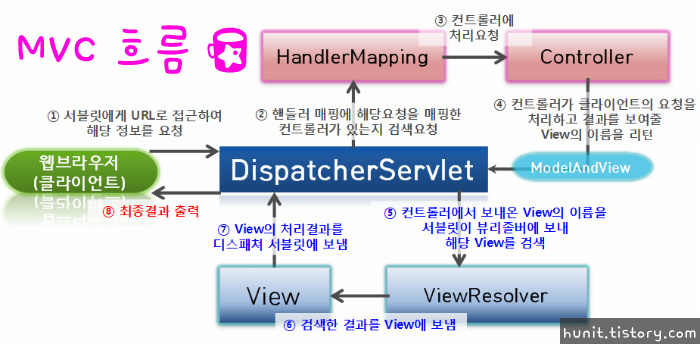
**DEV #10 – 2020.05.22**

>> Spring Web MVC 흐름을 조사하시오.

* Client 요청에서 부터 Client 응답까지의 흐름.



1. 웹 브라우저에게 정보요청을 받은 디스패쳐서블릿은 어느 컨트롤러에 해당 요청을

전송할지 결정

1. 디스패쳐 서블릿은 핸들러 매핑에 어느 컨트롤러를 사용할건지 물어봄 (URL로 링크)
2. 결정된 컨트롤러는 해당 요청을 수행하게 됨
3. 해당 요청을 처리한 컨트롤러는 디스패쳐서블릿에 결과를 보냄, 이 과정에서 Model이

생성되고 View(JSP)에서 같이 사용됨

1. ModelAndView는 실제 JSP정보를 갖고 있지 않기 때문에 ViewResolver가

실제 JSP 이름으로 변환하여 해당 view를 검색함

1. 검색한 결과를 View에 전송
2. View는 모든 과정에서 처리된 결과를 화면으로 표현
3. 디스패쳐서블릿이 웹 브라우저에 최종결과를 출력

>> Spring DI(Dependency Inject) 방식을 조사하시오.

1. Field Injection

가장 간단한 방법으로 Bean으로 등록된 객체를 사용하고자 하는 클래스에 Field로 선언한 뒤 @Autowired 키워드를 붙여주면 자동으로 주입됨

동일한 Class의 Bean이 여러 개

존재할 경우 @Qualifier를

통해서 bean이름을 지정하여 주입 받는 것이 가능.

많이 사용되는 방법이지만 권장되지 않는다. 이유는 너무 추상적인 Injection 기법 때문이다. 의존성 주입이 쉽기 때문에 Dependency 관계가 복잡해질 우려가 있고

[Framework 사용 목적에 위배]

* 너무나 쉬운 의존성 주입은 하나의 클래스가 지나치게 많은 기능을 하게 됨

🡪 초기 설계의 목적성 위배 (객체는 그에 맞는 동작만을 한다.)

* 추상화된 의존관계는 의존성을 검증하기 힘들게 함.
* Field Injection을 사용하면 해당 클래스를 곧바로 Instant화 시킬 수 없다.

이 부분 때문에 Constructor Injection이 권장되는 이유. (생성자)

DI Framework는 Field Injection 된 클래스의 Instance화에 대해서

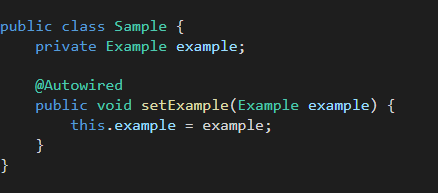
Null Pointer Exception을 만들 것이다.

* Field Injection된 객체는 final을 선언할 수 없으므로 가변적(Mutable)이다.

가변적이기 때문에 객체가 변경될 수 있으며 나중에 대응에 큰 비용이 듬.

1. Setter Injection

선택적인 의존성을 주입할 경우 유용, Spring 3.x 대까지 가장 권장하던 방식.



