Spring

目次

[DIとSpring Framework 2](#_Toc502767912)

[DIとは何か？ 2](#_Toc502767913)

[Spring Frameworkの内容 2](#_Toc502767914)

[Springのファイル構成 2](#_Toc502767915)

[Spring利用の基本 2](#_Toc502767916)

[Springの基本 3](#_Toc502767917)

[Beanクラスの定義 3](#_Toc502767918)

[Bean定義ファイルの作成 3](#_Toc502767919)

[サーブレットからBeanを利用する 4](#_Toc502767920)

[プロパティの利用 5](#_Toc502767921)

# DIとSpring Framework

## DIとは何か？

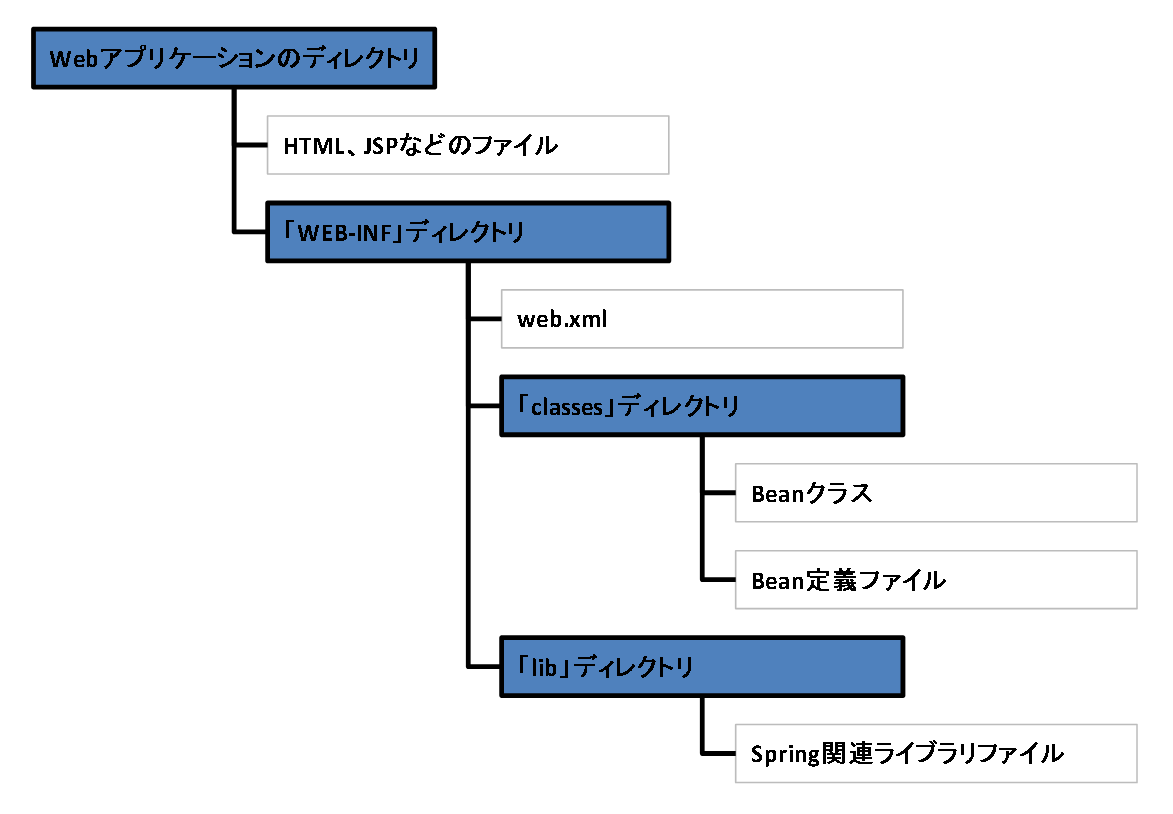
Dependency Injection（依存性の注入）

→外部のオブジェクトに依存する部分を中に含めず、実行する際に外部から注入する。

## Spring Frameworkの内容

|  |  |
| --- | --- |
| Spring Core | Springの根幹技術。DIに必要なオブジェクトの生成や関連付け、  それに必要なBeanコンテナと呼ばれる機能等を提供 |
| Spring Context | JavaBeanにアクセスするための機能 |
| Spring AOP | アスペクト指向プログラミングのためのフレームワーク |
| Spring Web MVC | Webアプリケーション開発のためのMVCアーキテクチャのフレームワーク |
| Spring Web | Webアプリケーション開発のためのStruts等のフレームワークを統合するための機能 |
| Spring DAO | JDBCを抽象化したDAO機能 |
| Spring ORM | Hibernate等のORMを統合するための機能 |

## Springのファイル構成



## Spring利用の基本

必要なライブラリファイルは以下の通り

・spring.jar

・「maven」フォルダ

・「module」フォルダ

・「resources」フォルダ

・「weaving」フォルダ

# Springの基本

## Beanクラスの定義

▼sample01.SampleBean

**package** sample01;

**import** java.io.PrintWriter;

**public** **class** SampleBean {

**public** **void** doTest(PrintWriter out){

out.println("<h3>Hello Spring Framework!</h3>");

out.println(**this**.getClass().getName());

}

}

## Bean定義ファイルの作成

▼bean-conf.xml

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

<!DOCTYPE beans PUBLIC

"-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id="sample01.test" class="sample01.SampleBean" />

</beans>

|  |  |
| --- | --- |
| beansタグ | 内部にbeanタグを記述 |
| beanタグ | Springで使用するタグを定義  　id属性　　　→beanの命名  class属性　→対応するJavaクラスを記述 |

