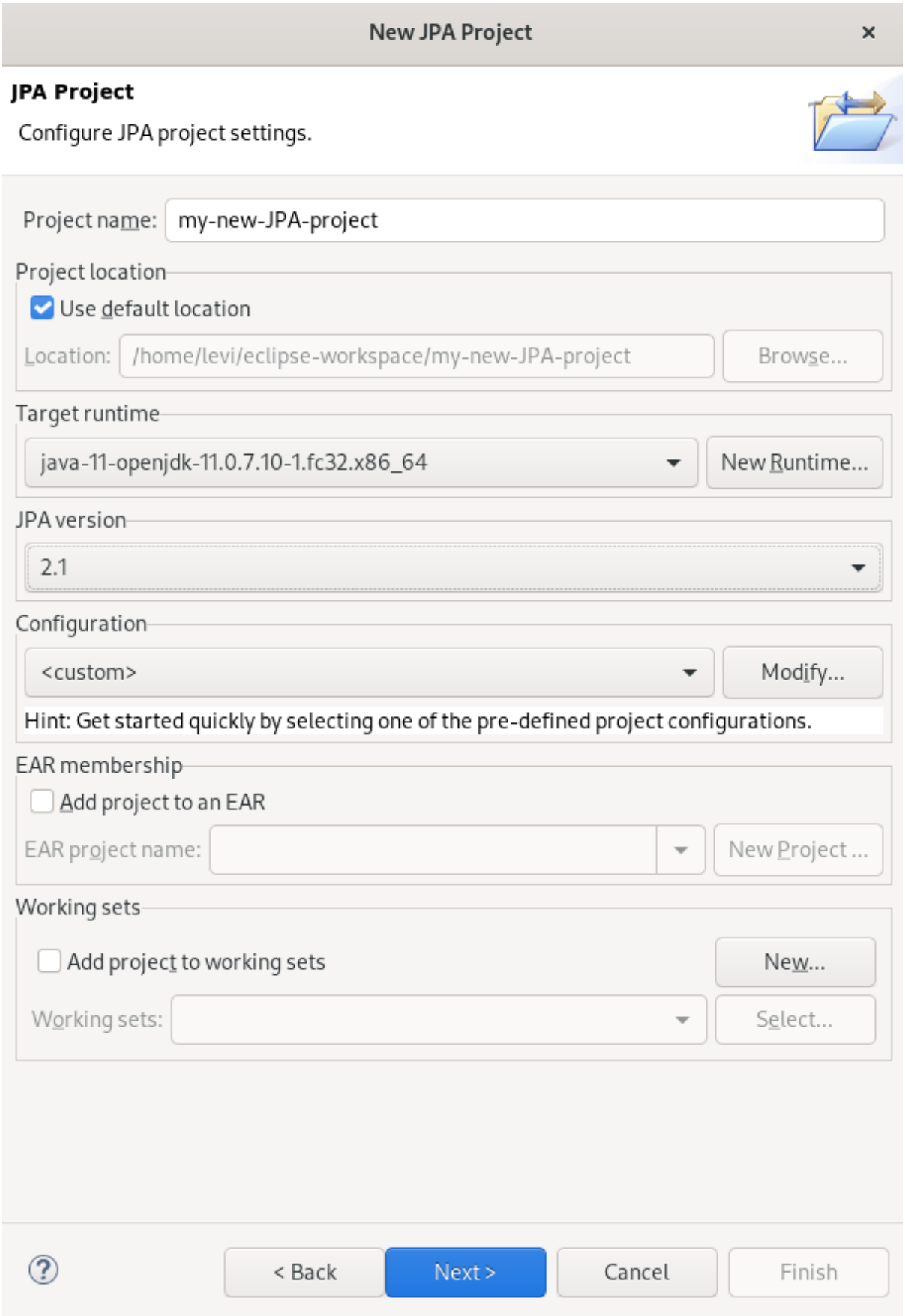
CodeReady Studio で新規 JPA プロジェクトを作成する方法を説明します。ここでは、前提条件に記載されている手順が完了済みであることを仮定しています。

手順

1. CodeReady Studio を起動します。
2. **Ctrl**+**N** キーを押します。
   Select a Wizard ウインドウが表示されます。

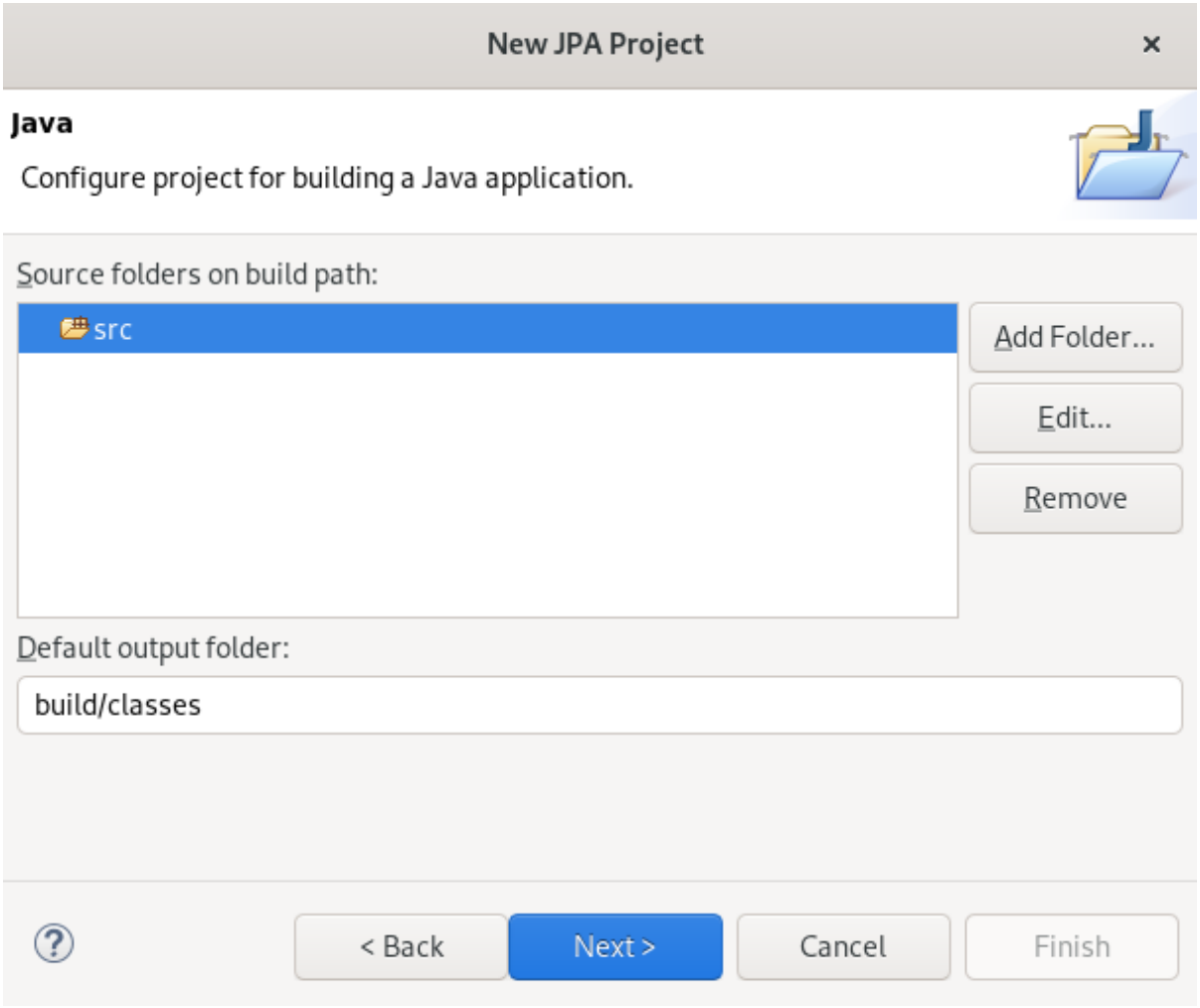


3. 検索フィールドに **JPA** と入力します。
4. **JPA Project** を選択します。
5. **Next** をクリックします。
   New JPA Project ウインドウが表示されます。

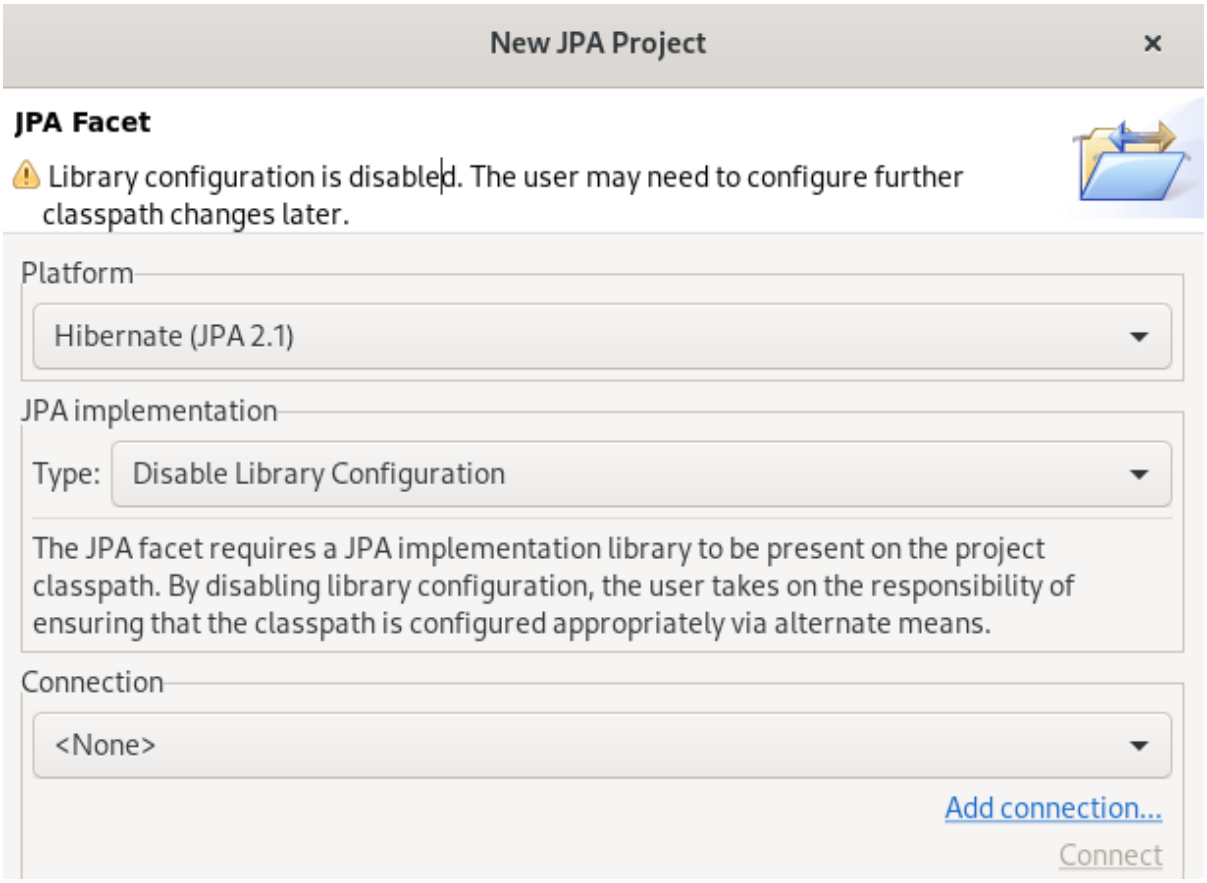


6. プロジェクトに名前を付けます。
7. プロジェクトの場所を選択します。
8. **Target runtime** フィールドで下矢印をクリックし、ランタイムサーバーを選択します。

9. **JPA version** を 2.1 に設定します。
10. **Next** をクリックします。
    Java ウィンドウが表示されます。



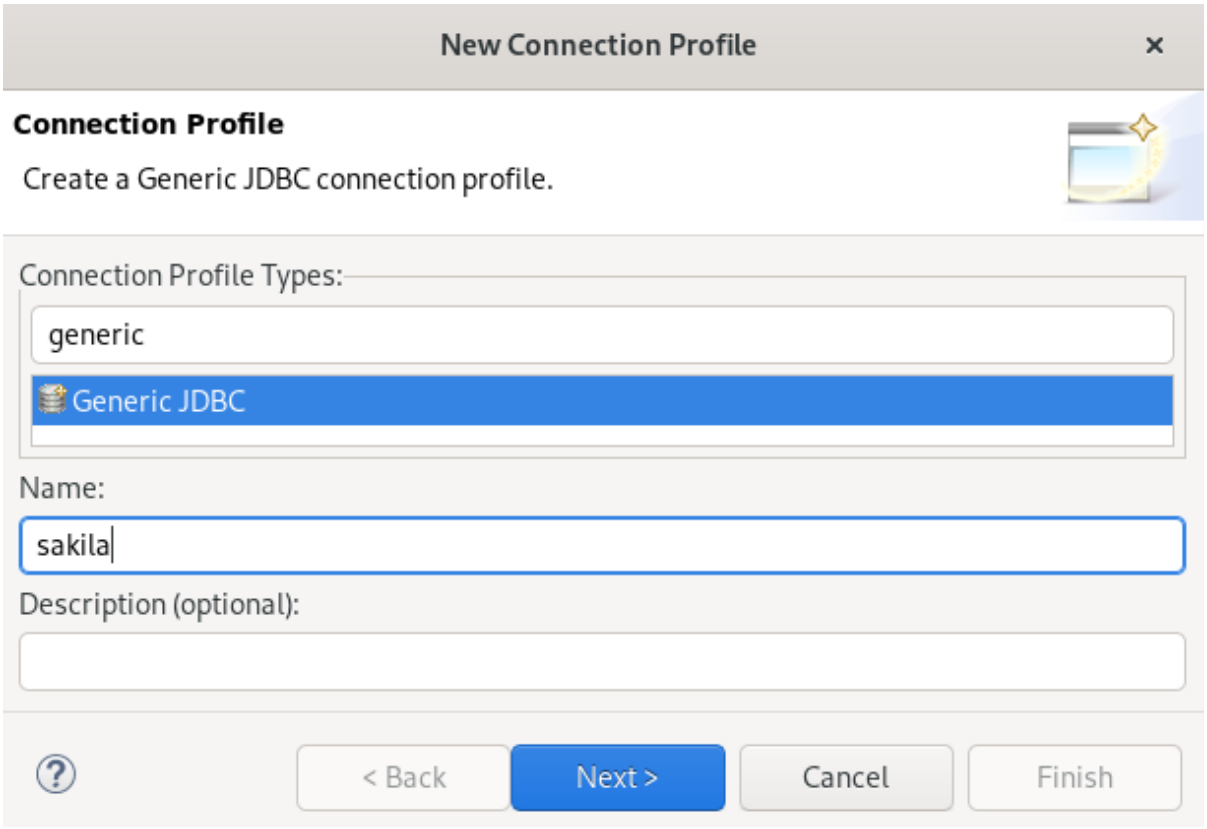
11. ソースフォルダーを選択します。
12. **Next** をクリックします。
    JPA Facet ウィンドウが表示されます。



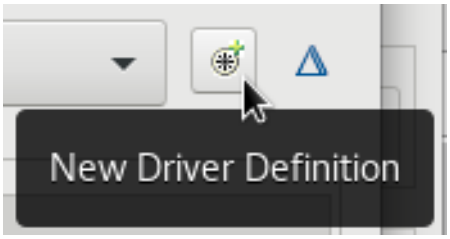
13. **Platform** フィールドの下矢印をクリックし、**Hibernate (JPA 2.1)** を選択します。

14. ユーザーライブラリーを追加するか、**JPA Implementation Type** を **Disable Library Configuration** に設定します。
ユーザーライブラリーの設定方法の詳細は、「ライブラリーの追加」 を参照してください。

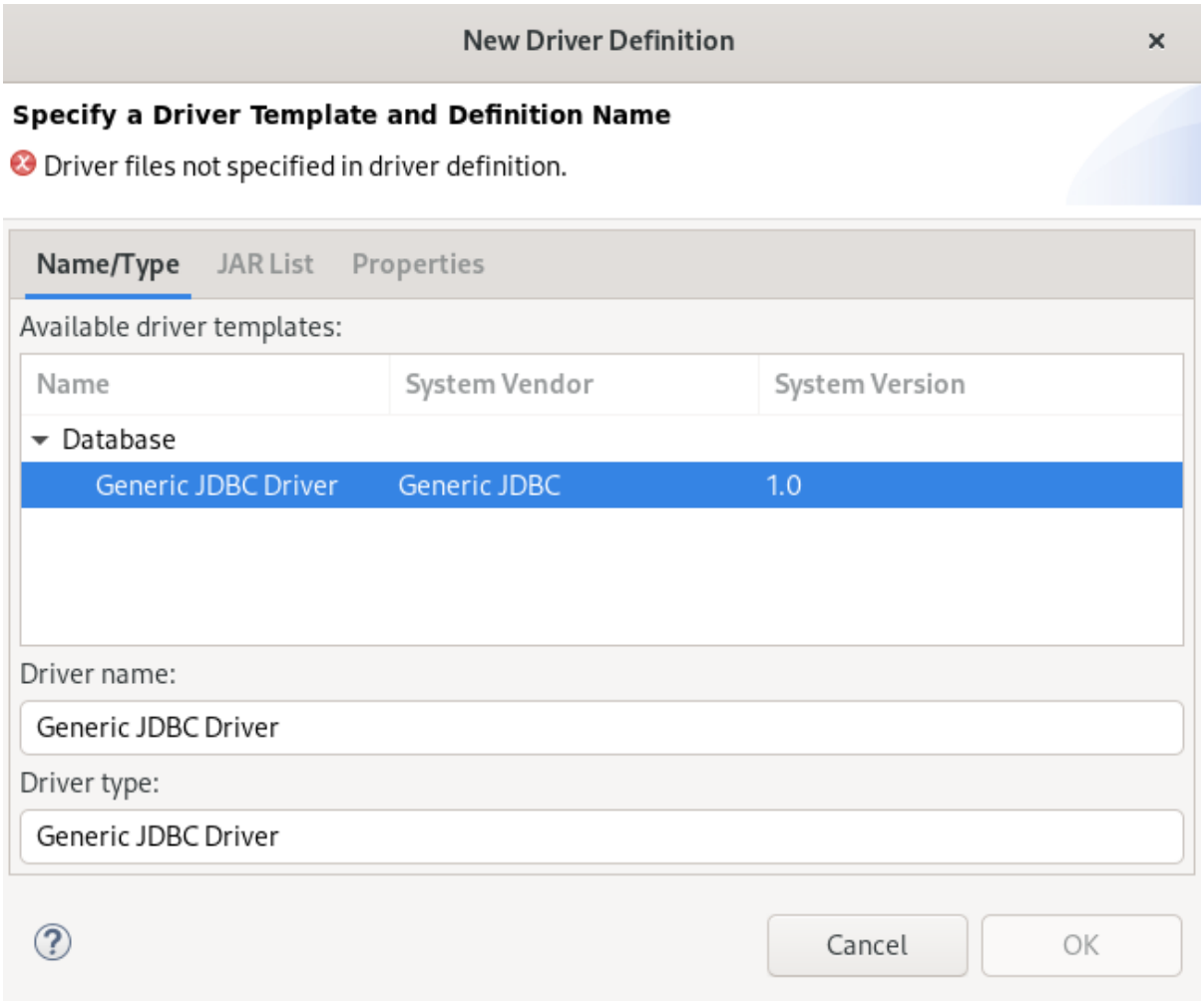
15. **Add connection** をクリックします。
Connection Profile ウインドウが表示されます。