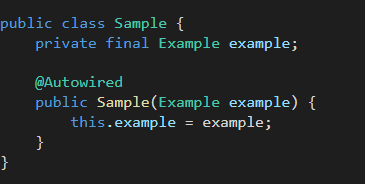
Field Injection으로 인한 패턴적 위험성을 상당 부분 해소한다.

Optional Injection의 경우 권장되는 방식.

@Required 어노테이션을 이용하면 의존성이 필요한 Settter를 만들 수 있다.

1. Constructor Injection

Spring 4.x 이상부터 권장되는 방식.



final 선언이 가능하며 Immutability(생성 후 변경 불가한 객체)에 대한

해소가 가능하며 의존성이 순환 참조(Circular Dependency)에 대한 예방이 가능.

Spring 4.3 이상부터는 생성자가 하나인 경우 @Autowired를 사용하지 않아도 됨.

>>Spring Bean의 개념을 조사하시오.

* Spring에서 POJO(plain, old java object)를 “Beans” 라고 부른다.
* Beans는 애플리케이션의 핵심을 이루는 객체이며, Spring IoC(Inversion of Control)

컨테이너에 의해 인스턴스화, 관리, 생성된다. (컨테이너가 관리하는 자바 객체)’

* ApplicationContext.getBean()으로 얻어질 수 있는 객체는 bean이다.
* Beans는 우리가 컨테이너에 공급하는 설정 메타 데이터(XML 파일)에 의해 생성된다.

🡪 컨테이너는 이 메타 데이터를 통해 Bean의 생성, Bean Life Cycle,

Bean Dependency(종속성) 등을 알 수 있다.

>> Maven Project를 생성하고 Spring web-mvc를 적용하시오.

* Spring web mvc dependency 추가 (5.2.4.RELEASE)
* logback dependency 추가 (1.2.3)
* javax-servlet-api dependency 추가 (3.1.0)
* maven-compiler-plugin 설정 (jdk version 1.8)
* web.xml 생성 – spring 전용 서블릿 / 서블릿 매핑
* handler mapping, view resolver, bean등록
* componet-scan, annotation 사용 설정
* Controller Bean 등록 및 View 생성
* Annotatino 방식 Controller지정

>> Spring Controller에 정의된 메소드의 매개변수로 올수 있는 type을 조사하시오.

* HttpServletRequest / HttpServletRespons / ServletRequest / ServletResponse
* HttpSession

HttpServletRequest를 통해 가져올 수 도 있지만 HttpSession만 필요할 경우 파라미터에

HttpSession을 받아서 사용하는게 낫다.

* WebRequest / NativeWebRequest

WebRequest는 HttpServletRequest의 요청 정보를 대부분 그대로 갖고 있고 서블릿 API에 종속적이지 않은 오브젝트 타입.

* Locale

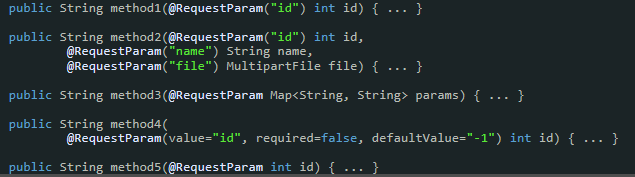
DispatcherServlet의 지역정보 리졸버가 결정한 Locale 오브젝트를 제공 받을 수 있다.

* InputStream / Reader
* OutputStream / Writer
* @PathVariable



* @RequestParam \*\*\*\*자주 사용

태그의 name 속성으로 찾음.



* @CookieValue
* @RequestHeader
* Map / Model / ModelMap
* @ModelAttribute

>> Spring Controller에 정의된 메소드의 리턴Type별로 적용되는 View를 조사하시오.

거의 void 아니면 String으로 리턴함

@ResponseBody

body태그 안에 있는 내용만 던져준다. (json 탑입으로 받아서 객체로 변환) ->

대부분 ajax를 사용할 때 사용함

>> JSP 파일 include 방식을 조사하시오.