step2 7주차 (mvcHanq2 프로젝트에서 수업 진행했음)

- @ModelAttribute

: WelcomeWebVO안에 필드의 명과

**private** String age;

화면에서 올라온 네임의 객체(화면에서 올린 인풋히든의 네임)가

<input type=*"hidden"* id=*"age"* name=*"age"* value=*11*/>

똑같으면 그 필드안에 화면에서 올린 네임의 객체의 밸류를 담아준다.

따라서 WelcomeWebVO.getAge 라고 적으면 화면에서 올린 벨류 11을 가져온다.

VO안에 필드가 여러 개면 정상적으로 안나오고 16진수로 나오게 된다. 나중에 한번에 볼 수 있는 방법을 알려줄 것 지금은 VO안에 어떠한 필드 안에 어떠한 값을 보고싶으면 일단은 해당 필드의 getAge를 이용해서 확인 해야 한다.

: 지금 해당 메서드의 타입은 문자열임, VO파일에 들어가면 String으로 되어 있기 때문에

model.addAttribute("welcomeWebVO", welcomeWebVO.getAge());

화면에서 쓰기 위해서 위에 줄을 사용하는데

만약 model.addAttribute("welcomeWebVO", welcomeWebVO); 로 사용하게 되면

${welcomeWebVO.age} 를 화면에서 사용해야 정상적으로 사용됨,

model.addAttribute("welcomeWebVO", welcomeWebVO);

여기서 뒤의 웰컴웹VO는 인스턴스변수라고 부른다. 객체를 담았기 때문에, 이 객체가 가리키는 게 있으니 사용하고 있는건데.. 각 스트링에 대한 스택이 생성된거고 각 스택들은 각각의 힙을 가르키고 그 힙의 참조값 가리키는게 인스턴스변수..

: 해당 model.addAttribute("welcomeWebVO", welcomeWebVO); 를 지웠음에도 화면에는 계속 11이 뜬다.

왜냐하면 @ModelAttribute 숨겨진 기능으로, 어노테이션 붙은 클래스 뒤에 있는 매개변수(갈색)를 model 안에 넣어준다. 현재 모델안에는 총 세개가 들어가있음 왜냐하면 위에서 먼저 WelcomeWebVO가 있었기 때문에 똑같은걸 하나 더 만들었던 것이라 삭제해도 화면에는 11이 나왔던 이유

: welcomeWebVO를 단순히 vo로 바꾸면 화면에 나오나 ?

아니 안나온다, 받아들이는 케이스의 차이로 발생하는 문제

@ModelAttribute("vo") 알리아스를 적어주면 적용된다.

: modelMap은 addAtrribute메소드를 사용하기 위해서 사용한 것 뿐이고, 따라서 model 자체는 화면에 값을 내리고 싶어서 사용했을 뿐, 화면에 내릴 필요가 없으면 modelMap을 안쓰는 것 뿐이지 @ModelAtrribute를 사용하는 것이라고 해서 꼭 modelMap 사용하지 않는게 아님을 유의

-- 스프링 –

- 전자정부 프레임워크와 비슷하게 수요가 많은게 STS 둘다 스프링이기 때문에 두개가 비슷하다고 생각할 수 있으나, 사실은 많이 다르다.

- 스프링은 @ 어노테이션 기반으로 만들어졌음

어노테이션은 인터페이스로 이루어져있고,

@컨트롤러를 쓰는 이유 : 컨트롤러 어노테이션의 하위 어노테이션을 쓰기 위해서

ex) @리퀘스트매핑을 쓰기 위해서

@리퀘스트매핑을 쓰는 이유 : 화면에서 올린 논리적 주소를 찾아가게 하기 위해서

안쓰면 어떻게 되요? : 화면에서 보낸 논리적 주소를 찾아갈 수 없게 됨

- 이번주업 전까지 우리한테 설명했던 내용

: jsp 화면에서 논리적 주소를 주면 컨트롤러에서 받는다고 했으나, 사실은 더 복잡함

컨트롤러와 뷰의 사이에는 **dispatcher-servlet**이 있다는 사실을 안알려줌

dispatcher-servlet에 들어가면 **RequestMappingHandlerAdapter** 라는 어뎁터가 있고 이 어뎁터가 @컨트롤러에 있는 @리퀘스트매핑을 훑고 해당 하는게 있으면 메서드 안으로 들어가는 것

dispatcher-servlet를 front controller라고 부른다.

- @컴포넌트

: 스프링은 다짜고짜 @를 쓴다고해서 읽어들이지 않는다. 스프링의 중요한 개념중에는 IOC라는 개념이 있다. 이것을 제어의 역전이라고 부른다.

우리가 A라는 클래스에서 B라는 클래스안에 있는 C라는 메서드를 사용하고 싶으면, 우리는 A에서 인스턴스화를 A클래스에서 하는데, IOC의 개념은 하나의 xml이 있고 이 xml을 bean factory라고 부르고, 이 xml에서 인스턴스화를 한다. 이 인스턴스화 한 것을 이 xml에서 B를 D로 인스턴스 변수화 했고, 이 D를 A에서 사용하게 하는 @리소스 를 통해 사용하게 해준다.

