Red Hat CodeReady Studio 12.19.1

『Getting Started with CodeReady Studio Tools』

Red Hat CodeReady Studio ツールの使用方法
Enter your first name here. Enter your surname here.
Enter your organisation’s name here. Enter your organisational division here.
Enter your email address here.
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。
<table>
<thead>
<tr>
<th>章目</th>
<th>トピック</th>
<th>頁数</th>
</tr>
</thead>
<tbody>
<tr>
<td>第1章</td>
<td>CODEREADY STUDIO の GIT の基本</td>
<td>5</td>
</tr>
<tr>
<td>1.1</td>
<td>GIT PERSPECTIVE の設定</td>
<td>5</td>
</tr>
<tr>
<td>1.2</td>
<td>GIT PERSPECTIVE でのリポジトリの管理</td>
<td>7</td>
</tr>
<tr>
<td>1.2.1</td>
<td>新規 Git リポジトリの作成</td>
<td>7</td>
</tr>
<tr>
<td>1.2.2</td>
<td>既存のローカル Git リポジトリの追加</td>
<td>8</td>
</tr>
<tr>
<td>1.2.3</td>
<td>既存の Git リポジトリのクローン作成</td>
<td>9</td>
</tr>
<tr>
<td>1.2.4</td>
<td>リポジトリのリモートの追加</td>
<td>11</td>
</tr>
<tr>
<td>1.3</td>
<td>GIT PERSPECTIVE でのブランチの管理</td>
<td>16</td>
</tr>
<tr>
<td>1.3.1</td>
<td>新規ブランチの作成</td>
<td>16</td>
</tr>
<tr>
<td>1.3.2</td>
<td>ブランチの使用</td>
<td>19</td>
</tr>
<tr>
<td>1.3.3</td>
<td>ローカルリポジトリの更新</td>
<td>20</td>
</tr>
<tr>
<td>1.4</td>
<td>変更のコミットおよびプッシュ</td>
<td>21</td>
</tr>
<tr>
<td>第2章</td>
<td>CODEREADY STUDIO における MAVEN の基本</td>
<td>24</td>
</tr>
<tr>
<td>2.1</td>
<td>新規 MAVEN プロジェクトの作成</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>既存の MAVEN プロジェクトのインポート</td>
<td>29</td>
</tr>
<tr>
<td>2.2.1</td>
<td>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</td>
<td>29</td>
</tr>
<tr>
<td>2.2.2</td>
<td>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</td>
<td>32</td>
</tr>
<tr>
<td>2.3</td>
<td>新しい MAVEN モジュールの作成</td>
<td>34</td>
</tr>
<tr>
<td>2.4</td>
<td>MAVEN 依存関係の MAVEN プロジェクトへの追加</td>
<td>38</td>
</tr>
<tr>
<td>2.5</td>
<td>MAVEN サポートを MAVEN 以外の既存プロジェクトへ追加</td>
<td>41</td>
</tr>
<tr>
<td>2.6</td>
<td>関連情報</td>
<td>42</td>
</tr>
<tr>
<td>第3章</td>
<td>CODEREADY STUDIO でのアプリケーションデプロイメント</td>
<td>43</td>
</tr>
<tr>
<td>3.1</td>
<td>ローカルサーバーの設定</td>
<td>43</td>
</tr>
<tr>
<td>3.2</td>
<td>リモートサーバーの設定</td>
<td>47</td>
</tr>
<tr>
<td>3.3</td>
<td>アプリケーションのデプロイ</td>
<td>53</td>
</tr>
<tr>
<td>第4章</td>
<td>CODEREADY STUDIO の JBOSS EAP および JBOSS WFK の基本</td>
<td>56</td>
</tr>
<tr>
<td>4.1</td>
<td>MAVEN リポジトリの設定</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>JBOSS EAP の設定</td>
<td>60</td>
</tr>
<tr>
<td>第5章</td>
<td>CODEREADY STUDIO での OPENSHIFT の基本</td>
<td>68</td>
</tr>
<tr>
<td>5.1</td>
<td>OPENSHIFT APPLICATION EXPLORER ビューの設定</td>
<td>68</td>
</tr>
<tr>
<td>5.2</td>
<td>OPENSHIFT APPLICATION EXPLORER を使用した OPENSHIFT クラスターへの接続</td>
<td>71</td>
</tr>
<tr>
<td>5.3</td>
<td>ブラウザベースのトークンの取得を使用した OPENSHIFT クラスターへの接続</td>
<td>72</td>
</tr>
<tr>
<td>5.3.1</td>
<td>login コマンドの貼り付け</td>
<td>72</td>
</tr>
<tr>
<td>5.3.2</td>
<td>トークンの取得</td>
<td>74</td>
</tr>
<tr>
<td>5.4</td>
<td>OPENSHIFT ソールを使用した DEVELOPER SANDBOX の設定</td>
<td>75</td>
</tr>
<tr>
<td>5.5</td>
<td>DEVFILE 基づくアプリケーションのビルド</td>
<td>78</td>
</tr>
<tr>
<td>5.5.1</td>
<td>devfile レジストリーの管理</td>
<td>84</td>
</tr>
<tr>
<td>5.5.1.1</td>
<td>devfile レジストリーの追加</td>
<td>84</td>
</tr>
<tr>
<td>5.5.1.2</td>
<td>devfile レジストリーの削除</td>
<td>86</td>
</tr>
<tr>
<td>5.5.1.3</td>
<td>devfile レジストリーの編集</td>
<td>88</td>
</tr>
<tr>
<td>5.5.1.4</td>
<td>devfile レジストリーからのコンポーネントの作成</td>
<td>88</td>
</tr>
<tr>
<td>5.6</td>
<td>S2I ファイルを基にしたアプリケーションのビルド</td>
<td>91</td>
</tr>
<tr>
<td>5.7</td>
<td>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</td>
<td>97</td>
</tr>
<tr>
<td>5.8</td>
<td>OPENSHIFT APPLICATION EXPLORER を使用した外部アクセス URL の定義</td>
<td>98</td>
</tr>
<tr>
<td>5.9</td>
<td>OPENSHIFT APPLICATION EXPLORER を使用したクラスターでのアップリケーションのデバッグ</td>
<td>100</td>
</tr>
<tr>
<td>第6章</td>
<td>CODEREADY STUDIO の QUARKUS ツールの基本</td>
<td>103</td>
</tr>
</tbody>
</table>
6.1. 新しい QUARKUS プロジェクトの作成
6.2. QUARKUS アプリケーションの実行
6.3. QUARKUS アプリケーションのデバッグ
6.4. CODEREADY STUDIO での言語サポートの使用
   6.4.1. Quarkus のコンテンツアシストの使用
   6.4.2. MicroProfile REST Client プロパティーの言語サポートの有効化

