目次

DIとSpring Framework

DIとは何か？

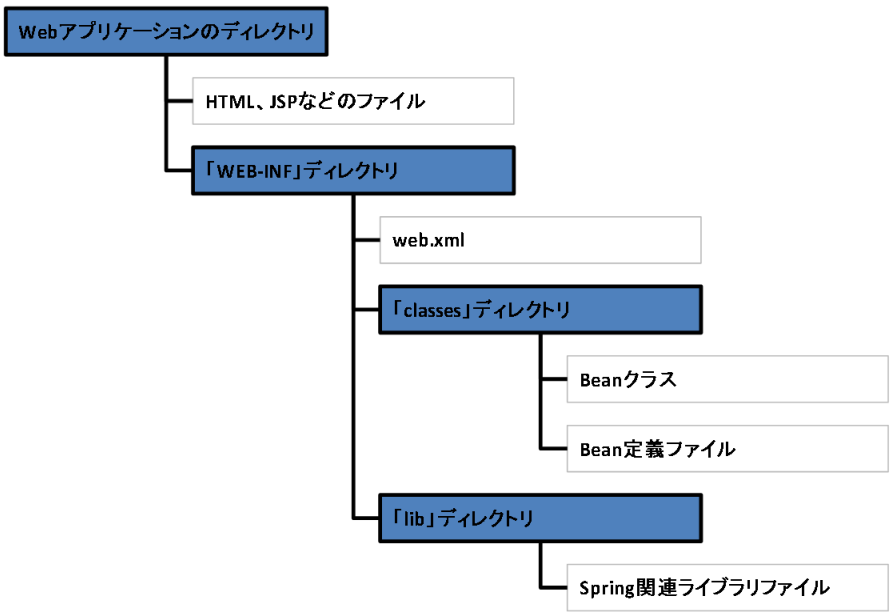
Dependency Injection（依存性の注入）

→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

② 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

③ 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