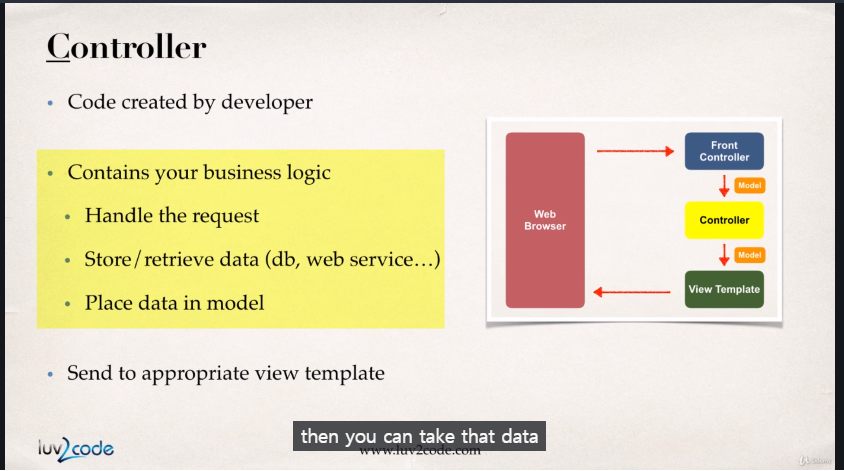
DispatcherServlet의 역할

Servlet Container에서 HTTP프로토콜을 통해 들어오는 모든 요청을 프레젠테이션 계층의 제일앞에 둬서 중앙집중식으로 처리해주는 프론트 컨트롤러(Front Controller)

<https://mangkyu.tistory.com/18>

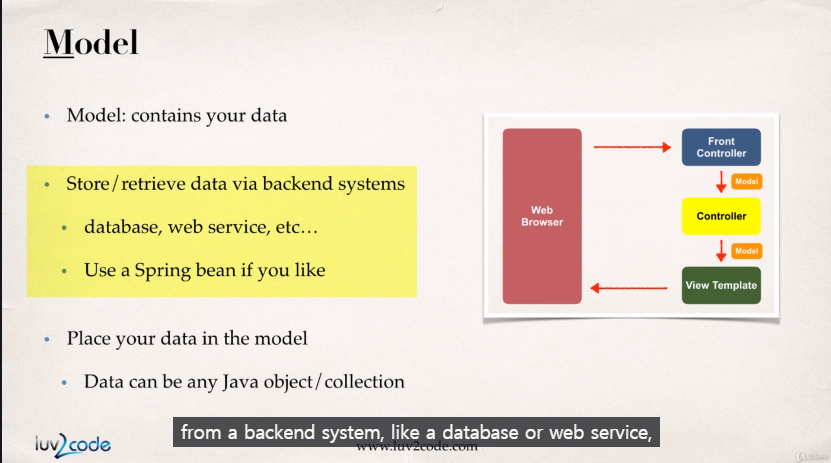


Controller는 우리가 가진 Business logic을 담당한다.

데이터를 전달 혹은 저장하고, 리퀘스트를 핸들링하는 작업을 한다.

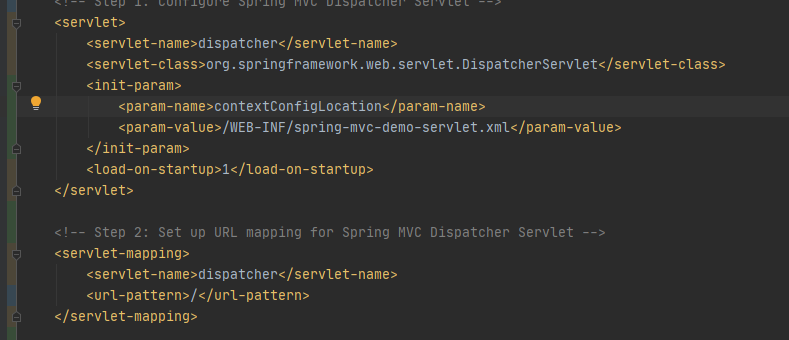
Front Controller는 Spring Framework에서 정의된 DispatcherServlet을 이용하며, 이에대한 레퍼런스 문서 언급은 아래와 같다.

Spring MVC, as many other web frameworks, is designed around the front controller pattern where a central Servlet, the DispatcherServlet, provides a shared algorithm for request processing, while actual work is performed by configurable delegate components. This model is flexible and supports diverse workflows.



Web.xml에서 지정되는 servlet 같은 경우 아래와 같은 Process를 따라 생성된다.

The DispatcherServlet, as any Servlet, needs to be declared and mapped according to the Servlet specification by using Java configuration or in web.xml. In turn, the DispatcherServlet uses Spring configuration to discover the delegate components it needs for request mapping, view resolution, exception handling, [and more](https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#mvc-servlet-special-bean-types).