16. 検索フィールドに **Generic** と入力します。
17. **Generic JDBC** を選択します。
18. **Name** フィールドに **sakila** と入力します。
19. **Next** をクリックします。
    Specify a Driver and Connection Detailsウィンドウが表示されます。
20. **New Driver Definition** アイコンをクリックします。

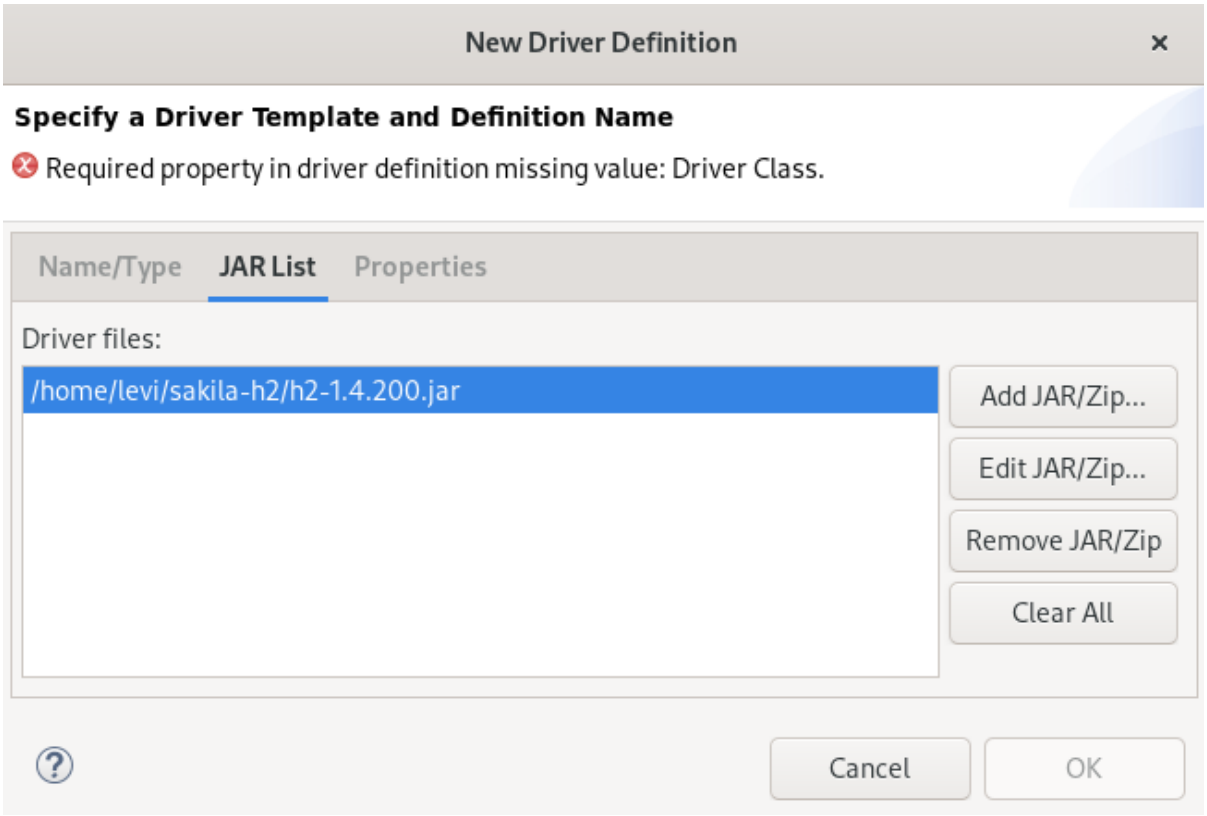


New Driver Definition ウインドウが表示されます。

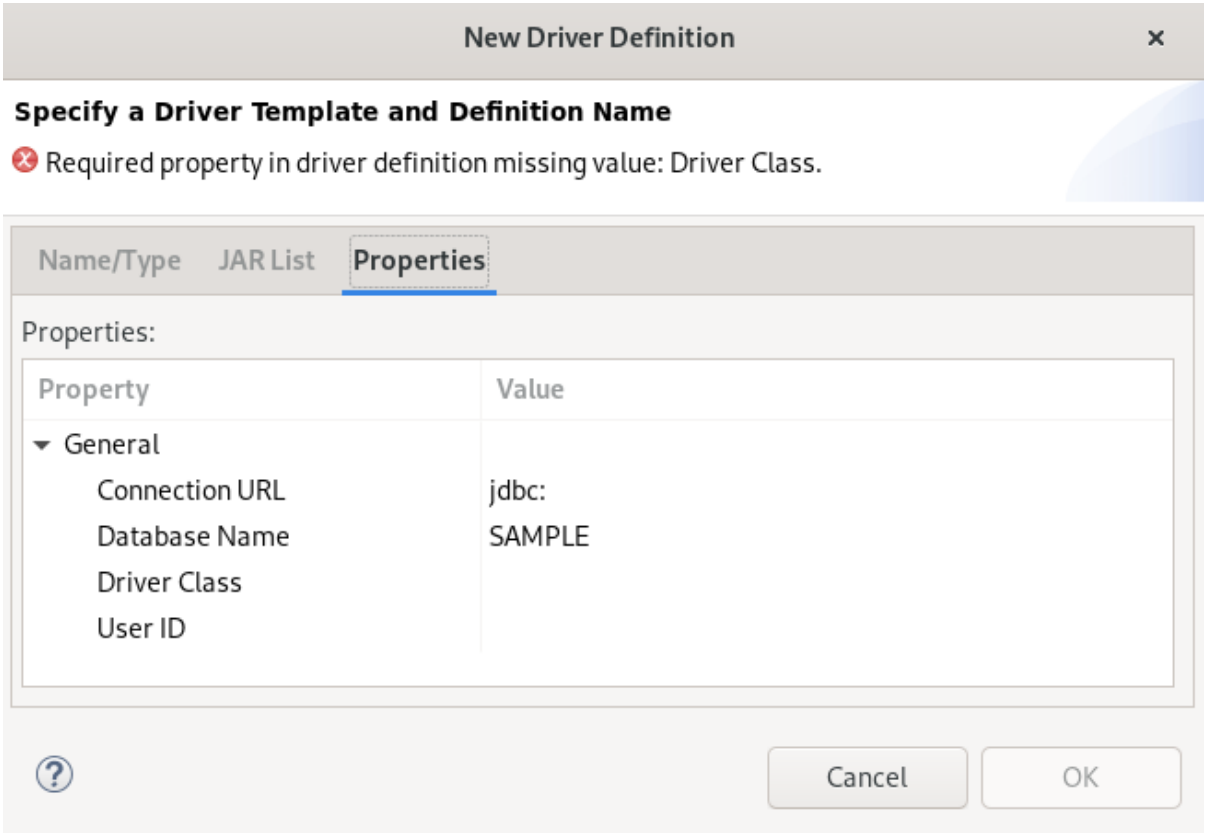


21. Generic JDBC Driver を選択します。

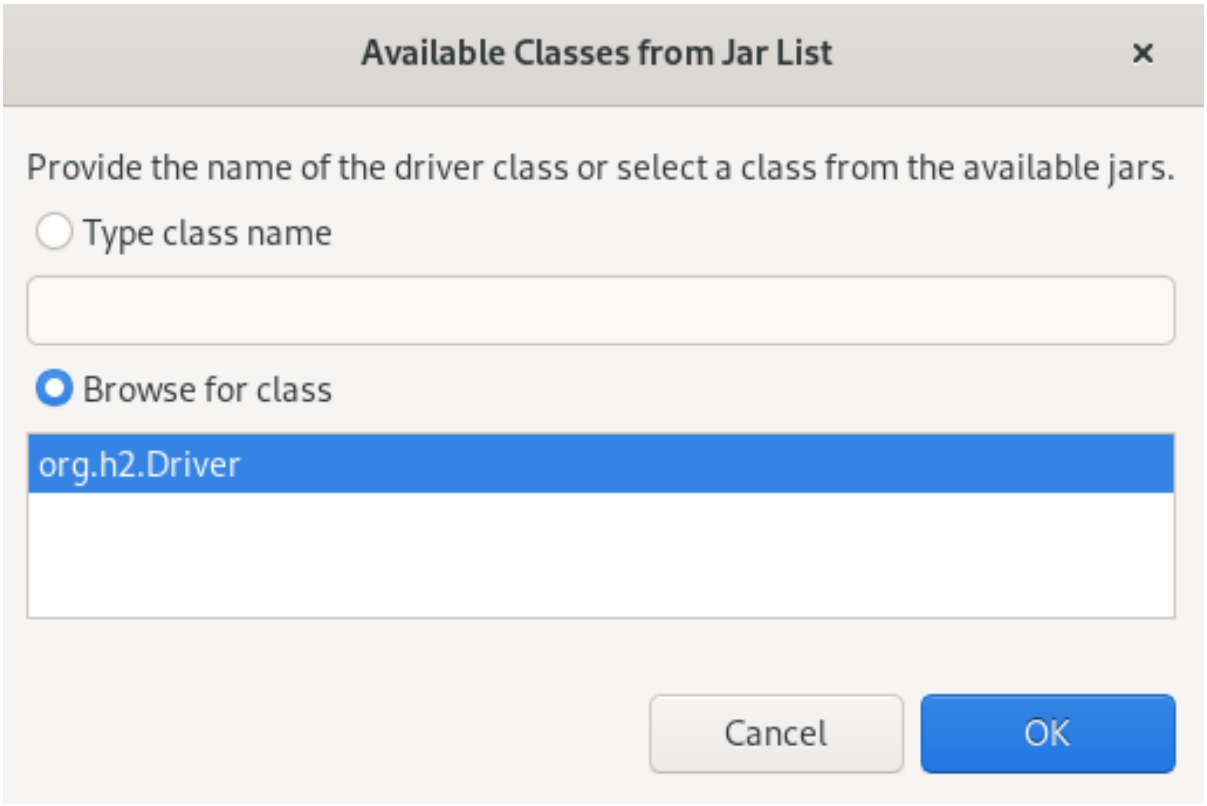
22. JAR List タブをクリックします。



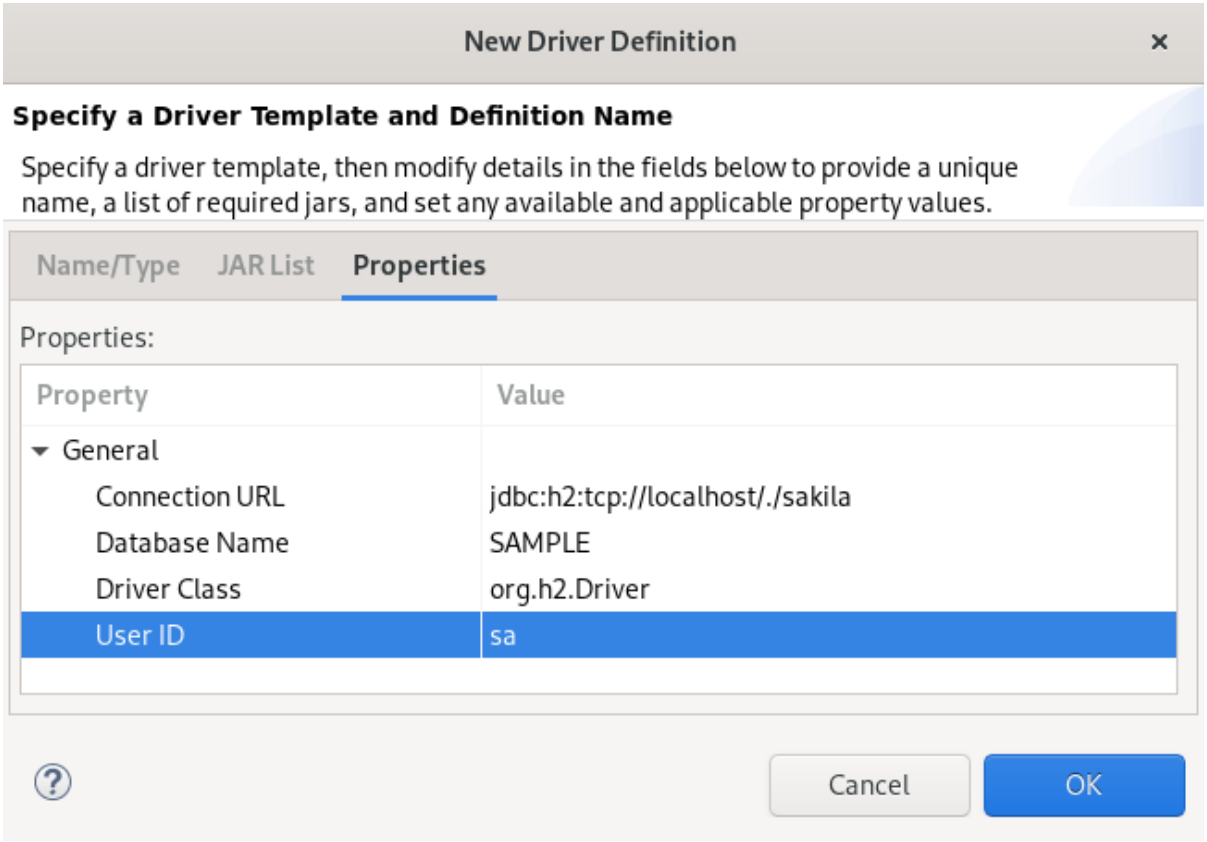
23. **Add JAR/Zip** ボタンをクリックします。
24. Sakila データベースの **.jar** ファイルを選択します。
25. **Properties** タブをクリックします。



26. **Connection URL** フィールドに **jdbc:h2:tcp://localhost/./sakila** を追加します。
27. **Driver Class** フィールドをクリックします。
28. **Driver Class** フィールドの末尾にある、点 3 つのアイコンをクリックします。
Available Classes from Jar Listウインドウが表示されます。



29. **Browse the Class**オプションを選択します。
30. **org.h2.Driver** を選択します。
31. **OK** をクリックします。
32. **User ID** フィールドに **sa** と入力します。



33. **OK** → **Finish** → **Finish** とクリックします。

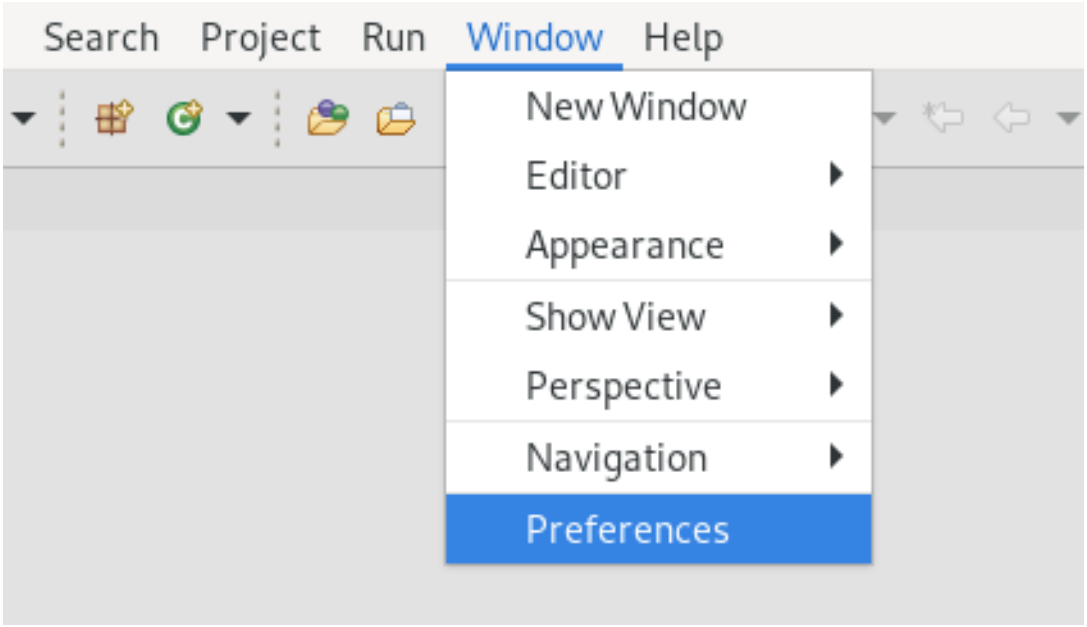
新たに作成された JPA プロジェクトが **Project Explorer** に表示されます。

7.2. ライブラリーの追加

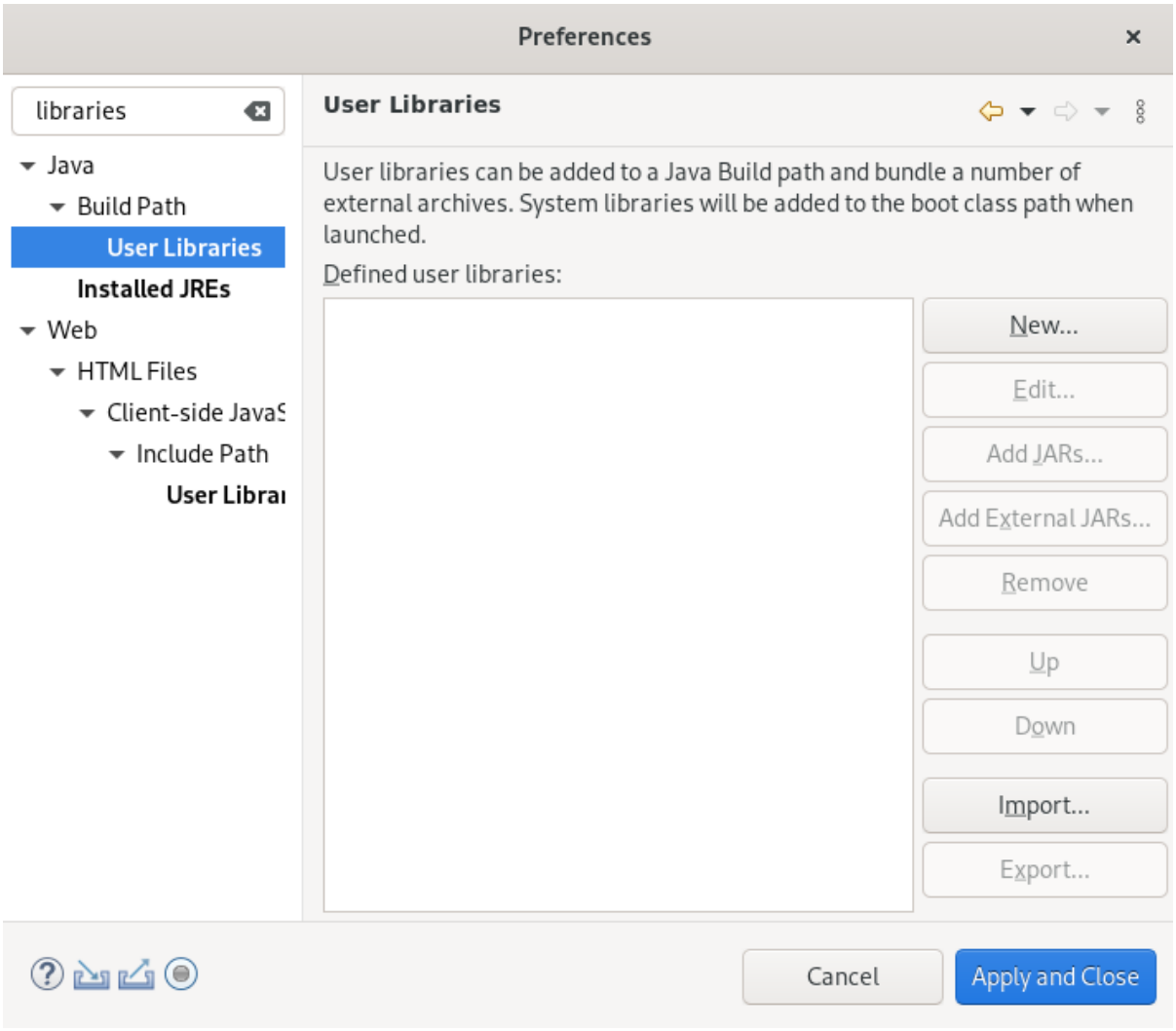
CodeReady Studio で、Hibernate プロジェクトにライブラリーを追加する方法を説明します。

