Agenda

- Spring Projectとは
- Spring Core
- Spring MVC
- Spring Boot
- まとめ

フレームワークとは

フレームワークとは

- システム開発で頻繁に使われる処理を提供してくれるもの
    - データアクセス、セッション管理、認証 etc.
- 利用することでビジネスロジックに集中できる

Spring Project

Spring Project

- Java向けのオープンソースフレームワーク
- 複数のコンポーネントの組み合わせでできている
- 公式サイト👉 https://spring.io/

Spring Project の一部

- Spring Framework
- Spring Data
- Spring Security
- Spring Batch
- Spring Boot

Spring Framework の一部

- Spring Core
    - Dependency Injection (DI)
        - Aspect Oriented Programming (AOP)
- Spring MVC
- Spring Testing

今回扱う部分

- Spring Core
    - Dependency Injection (DI)
- Spring MVC
- Spring Boot

Spring Core

Dependency Injection (DI)

- 直訳すると「依存性の注入」
- 利用したいインスタンスを利用する側で生成するのではなく、外から設定してあげること
- （注意）DI自体はSpringだけの機能ではない

Dependency Injection (DI)

- Springが管理するインスタンス = Bean
- Bean定義したらDIコンテナに登録される
- 利用する側のクラスでBeanをインジェクションして使う

Dependency Injection (DI)

- Springが管理するインスタンス = Bean

- Bean定義したらDIコンテナに登録される

- 利用する側のクラスでBeanをインジェクションして使う

Bean定義とは

- Springに管理してほしいインスタンスだとわかるように定義すること

Bean定義の方法3つ

- XMLベース
- Java Configベース 👈 今回はこれ
- アノテーションベース 👈 今回はこれ

Bean定義 - Java Config

- @ConfigurationをつけたJavaクラスを用意
- そのクラスに@Beanをつけたメソッドを作成
- そのメソッドはBeanとして定義したいクラスをnewしたものを戻り値にとる

＼具体例はハンズオンパートにて／

Bean定義 - アノテーション

- Bean登録したいクラスだとわかるような目印（=アノテーション）をつける
- アノテーションの種類は色々ある

  @Controller, @Service, @Repository, @Component, etc.

＼具体例はハンズオンパートにて／

Dependency Injection (DI)

