|  |
| --- |
| Spring 공부 내용 정리  2015년 5월 22일 |
| 통합ICTO팀 이영주 사원 |

**Spring Framework** 일반 자바 객체의 생성 -> 사용 -> 소멸 자체를 스프링 프레임워크에게 위임(xml, 어노테이션 설정). servlet / jsp은 web container

**Java bean = DTO = VO** 스프링에서의 자바 빈 = 스프링 프레임워크 관리하는 모든 자바 객체들. 설정파일은 <bean> tag로 사용

**Spring Framework 특징**

1. Entrerprise Application에서 필요로 하는 기능 제공
2. 경량(lightweight) 애플리케이션 컨테이너
3. Dependency Injection[DI] 지원
   * Spring은 설정 파일이나 어노테이션을 통해서 객체간의 의존 관계를 설정 할 수 있도록 함. 객체는 의존하고 있는 객체를 코드 상에서 직접 생성하거나 검색할 필요 없음
4. Aspect Oriented Programming[AOP] 지원
   * 자체적으로 AOP를 지원하고 있기 때문에 트랜잭션이나 로깅, 보안과 같이 여러 비즈니스 모듈에서 공통적으로 필요로 하는 공통관심 사항을 핵심 로직과 분리시켜 각 모듈에 적용할 수 있음 => 중복 코드 삭제
5. Plane Old Java Object[POJO] 지원 –일반 자바 클래스들
6. Servlet 상속받은 사용자정의 Servlet클래스. POJO 아님
7. 트랜잭션 처리를 위한 일관된 방법 제공
8. 영속성과 관련된 다양한 API 지원 및 연동 지원
   * Spring은 JDBC API를 비롯하여 iBatis/Hibernate, JPA등 데이터베이스 처리를 위해 널리 사용되는 library들과의 연동을 지원

**Spring Framework 장점**

1. 개발자들이 개발하고자 하는 애플리케이션 로직 개발에만 집중할 수 있음
2. 개발이 단순해짐
3. POJO 방식의 기술 사용이 가능

**Spring Framework의 모듈**

|  |  |
| --- | --- |
| 모듈 이름 | 설명 |
| Beans | BeanFactory 인터페이스를 통해 구현됨 |
| Core | 프레임워크의 가장 기본적인 부분  컨테이너 기능을 수행하기 위해 의존성 주입[DI] 기능을 제공 |
| Context | spring-core, spring-beans 모듈을 확장해서 국제화, 이벤트 처리, 리소스 로딩, 서블릿 컨테이너를 위한 컨텍스트 생성 등의 기능을 추가로 제공. ApplicationContext 인터페이스를 통해 구현됨 |
| Expression Language | 객체에 접근하고 객체를 조작하기 위한 표현 언어를 제공 |
| AOP | AOP Alliance에 호환되는 AOP 구현을 제공 |
| Aspects | AspectJ와의 통합을 제공 |
| Web(MVC/Remoting) | Spring MVC를 제공하며 struts와도 연동 기능 제공등 웹 관련 기능 지원 |
| Data Access/Integration | JDBC를 위한 템플릿 제공. 따라서 간결한 코드로 JDBC 프로그램 가능, iBatis 및 하이버네이트 등의 ORM api를 위한 통합 레이어 제공. Spring이 제공하는 트랜젝션과의 연동 지원 |

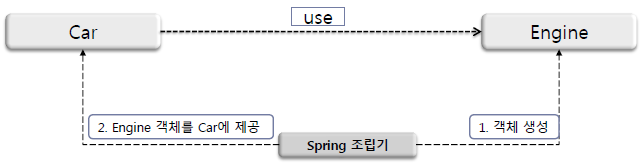
**Spring Project를 위한 구성**

1. Spring Framework library
2. 설정 메타정보 파일(applicationContext.xml)
   * 이 설정 메타데이터는 Spring 컨테이너에 객체 생성 및 관계 설정 내용을 XML 또는 properties[프로퍼티] 파일, 소스코드 어노테이션과 같은 외부 리소스로 작성
3. 자바 소스들
   * Spring Bean
   * Spring Bean 사용 클래스들
   * 이외의 자바 클래스들

**Dependency Injection[DI]** 의존성 주입. Spring Framework가 지원하는 핵심 기능. 객체 사이의 의존 관계가 객체 자신이 아닌 외부(조립기)에 의해 설정됨. 의존 관계는 방향성이 부여되어 있다.