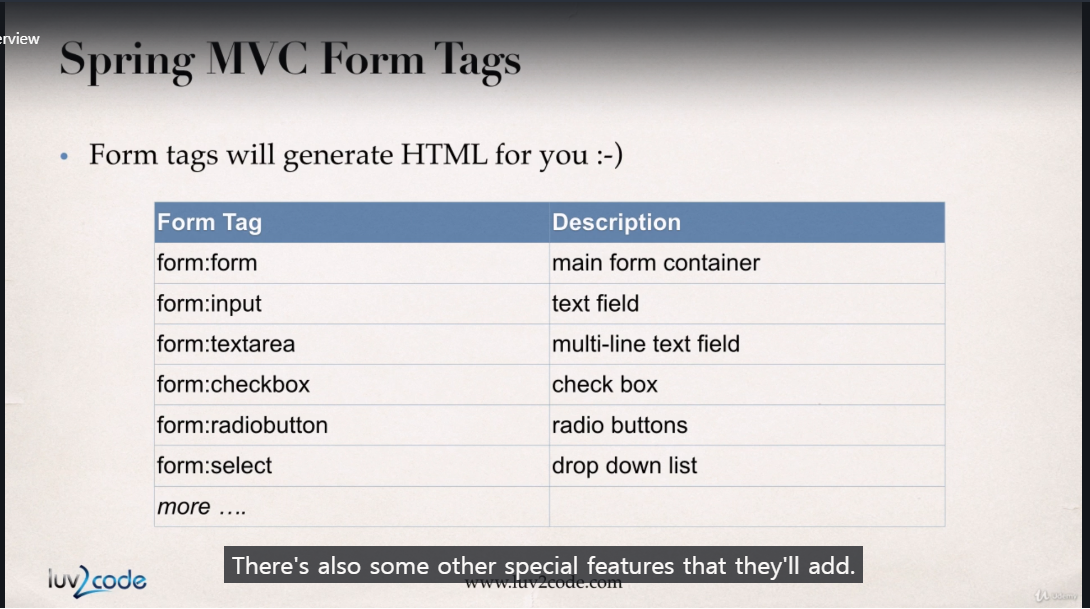


Web.xml에서 등록된 servlet-name과 mapping 당시에 사용되는 servlet-name이 일치해야 Springframework에서 제공하는 DespatcherServlet을 이용할 수 있다.

@Controller는 @Component를 상속한 클래스이다.

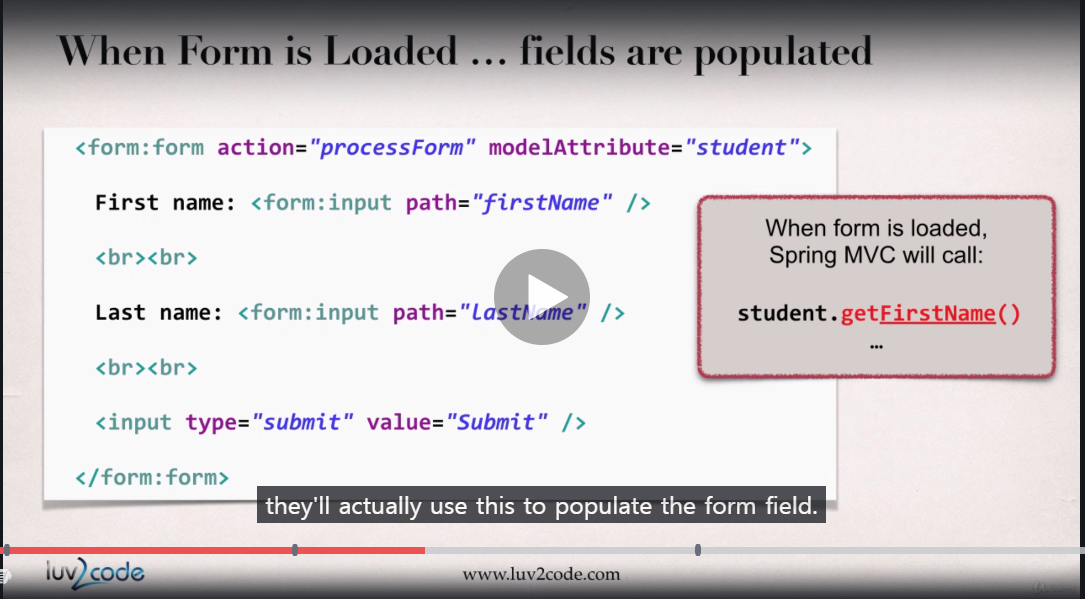
JSP 상에서 html param filed에 저장되어 있는 것들을 불러오기 위해서는

${param.valueName} 으로 사용 가능하다.