- Springが管理するインスタンス = Bean

- Bean定義したらDIコンテナに登録される

- 利用する側のクラスでBeanをインジェクションして使う

DIコンテナとは

- Springは起動時にBean定義の目印がついたものを探す
- 見つかったらDIコンテナに登録する
- 目印を探すことをコンポーネントスキャンという

DIコンテナ登録の図 ①

開発者がアノテーションをつけて

Bean定義する

@Service

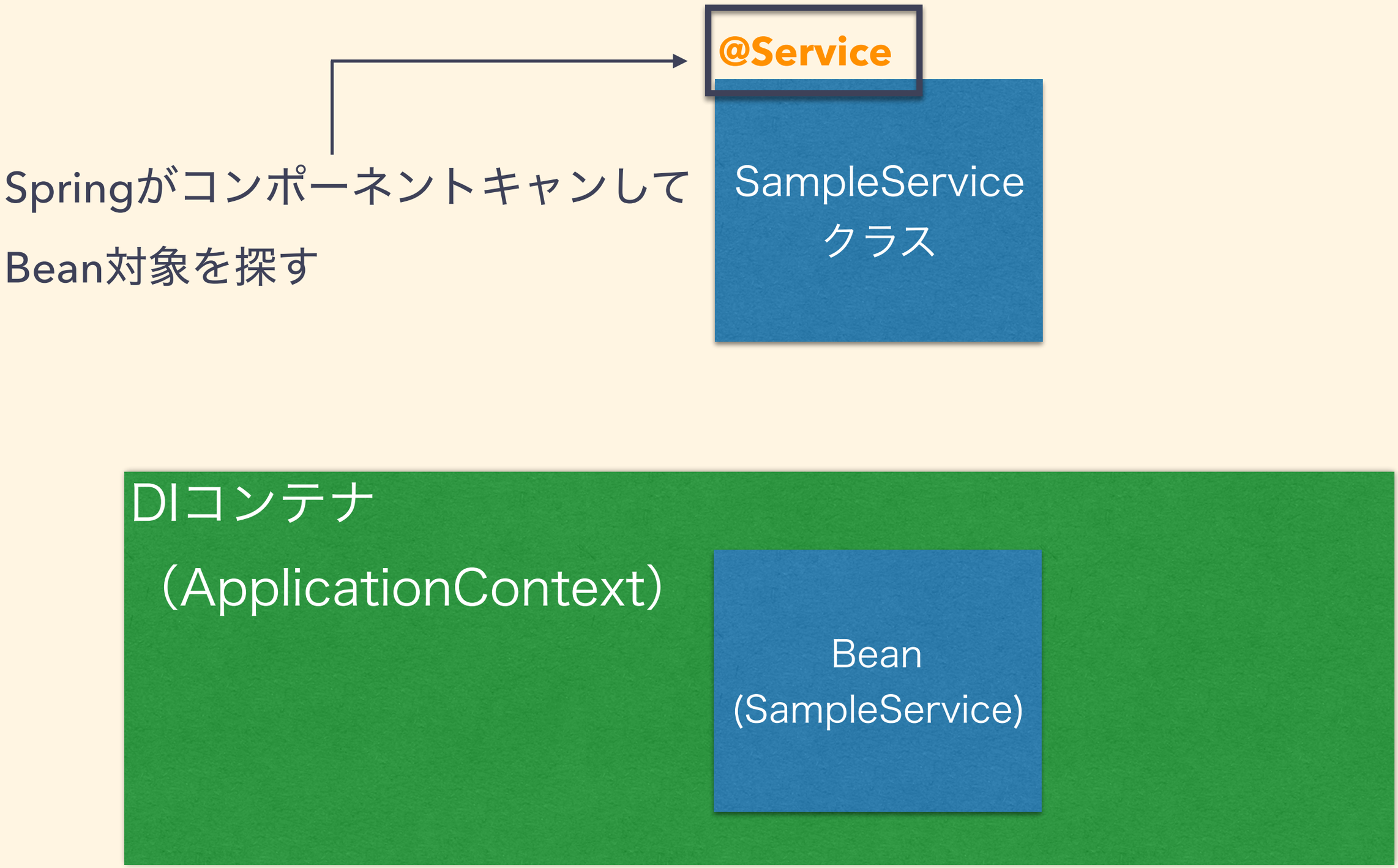
SampleService
クラス

DIコンテナ

（ApplicationContext）

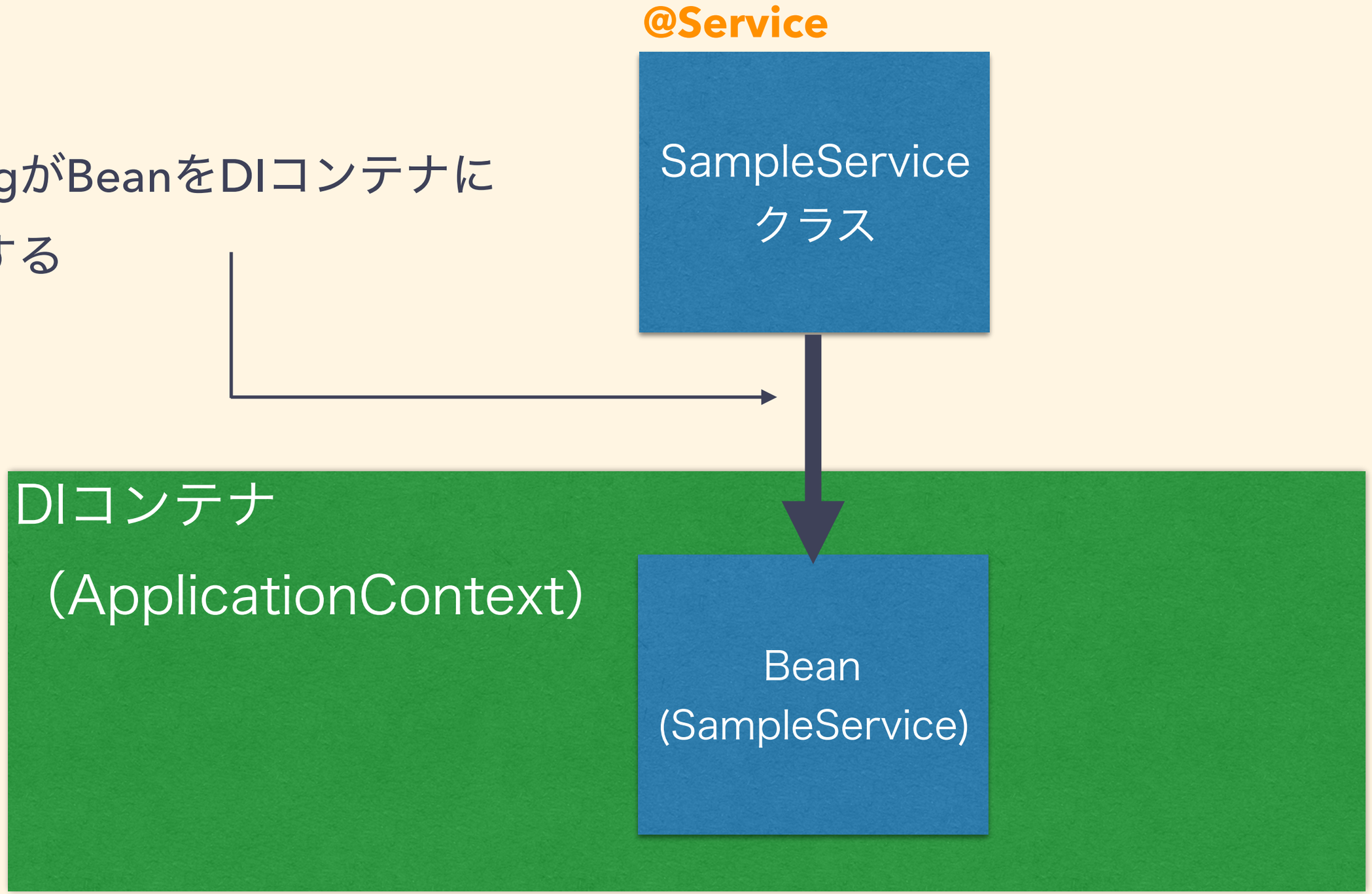
Bean
(SampleService)