第7章 CODEREADY STUDIO の HIBERNATE TOOLS の基本 .......................... 119
  7.1. 新規 JPA プロジェクトの作成
  7.2. ライブラリーの追加
  7.3. エンティティーからのテーブルの生成
  7.4. HIBERNATE マッピングファイルの作成
  7.5. HIBERNATE 設定ファイルの作成
  7.6. HIBERNATE コンソール設定ファイルの作成
  7.7. HIBERNATE プロジェクト設定の編集
  7.8. HIBERNATE ランタイム実装の設定

第8章 CODEREADY STUDIO での MOBILE WEB TOOLS の基本 ........................ 156
  8.1. HTML5 プロジェクトの作成
  8.2. 新しい HTML5 JQUERY MOBILE ファイルの追加
  8.3. 新しいモバイルページの追加
多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージをご覧ください。
第1章 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 でのリポジトリの管理

1.2.1. 新規 Git リポジトリの作成

Git Perspective を使用して新規 Git リポジトリ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 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。

Create a New Git Repository

Determine the directory for the new repository

Repository directory: /home/user/git/repository

Cancel Create
デフォルトの Repository ディレクトリへのパスが自動的に生成されます。リポジトリの保存先へのパスを選択し、リポジトリ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
ベアリポジトリは中央リポジトリに推奨されますが、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、リポジトリの Git リビジョンの履歴は、a.git サブフォルダーではなく、ルート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 リポジトリの追加

Git Perspective を使用して、ローカル Git リポジトリを CodeReady Studio 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 ワイ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表示するチェックボックスを選択します。

6. 追加 をクリックします。

ローカルリポジトリが 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 (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が、CodeReady Studio の Git Repository ビューに表示されます。

1.2.4. リポジトリのリモートの追加
Git Perspectiveにリポジトリを設定したら、リポジトリ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. リポジトリを展開します。
4. Remotes → Create Remoteと右クリックします。
   New Remoteウィ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. 変更 をクリックします。
Select a URI ウィンドウが表示されます。
ソースリポジトリの 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。