**DI에서 컨테이너의 역할** A 객체가 필요로 하는 의존 관계에 있는 다른 객체 B 객체를 직접 생성하여 A 객체로 주입(설정)해주는 역할을 담당

**DI(Dependency Injection)이란?** 구체적인 의존 객체와 그것을 사용할 객체를 런타임에 연결해주는 작업



**Dependency Injection 장점** 설계가 쉽고, 추후 이미 개발되어 있는 프로그램에 변경 사항이 발생했을 경우라 하더라도 변경 내용 적용이 용이하므로 확장성이 매우 좋다.

**Spring 빈(bean) 이란?** Spring이 제어권을 가지고 생성 및 객체간의 관계를 관리하는 객체를 의미. 형태는 제한이 없고 기존 일반 자바빈(DTO=VO)와는 다른 개념이다.

**BeanFactory OR ApplicationContext** Spring에선 빈의 생성과 관계 설정, 사용, 제거 등의 기능을 담당하는 컨테이너를 의미

**ApplicationContext 컨테이너** BeanFactory 인터페이스 상속받은 하위 인터페이스. BeanFactory의 빈관리 기능 이외에 여러 컨테이너 기능 추가됨. 장점 => Spring과 Spring이 관리하는 리소스를 선언적으로 설정하고 관리 할 수 있게 해 주며, 일반적으로 Spring 컨테이너라 하면 애플리케이션컨텍스트를 의미

주요 자식 API

1. ClassPathXmlApplicationContext
   * 클래스패스에 위치한 XML파일로부터 설정 정보 로딩
2. FileSystemXmlApplicationContext
   * 파일 시스템에 위치한 XML파일로부터 설정 정보 로딩
3. XmlWebApplicationContext
   * 웹 애플리케이션에 위치한 XML파일로부터 설정 정보 로딩

**Spring의 DI 방법** 객체간의 의존성을 설정 파일로 simple하게 설정 및 관리

1. Constructor Injection
   * 생성자를 이용한 의존 관계 설정. 매개변수가 있는 생성자 호출
   * 설정 파일에 <constructor-arg> 태그 이용
2. Setter Injection
   * Setter 메소드를 이용한 의존 관계 설정. Set 메소드로 의존객체 주입
   * 설정파일에 <property> 태그를 이용한다.
   * 객체인 경우 <ref>, 문자열이나 기본데이터 타입이라면 <value> 태그 이용

<property name=“engine">

<ref bean=“engine" />

</property>

**의존관계 자동 설정** 의존하는 빈객체의 타입이나 이름을 이용하여 의존객체를 자동으로 설정할 수 있는 기능으로 autowire 속성을 이용한다.

**어노테이션(Annotation)** 메타데이터를 XML등의 문서에 설정하는 것이 아니라 소스 코드에 “@어노테이션”의 형태로 표현하며 클래스, 필드, 메소드의 선언부에 적용 할 수 있는 특정 기능이 부여된 표현법. 복잡한 환경설정을 간단하게 만든다.

|  |  |
| --- | --- |
| 어노테이션 | 설명 |
| @Required | Setter 주입방식을 이용 |
| @Autowired | 타입을 기준으로(byType) 빈을 찾아 주입 (의존하는 객체를 삽입) |
| @Resource | 이름을 기준으로(byname) 빈을 찾아 주입 |
| @Component | 빈 스캐닝 기능을 이용  <context:component-scan> 태그를 설정파일에 추가하면 해당 어노테이션이 붙은 클래스를 빈에 추가한다. 검색된 클래스를 빈으로 등록할 경우 클래스의 이름을 빈의 이름으로 사용(첫 글자는 소문자로 변경)  <context:component-scan base-package=“package명"/>  => 클래스 검색할 패키지 지정 |
| @Qualifier | 특정 빈 삽입 |
| @Service(“ “) | 비지니스 로직이 들어가는 Service로 등록이 된다 |
| @Repository | 일반적으로 DAO에 사용되며 DB Exception을 DataAccessException으로 변환 |