DIコンテナ登録の図 ②



DIコンテナ登録の図 ③

SpringがBeanをDIコンテナに
登録する



Dependency Injection (DI)

- Springが管理するインスタンス = Bean
- Bean定義したらDIコンテナに登録される
- 利用する側のクラスでBeanをインジェクションして使う

インジェクションとは

- クラス間の依存関係を外から入れてあげること

インジェクションの方法3つ

- フィールドインジェクション
- セッターインジェクション
- コンストラクタインジェクション 👈 今回はこれ

コンストラクタインジェクション

- Beanを利用したいクラスのコンストラクタの引数にインジェクションしたいBeanを渡し、自身のフィールドにセットしていく

- コンストラクタに@Autowiredをつける（コンストラクタが一つの時は省略可能）

＼具体例はハンズオンパートにて／

インジェクションの図 ①

@Controller

SampleController

private final SampleService sampleService

@Autowired

```
SampleController(SampleService sampleService) {
    this.sampleService = sampleService
}
```

@Service

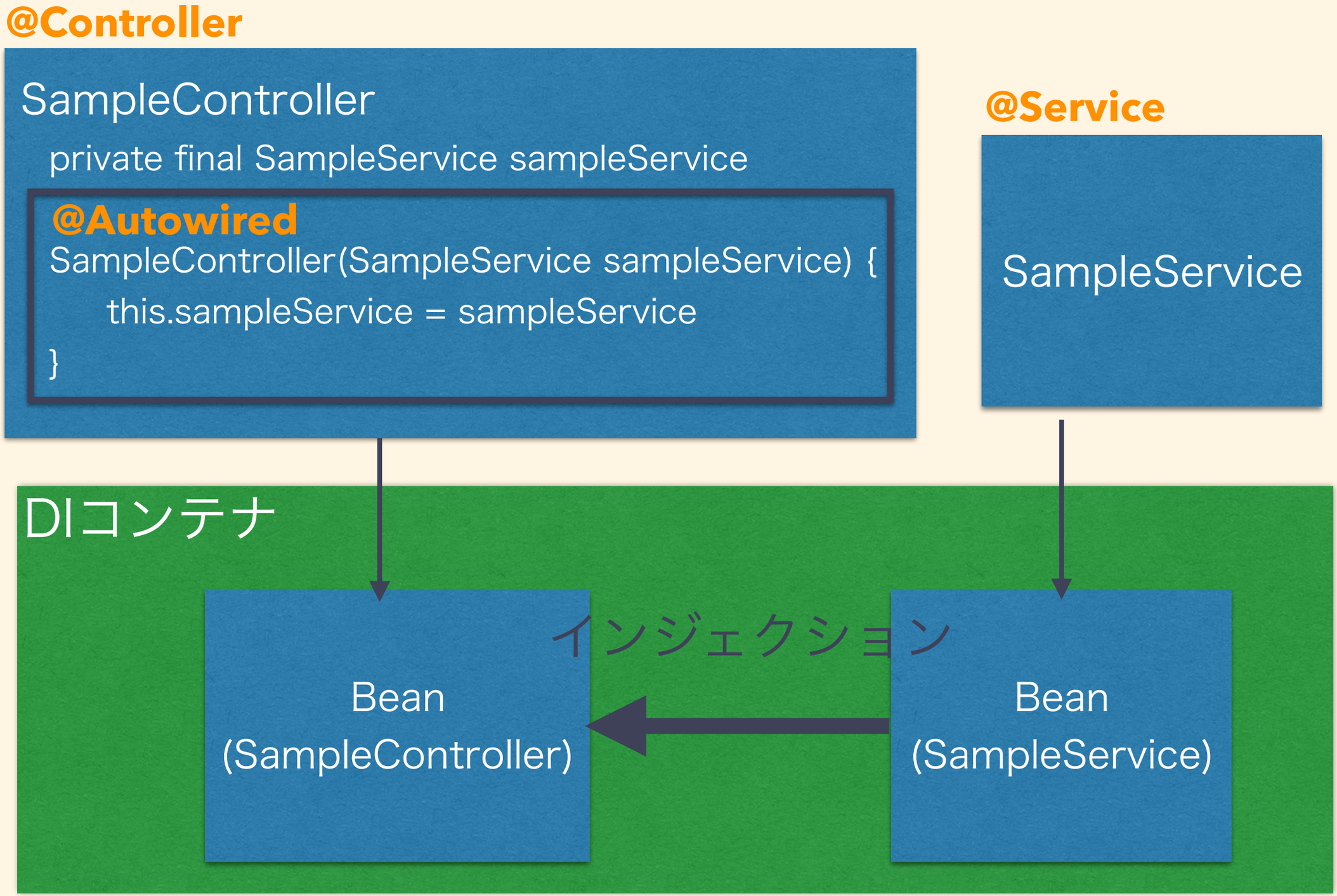
SampleService

DIコンテナ

Bean
(SampleController)

Bean
(SampleService)

インジェクションの図 ①



DIで何がうれしいか

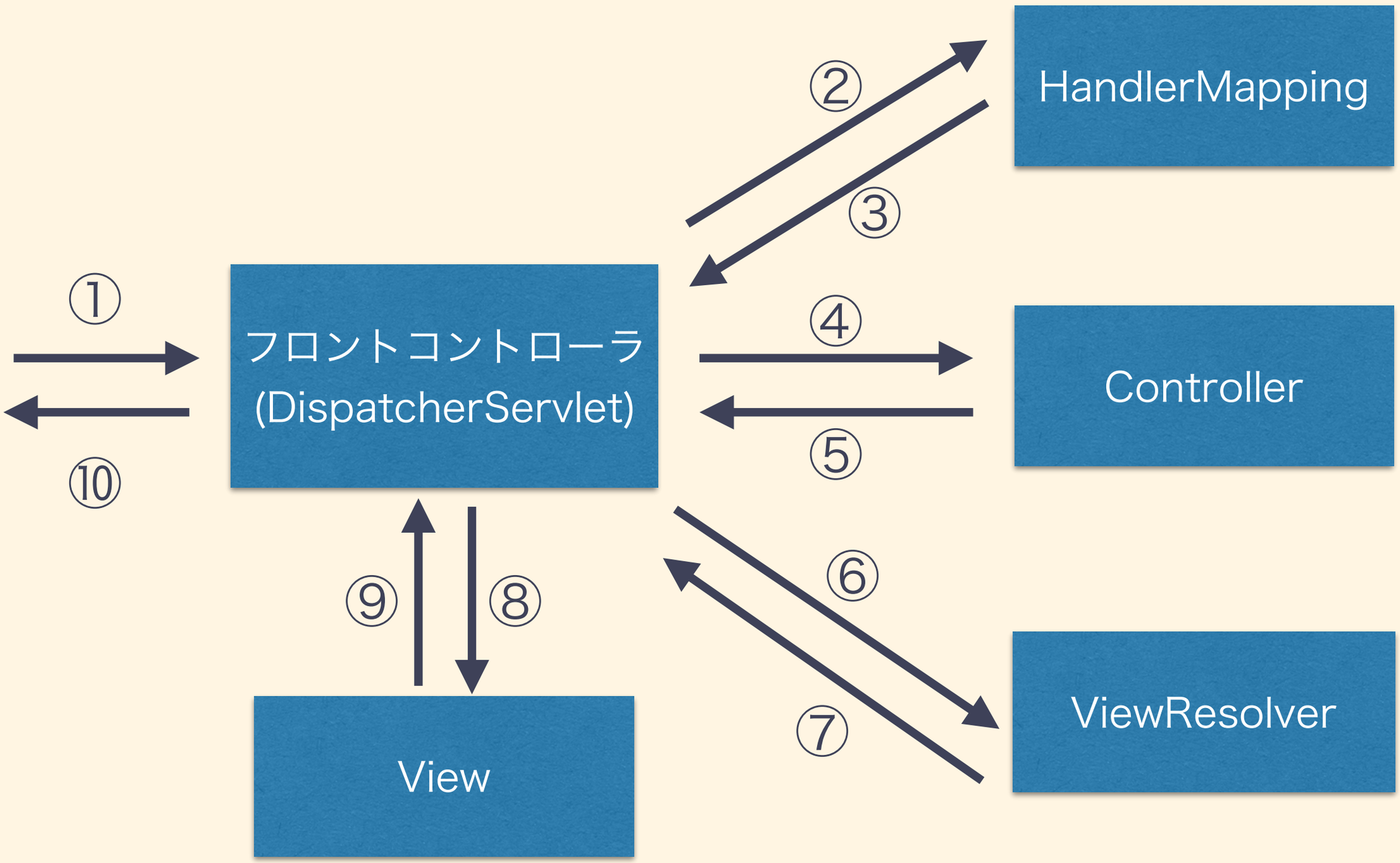
- インスタンスの作成や初期化処理などを
  切り出せるのでコードの見通しが良くなる
- DIコンテナにインスタンスのスコープや
  ライフサイクルを管理してもらえる
- テストでインスタンスの差し替えがしやすく
  なる