9. **Finish** をクリックします。

10. **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. リポジトリを展開します。
4. branches → Remote Tracking で、master → Create Branch を右クリックします。
Create Branch ウィ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branches → Local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 で埋め込みのターミナルを開き、作成したブランチを使用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. Shift+Ctrl+Alt+T キーを押します。
   
   Launch Terminal ユ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** ウィ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. ローカルリポジトリの更新

特に共有リポジトリで作業を行う場合に、マージの競合を防ぐには、変更をマージする前にローカル
リポジトリを更新します。

ローカルリポジトリを更新するには、リモートリポジトリから変更をプルし、ローカルリポジト
リーにマージします。

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 ウィンドウが表示されます。

4. 閉じる をクリックします。

これで、リモートリポジトリからの変更がローカルリポジトリにマージされます。

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、リポジトリーオ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。

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 ユインドウが表示されます。
3. 検索フィールドに Project Explorer と入力します。

4. Project Explorer を選択します。

5. Open をクリックします。
   Project Explorer ビューが表示されます。

6. Ctrl+N キーを押します。
   Select a wizard メニューが表示されます。
7. WizardsフィールドにMavenと入力します。

8. Maven Projectを選択します。

9. Nextをクリックします。
   New Maven Projectウィンドウが表示されます。
10. **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の選択が省略されます。また、プロジェクトタイプは、マルチモジュールのMavenプロジェクトに必要なPOM(Project Object Model)に自動的に設定されます。

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、画面の指示に従ってパッケージオプションをjarまたはwarに設定します。

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

12. **Next**をクリックします。
13. グループ ID とアーティファクト ID を入力します。

注記
ID の値には、スペースや特殊文字を使用できません。使用できる特殊文字は、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。

14. Packaging を pom、jar、または war に設定します。

15. 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 Project Explorer 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 ウィンドウが表示されます。
Select an import wizard フィールドに Maven と入力します。

Existing Maven Projects を選択します。

Next をクリックします。

Import Maven Project ウィ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. Add project(s) to working set チェックボックスを選択します。

8. Finish をクリックします。

ローカルの Maven プロジェクトが Project Explorer 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。

Target Location
Select target location and revision

SCM URL: git https://github.com/user/maven-project.git

Check out Head Revision
Revision: 

Check out All Projects
Advanced

Find more SCM connectors in the m2e Marketplace

Finish
4. ソースリポジトリのアドレスを 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 ウィンドウが表示されます。

Check out as Maven project from SCM

Select Project Location
Select project location and working set

- Use default Workspace location
  Location: [入力]
  Browse...
- Add project(s) to working set
  Working set: [入力]
  More...
  Advanced

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 Repositories ビューに表示されます。

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。

Select a wizard
Create a Maven module

Wizards:

- maven
  - Configure Maven Repositories
  - Maven
    - Check out Maven Projects from SCM
    - Maven Module
    - Maven Project
  - Red Hat Fuse
    - Fuse Integration Project
    - Fuse Online Extension Project

Show All Wizards.
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をクリックします。

New Maven Moduleウィ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の選択が省略されます。また、プロジェクトタイプは、マルチモジュールのMavenプロジェクトに必要なPOM(Project Object Model)に自動的に設定されます。

代わりにスタンドアロンのMavenプロジェクトを作るには、Create a simple projectチェックボックスを未選択にし、画面の指示に従ってパッケージオプションをjarまたはwarに設定します。
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 ウィンドウが表示されます。

   ![Configure Project Window](image)

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Group Id: com.example.maven-project</th>
<th>Artifact Id: new-maven-module</th>
<th>Version: 1.0-SNAPSHOT</th>
<th>Packaging: pom</th>
<th>Name:</th>
<th>Description:</th>
</tr>
</thead>
</table>

**Parent Project**

<table>
<thead>
<tr>
<th>Group Id:</th>
<th>com.example.maven-project</th>
<th>Artifact Id: maven-project</th>
<th>Version: 1.0-SNAPSHOT</th>
</tr>
</thead>
</table>

10. **Packaging** を pom、jar、または war 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。
Finish をクリックします。

新たに作成された Maven モジュールが Maven プロジェクトの下に表示されます。

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 (*)**:

   org.company-name_project-project-name

   - Index downloads are disabled, search results may be incomplete.