手順

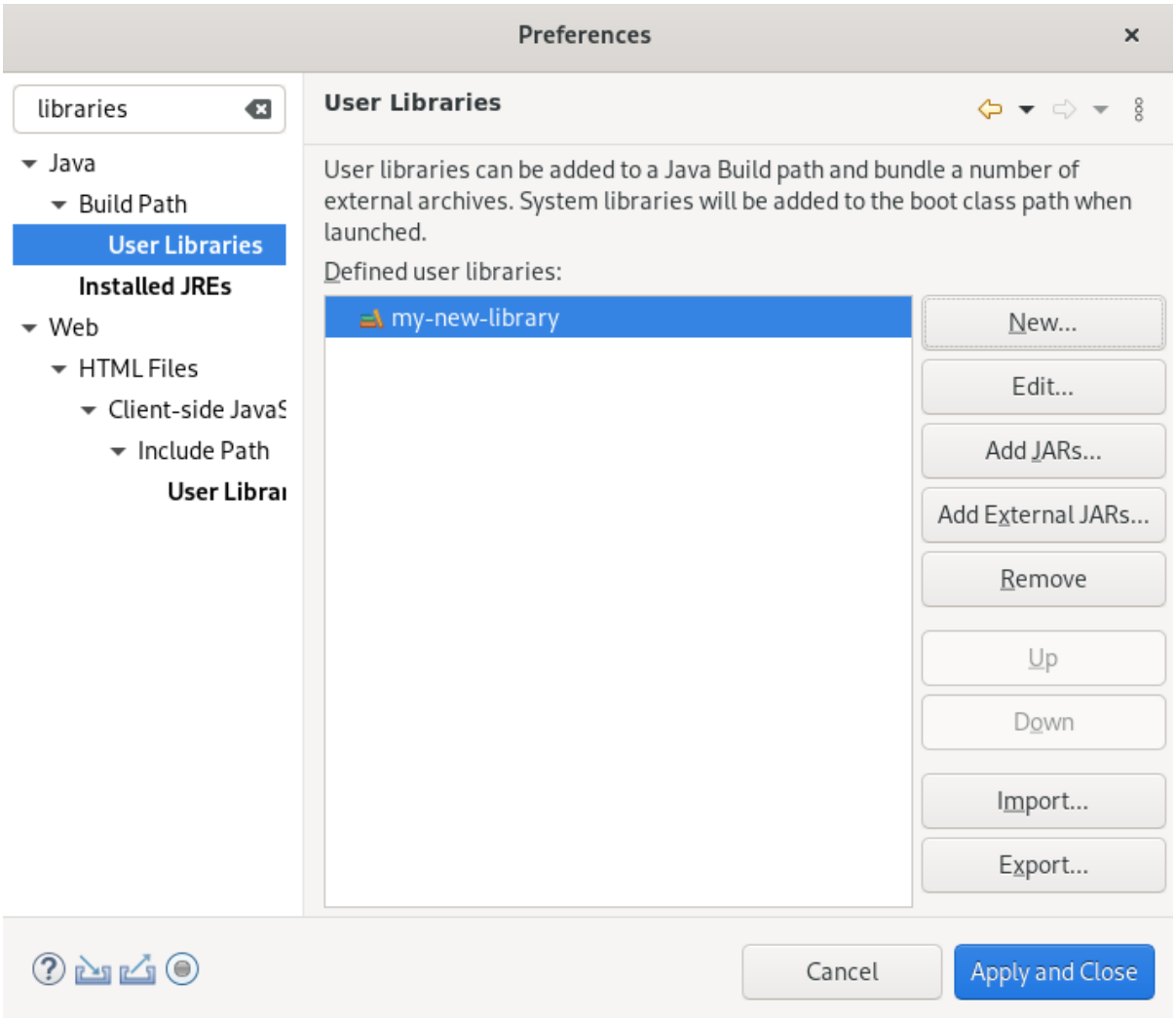
1. Hibernate ORM をダウンロードします。
2. ファイルを展開します。
3. CodeReady Studio を起動します。
4. **Window → Preferences** をクリックします。



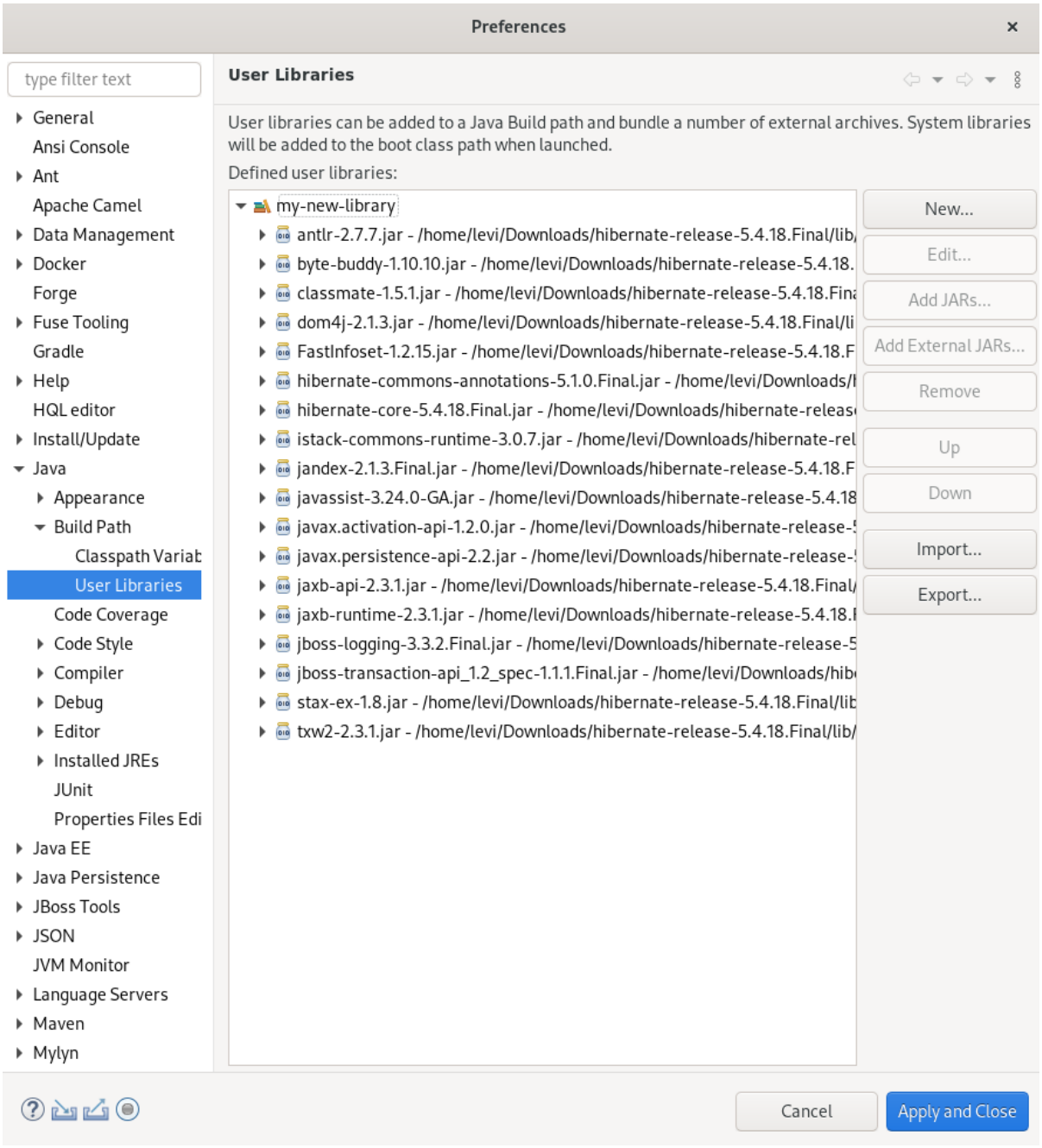
Preferences ウィンドウが表示されます。



5. 検索フィールドに **Libraries** と入力します。
6. **Java** で **User Libraries** を選択します。
7. **New** ボタンをクリックします。
   New User Library ウィンドウが表示されます。
8. ユーザーライブラリーに名前を付けます。
9. **OK** をクリックします。
10. 新しいユーザーライブラリーを選択します。



11. **Add External JARs** ボタンをクリックします。
12. Hibernate ORM ファイルを展開したディレクトリーを選択します。
13. **/lib/required/** ディレクトリーに移動します。
14. **.jar** ファイルを選択します。
15. **Open** をクリックします。
選択した **.jar** ファイルがユーザーライブラリーに表示されます。



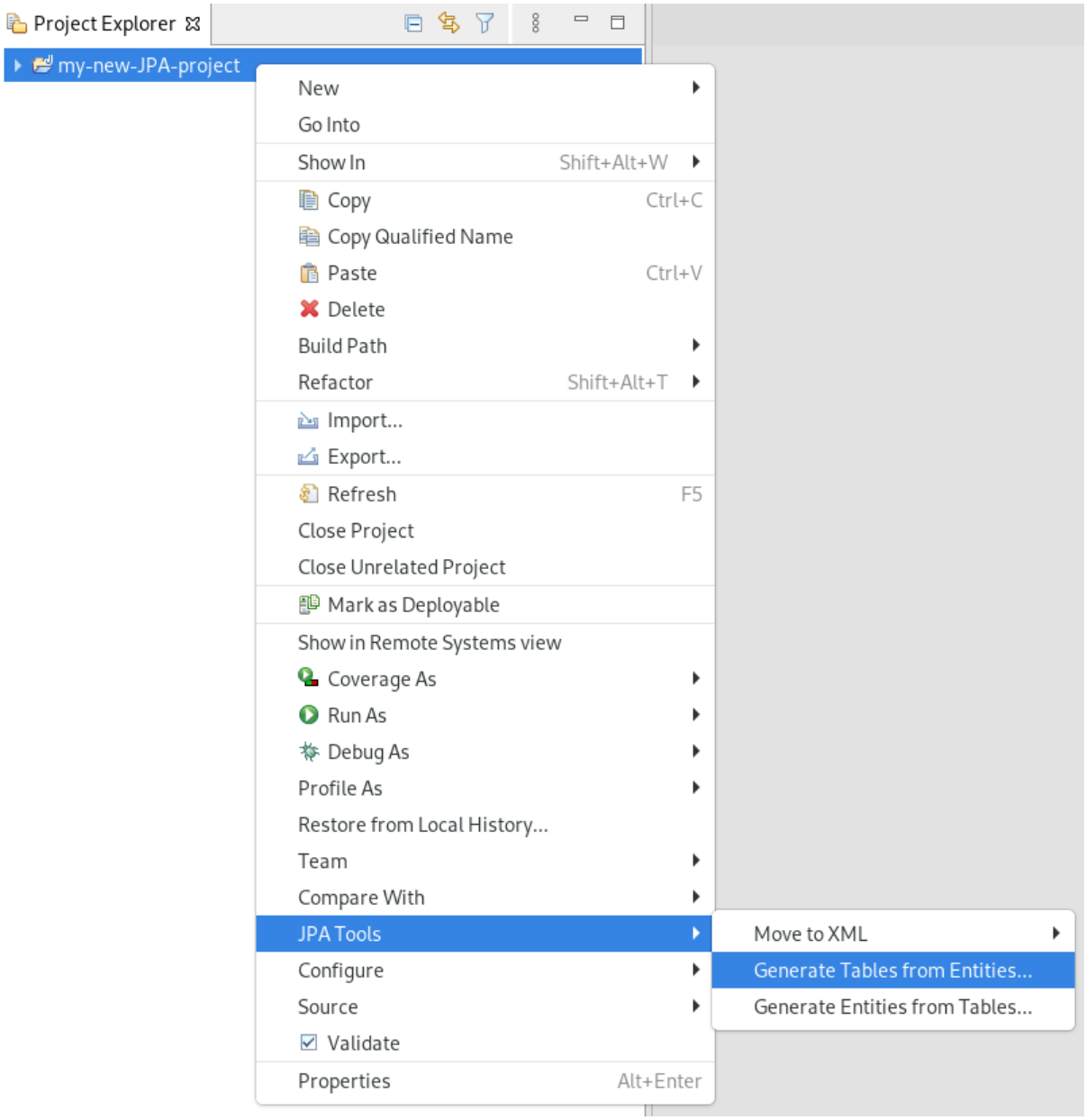
16. **Apply and Close** をクリックします。

7.3. エンティティーの生成

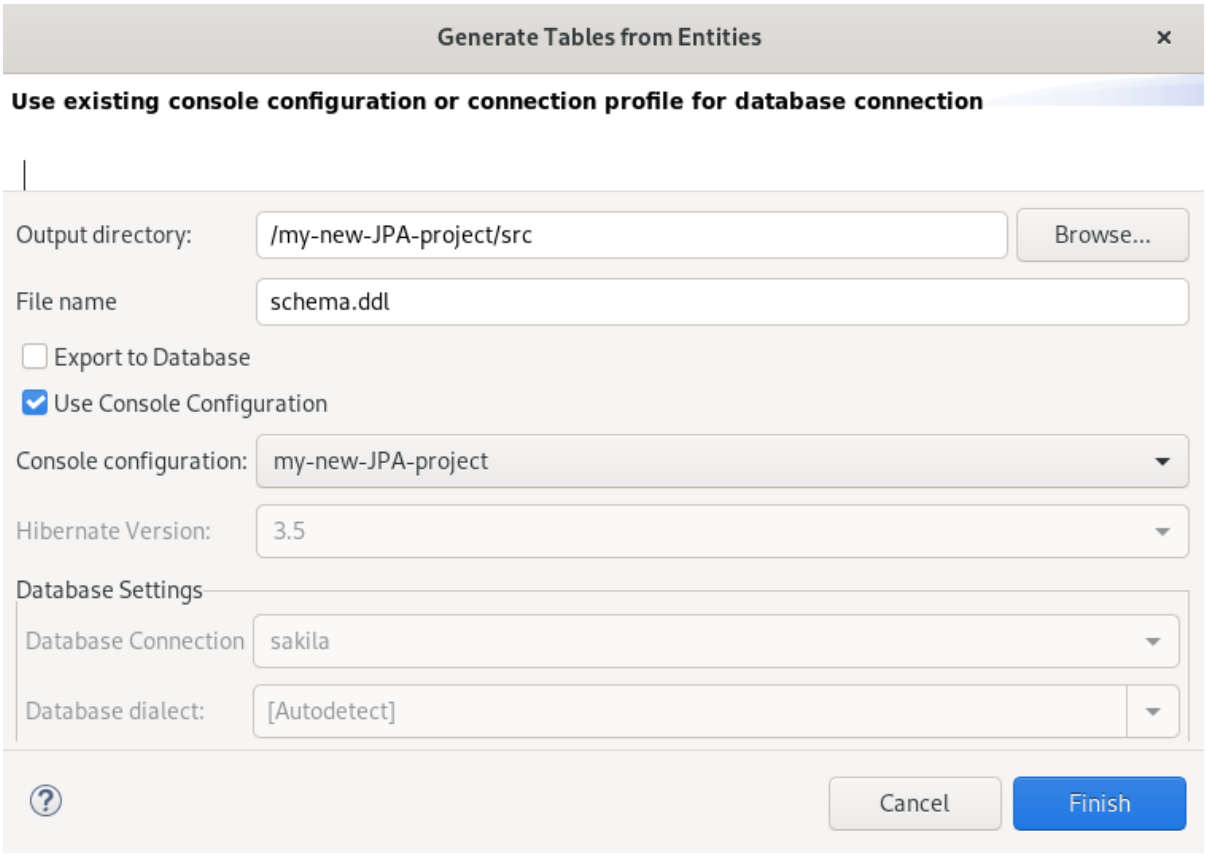
CodeReady Studio で Hibernate プロジェクトのエンティティーを生成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Project Explorer** を開きます。
3. **JPA project → JPA Tools → Generate Tables from Entities** を右クリックします。