Spring MVC

Spring MVC

- SpringでWebアプリケーションを開発する際に利用するフレームワークの一つ
- MVCパターンを採用
- リクエストマッピングなどをアノテーションを利用して設定できる

Spring MVC の図



Spring MVC の内部処理

① 外部からリクエストがやってくる

② DispatcherServletがHandlerMappingにリクエストする

③ HandlerMappingがリクエストに応じたControllerを返してくれる

④ DispatcherServletがControllerに処理を委譲する

⑤ ControllerがView名とModelを返す

⑥ DispatcherServletがView名をViewResolverに渡す

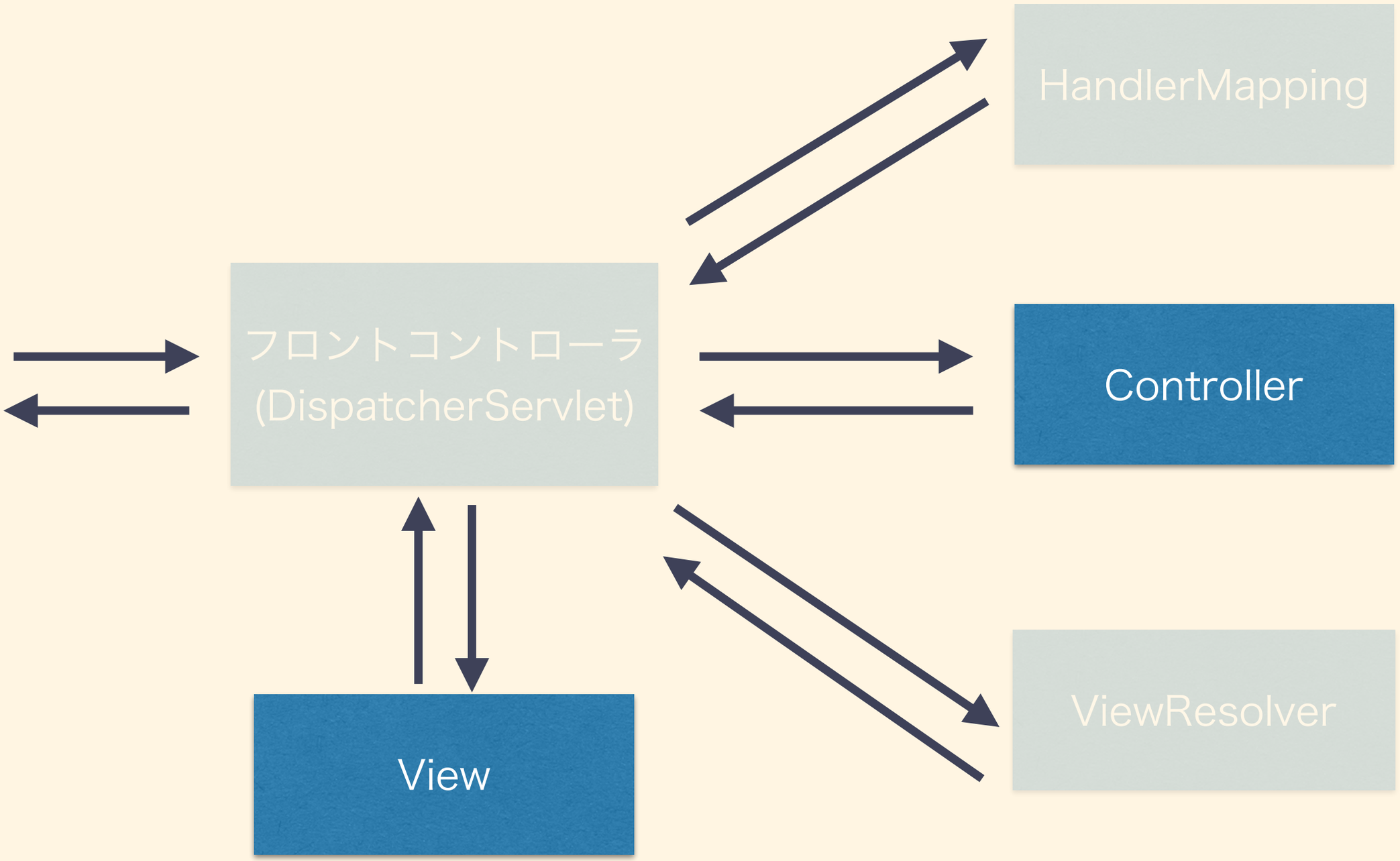
⑦ ViewResolverがViewの完全パスを返す

⑧ DispatcherServletがViewにModelを渡す

⑨ ViewはModelを元にHTMLを組み立てて返す

⑩ 外部にHTMLレスポンスを返す

今回扱う部分


HandlerMapping
フロントコントローラ
(DispatcherServlet)
Controller
View
ViewResolver

Controllerについて

- HTTPリクエストをマッピングしたり
  例外ハンドリングを行う部分
- @Controller, @RestControllerアノテーション
  をつける