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。使用できる特殊文字は、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。

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 Java プロジェクトに表示されます。

2.6. 関連情報

- Maven ソフトウェアプロジェクトの管理および解釈のツールに関する詳細は、「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

CODEREADY Studio 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、CODEREADY Studio 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てきます。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 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てください。

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。

No new runtime found.
Searching runtimes is finished.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Version</th>
<th>Errors / QuickFix</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat JBoss EAP 7.2</td>
<td>EAP</td>
<td>7.2</td>
<td></td>
<td>/home/user/jboss-eap-</td>
</tr>
</tbody>
</table>

- Hide already created runtimes
- Add runtime path to list of paths searchable on startup

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 をクリックします。

Define a New Server ウインドウが表示されます。
6. 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. 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。
第3章 CODEREADY STUDIOでのアプリケーションデプロイメント

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 CodeReady Studio 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 CodeReady Studio 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。
Run On Server

Select which server to use

How do you want to select the server?
- Choose an existing server
- Manually define a new server

Select the server that you want to use:

<table>
<thead>
<tr>
<th>Server</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost</td>
<td></td>
</tr>
<tr>
<td>Red Hat JBoss EAP 7.0</td>
<td>Started</td>
</tr>
<tr>
<td>Red Hat JBoss EAP 7.3</td>
<td>Stopped</td>
</tr>
<tr>
<td>WildFly 18.0</td>
<td>Stopped</td>
</tr>
<tr>
<td>WildFly 19</td>
<td>Stopped</td>
</tr>
</tbody>
</table>

Red Hat JBoss Enterprise Application Platform (EAP) 7.0

Always use this server when running this project

3. **Choose an existing server** チェックボックスを選択します。

4. デプロイする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 リポジトリおよび JBoss WFK リポジトリへの IDE のアクセスが必要になります。

4.1. MAVEN リポジトリの設定

Maven リポジトリ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 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。

Maven リポジトリの設定方法に関する詳細は、「Maven リポジトリの設定」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。
必要な JBoss EAP 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. 追加をクリックします。

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。
Download Runtime

URL: https://www.jboss.org/download-manager/jdf/file/jboss-eap-7.0.0.zip
Install folder: /home/user
Download folder: /tmp
☑️ Delete archive after installing

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、デバッグも容易および迅速に行うことができます。

OpenShift Application Explorer は、CodeReady Studio でデフォルトのビューとして設定されています。手動で開く必要がある場合は、「OpenShift Application Explorer ビューの設定」を参照してください。

前提条件

- 実行中の OpenShift クラスター。

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 ウィ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。

Procedure

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 ウィンドウが表示されます。
4. URL フィールドに OpenShift API URL を貼り付けます。

OpenShift API URL を使用したクラスターへのアクセスについての詳細は、Red Hat OpenShift のサービスへのアクセスについて参照してください。

5. ユーザー名とパスワードまたはトークンを入力します。

6. Finish をクリックします。

5.3. ブラウザーベースのトークンの取得を使用した OPENSHIFT クラスターへの接続

OpenShift Application Explorer にユーザー名とパスワードまたはトークンを提供する場合は、ブラウザーベースのトークンの取得を使用して OpenShift クラスターにログインします。ログインオプションを貼り付けたり、トークンを取得する 2 つのログインオプションがあります。

5.3.1. login コマンドの貼り付け

Procedure
1. URL フィールドに OpenShift API URL を貼り付けます。

OpenShift API URL を使用したクラスターへのアクセスについての詳細は、Red Hat OpenShift のサービスへのアクセスについて参照してください。

2. OpenShift Container Platform の Web UI にアクセスします。

3. 右上隅のドロップダウンメニューをクリックします。

4. Copy Login Command をクリックします。

5. Display Token をクリックします。

6. ログインコマンドをコピーします。

7. Sign in to OpenShift ウィンドウで Paste login command をクリックします。
8. 

Finish をクリックします。

注記

OpenShift 3 では、ログインコマンドは自動的にクリップボードにコピーされます。

5.3.2. トークンの取得

Procedure

1. 

URL フィールドに OpenShift API URL を貼り付けます。

OpenShift API URL を使用したクラスターへのアクセスについての詳細は、Red Hat OpenShift のサービスへのアクセスについて参照してください。

2. 

