DIコンテナ in Spring Boot

Introduction

DIコンテナ

DI（Dependency Injection）を自動で行なってくれるフレームワーク。

DIは、『依存性の注入』と訳される。
意味としては、『使用するオブジェクトを、DIコンテナが注入してくれる』と理解すると良い。

オブジェクトの使用時に、newによるインスタンス化ではなく、DIコンテナによるインスタンス化を用いる。
それにより、依存性が低下し、疎結合なプロジェクトを作成出来る。

目次

1, DIコンテナのメリット

2, DIコンテナ 未使用&使用時の違い

3, 主なアノテーション

4, DIコンテナ 未使用&使用サンプル

1, DIコンテナのメリット

メリット

疎結合なプロジェクトを作成することが出来る。
それにより、**単体テストが行いやすくなる。**

密結合なプロジェクト

単体テスト対象が、他の完成していないプログラムに依存している場合、
単体テストを実施することが出来ない。

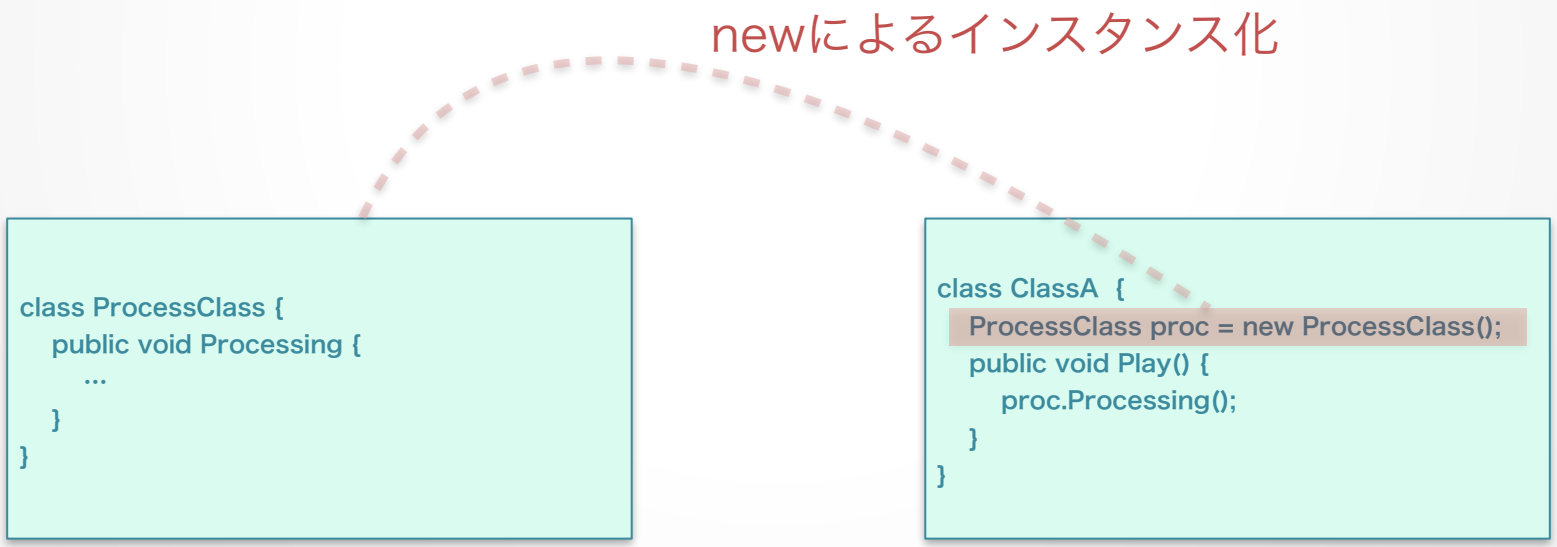
疎結合なプロジェクト

単体テスト対象が、他のプログラムに依存していないため、
容易に単体テストを実施することが出来る。

2, DIコンテナ 未使用&使用時の違い

DIコンテナ未使用の場合

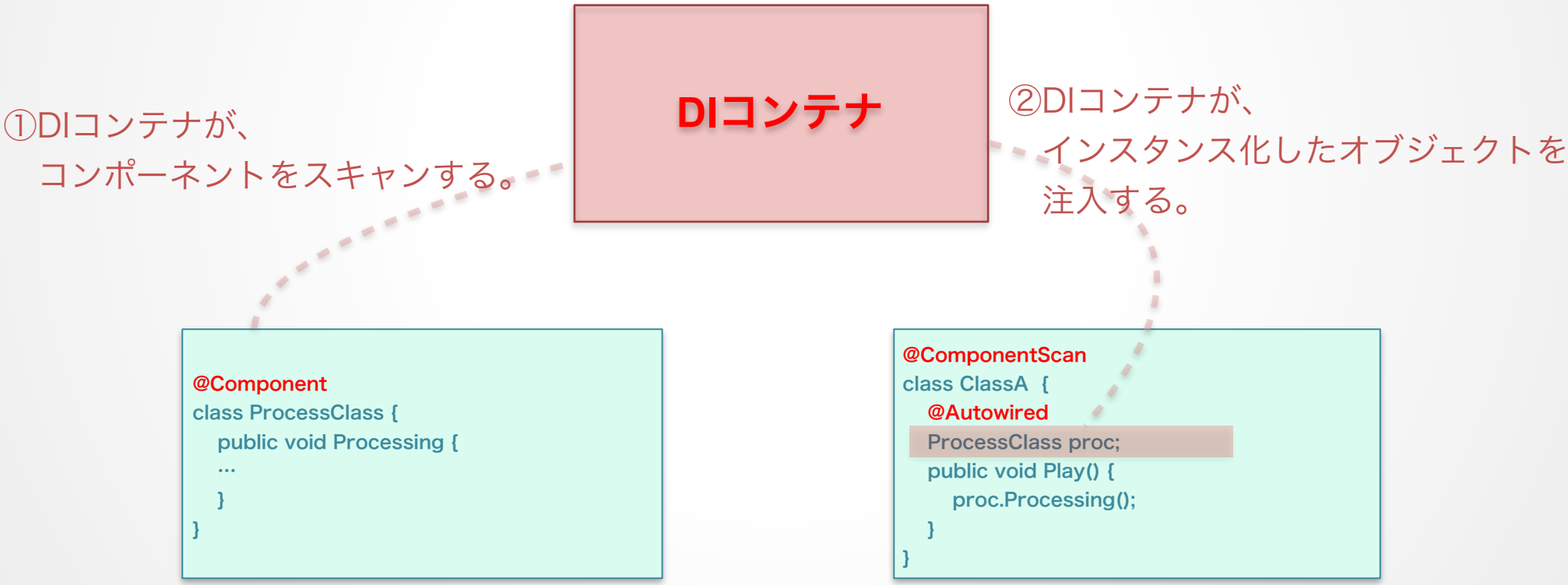
ClassAは、ProcessClassに依存している。
つまり、クラス間の結合度が高い状態である。