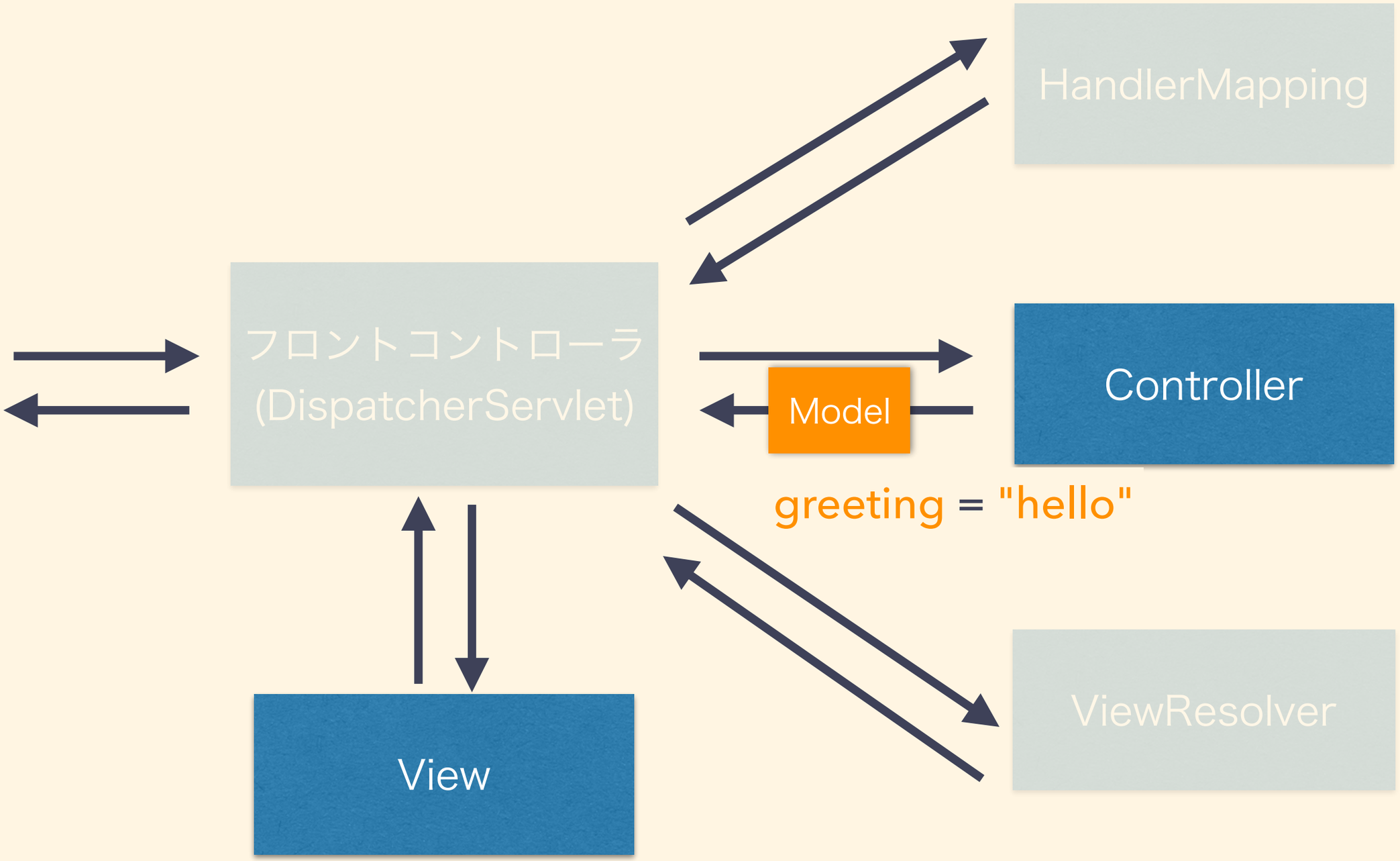
RequestMapping

- Controllerに用意したメソッドにつけるアノテーション
- このURLにこのHTTPメソッドのリクエストが来たらこの処理、と振り分けるための設定

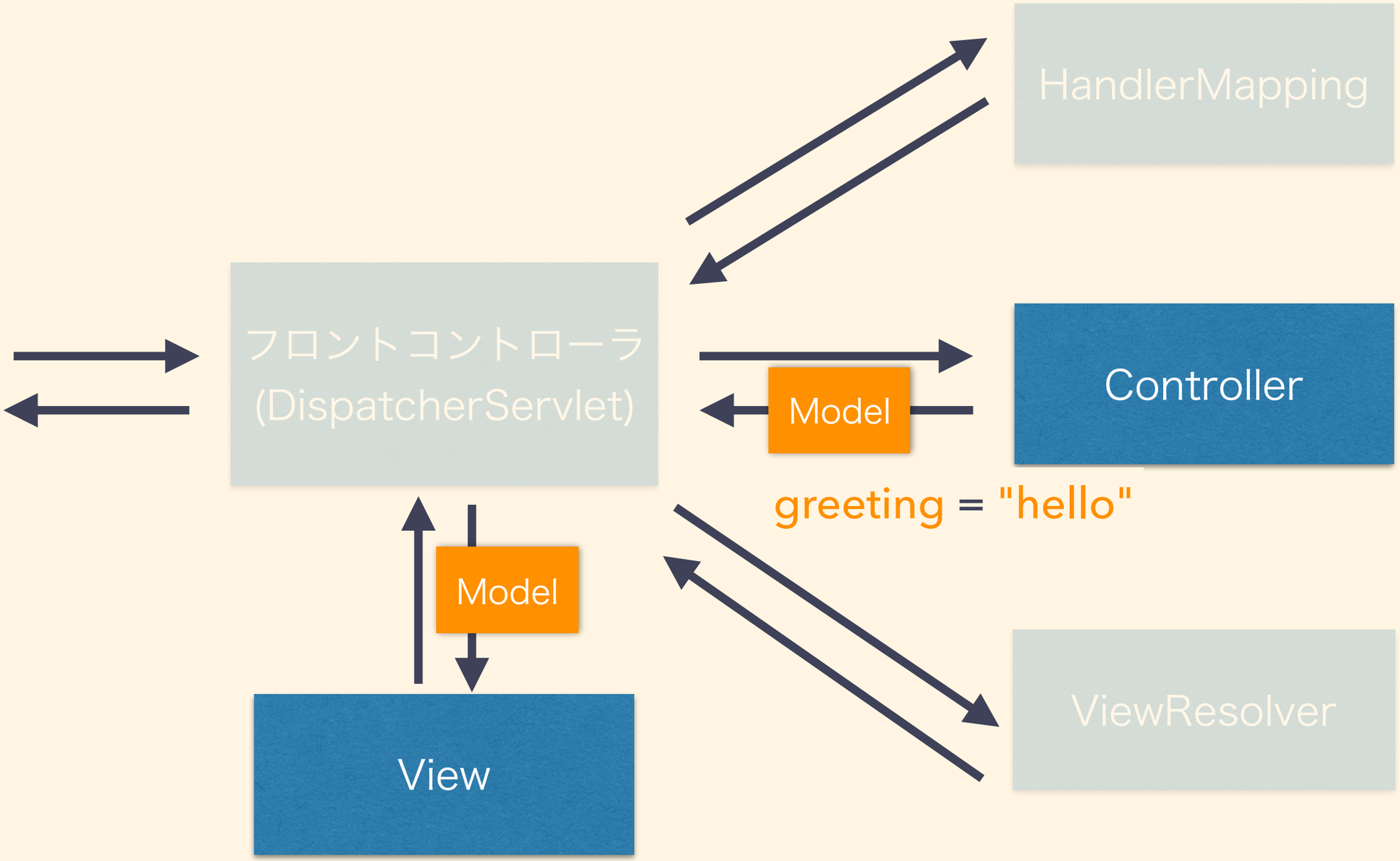
Modelについて

- Viewテンプレートに値を受け渡すために使用される
- ControllerでModelに情報を詰めて返す
- Modelの中の情報はViewが見れる位置に置かれる