**프론트 Controller 패턴** 적적한 세부 Controller로 작업 위임

**프론트 Controller 패턴 실행 process**

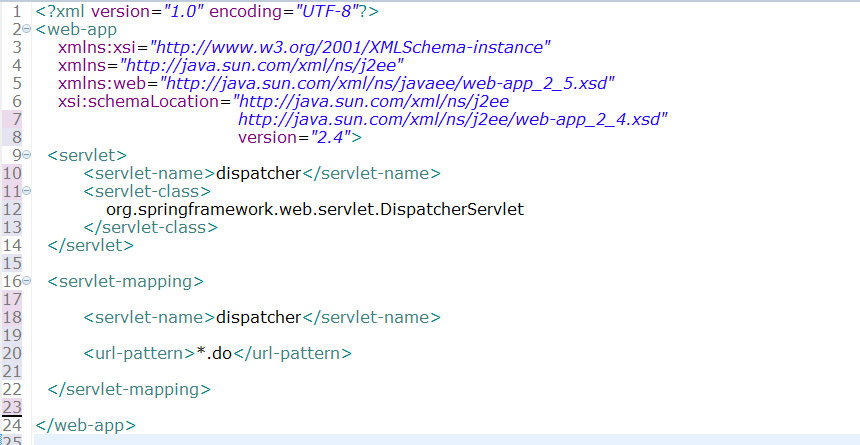
1. DispatcherServlet이 HTTP 요청 받음
2. DispatcherServlet에서 서브 Controller로 HTTP 요청 위임
3. Controller의 모델 생성과 정보를 등록하고 클라이언트 요청 경과 리턴
4. DispatcherServlet의 모델로 뷰 생성
5. HTTP 응답 돌려주기

**Spring MVC의 주요 구성 요소**

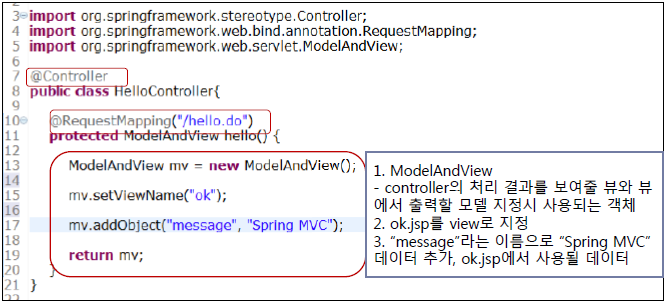
|  |  |
| --- | --- |
| 구성 요소 | 설명 |
| DispatcherServlet | 클라이언트의 요청을 받아서 Controller에게 클라이언트의 요청을 전달하고, 리턴한 결과값을 View에게 전달하여 알맞은 응답을 생성 |
| HandlerMapping | URL과 요청 정보를 기준으로 어떤 핸들러 객체를 사용할지 결정하는 로직의 객체. DispatcherServlet은 하나 이상의 핸들러 매핑을 가질 수 있음 |
| Controller | 클라이언트의 요청을 처리한 뒤 그 결과를 DispatcherServlet에게 알려 줌  Struts의 Action과 동일한 기능 |
| ModelAndView | Controller가 처리한 결과정보(Model) 및 뷰 선택에 필요한 정보를 보유한 객체 |
| ViewResolver | Controller가 리턴한 뷰 이름을 기반으로 Controller 처리 결과를 생성할 뷰를 결정 |
| View | Controller의 처리 결과 화면을 생성함 |

**Spring MVC 웹 어플리케이션 개발 단계**

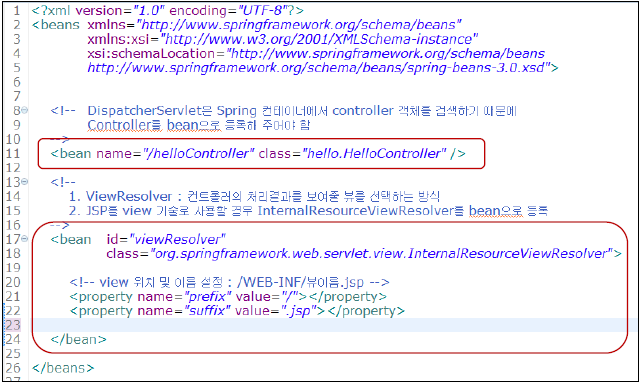
1. client의 요청을 받을 DispatcherServlet 을 web.xml 파일에 설정