Retrieve token をクリックします。
3. ユーザー名とパスワードを入力します。

4. Log in をクリックします。

5. Display Token をクリックします。

6. Finish 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。

5.4. OPENSHIFTツールを使用した DEVELOPER SANDBOX の設定

ここでは、CodeReady Studio で Developer Sandbox にブートストラップを実行し、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. OpenShift コネクションを右クリックします。

4. Login をクリックします。

5. Red Hat Developer Sandbox をクリックします。

6. Red Hat アカウントのクレデンシャルを入力し、Log in をクリックします。
Develop Sandbox がブートストラップされました。

Login to Red Hat Developer Sandbox ウィンドウが表示されます。

1. Next をクリックします。
2. DevSandbox をクリックします。
3. Red Hat アカウントのクレデンシャルを再度入力し、Log in をクリックします。
4. Display Token をクリックします。
5. Finish をクリックします。

トークンが Sign in to OpenShift ウィンドウに表示されます。
12. Finish をクリックします。
これで Developer Sandbox にログインした状態となります。

Developer Sandbox が OpenShift Application Explorer ビューに表示されます。

5.5. DEVFILE に基づくアプリケーションのビルド

devfile を基にしてアプリケーションをデプロイするには、ローカルのワークスペースに空のプロジェクトが必要です。さらに、OpenShift にも devfile コンポーネントを作成するための空のプロジェクトが必要です。コンポーネントの確立後、プロジェクトが更新され、OpenShift で作成されたローカルおよびリモートアーティファクトも更新されます。

Procedure

1. CodeReady Studio を起動します。

2. Ctrl+M キーを押します。

Select a wizard ウィンドウが表示されます。
3. **General → Project**を選択します。

4. **Next**をクリックします。

   *New Project* ウインドウが表示されます。
5. プロジェクトに名前をつけます。

6. プロジェクトの場所を選択します。

7. Finish をクリックします。

   新たに作成した空のプロジェクトが Project Explorer ビューに表示されます。

8. OpenShift Application Explorer を起動します。

9. 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。

プロジェクトに名前を付けます。

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。

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。
プロジェクトに名前を付けます。

14. Browse をクリックして Eclipse Project を選択します。

15. Component type を java-vertx に設定します。
16. Project スターターを java-vertx に設定します。

17. アプリケーションに名前を付けます。

18. Push after create チェックボックスを未選択にします。

19. 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。

devfile をベースとするアプリケーションがビルドされます。

5.5.1. devfile レジストリーの管理

CodeReady Studio で OpenShift Application Explorer を使用して devfile レジストリーを作成、削除、および編集する方法を説明します。

5.5.1.1. devfile レジストリーの追加

前提条件

- 実行中の OpenShift クラスター。

Procedure

1. CodeReady Studio を起動します。

2. OpenShift Application Explorer を起動します。
3. devfile レジストリーは、Devfile registries ノードに表示されます。

4. devfile レジストリーノードを拡張すると、そのレジストリーのすべての devfile が表示されます。

新規 devfile レジストリーを追加するには、Devfile registries を右クリックし、new をクリックします。
5. devfile レジストリーに名前を付けます。

6. devfile URL を貼り付けます。

7. Finish をクリックします。

新たに作成された devfile レジストリーが Devfile registries 下の OpenShift Application Explorer ビューに表示されます。

5.5.1.2. devfile レジストリーの削除

前提条件

- 実行中の OpenShift クラスター。

Procedure

1. CodeReady Studio を起動します。
第5章 CODEREADY STUDIOでのOPENSIFTの基本

2. OpenShift Application Explorerを起動します。

3. devfileレジストリーは、Devfile registriesノードに表示されます。

4. 展開されたdevfileレジストリーノードには、そのレジストリーのすべてのdevfileが表示されます。

devfileレジストリーを削除するには、devfileレジストリーのノードを右クリックして、deleteをクリックします。
devfile レジストリーが削除されます。

5.5.1.3. devfile レジストリーの編集

前提条件

- 実行中の OpenShift クラスター。

Procedure

- devfile レジストリーを編集するには、YAML エディターを使用します。YAML エディターは構文の検証およびコンテンツ補助を提供します。

5.5.1.4. devfile レジストリーからのコンポーネントの作成

devfile レジストリーから新規コンポーネントを作成するには、コンポーネントタイプまたはスタータープロジェクトから新規コンポーネントを作成できます。

前提条件

- 既存の Eclipse プロジェクト。
- 既存の OpenShift プロジェクト。
- OpenShift クラスターに接続されている。

Procedure

1. Devfile レジストリーノードで devfile レジストリーを展開します。

2. コンポーネントタイプまたはスタータープロジェクトを右クリックし、Project → New → Component をクリックします。
ウィンドウが表示されます。

Create component ウィンドウが表示されます。

- https://api.crc.testing:5443/
- Devfile registries
  - DefaultDevfileRegistry https://registry.devfile.io
    - Django Python3.7 with Django
    - Maven Java Upstream Maven and OpenJDK 11
  - springbootproject
    - Node.js Runtime
    - Open Liberty Java application stack using Open Liberty runtime
    - Python Python Stack with Python 3.7
3. コンポーネントに名前を付けます。