## サーブレットからBeanを利用する

▼sample01.SampleServlet

**package** sample01;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.ClassPathResource;

**import** org.springframework.core.io.Resource;

**public** **class** SampleServlet **extends** HttpServlet {

**private** **static** **final** **long** **serialVersionUID** = 1L;

**public** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** IOException{

response.setContentType("text/html; charset=utf-8");

request.setCharacterEncoding("utf-8");

Resource **resorce** = **new** ClassPathResource("bean-conf.xml");

XmlBeanFactory **beanFactory** = **new** XmlBeanFactory(resorce);

SampleBean **test** = (SampleBean)beanFactory.getBean("sample01.test");

PrintWriter **out** = response.getWriter();

out.println("<html><head></head>");

out.println("<body>");

out.println("<p>" + **this**.getClass().getName() + "</p>");

test.doTest(out);

out.println("</body></html>");

}

}

1. Resourceクラスの準備

ClassPathResourceクラスをBean定義ファイルを引数に生成する

1. Factoryクラスを準備
   1. で準備したResourceクラスを元にBean作成のためのFactoryクラスを生成する
2. FactoryクラスからBeanを取得

Bean定義ファイルで設定したidを元にBeanクラスを生成する

## プロパティの利用

▼bean-conf.xml

<bean id="sample02.bean" class="sample02.SampleBean">

<property name="title">

<value>TITLE</value>

</property>

<property name="message">

<value>message.</value>

</property>

</bean>

|  |  |
| --- | --- |
| propertyタグ | name属性　→Beanクラスで定義したフィールド属性を指定  内部にrefタグもしくはvalueタグを記述 |
| refタグ | bean属性　→Beanのidを指定 |
| valueタグ | 内部で属性に設定する値を記述する |

# Spring MVC

▼web.xml

<servlet>

<servlet-name>controller</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>controller</servlet-name>

<url-pattern>\*.spg</url-pattern>

</servlet-mapping>

基本的に通常の記述方法と同一。

|  |  |
| --- | --- |
| servlet-classタグ | SpringのDispatcherServletを指定する |
| url-patternタグ | Bean名.spgで記述（上記では＊で記述） |

## Controllerの作成

▼sample05.SampleController

**package** sample05;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.web.servlet.ModelAndView;

**import** org.springframework.web.servlet.mvc.Controller;

**import** common.HtmlTags;

**public** **class** SampleController **extends** HtmlTags **implements** Controller {

**private** SampleBean bean;

**public** SampleBean getBean() {

**return** bean;

}

**public** **void** setBean(SampleBean bean) {

**this**.bean = bean;

}

**public** ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response) **throws** IOException{

response.setContentType("text/html; charset=utf-8");

request.setCharacterEncoding("utf-8");

bean.setText(request.getParameter("text"));

PrintWriter **out** = response.getWriter();

out.println(

html(

head(**META**),

body(

p(**this**.getClass().getName()),

bean.getText())

)

);

**return** **null**;

}

}

ポイント！　　Controllerインタフェースを実装する。（handleRequestメソッドをオーバーライド）

※ModelAndViewクラスはModelとViewの名前を管理するためのクラス

## Bean定義ファイルと表示用JSPの作成

▼controller-servlet.xml

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

<!DOCTYPE beans PUBLIC

"-//SPRING//DTD BEAN//EN"

"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean name="/SampleController.spg"

class="sample05.SampleController">

<property name="bean">

<ref bean="sample05.bean" />

</property>

</bean>

<bean id="sample05.bean" class="sample05.SampleBean"/>

</beans>

※「WEB-INF」内に配置

※「web.xmlのサーブレット名-servlet.xml」という名前で作成する

|  |  |  |
| --- | --- | --- |
| beanタグ | name属性 | URLパターンを記述 |
| class属性 | 作成したコントローラークラスを指定 |
| propertyタグ | ※前述の通り |

# Strutsとの連携

※準備中

# JDBCを使ったデータベースアクセス

## データベースアクセスのBeanクラス作成

▼sample07.SampleBean

**package** sample07;

**import** java.util.List;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.jdbc.datasource.DriverManagerDataSource;

**public** **class** SampleBean {

**private** DriverManagerDataSource datasource;

**public** DriverManagerDataSource getDatasource(){

**return** **this**.datasource;

}

**public** **void** setDatasource(DriverManagerDataSource datasource){

**this**.datasource = datasource;

}

**public** List getAll(){

JdbcTemplate **jt** = **new** JdbcTemplate(**this**.datasource);

**final** String **sql** = "select \* from member";

**return** jt.queryForList(sql);

}

}

ポイント①

→フィールドにDriverManagerDataSourceクラス（DateSourceを実装）を持たせる

ポイント②

→JDBCにアクセスするためのテンプレートクラス（JdbcTemplate）を取得

ポイント③

→JdbcTemplateのインスタンスでクエリを実行し、結果を取得

　※取得するのはList<Map<String column, Object value>> 1レコードが1つのマップで返ってくるイメージ

## DriverManagerDataSourceのBean定義

▼bean-conf.xml

<bean id="sample07.bean" class="sample07.SampleBean">

<property name="datasource">

<ref local="sample07.datasource" />

</property>

</bean>

<bean id="sample07.datasource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName">

<value>oracle.jdbc.driver.OracleDriver</value>

</property>

<property name="url">

<value>jdbc:oracle:thin:@localhost:1521:XE</value>

</property>

<property name="username">

<value>TEST</value>

</property>

<property name="password">

<value>TEST</value>

</property>

</bean>

フィールドに持たせたDriverManagerDataSourceクラスの定義情報を記述する

|  |  |
| --- | --- |
| driverClassName | 各ベンダーのDriverクラスを記述 |
| url | アクセス先のURLを記述 |
| username | 接続時のユーザー名を記述 |
| password | 接続時のパスワードを記述 |