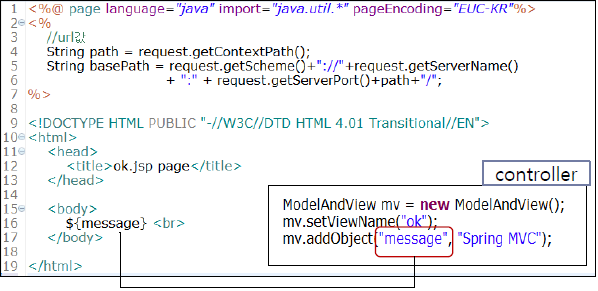
1. client의 요청을 처리할 Controller 작성



1. Spring 빈으로 controller 등록 & ViewResolver 설정

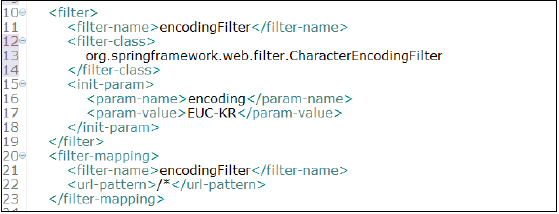


1. JSP를 이용한 뷰 영역의 코드 작성



1. 실행

**캐릭터 인코딩 처리를 위한 필터 설정**



**Controller 구현**

|  |  |
| --- | --- |
| 구성 요소 | 설명 |
| @Controller | Controller 클래스 정의 |
| @RequestMapping | HTTP 요청 URL을 처리할 Controller 메소드 정의 |
| @RequestParam | HTTP 요청에 포함된 파라미터 참조 시 사용 |
| @ModelAttribute | HTTP 요청에 포함된 파라미터를 모델 객체로 바인딩 @ModelAttribute의 ‘name’으로 정의한 모델 객체를 다음 뷰에게 사용 가능 |

**Controller의 처리 결과를 보여줄 view page 지정 방법** 메소드 반환 값에 따라 다르다.

1. ModelAndView인 경우
   * serViewName() 메소드 파라미터로 설정
2. String인 경우
   * 메소드의 리턴값

**Spring의 데이터베이스 연동** simple하게 JDBC뿐만 아니라 ORM 프레임워크들과의 연동도 매우 쉬움. 손쉽게 DAO클래스를 구현할 수 있도록 지원. DaoSupport 클래스(ex. iBatis)를 상속받아 DAO클래스를 구현한 뒤, DaoSupport 클래스가 제공하는 기능을 사용하여 보다 편리하게 코드를 작성할 있음

**<param-value>가 없으면?** (서블릿 이름)-servlet.xml로 찾는다!

**ContextLoaderListener** contextConfigLocation이 가지고 있는 값을 읽어서 bean을 초기화한다.

**@RequestParam** vo에 넘어오는 값이 없을 경우 @RequestParam("sample1") String sample1 으로 정의해놓으면 sample1으로 사용 가능

**Model 객체** 데이터만 담는 Model 객체를 만들어 md.addAttribute 사용

**컨트롤러가 여러 개?** ServiceFactory를 하나 만들어서 어떤 것을 쓸지 지정

**selectList와 selectMap** DB의 coulmn name과 VO 객체의 getter, setter의 get, set을 제외하고 첫 문자를 소문자로 변경한 이름과 동일하면 자동으로 바인딩(값 대입)을 해준다. 넣어준 값이 1개 이상이면 호출하는 곳에서 호출하는 방식에 따라 필요한 객체를 생성하고 리턴한다.

selectList => List에 담아서 리턴

selectMap => Map에 담아서 리턴

**Web INF참조** web-inf에 넣는 이유는 보안때문에 외부에서 접근할 수 없기 때문에