4. **Browse** をクリックして **Eclipse プロジェクト**を選択します。

5. **Component type** を選択します。

6. プロジェクトスターターを選択します。
アプリケーションに名前をつけます。

8. Finish をクリックします。

新規コンポーネントは devfile レジストリーから作成されます。

関連情報

devfile の詳細は、「Introduction to Devfile」を参照してください。

5.6. S2I ファイルを基にしたアプリケーションのビルド

S2I ファイルを基にしてアプリケーションをデプロイするには、ローカルのワークスペースに空のプロジェクトが必要です。さらに、OpenShift にもコンポーネントを作成するための空のプロジェクトが必要です。コンポーネントの確立後、プロジェクトが更新され、OpenShift で作成されたローカルおよびリモートアーティファクトも更新されます。

Procedure

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 ウィンドウが表示されます。
6. **Mission** は rest-http に設定します。

7. **Runtime** tovert.x community に設定します。
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 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。

11. OpenShift Application Explorer を起動します。

12. 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。
13. プロジェクトに名前を付けます。

14. 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。

15. 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。
プロジェクトに名前を付けます。

17. Browse をクリックして Eclipse Project を選択します。
S2I の横にある矢印をクリックし、Component type を java に設定します。

Component version を latest に設定します。

アプリケーションに名前を付けます。

Push after create チェックボックスを未選択にします。

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。

S2I ファイルをベースとするアプリケーションが構築されている。

5.7. 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. \text{component} \rightarrow \text{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。

5.8. 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. \text{component} \rightarrow \text{New} \rightarrow \text{URL} を右クリックします。
Create URL ウィンドウが表示されます。

Create url
 Specify a name for the url, choose a port to bind to and select a secure (https) scheme or not.

Name: my-new-url
Port: 8080
Secure: 

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。

5.9. 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、クラスターで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. Project Explorer ビューで、MainVerticle.java(devfiles)または HttpApplication.java(S2I)ファイルを見つけ、ダブルクリックして開きます。

3. 左側がルーラーになっている列をクリックし、ブレークポイントを設定します。
4. OpenShift Application Explorerを開きます。

5. プロジェクトを展開します。

6. アプリケーションを展開します。

7. component → Debugを右クリックします。

Consoleビューが表示されます。
8. OpenShift Application Explorer で、コンポーネントを展開します。

9. URL → Open in Browser を右クリックします。

Confirm Perspective Switch ウィンドウが表示されます。

10. 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、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。詳細は、「JBoss Tools Downloads」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. 必要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 ウィンドウが表示されます。
プロジェクトに必要な 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。

プロジェクトに必要な 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。

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。
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。

注記

環境変数を Quarkus プロジェクトに追加できます。新しい環境変数を追加するには、Environment → add とクリックし、名前と値を選択します。

7. 適用をクリックします。

8. **Run** をクリックします。

**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](image)

   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
Quarkus 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、MicroProfile REST Client プラットフォーム、および MicroProfile Health アーキテクチャで使用できるようになりました。

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

6.4.1. Quarkus のコンテンツアシストの使用

CodeReady Studio で Quarkus application.properties コンテンツ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。
5. 空の行に移動します。