Generate Table from Entities ウィンドウが表示されます。



4. **Use Console Configuration** チェックボックスを選択します。
5. **Finish** をクリックします。

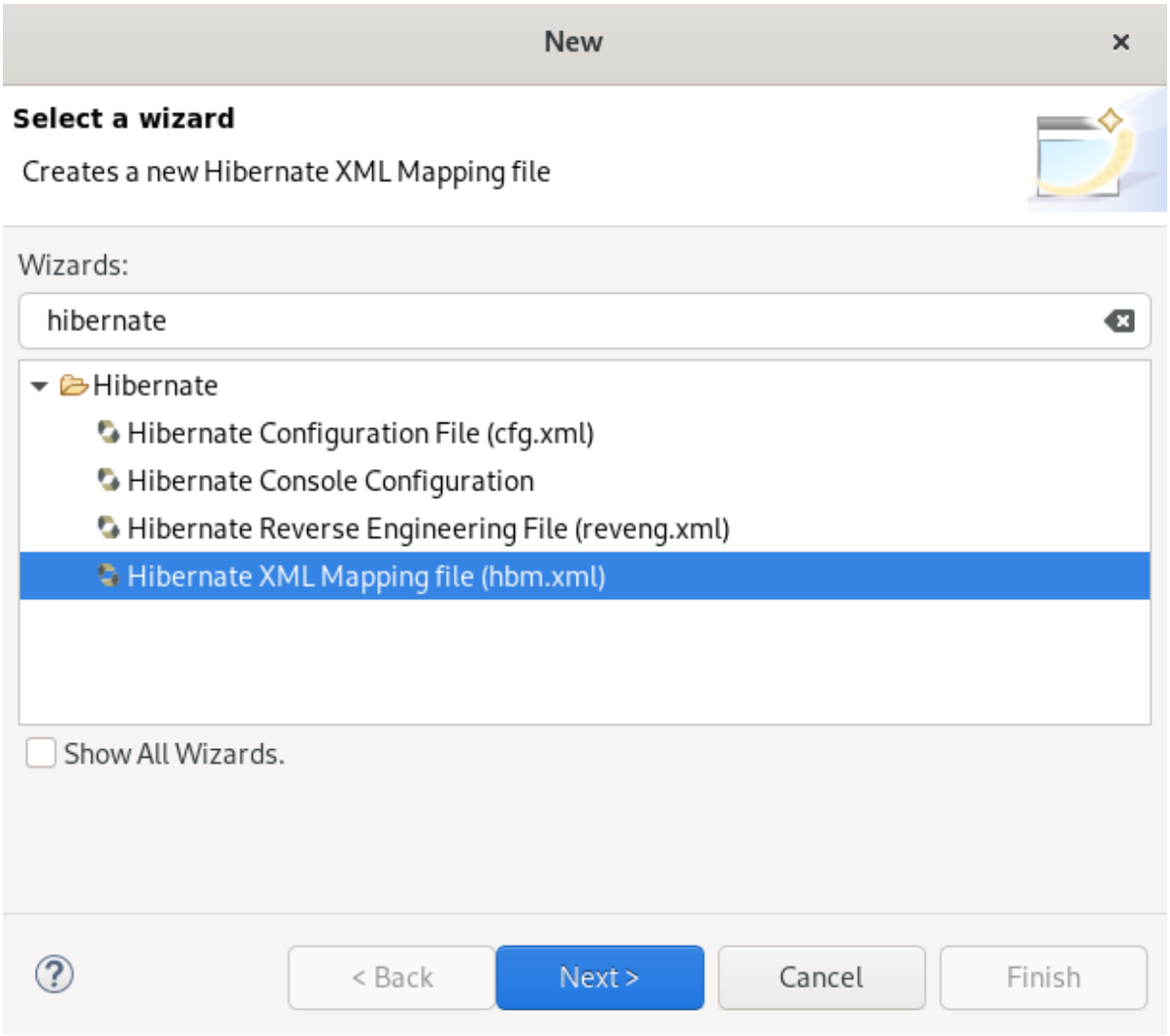
7.4. HIBERNATE マッピングファイルの作成

Hibernate マッピングファイルは、オブジェクトがどのようにデータベーステーブルに関連するかを指定します。

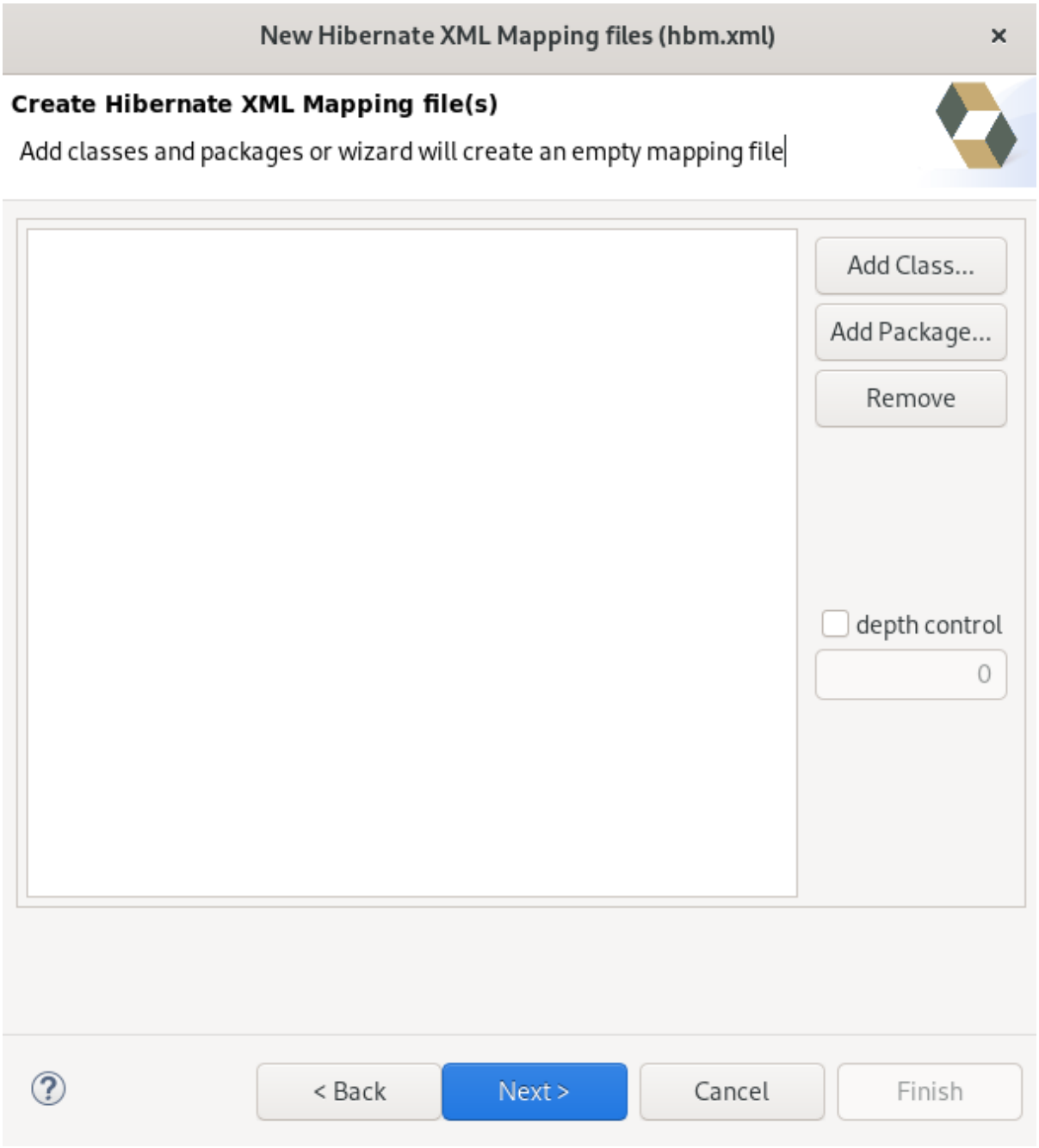
CodeReady Studio で Hibernate マッピングファイルを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Ctr+N** キーを押します。
   Select a wizard ウインドウが表示されます。



3. 検索フィールドに **Hibernate** と入力します。
4. **Hibernate XML Mapping file (hbm.xml)** を選択します。
5. **Next** をクリックします。
Create Hibernate XML Mapping fileウインドウが表示されます。

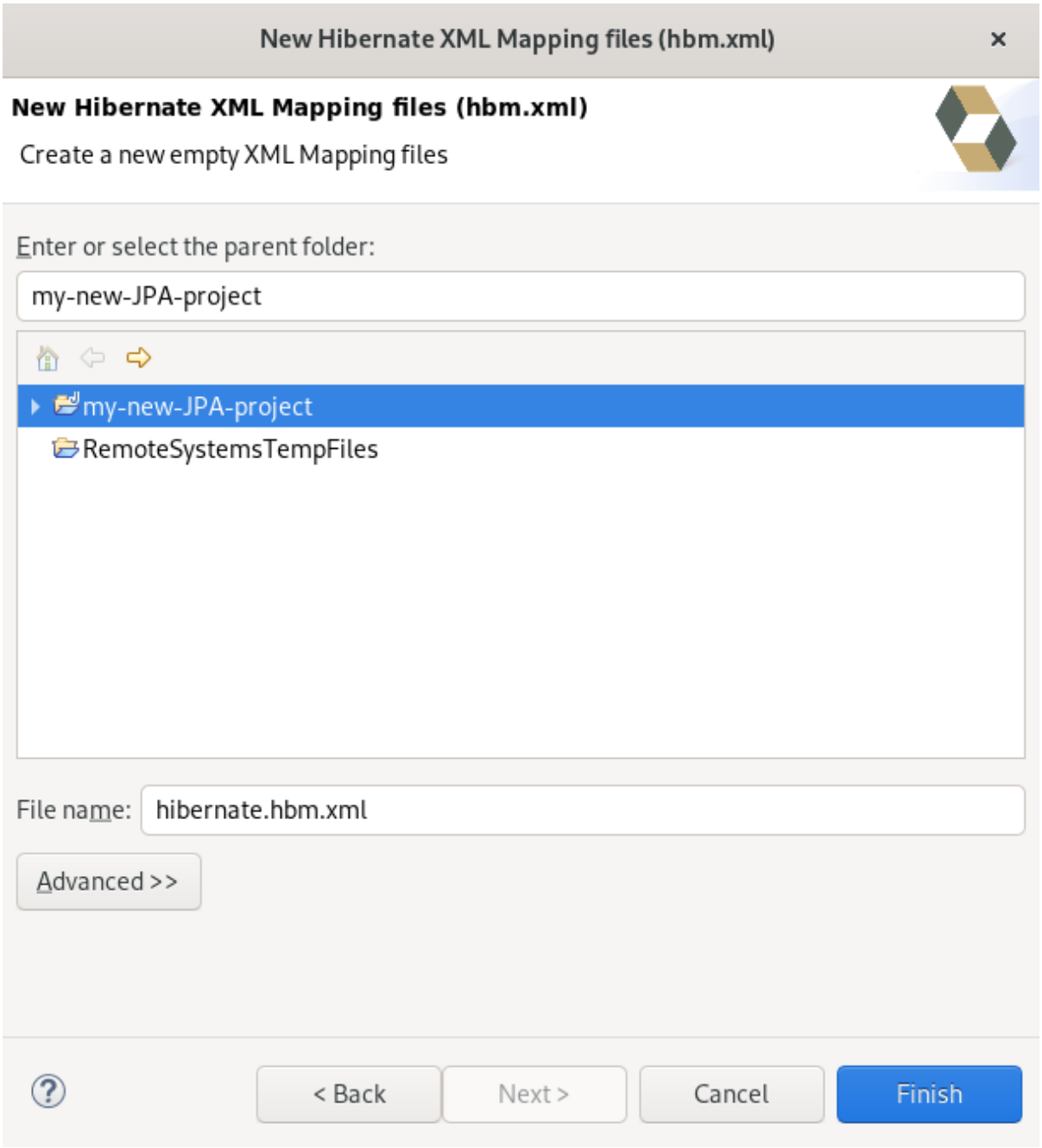


6. **Add Class** ボタンをクリックしてクラスを追加します。

7. **Add Package** ボタンをクリックしてパッケージを追加します。
   または、パッケージまたはクラスを選択せずに空の **.hbm.xml** ファイルを作成することもできます。

8. **depth control** チェックボックスを選択し、クラスの選択時に使用される依存関係の深さを定義します。

9. **Next** をクリックします。
   New Hibernate XML Mapping files ウインドウが表示されます。



10. 親ディレクトリーを選択します。
11. **.hbm.xml** ファイルに名前を付けます。
12. **Finish** をクリックします。

7.5. HIBERNATE 設定ファイルの作成

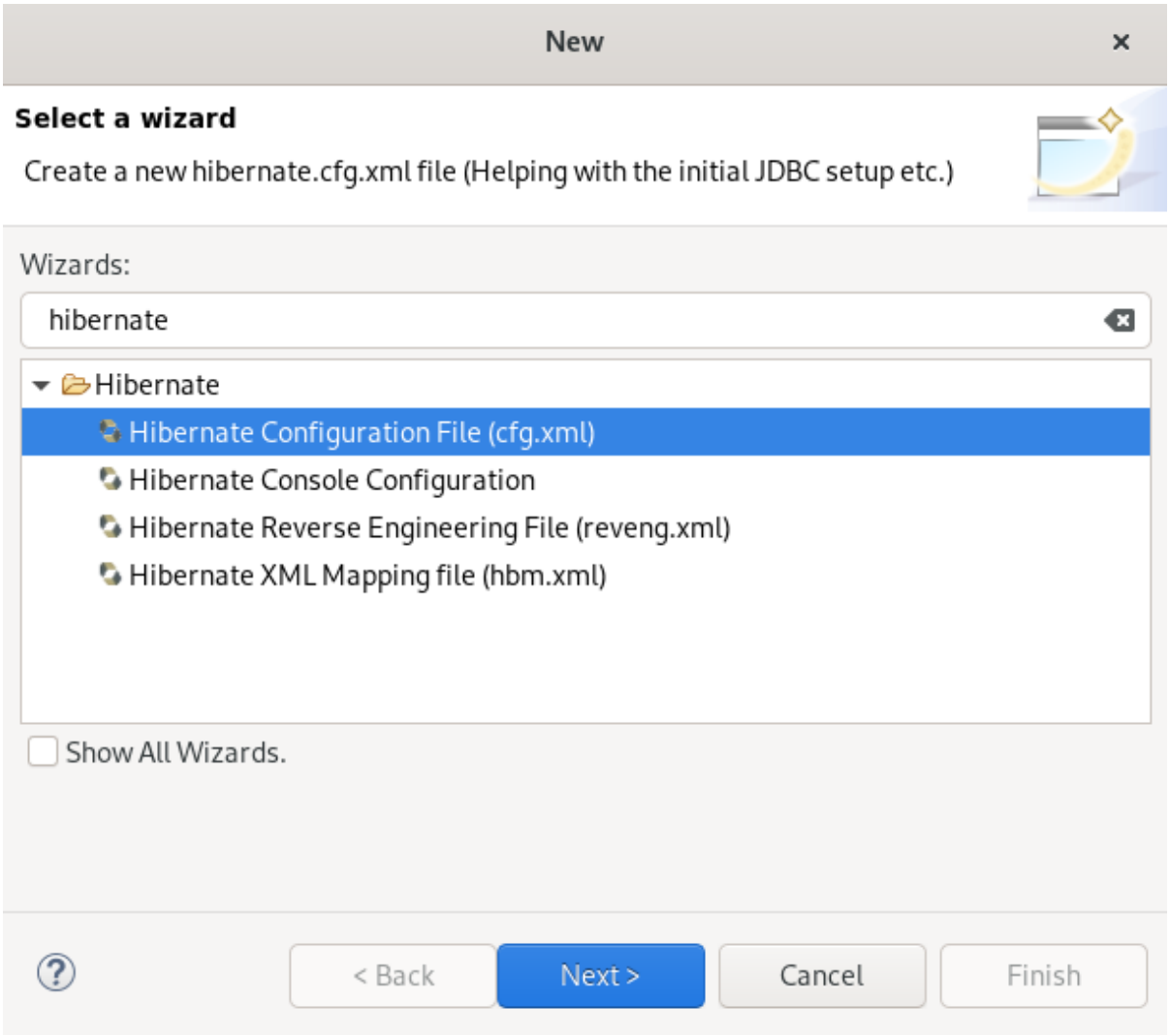
リバースエンジニアリング、プロトタイプクエリー、または Hibernate Core の使用には、**hibernate.properties** または **hibernate.cfg.xml** ファイルが必要になります。CodeReady Studio には、**hibernate.cfg.xml** ファイルを生成するウィザードがあります。

CodeReady Studio で Hibernate 設定ファイルを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。

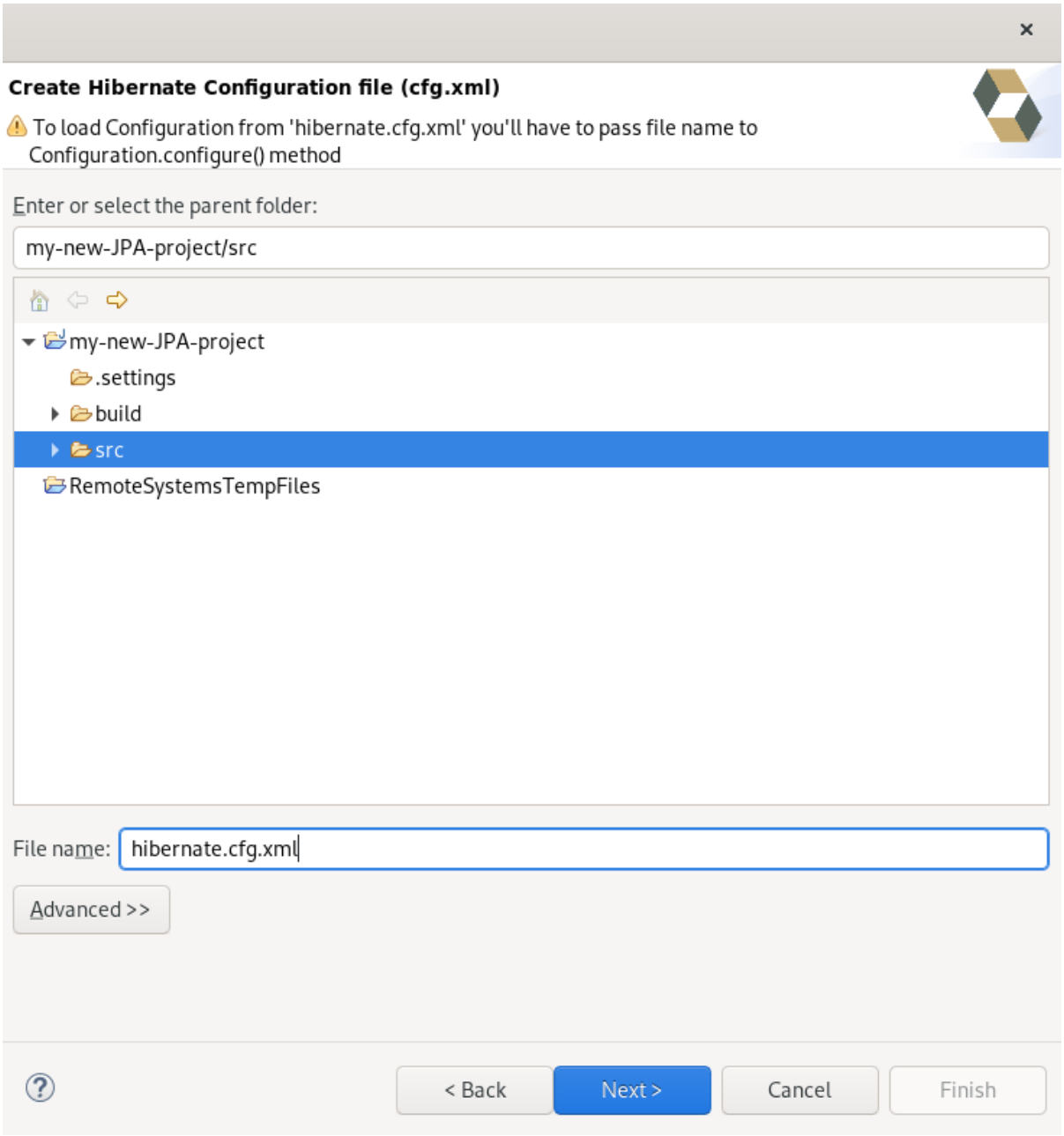
2. **Ctr+N** キーを押します。
Select a wizard ウインドウが表示されます。



3. 検索フィールドに **Hibernate** と入力します。

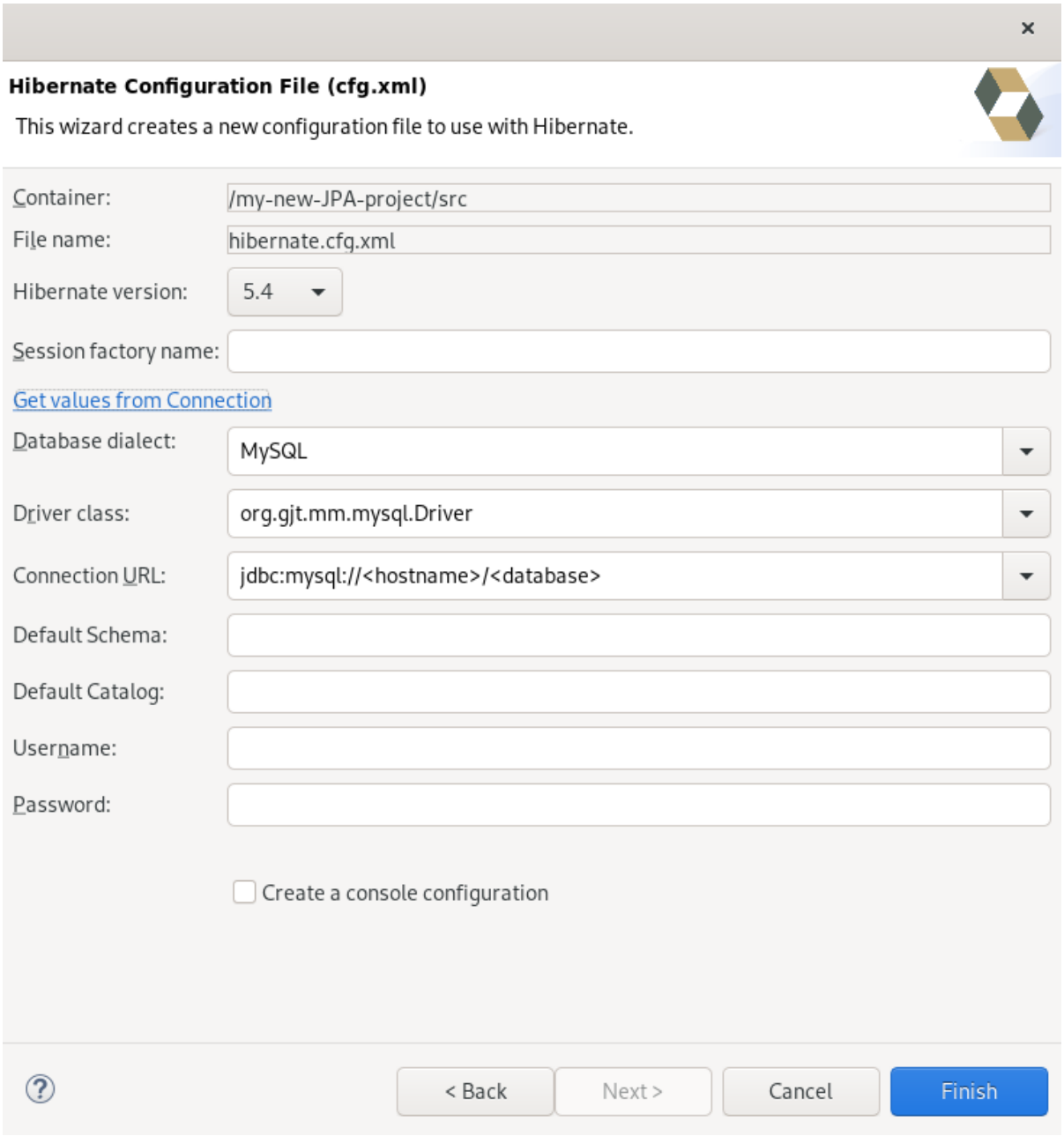
4. **Hibernate Configuration file (cfg.xml)** を選択します。