스프링은 jsp 파일에서 Form 태그를 지원하며 이를 통해 쉽게 attribute를 조작 가능하다.

아래 슬라이드에 코드는 그 사용 예제이다.



jsp form이 처음으로 로딩되었을 때 스프링 MVC는 이를 사용하기 위해 path를 method 이름으로 포함하는 get method를 호출한다. null일 경우 단순히 그 field는 empty인 것이다.

만약 위의 예제와 같이 form이 submit될 경우 단순히 spring은 set method를 호출한다

But the key here is that Spring will actually take all that form data, bind it **automatically** to your object, and then pass it into your controller so you can make use of it. So there's no need for you to do the **manual** **request.get parameter** for each one of the fields. Spring will handle all of that work for you, and that's the real benefit **of using the Spring MVC form tags,**

form tag의 path는 우리가 선언한 getter과 setter method에 의해서 결정되며 getFirstName을 선언 후 attribute를 조작할 수 있다.

First Name: <**form:input** path="firstName"/>

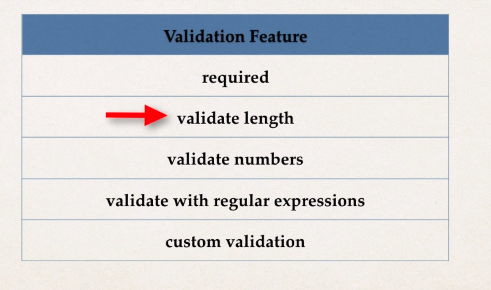
spring form tag의 method 기본값은 POST이다.

<**form:form** action="processForm" modelAttribute="student" method="GET">

addAttribute의 사용법 알아두기

Java Spring Form Validation

beanvalidiation.org



@Valid: Perform validiation rules on Some Paramerter

Spring은 background에서 BindingResult에 Validiation에 대한 결과를 받는다

Special Note about BindingResult Parameter Order

When performing Spring MVC validation, the location of the BindingResult parameter is very important. In the method signature, **the BindingResult parameter must appear immediately after the model attribute**.

MVC 유효성 검사에서는 BindingResult 파라미터가 매우 중요하다. 메소드선언에서 BindingResult는 반드시 model attribute뒤에 나와야 한다

아래 코드는 이에 대한 예시이다.

1. @RequestMapping("/processForm")
2. public String processForm(
3. @Valid @ModelAttribute("customer") Customer theCustomer,
4. BindingResult theBindingResult) {
5. ...
6. }

Validiation form에 message를 설정하지 않는다면 default message 형식의 메시지가 나가게 된다.

@InitBinder

전처리기

모든 컨트롤러 request에서 먼저 동작된다.

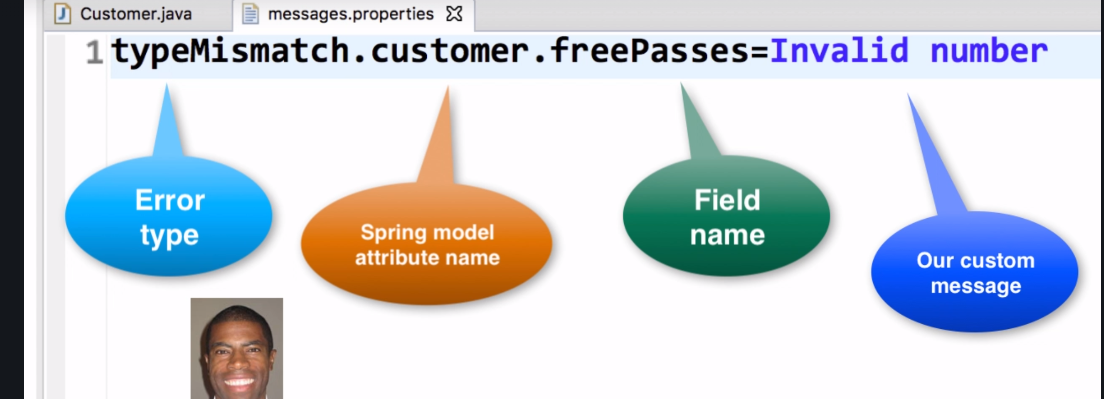
*// input 스트링으로 들어오는 String 데이터들의 white space를 trim해주는 역할을 한다.*

*// 모든 요청이 들어올때마다 해당 method를 거침 (node의 middleware 같은 것 )* @InitBinder public void