6. Ctrl+Space 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。
関連情報

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

6.4.2. MicroProfile REST Client プロパティー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 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。

```java
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
  ("/greet")
  Response greet();
}

Ctrl+Sキーを押して変更を保存します。

その他のリソース

• 言語サポートの使用方法の詳細は、「Quarkus のコンテンツアシストの使用」 を参照してください。

• 言語サポートの調整については、Quarkus - REST クライアントを使用したインターフェースの作成について参照してください。
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。

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。

前提条件

- Sakila データベースサーバーが起動します。
  Sakila データベースを起動するには、以下を実行します。

  a. h2 バージョンの Sakila データベースをダウンロードします。

  b. runh2.sh ファイルが含まれるディレクトリーに移動します。

  c. runh2.sh ファイルを実行します。

    $ ./runh2.sh

Procedure

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。
### JPA Project

Configure JPA project settings.

<table>
<thead>
<tr>
<th>Project name:</th>
<th>my-new-JPA-project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project location:</td>
<td>Use default location</td>
</tr>
<tr>
<td>Location:</td>
<td>/home/levi/eclipse-workspace/my-new-JPA-project</td>
</tr>
<tr>
<td>Target runtime:</td>
<td>java-11-openjdk-11.0.7.10-1.fc32.x86_64</td>
</tr>
<tr>
<td>JPA version:</td>
<td>2.1</td>
</tr>
<tr>
<td>Configuration:</td>
<td>&lt;custom&gt;</td>
</tr>
<tr>
<td>Hint:</td>
<td>Get started quickly by selecting one of the pre-defined project configurations.</td>
</tr>
</tbody>
</table>

Ear membership

- Add project to an EAR

Ear project name: |

Working sets

- Add project to working sets

Working sets: |

---

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。

<table>
<thead>
<tr>
<th>New JPA Project</th>
</tr>
</thead>
</table>

### JPA Facet

⚠️ Library configuration is disabled. The user may need to configure further classpath changes later.

#### Platform

- Hibernate (JPA 2.1)

#### JPA implementation

- Type: Disable Library Configuration

The JPA facet requires a JPA implementation library to be present on the project classpath. By disabling library configuration, the user takes on the responsibility of ensuring that the classpath is configured appropriately via alternate means.

#### Connection

- <None>

Add connection...  Connect

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 ウィ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 ウィ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. jdbc:h2:tcp://localhost/./sakila を Connection URL フィールドに追加します。

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 ウィンドウが表示されます。
29. Browse for 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。
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 ビューに表示されます。

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 ヴ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. a.jar ファイルを選択します。

15. 開くをクリックします。

selected.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. エンティティーからのテーブルの生成

   CodeReady StudioでHibernateプロジェクトのエンティティーからテーブル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 ワイ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. Ctrl+N キーを押します。

   **Select a wizard** ウィ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、パッケージまたはクラスを選択せずに empty.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。

![New Hibernate XML Mapping files (hbxml)](image)

**New Hibernate XML Mapping files (hbxml)**

Create a new empty XML Mapping files

Enter or select the parent folder:

- `my-new-JPA-project`

File name: `hibernate.hbm.xml`

10. 親ディレクトリーを選択します。

11. `your.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、設定ファイル hibernate.cfg.xml 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. Ctrl+N キーを押します。

Select a wizard ウインドウが表示されます。

<table>
<thead>
<tr>
<th>Wizards:</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibernate</td>
</tr>
</tbody>
</table>

Hibernate

- Hibernate Configuration File (cfg.xml)
- Hibernate Console Configuration
- Hibernate Reverse Engineering File (reveng.xml)
- Hibernate XML Mapping file (hbm.xml)

Show All Wizards.
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) ウィ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。
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. Ctrl+N キーを押します。

Select a wizard ウインドウが表示されます。

Finish をクリックします。
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. 正しい 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 ウィ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 ウィ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) ウィンドウが表示されます。

Create Hibernate Properties file (.properties)
Create a new properties file

Enter or select the parent folder:

my-new-JPA-project

File name: hibernate.properties
14. 親ディレクトリを選択します。

15. your.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. target.cfg.xmlファイルへのパスを選択します。

Setup configuration fileウィ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)ウィ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) ウィンドウが表示されます。
### Hibernate Configuration File (cfg.xml)

このウィザードは、Hibernateを使用するための新しい構成ファイルを作成します。

<table>
<thead>
<tr>
<th>Container:</th>
<th>/my-new-JPA-project/src</th>
</tr>
</thead>