5. **Next** をクリックします。
Create Hibernate Configuration file (cfg.xml) ウインドウが表示されます。



6. 親ディレクトリーを選択します。

7. **Next** をクリックします。
Hibernate Configuration File(cfg.xml) ウインドウが表示されます。



8. **Database dialect** フィールドの下矢印をクリックして、データベースを選択します。
9. **Driver class** フィールドの下矢印をクリックして、ドライバーを選択します。
10. **Connection URL** フィールドの下矢印をクリックし、URL を選択します。
11. **Finish** をクリックします。

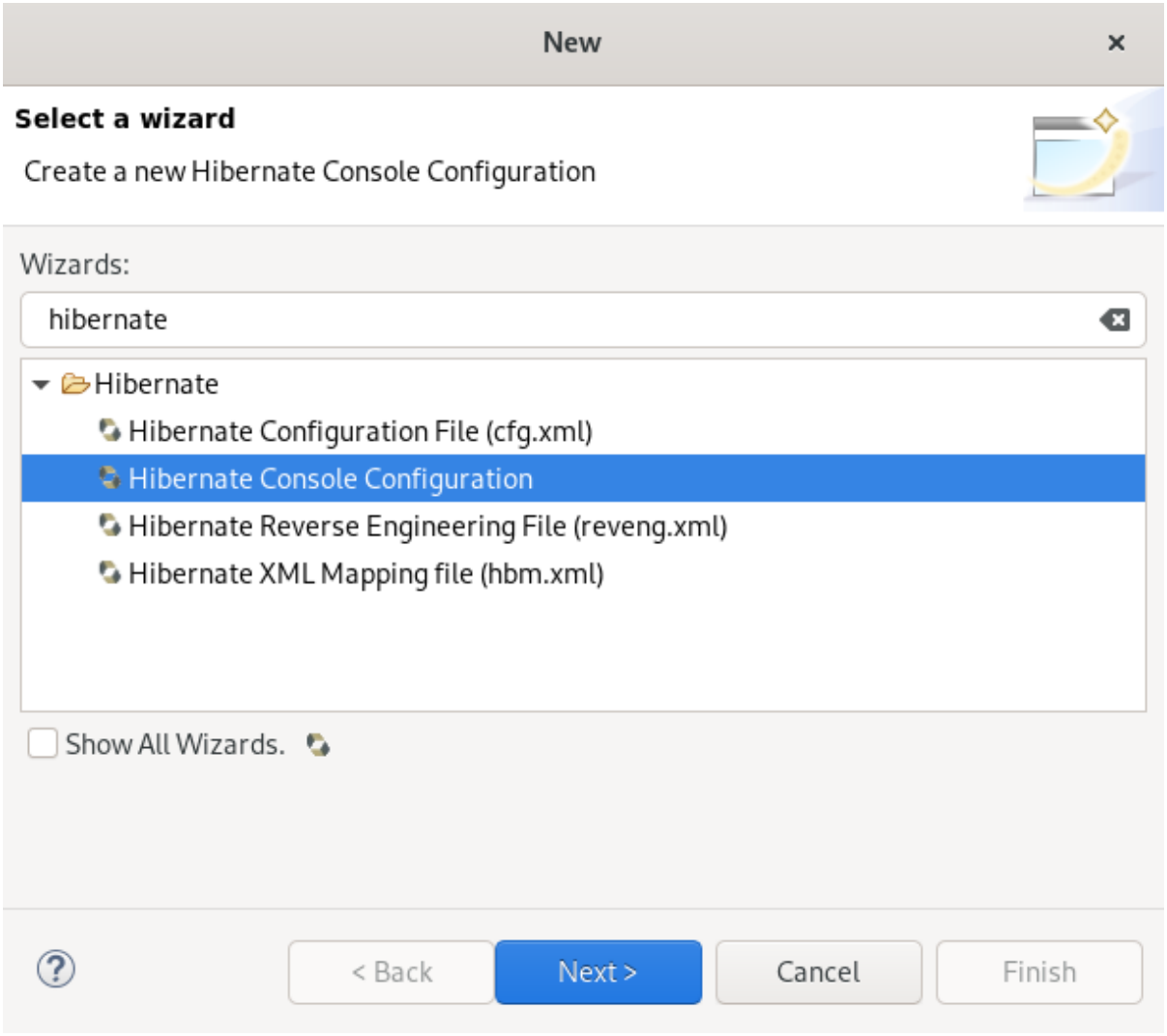
7.6. HIBERNATE コンソール設定ファイルの作成

コンソール設定ファイルには、Hibernate プラグインがどのように Hibernate を設定するかが記述されています。また、POJO や JDBC ドライバーなどのロードに必要な設定ファイルおよびクラスパスも記述されています。これは、クエリープロトタイピング、リバースエンジニアリング、およびコードの生成を使用するために必要です。プロジェクトごとに複数のコンソール設定を指定できますが、1 つの設定で十分です。

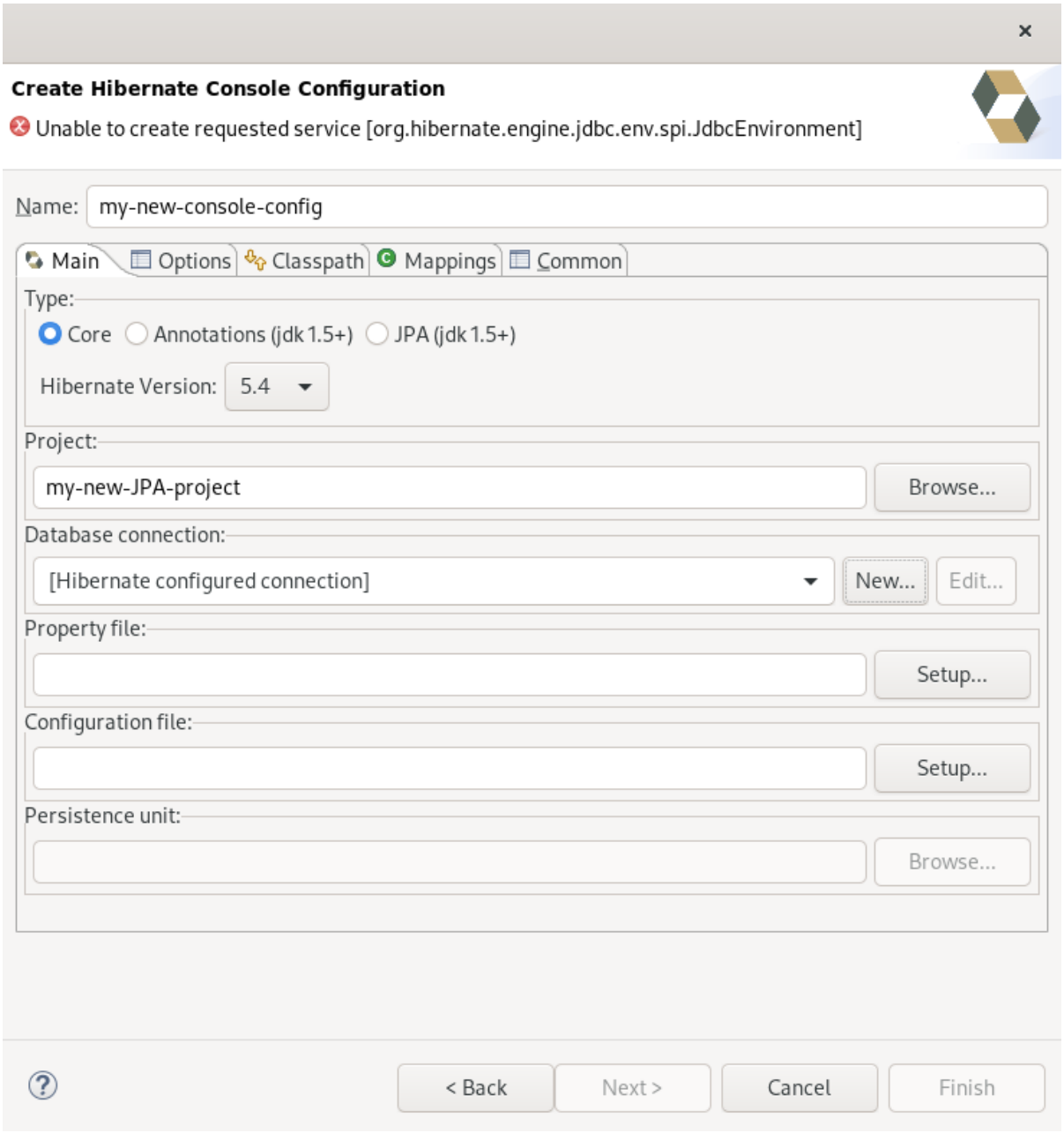
CodeReady Studio で Hibernate コンソール設定ファイルを作成する方法を説明します。

手順

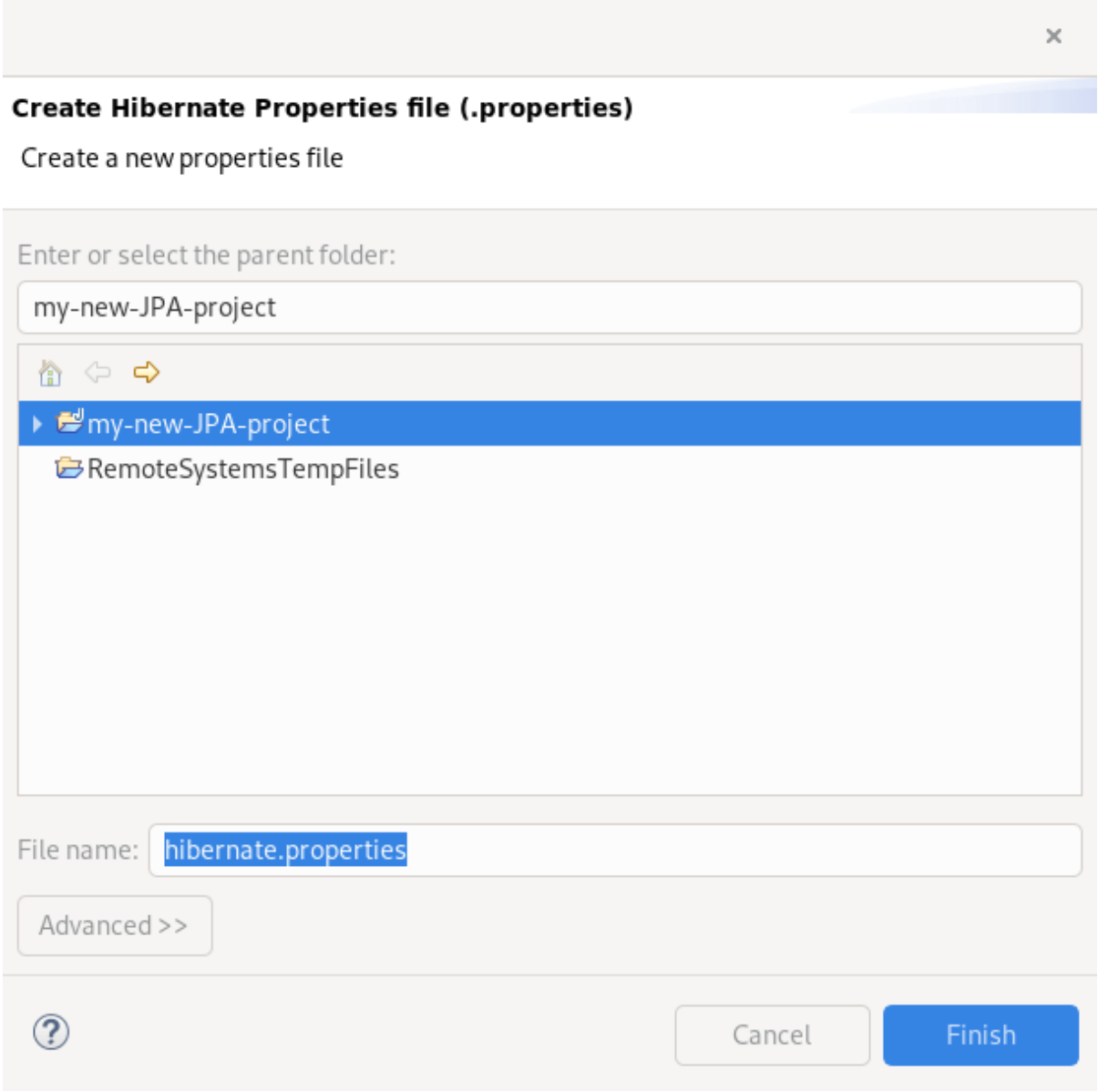
1. CodeReady Studio を起動します。
2. **Ctr+N** キーを押します。
   Select a wizard ウインドウが表示されます。



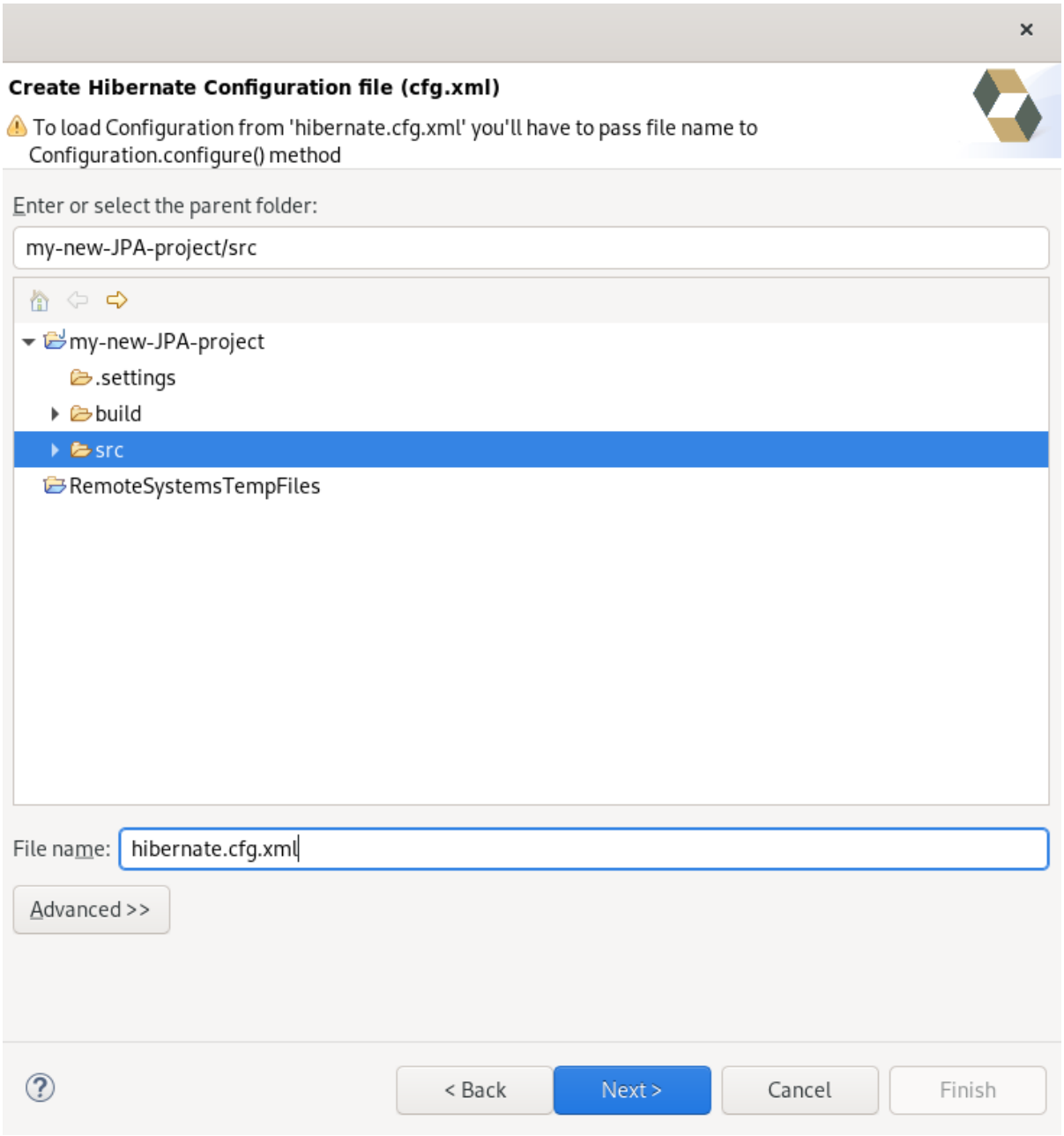
3. 検索フィールドに **Hibernate** と入力します。
4. **Hibernate Console Configuration** を選択します。
5. **Next** をクリックします。
   Create Hibernate Console Configuration ウィンドウが表示されます。



6. 設定ファイルに名前を付けます。
7. **Type** が **Core** に設定されていることを確認します。
8. **Hibernate version** を選択します。
9. **Browse** をクリックしてプロジェクトを見つけます。
10. **New** をクリックして、新しい **Database connection** を設定します。
New Connection Profile ウインドウが表示されます。
11. **Data Connection** を選択するか、新たに作成します。
12. **Setup** をクリックし、**Property file** を設定します。
Setup property file ウインドウを表示します。
13. **Create new** をクリックします。
Create Hibernate Properties file (.properties) ウインドウが表示されます。