2, DIコンテナ 未使用&使用時の違い

DIコンテナ使用の場合

ClassAは、ProcessClassに依存していない。
つまり、クラス間の結合度が低い状態である。



3, 主なアノテーション

DIコンテナを使用する際、以下の3つを用いればDIを実現出来る。
細かい調整が必要な場合は、他のアノテーションを用いる。

@Autowired

DIコンテナからオブジェクトを注入したい時に付ける。
DIコンテナがインスタンス化してくれる。

@Component

コンポーネントとして定義させたいクラスに付ける。
それにより、DI対象のクラスになる。

@ComponentScan

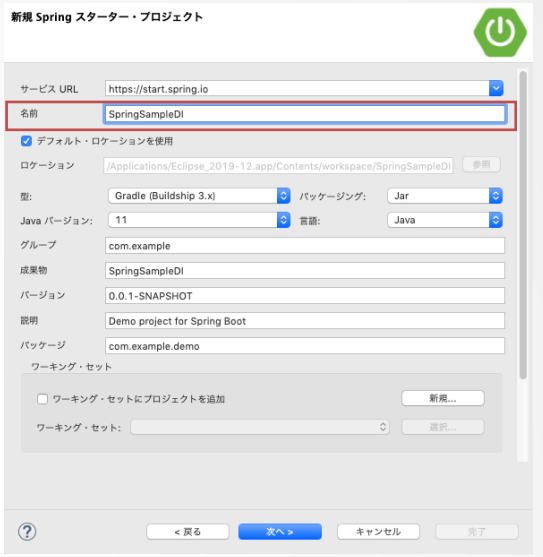
依存性を注入（DI）させたいクラスに付ける。
コンポーネントを自動でスキャンしてくれる。

4, DIコンテナ 未使用&使用サンプル

4-1, プロジェクトの作成

1, Spring スターター・プロジェクト作成

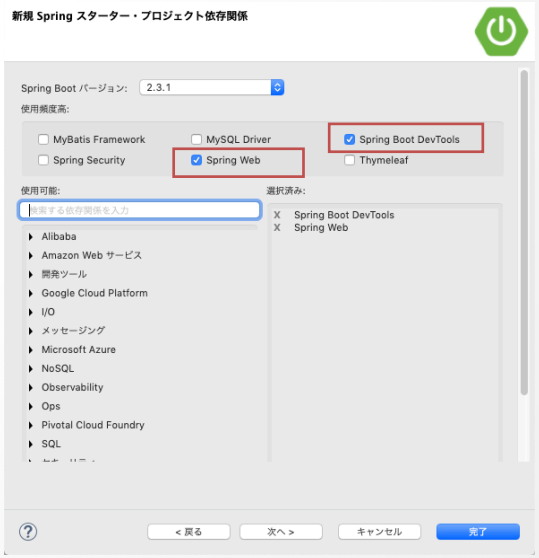
名前：SpringSampleDI
型：Gradle (Buildship 3.x)
Java バージョン：11



