**Spring**

**Springプロジェクト**

・springスタータープロジェクト spring boot を使ったプロジェクト

・springレガシープロジェクト spring boot を使わず個々のフレームワークを個別に組み込むプロジェクト

**Templstes**

・Simple Java

・Simple Spring Maven

・Simple Spring Web Maven

**Mavenプロジェクト**

基本的にアーキタイプを選択し、maven-archetype-quickstartを選択する

pomにjunitやエンコーディングなどの情報が最初から記述されている

**MavenプロジェクトのJREシステムライブラリを1.8にする**

・propertiesタグに追加

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

・maven-compiler-pluginを追加

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

**DI**

**DIコンテナへのBeanの登録**

|  |  |
| --- | --- |
| **XMLベース** | **<bean>タグ** |
| **Javaベース** | **@Configurationと@Bean** |
| **アノテーションベース** | **@Componentとcomponent-scan** |

**XMLベース**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id=" userRepository" class="com.example.demo.UserRepositoryImpl" />

<bean id=" passwordEncoder " class="com.example.demo.BCriptPasswordEncoder " />

<bean id=" userService" class="com.example.demo.UserServiceImpl">

<constructor-arg ref=”userRepository”>

<constructor-arg ref=”passwordEncoder”>

</bean>

</beans>

・idがBean名

・classがインスタンス(完全修飾クラス名)

**Javaベース**

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

UserRepository userRepository () {

return new UserRepository ();

}

@Bean

PasswordEncoder passwordEncoder () {

return new PasswordEncoder ();

}

@Bean

UserService userService () {

return new UserService ();

}

}

・メソッド名がBean名となる

**アノテーションベース**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype

@Component

public class UserRepositoryImpl implements UserRepository {

// ・・・

}

Bean定義(Java)

import org.springframework.context.annotation.ComponentScan

@Configuration

@ComponentScan(“com.example.demo”)

public class AppConfig {

}

Bean定義(XML)

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="com.example.demo" />

</beans>

**ApplicationContext**

|  |  |
| --- | --- |
| ClassPathXmlApplicationContext | XMLベース |
| AnnotationConfigApplicationContext | Javaベース |
| アノテーションベース |

ApplicationContext context = new ClassPathXmlApplicationContext(applicationContext.xml);

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.java);

・Bean DIコンテナに登録するコンポーネント

・Bean定義 Configuration

・ルックアップ DIコンテナからBeanを取得すること

getBean(class)

getBean("id")

getBean("id", class)

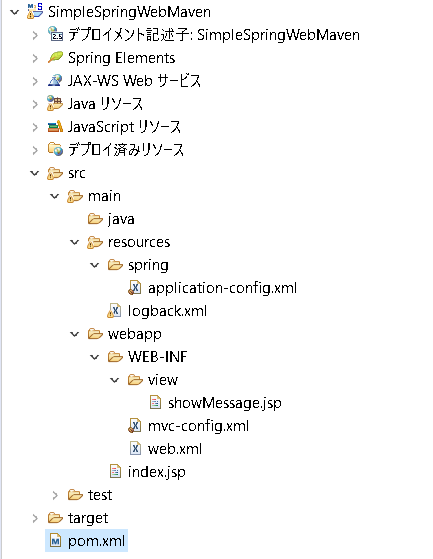
**インジェクション**

・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

・フィールドインジェクション

**Spring Web プロジェクト**



POMデフォルト

<project

xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.springframework.samples.service.service</groupId>

<artifactId>SpringWeb</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<!-- Generic properties -->

<java.version>1.6</java.version>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<!-- Web -->

<jsp.version>2.2</jsp.version>

<jstl.version>1.2</jstl.version>

<servlet.version>2.5</servlet.version>

<!-- Spring -->

<spring-framework.version>3.2.3.RELEASE</spring-framework.version>

<!-- Hibernate / JPA -->

<hibernate.version>4.2.1.Final</hibernate.version>

<!-- Logging -->

<logback.version>1.0.13</logback.version>

<slf4j.version>1.7.5</slf4j.version>

<!-- Test -->

<junit.version>4.11</junit.version>

</properties>

<dependencies>

<!-- Spring MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring-framework.version}</version>

</dependency>

<!-- Other Web dependencies -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>${servlet.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>${jsp.version}</version>

<scope>provided</scope>

</dependency>

<!-- Spring and Transactions -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring-framework.version}</version>

</dependency>

<!-- Logging with SLF4J & LogBack -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>${logback.version}</version>

<scope>runtime</scope>

</dependency>

<!-- Hibernate -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>${hibernate.version}</version>

</dependency>

<!-- Test Artifacts -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring-framework.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**デフォルト依存関係**

<artifactId>

spring-context

spring-tx

slf4j-api

logback-classic

hibernate-entitymanager

spring-test

junit

**pom修正点**

・Javaのバージョン

・Springのバージョン

・servlet-apiのアーティファクトID

　servlet-api ⇒ javax-servlet-api

**web.xml修正点**

・バージョン変更

・application-configの場所

・ContextLoaderListener

・エンコーディングフィルター

**リソース**

**StaticApplicationContext**

とりあえずApplicationContextを用意したい場合

**Resource**

Resource res = app.getResource(“classpath:sample.txt”);

・getURI()

・getFilename()

・getInputStream()

**リソース文字列**

|  |  |
| --- | --- |
| classpath: | src/main/resources フォルダ（相対パス） |
| file: | ファイルシステム（絶対パス） |
| http: | ネットワーク上のリソース |

・src/main/javaとsrc/main/resourcesはビルドすると同じ場所にまとめられる

**ClassPathXmlApplicationContext**

src/main/resourcesから読み込む場合

**FileSystemXmlApplicationContext**

ファイルシステムから読み込む場合

**データアクセス**