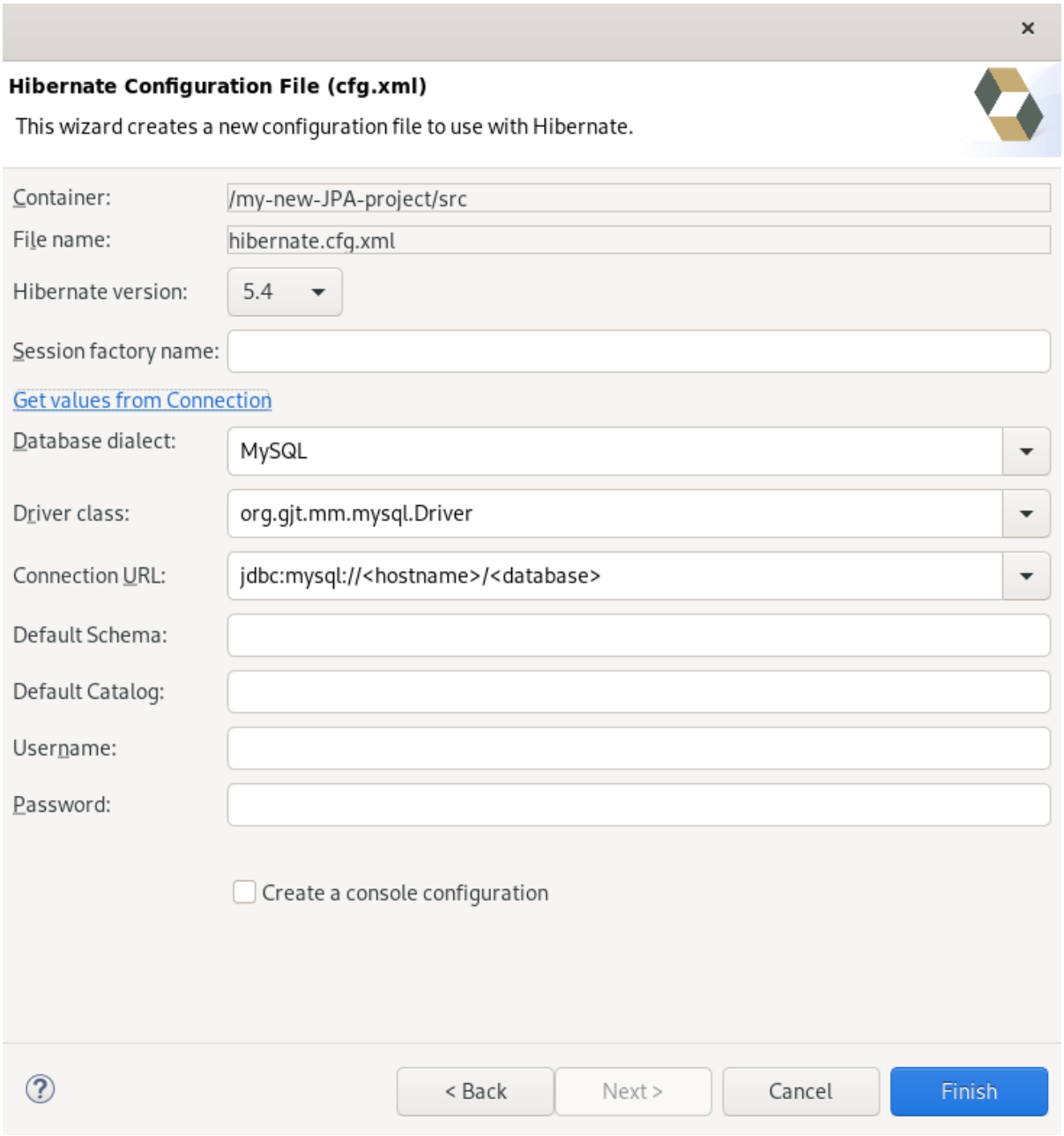


14. 親ディレクトリーを選択します。
15. **.properties** ファイルに名前を付けます。
16. **Finish** をクリックします。
17. **Setup** をクリックし、**Configuration file** を設定します。
18. ターゲット **.cfg.xml** ファイルへのパスを選択します。
    Setup configuration file ウインドウが表示されます。
19. **Create new** をクリックします。
    Create Hibernate Configuration file (cfg.xml)ウインドウが表示されます。

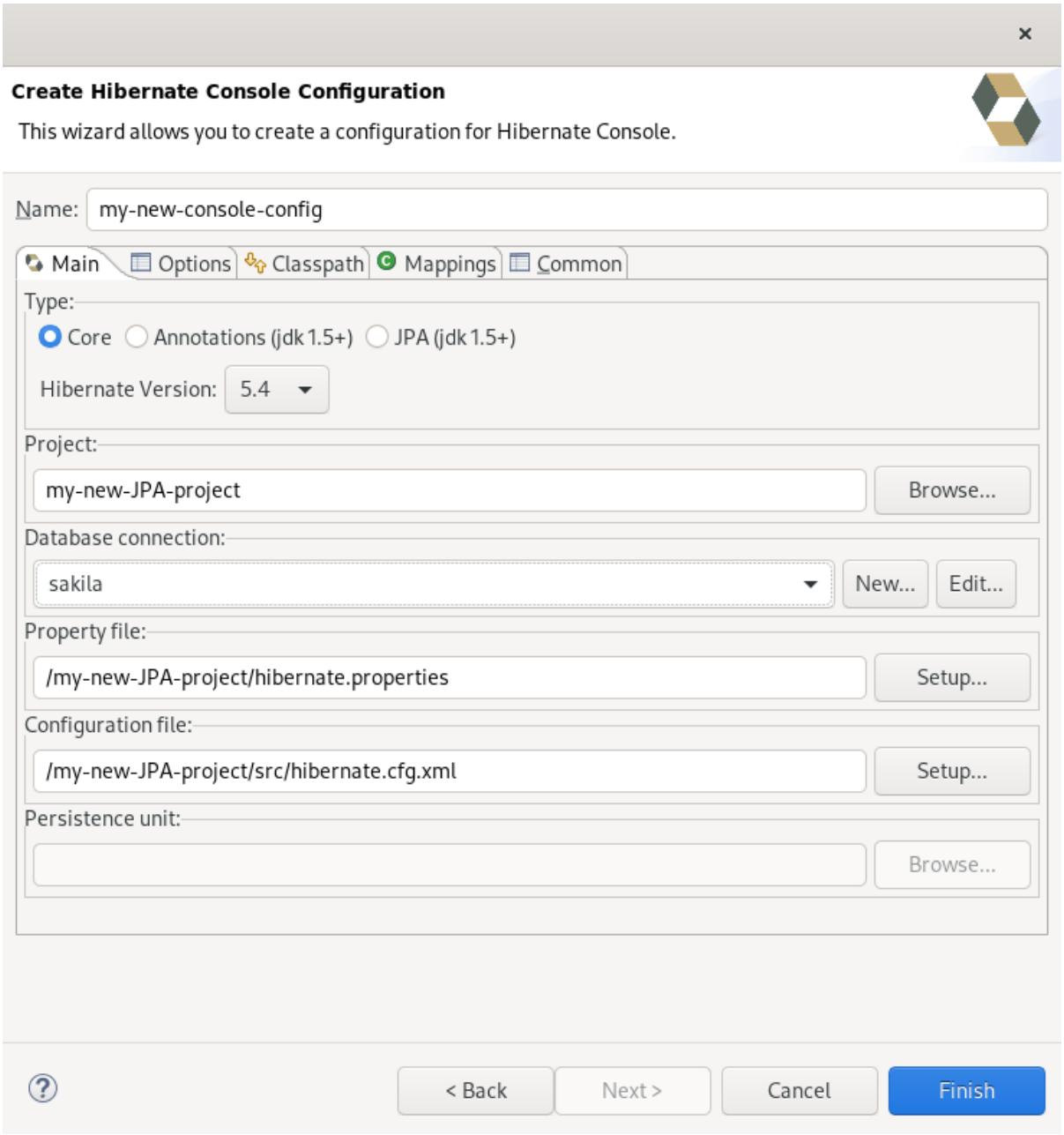


20. 親ディレクトリーを選択します。

21. **Next** をクリックします。
Hibernate Configuration File(cfg.xml) ウインドウが表示されます。



22. **Database dialect** フィールドの下矢印をクリックして、データベースを選択します。
23. **Driver class** フィールドの下矢印をクリックして、ドライバーを選択します。
24. **Connection URL** フィールドの下矢印をクリックし、URL を選択します。
25. **Finish** をクリックします。



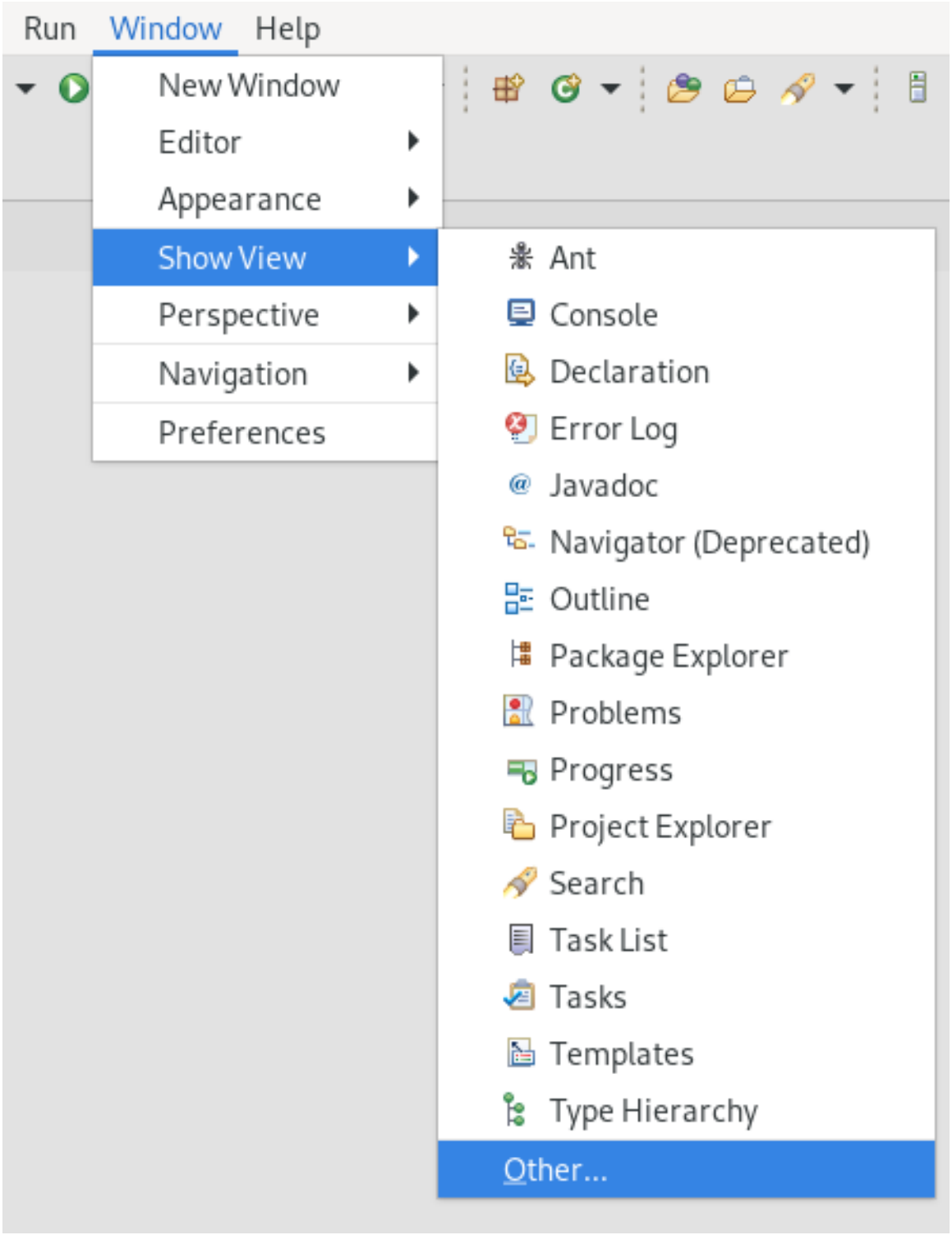
26. **Finish** をクリックします。

7.7. HIBERNATE プロジェクト設定の編集

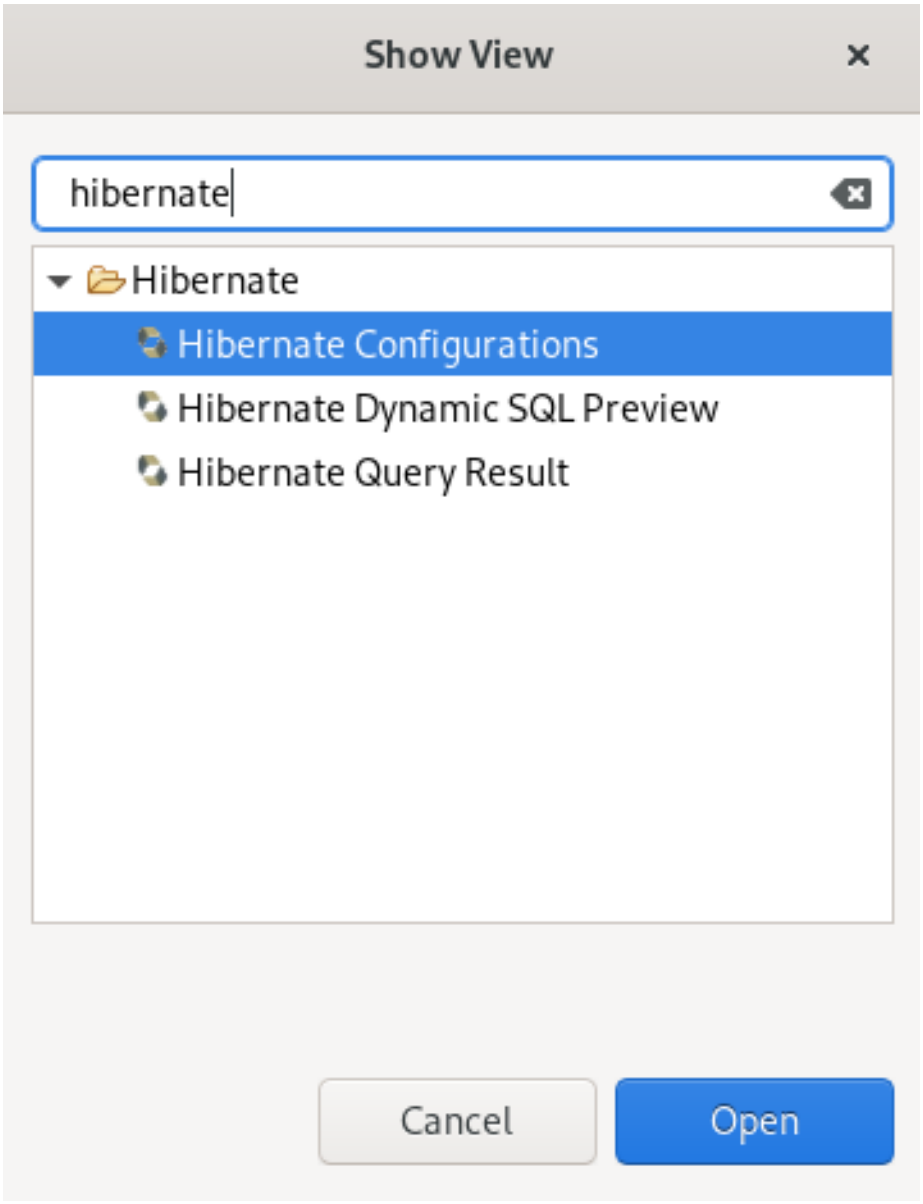
CodeReady Studio で Hibernate プロジェクトの設定を編集する方法を説明します。

手順

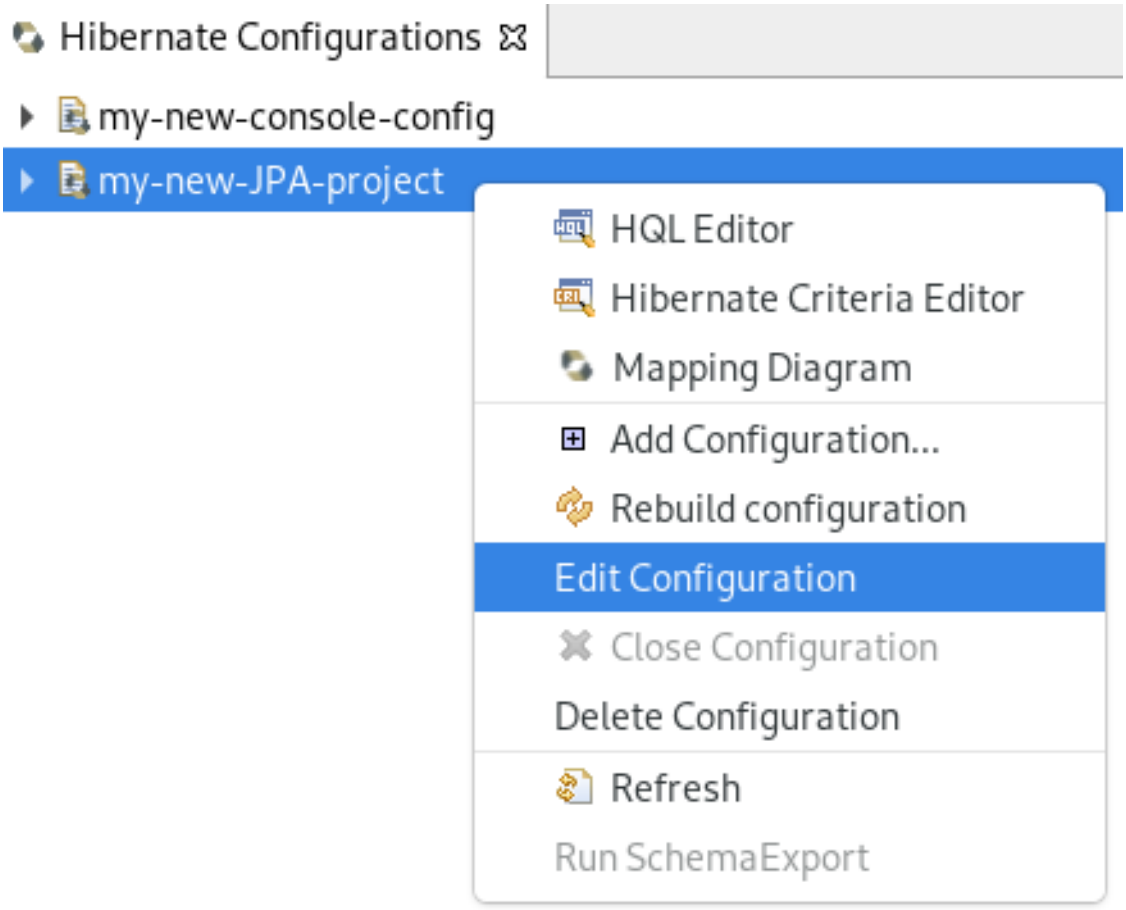
1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。