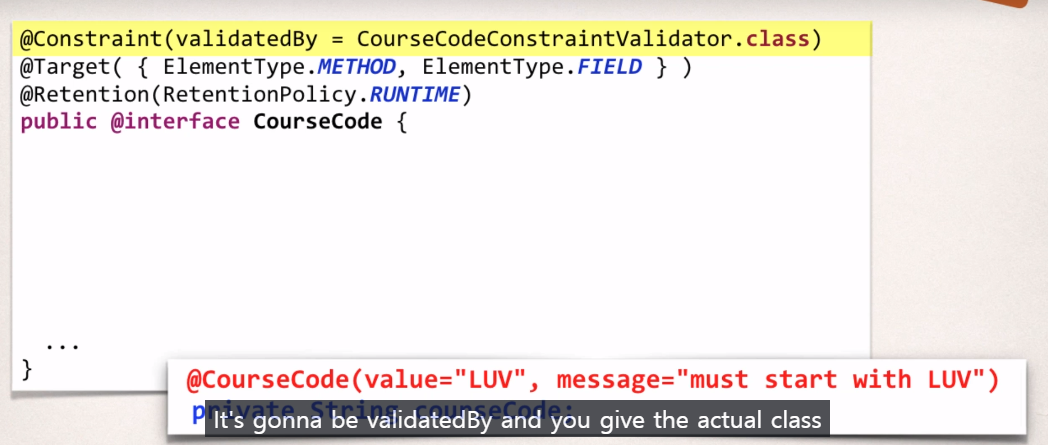
@InitBinder  
public void initBinder(WebDataBinder dataBinder){  
 StringTrimmerEditor stringTrimmerEditor = new StringTrimmerEditor(true);  
 dataBinder.registerCustomEditor(String.class, stringTrimmerEditor);  
}

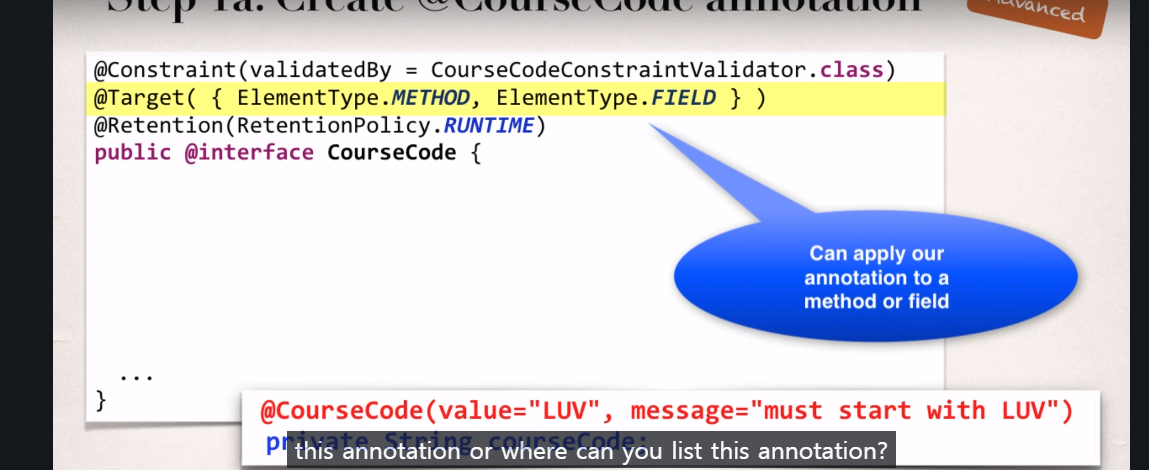
해당 method은 request로 들어오는 String model을 trim 해주는 역할을 한다 (즉 String 앞 뒤의 white space를 제거해줌)

properties 파일을 통해 custom message를 출력하는 방법과 그 파일 입력 폼.



custom Annotation을 사용하기 위해서는

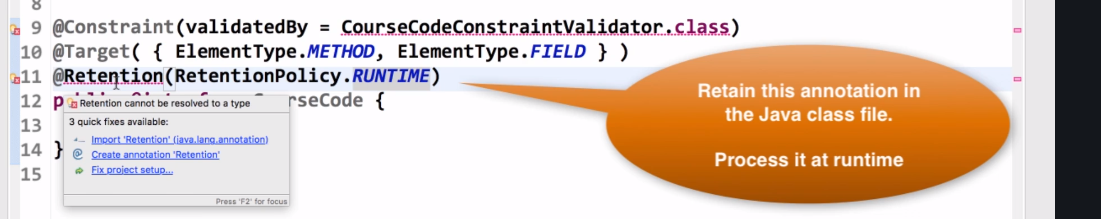
위와 같은 방식의 클래스 정의가 필요하다.



@target은 어떤 element의 사용할지를 가리키는 것.

@Constrait





public boolean isValid(String obj, ConstraintValidatorContext context) {  
 if(obj.startsWith(prefix)) return false;  
 else return true;  
   
}

ContraintValidatorContex는 추가 에러 메시지를 원할 때 사용.