2, 新規 Spring スターター・プロジェクト依存関係

下記の項目をチェック。
・Spring Web
・Spring Boot DevTools

※画面上部に表示されていない場合は、検索欄を使用。

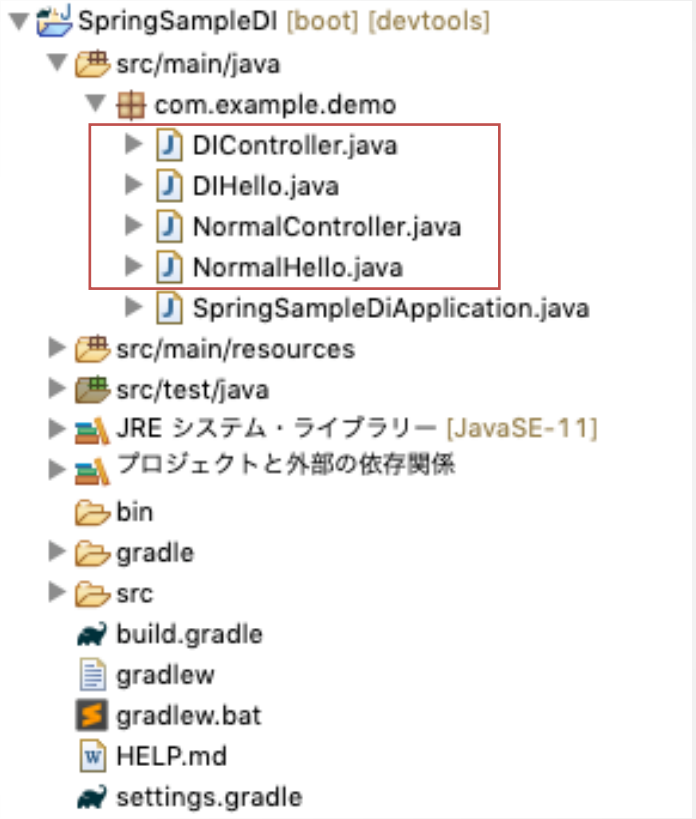


4, DIコンテナ 未使用&使用サンプル

4-1, プロジェクトの作成

3, 追加ファイル

Javaクラスファイル
- DIController.java
- DIHello.java
- NormalController.java
- NormalHello.java



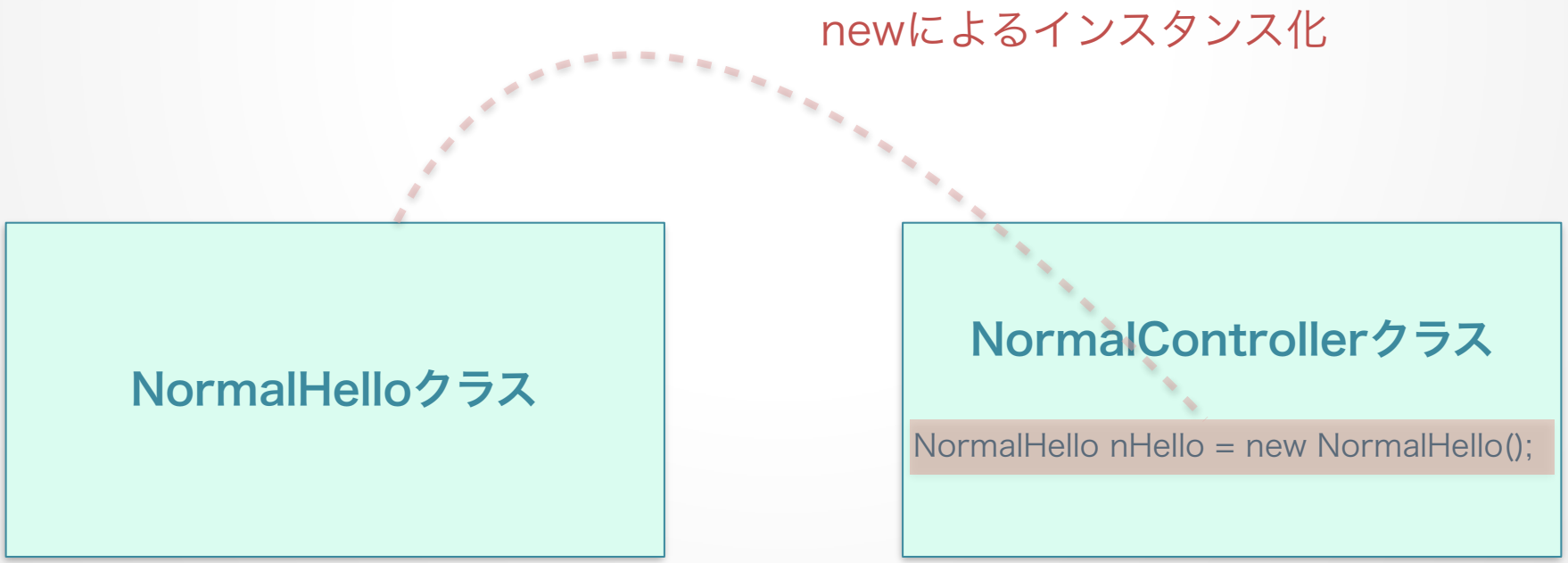
※ファイル内容は、サンプルファイルよりコピー&ペースト。

4, DIコンテナ 未使用&使用サンプル

4-2, プロジェクトの実施

1, Hello1関数実行の場合

1, プロジェクトを実行する。
2, "http://localhost:8080/hello1"へ、アクセスする。
3, ブラウザに、「Hello Hello Hello Normal World !」が出力される。