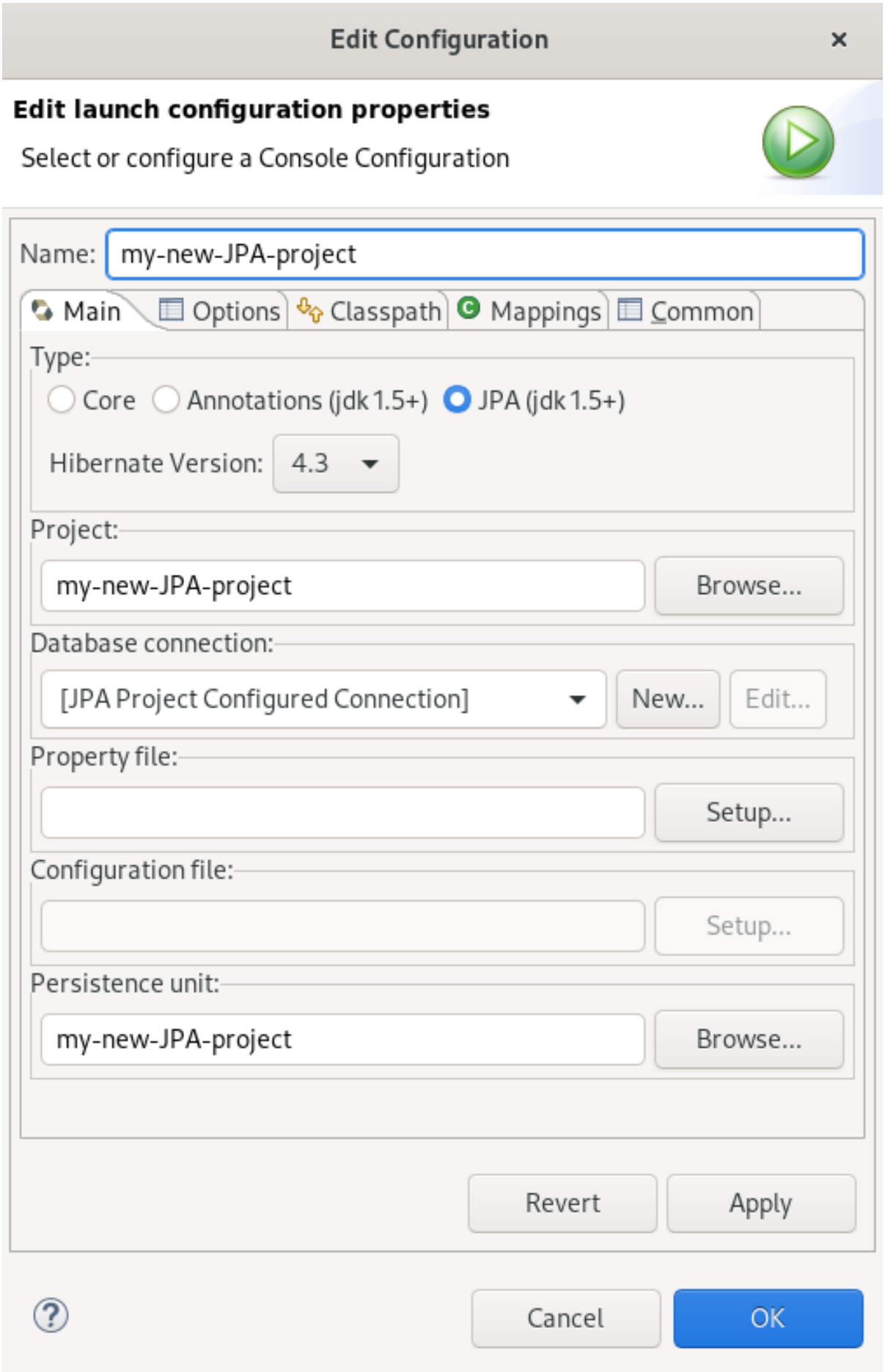
Show View ウインドウが表示されます。



3. 検索フィールドに **Hibernate** と入力します。
4. **Hibernate Configurations** を選択します。
5. **Open** をクリックします。
Hibernate Configurations ビューが表示されます。



6. **project → Edit Configuration** を右クリックします。
Edit Configuration ウインドウが表示されます。



7. 設定を編集します。
8. **Apply** をクリックします。

9. **OK** をクリックします。

第8章 CODEREADY STUDIO での MOBILE WEB TOOLS の基本

Mobile Web Tools は、モバイルデバイス向けに最適化された Web アプリケーションの作成を可能にする **HTML5 Project** ウィザードを提供します。**HTML5 Project** ウィザードは、IDE ですべての新規 HTML5 Web アプリケーションを作成するための便利なウィザードです。このウィザードは、Maven archetype からの REST リソースを使用して、そのままデプロイできる HTML5 モバイルアプリケーションのサンプルを生成します。

組み込みエディターを使用してアプリケーションをカスタマイズでき、組み込みブラウザーでアプリケーションをデプロイおよび表示できます。

IDE では、ユーザーが対話式の Web アプリケーションを作成できる **Mobile Web** パレットが提供されます。このパレットは、HTML5、jQuery Mobile、および Ionic タグなどの一般的な Web インターフェースフレームワーク機能を html ファイルに追加するドラッグアンドドロップウィジェットなど、幅広い機能を提供します。また、よりユーザーフレンドリーで効率的なアプリケーションを実現するための、**Panels**、**Pages**、**Lists**、**Buttons** などのウィジェットも含まれています。

前提条件

- 設定済みのサーバー。
  ローカルランタイムサーバーの設定およびアプリケーションのローカルランタイムサーバーへのデプロイメントに関する詳細は、「ローカルサーバーの設定」 を参照してください。

IDE は、アプリケーションをデプロイするすべてのサーバーに対して設定する必要があります。これには、アプリケーションサーバーの場所およびタイプ、カスタム設定、カスタム管理設定などが含まれます。ここでは、事前に設定が完了していることを前提としていますが、デプロイメント時に手順を完了することもできます。

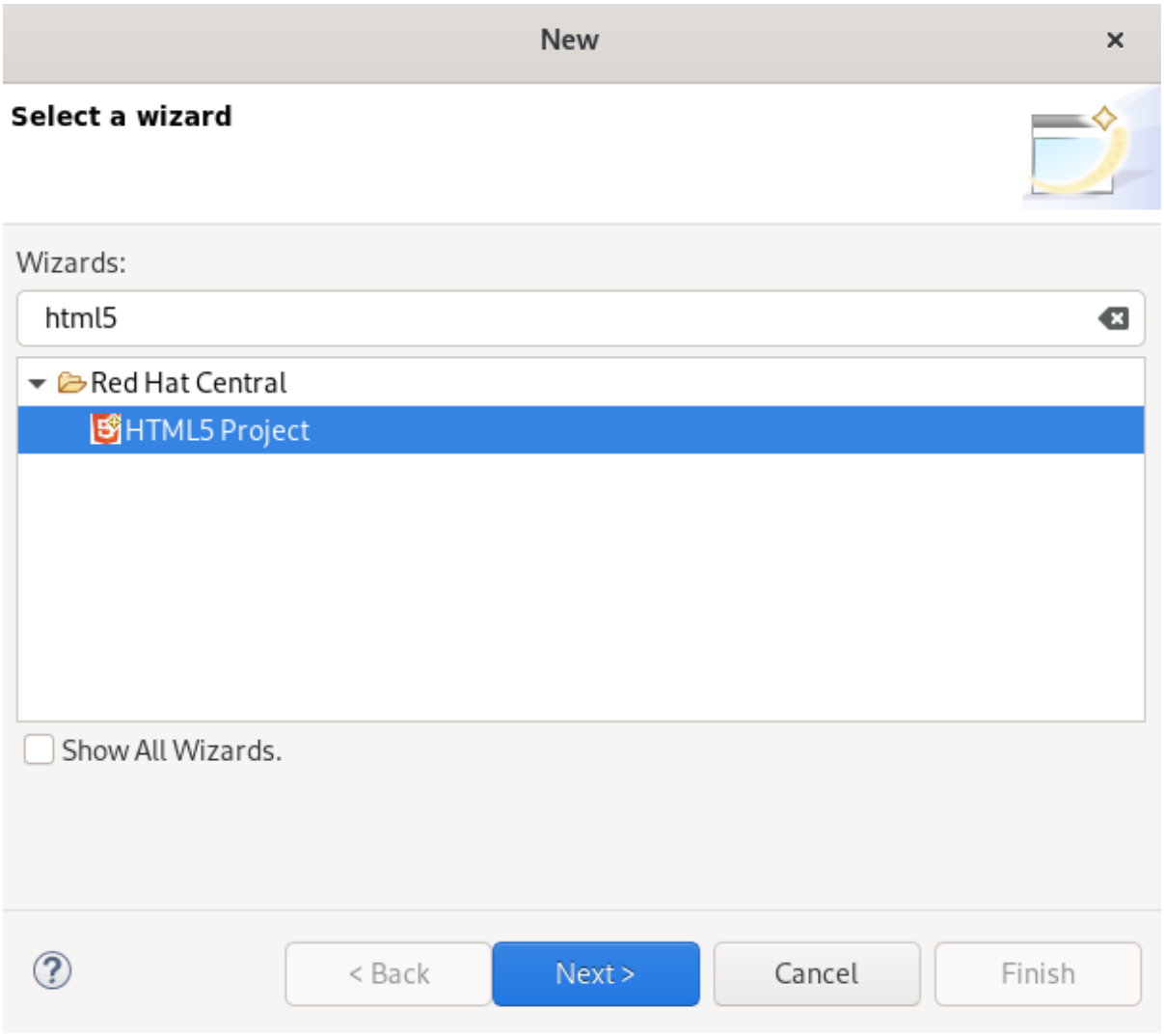
8.1. HTML5 プロジェクトの作成

HTML5 Project ウィザードは、Maven アーキタイプと、提供するプロジェクトおよびアプリケーション識別子を基にして、サンプルプロジェクトを生成します。Maven archetype バージョンは、ウィザードの最初のページにある **Description** フィールドに示されます。バージョンを変更できるため、ウィザード内でエンタープライズまたは非エンタープライズいずれかのターゲットランタイムを選択して、プロジェクトの見た目と依存関係も変更できます。

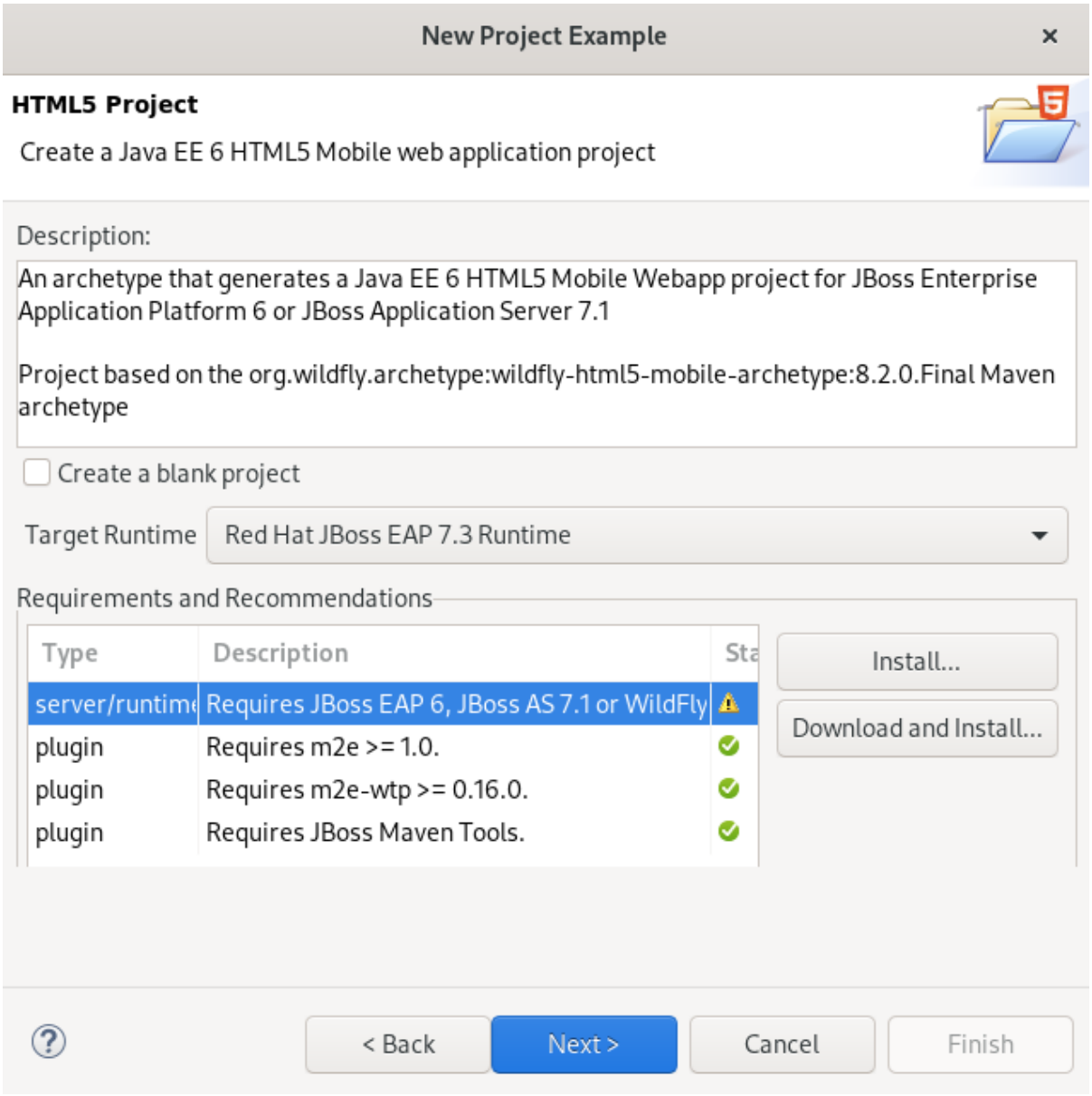
CodeReady Studio で HTML5 プロジェクトを作成する方法を説明します。

手順

1. CodeReady Studio を起動します。
2. **Ctr+N** キーを押します。
   Select a wizard ウインドウが表示されます。



3. 検索フィールドに **HTML5** と入力します。
4. **HTML5 Project** を選択します。
5. **Next** をクリックします。
   New Project Example ウィンドウが表示されます。



6. **Target Runtime** フィールドの下矢印をクリックします。
7. サーバーを選択します。
8. **Next** をクリックします。
9. プロジェクトとパッケージに名前を付けます。
10. プロジェクトの場所を選択します。
11. **Finish** をクリックします。
    プロジェクトの生成には時間がかかる場合があります。

    New Project Example ウィンドウが表示されます。
12. **Finish** をクリックします。

新たに作成されたプロジェクトが **Project Explorer** ビューに表示されます。

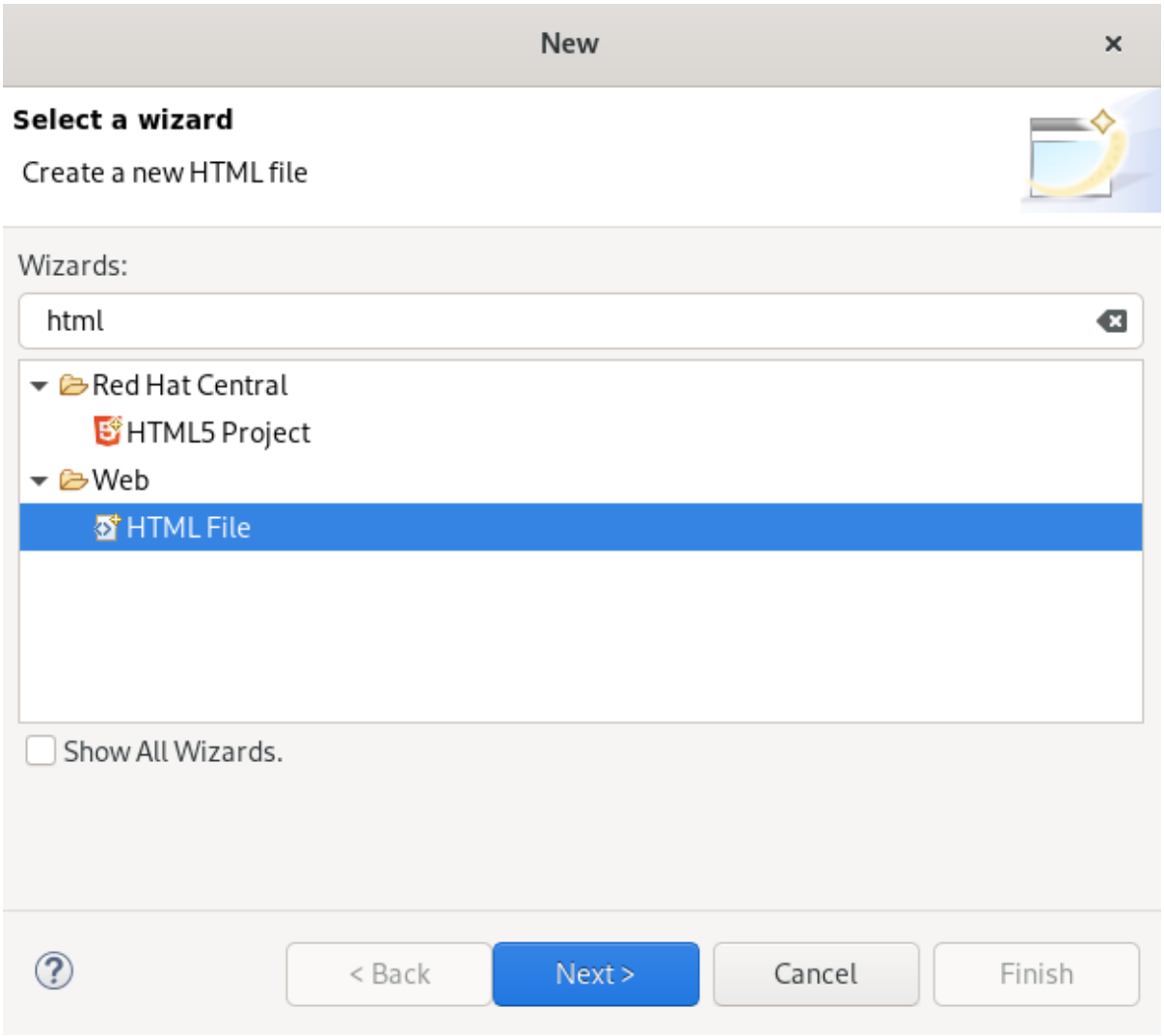
8.2. 新しい HTML5 JQUERY MOBILE ファイルの追加

HTML5 **jQuery Mobile** ファイルテンプレートは、ファイルの HTML ヘッダーに挿入される JavaScript および CSS ライブラリー参照で構成されます。このテンプレートは、**jQuery Mobile** ページのスケルトンや、ファイルの HTML ボディーの **listview** ウィジェットも挿入します。