Modelについての図

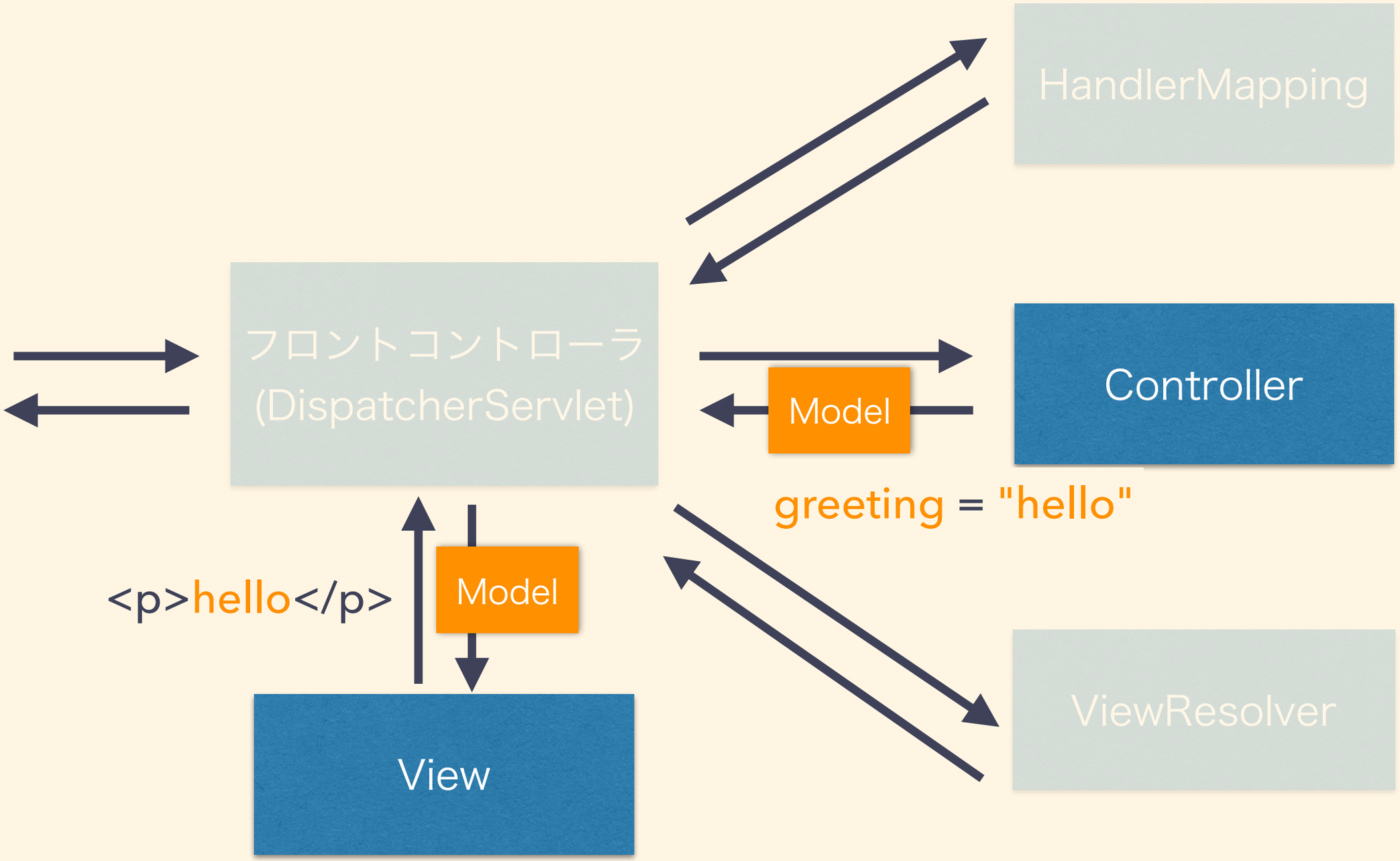


Modelについての図




```
<p th:text="${greeting}"></p>
```

Modelについての図




```
<p th:text="${greeting}"></p>
```

フロントコントローラの設定

- 自動でフロントコントローラを使えるようにするには、Java Configクラスに
@EnableWebMvcアノテーションをつける

Spring Boot

Spring Boot

- Springをより手軽に、最小限の設定で使えるようにしたフレームワーク

Spring Boot の特徴

- Auto Configuration
- Spring Boot Starter
- Stand-alone Spring Application
- etc.

Spring Boot の特徴

- Auto Configuration
- Spring Boot Starter
- Stand-alone Application

Auto Configuration

- Springを使用するには様々な設定が必要
- Spring Bootには多くの設定（Java Config）がデフォルトで用意されている
- @EnableAutoConfigurationを付けると必要な設定を自動で読み込んでくれる

Spring Boot の特徴

- Auto Configuration
- Spring Boot Starter
- Stand-alone Application

Spring Boot Starter

- Spring Bootには必要なライブラリをまとめてくれるStarterという仕組みがある
- 依存関係にspring-boot-starter-xxxを追加するだけで関連するライブラリをまとめて取得してくれる

今回使うStarter

- spring-boot-starter-web
  - webアプリケーションを作成するため
- spring-boot-starter-thymeleaf
  - ThymeleafというViewテンプレートエンジンを利用するため

Spring Boot の特徴

- Auto Configuration
- Spring Boot Starter
- Stand-alone Application

Stand-alone Application

- jarを生成する際に組み込みサーバを含んでくれるので単体で実行可能
- warを生成してサーバにデプロイしなくても実行できる

まとめ

Spring & Spring Boot まとめ

- 複数のコンポーネント（Spring Core, Spring MVC, etc.）の組み合わせでできている
- Springの根幹はDIにあり
- Springをより手軽に使えるようにしたのが Spring Boot

参考資料

参考資料（Web）

- Spring 公式サイト
  - https://spring.io/
- 初めてでも30分で分かるSpring 5 & Spring Boot 2オーバービュー
  - https://www.slideshare.net/masatoshitada7/30spring-5-spring-boot-2-103523666

参考資料（書籍）

- Spring徹底入門