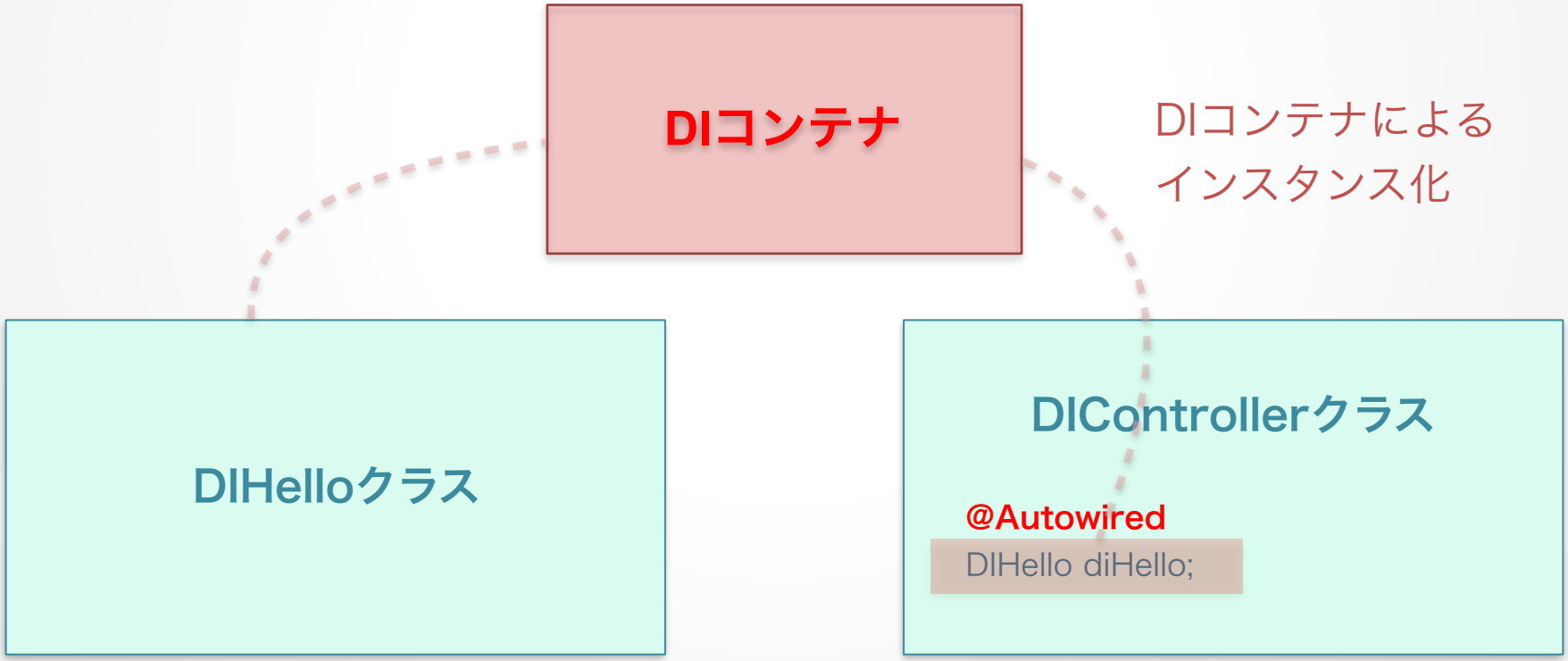


4, DIコンテナ 未使用&使用サンプル

4-2, プロジェクトの実施

2, Hello2関数実行の場合

1, プロジェクトを実行する。
2, "http://localhost:8080/hello2"へ、アクセスする。
3, ブラウザに、「Hello Hello Hello DI World !」が出力される。



DIコンテナ in Spring Boot

Fin.