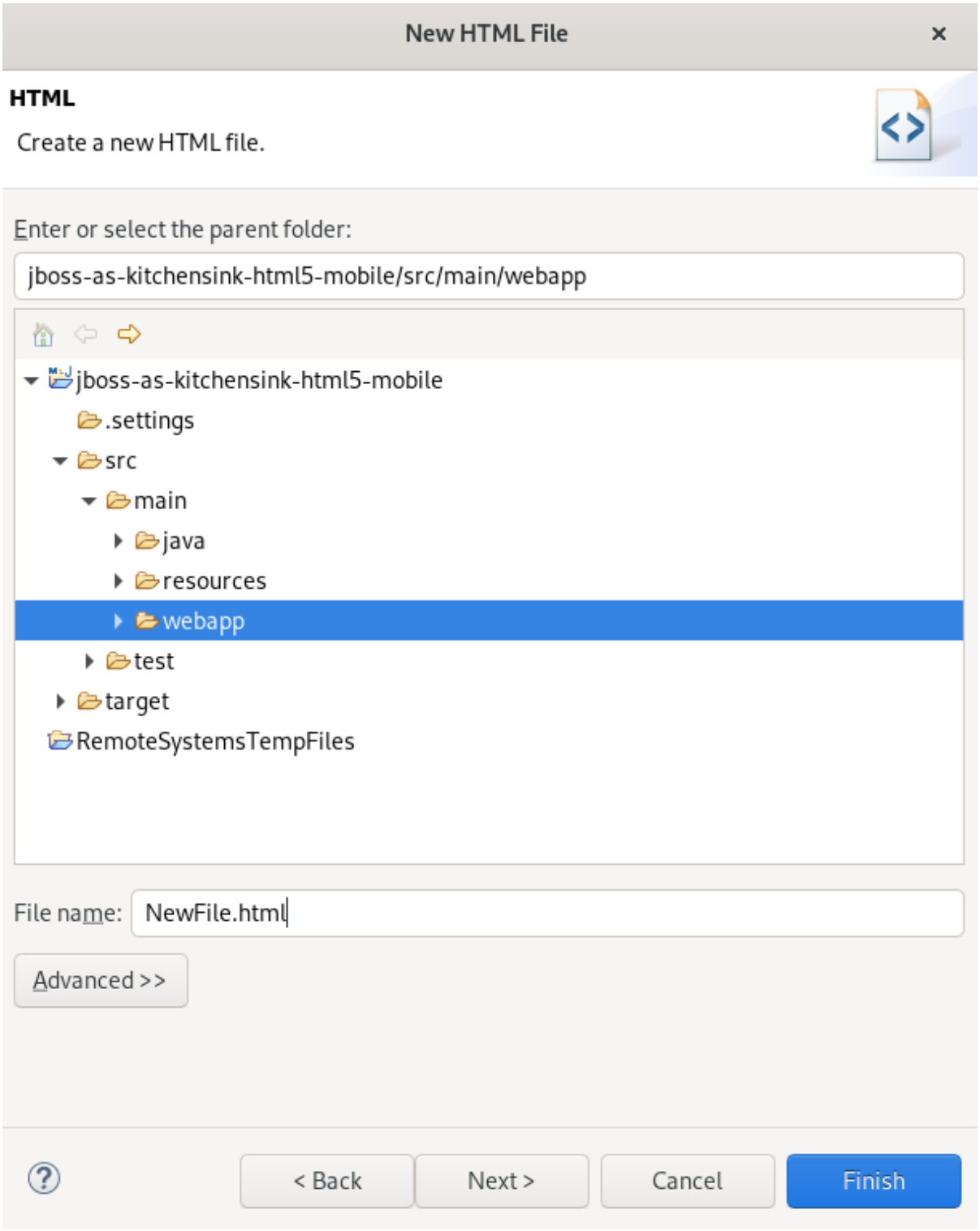
既存のプロジェクトに新しい HTML5 jQuery Mobile ファイルを追加する方法を説明します。

手順

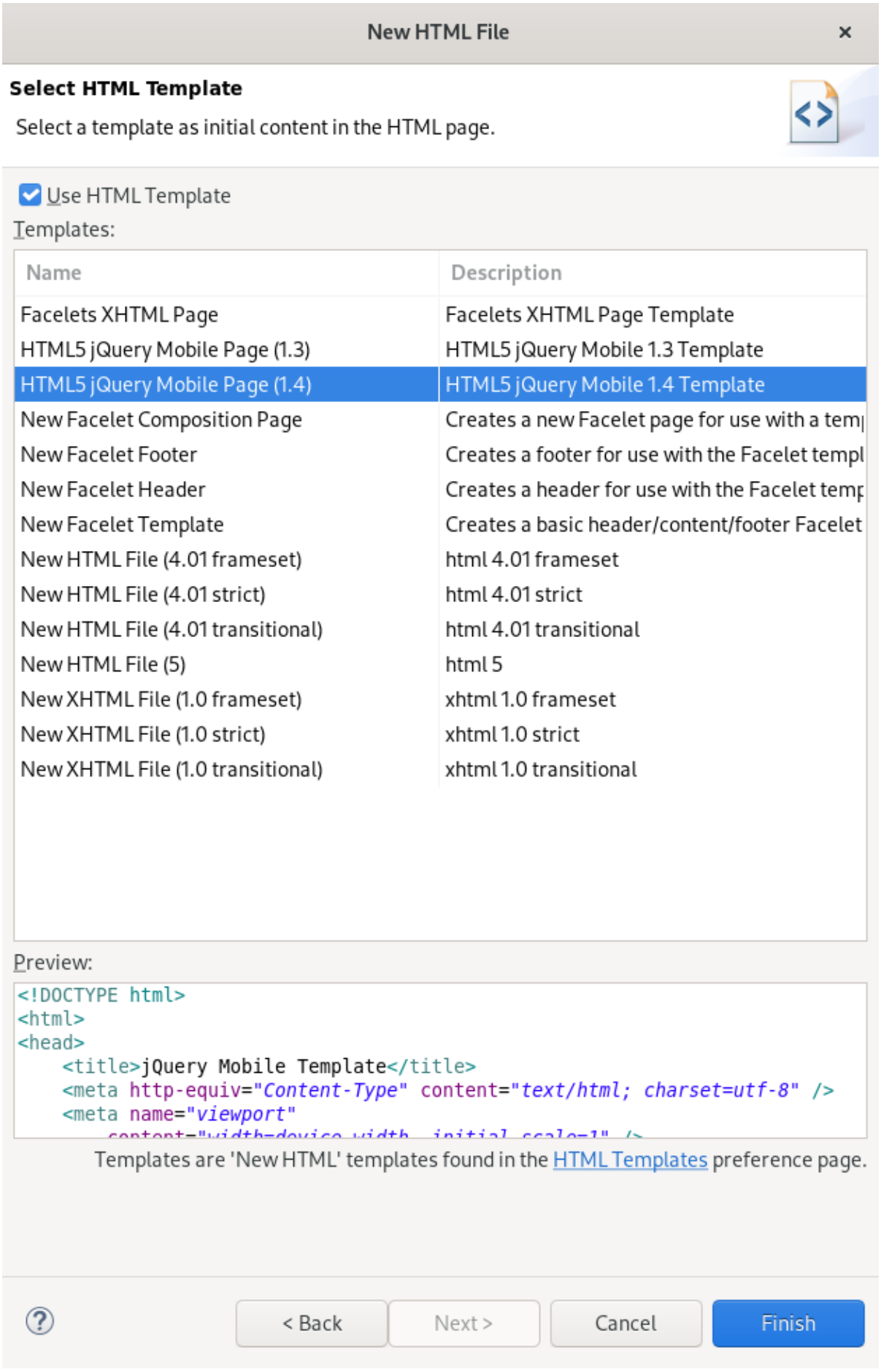
1. CodeReady Studio を起動します。
2. **Ctr+N** キーを押します。
   Select a wizard ウインドウが表示されます。



3. 検索フィールドに **HTML** と入力します。
4. **HTML File** を選択します。
5. **Next** をクリックします。
   New HTML File ウインドウが表示されます。



6. ファイルの場所を選択します。
7. ファイルに名前を付けます。
8. **Next** をクリックします。
   Select HTML Template ウインドウが表示されます。



9. テンプレートを選択します。
10. **Finish** をクリックします。

新たに作成された HTML ファイルが CodeReady Studio エディターに表示されます。

8.3. 新しいモバイルページの追加

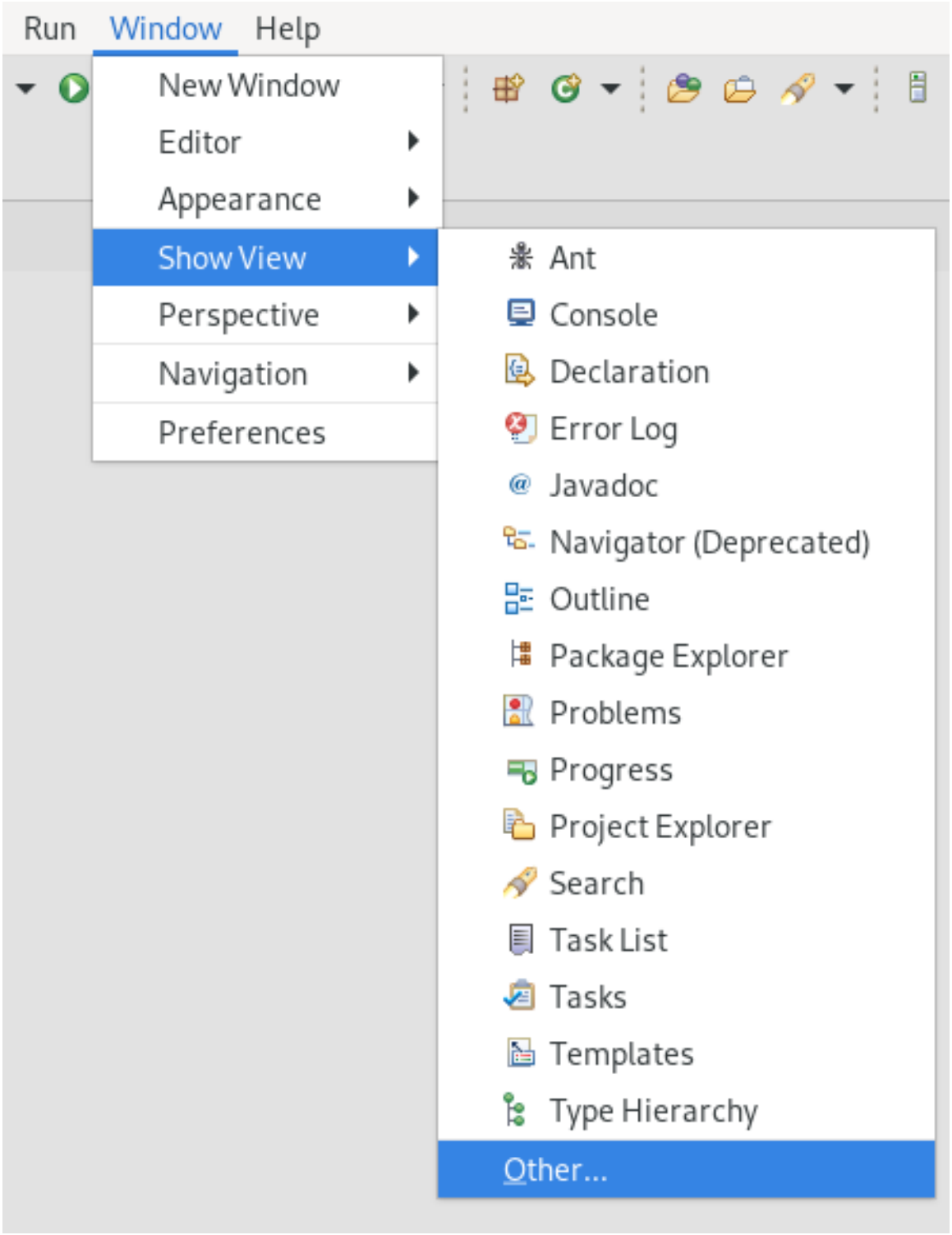
既存の Web アプリケーションに新しい jQuery Mobile ページを追加する方法を説明します。

前提条件

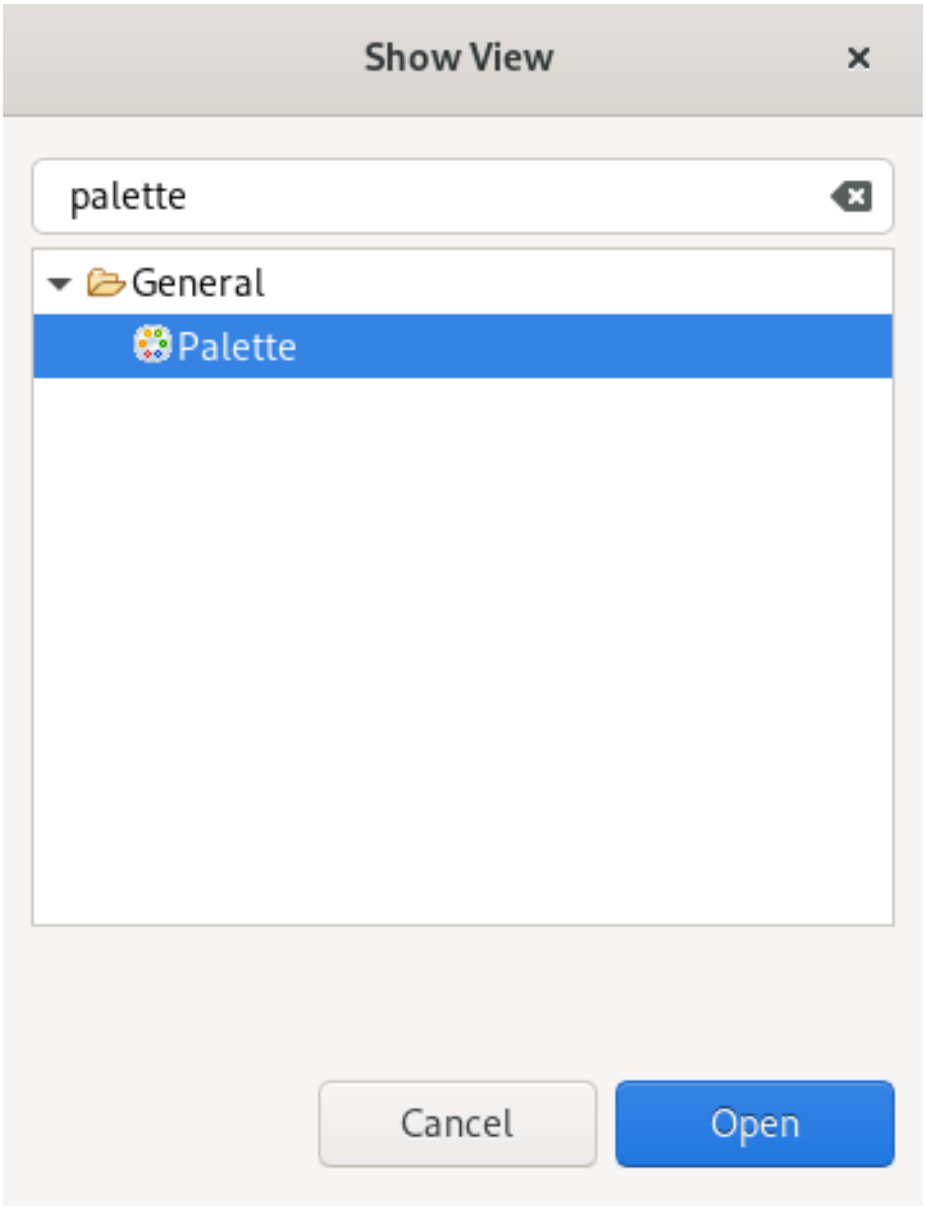
- HTML5 プロジェクト。
  HTML5 プロジェクトの作成方法についての詳細は、「HTML5 プロジェクトの作成」 を参照してください。

手順

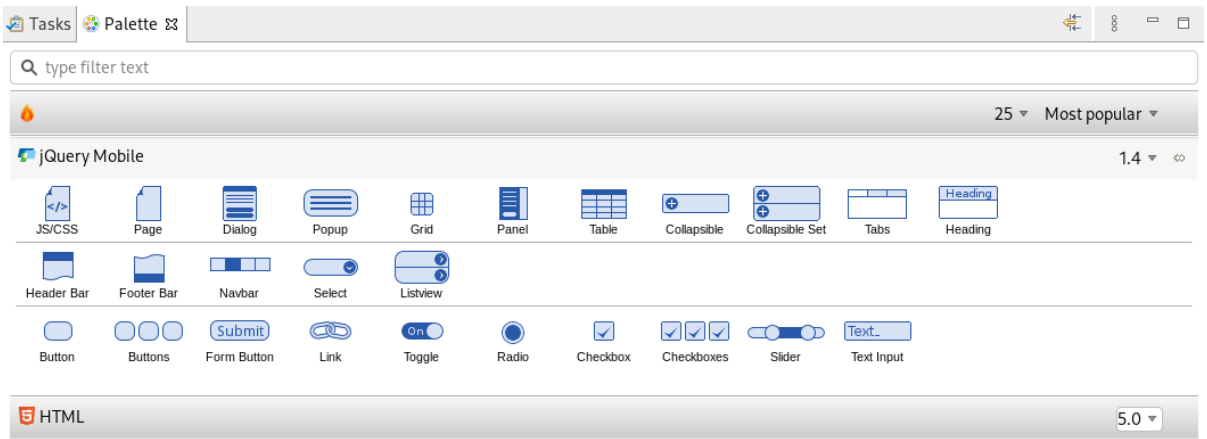
1. CodeReady Studio を起動します。
2. **Window → Show view → Other** とクリックします。



Show View ウインドウが表示されます。

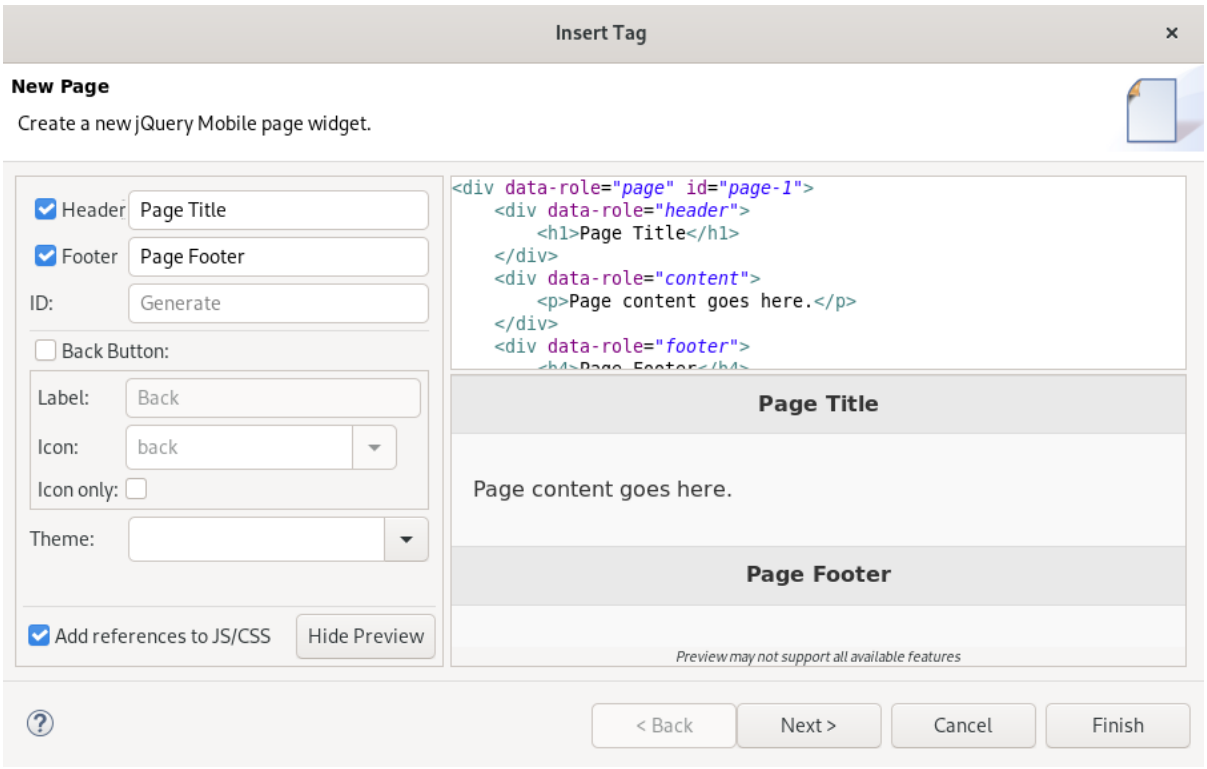


3. 検索フィールドに **Palette** と入力します。
4. **Palette** を選択します。
5. **Open** をクリックします。
Palette ビューが表示されます。



6. **Page** アイコンをクリックします。

Insert Tag ウィンドウが表示されます。



7. **Header** に名前を付けます。
8. **Footer** に名前を付けます。
9. **Finish** をクリックします。

新たに追加されたページが CodeReady Studio エディターに表示されます。

Palette ビューからのウィジェットを選択して、Web アプリケーションのページをカスタマイズする場合、同じワークフローが使用されます。