<tbody>
<tr>
<td>File name:</td>
<td>hibernate.cfg.xml</td>
</tr>
<tr>
<td>Hibernate version:</td>
<td>5.4</td>
</tr>
<tr>
<td>Session factory name</td>
<td></td>
</tr>
</tbody>
</table>

#### Get values from Connection

- **Database dialect:** MySQL
- **Driver class:** org.gjt.mm.mysql.Driver
- **Connection URL:** jdbc:mysql://<hostname>/<database>
- **Default Schema:**
- **Default Catalog:**
- **Username:**
- **Password:**

- Create a console configuration

---

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。

**Create Hibernate Console Configuration**ウィンドウが表示されます。
26. データベース接続を sakila に設定します。

27. 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。
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 ウィ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。

7.8. HIBERNATE ランタイム実装の設定

Hibernate preference ページでランタイム実装を有効または無効にできます。

以下のセクションでは、ランタイム設定を設定する方法を説明します。

Procedure

1. 
Window → Preferences をクリックします。

設定ページが開きます。
2. 検索フィールドに Hibernate と入力します。

3. Hibernate を選択します。

Hibernate プロジェクトのランタイムを無効にする場合は、有効またはチェックをチェックしてください。
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、CodeReady Studio 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。

CodeReady Studio 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、および Buttons などのウィジェットも含まれています。

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。

前提条件

- 設定済みのローカルサーバー。

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。

CodeReady Studio 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。
手順

1. CodeReady Studio を起動します。

2. Ctrl+N キーを押します。
   Select a wizard ウィンドウが表示されます。

   ![Select a wizard](image)

   Wizards:
   - html5
   - Red Hat Central
     - HTML5 Project

   Show All Wizards.

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。

### HTML5 Project

Create a Java EE 6 HTML5 Mobile web application project

**Description:**

An archetype that generates a Java EE 6 HTML5 Mobile Webapp project for JBoss Enterprise Application Platform 6 or JBoss Application Server 7.1

Project based on the org.wildfly.archetype:wildfly-html5-mobile-archetype:8.2.0.Final Maven archetype

![HTML5 Project Archetype](image)

- **Create a blank project**
- **Target Runtime**
  - Red Hat JBoss EAP 7.3 Runtime

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>server/runtime</td>
<td>Requires JBoss EAP 6, JBoss AS 7.1 or WildFly</td>
<td><img src="image" alt="Install..." /></td>
</tr>
<tr>
<td>plugin</td>
<td>Requires m2e &gt;= 1.0.</td>
<td><img src="image" alt="Download and Install..." /></td>
</tr>
<tr>
<td>plugin</td>
<td>Requires m2e-wtp &gt;= 0.16.0.</td>
<td><img src="image" alt="Install..." /></td>
</tr>
<tr>
<td>plugin</td>
<td>Requires JBoss Maven Tools.</td>
<td><img src="image" alt="Download and Install..." /></td>
</tr>
</tbody>
</table>

- **Next**をクリックします。

---

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。

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。

CodeReady Studio 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。

手順
1. CodeReady Studio を起動します。

2. Ctrl+N キーを押します。

   Select a wizard ウィンドウが表示されます。

   ![Select a wizard](Image)

   New Wizard

   Wizards:

   - html
   - Red Hat Central
     - HTML5 Project
   - Web
     - HTML File

   □ Show All Wizards.

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 ウィ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 ユインドウが表示されます。

Select HTML Template

Select a template as initial content in the HTML page.

- **Use HTML Template**
- **Templates:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facelets XHTML Page</td>
<td>Facelets XHTML Page Template</td>
</tr>
<tr>
<td>HTML5 jQuery Mobile Page (1.3)</td>
<td>HTML5 jQuery Mobile 1.3 Template</td>
</tr>
<tr>
<td>HTML5 jQuery Mobile Page (1.4)</td>
<td>HTML5 jQuery Mobile 1.4 Template</td>
</tr>
<tr>
<td>New Facelet Composition Page</td>
<td>Creates a new Facelet page for use with a template</td>
</tr>
<tr>
<td>New Facelet Footer</td>
<td>Creates a footer for use with the Facelet template</td>
</tr>
<tr>
<td>New Facelet Header</td>
<td>Creates a header for use with the Facelet template</td>
</tr>
<tr>
<td>New Facelet Template</td>
<td>Creates a basic header/content/footer Facelet template</td>
</tr>
<tr>
<td>New HTML File (4.01 frameset)</td>
<td>html_4.01 frameset</td>
</tr>
<tr>
<td>New HTML File (4.01 strict)</td>
<td>html_4.01 strict</td>
</tr>
<tr>
<td>New HTML File (4.01 transitional)</td>
<td>html_4.01 transitional</td>
</tr>
<tr>
<td>New HTML File (5)</td>
<td>html_5</td>
</tr>
<tr>
<td>New XHTML File (1.0 frameset)</td>
<td>xhtml_1.0 frameset</td>
</tr>
<tr>
<td>New XHTML File (1.0 strict)</td>
<td>xhtml_1.0 strict</td>
</tr>
<tr>
<td>New XHTML File (1.0 transitional)</td>
<td>xhtml_1.0 transitional</td>
</tr>
</tbody>
</table>

Preview:

```html
<!DOCTYPE html>
<html>
<head>
  <title>jQuery Mobile Template</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
</head>
```

Templates are 'New HTML' templates found in the HTML Templates preference page.
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 ウィ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。

注記

Palette ビューからウィジェットを選択して、同じワークフローを使用して Web アプリケーションのページをカスタマイズできます。