IOC의 장점은 pom.xml과 같음, pom을 meaven이라고 말하고, 여러 개의 jar파일을 한군데에서 관리 할 수 있는 것처럼, IOC도 동일한 이유처럼 하나의 xml만 봐도 해당 클래스가 어디서 상속받고 인스턴스화 되었는지, 등등의 프로젝트의 **생명주기를** 확인 할 수 있게 한다. 이게 바로 관리적인 측면에서 좋은 점, 이런식으로 xml에 인스턴스화 시키고 하는 것을 bean이라는 태그를 쓴다 빈을 추가한다, 디펜던스를 추가한다는 것을 인젝션한다고 하고, 이 인젝션 한 것을 사용하는 것을 lookup한다고 한다.

다시 위로 올라가 왜 컴포넌트가 있는가? 스프링은 @컴포넌트가 없으면 해당 bean을 등록하지 않는다. 즉 인젝션 하지 않는다. 컴포넌트라고 붙인다해서 읽는게 아니라 component-scan이 있어야 모든 컴포넌트가 등록되는 것이다. 이게 dispatcher-servlet에 살고 있는데 한 개의 프로젝트에 있는 모든 컴포넌트 어노테이션이 붙어있는 애들을 bean에 등록시키는 역할을 하기 때문 만약 이 component-scan가 없다면 컨트롤러 서비스 서비스임플 맵퍼 어노테이션이 모두 안읽혀질 것임, 왜냐하면 컴포넌트 어노테이션을 스프링에 등록시켜주는 애가 component-scan인데 가 없으니 안읽혀지는 것,

우리는 데이터를 내릴때도 dispatcher-servlet를 중간에 거치게 되는데

올릴때처럼 RequestMappingHandlerAdapter 를 쓰는게 아니라 view resolver를 타게 된다. 그게바로 JstlView로, JstlView를 사용하게 된다. 컨트롤러에서 우리는 ‘모델’과 리턴에 jsp폴더/jsp파일명이라는 ‘논리적 주소’를 내려줬고 이 내린 것을 화면이 받는게아니라 JstlView를 거쳐서 화면에 출력되는 것. 보면 우선순위가 2로 되어잇지만 1위인 tiles는 JstlView를 상속하기 때문에 같은 것

JstlView가 만약에 c포문을 돌리다가 welcomeWebList를 만났을 때 welcomeWebList가 무엇인지 모르니 뷰는 모델과 논리적 주소를 내려받음으로 이 뷰는 이 그린 것을 dispatcher-servlet에 주고 뷰가 만든 그림은 사람이 볼수 있는 상황이 아니므로 dispatcher-servlet의 Servlet Container를 통해 변환을 한다..

브라우저는 딱 한가지 통신만 하는데 그 통신은 HTTP통신, HTTP통신을 하기 위해서는 누군가 변환을 해줘야 하는데 Servlet Container가 한다. 이 컨테이너를 통해서 우리가 화면에서 볼 수 있다. 그리고 뷰가 떨어트릴 때 논리적 주소를 같이 보내주므로 이 화면을 해당 논리적 주소로 보내주는 것.

**==== 중요 ====**

정리하자면, 화면에서 서브밋 하면, Http 요청을 해서 요청한 것을 dispatcher-servlet이 받고, 그 중에서 **RequestMappingHandlerAdapter**가 받아서 @Controller를 먼저 찾고 하위 어노테이션인 @RequestMapping 을 찾고 거기있는 value속성의 문자열과 내가 주소창에 가지고 있는 문자열이 맞다면 그 안의 구현부(메서드)를 들어간다. 그 구현부 안에 들어가고 서비스, 서비스임플, 맵퍼, sql에서 가져온 것을 모델을 생성해서 담고, 담은것을 다시 dispatcher-servlet에 내려야 하는데 내릴 때 모델과 뷰 두가지를 내린다. dispatcher-servlet에서 **JstlView**가 받고 이 뷰가 모델을 참고하면서 그려내고 그 결과물을 dispatcher-servlet에 다시 보내고 **Servlet Container**가 받아서 Http한테 던지면 브라우저를 통해 우리가 확인 할 수 있는 것.

--- 스프링 설명 끝 ---

--- 배열 시작 ---

- 배열이 나온 이유는, 한 개의 변수에 여러 개의 값을 넣고 싶어서

키는 없고 벨류만 여러 개 담을 뿐, 이 벨류를 꺼낼때는 index를 쓰고

배열은 0부터 시작하므로 첫번째 값을 가져오고 싶으면 0을 사용해야 함

- 배열을 선언하고 초기화 하는 방식에는 두가지가 있음

리터럴 방식 선언 [ , ] 대괄호를 이용해서 값을 부여한다.

생성자 방식 선언 var arryClass = new Array(“a”,”b”) 메모리를 잡아먹고 타이핑이 많으므로 안씀

- 변수.length

: 배열이 있는 변수.length라고 적게되면 3이 나옴

값이 있는 배열을 나타내는 것이기 때문

- $("#arr").val(arryLiteral);

파라미터로 넣은 값을 해당 타겟의 value라는 속성에 넣는 것

- 컨솔창에 나온 a, b, c 는 스트링인가 배열인가? 는 Stirng이다.

request.getParameter의 반환값이 String이기 때문에,

사실은 우리가 jsp 에서부터 올릴때부터 String화 되어 있음

input에 담는 순간부터 String이 된 것이기 때문, 남들과 다른 코딩을 하기 위해서는 타입유의!!

alert(typeof $(“#arr”).val()); 타입오브를 씀으로써 해당 값이 어떤타입인지 알 수 있음

우리는 화면에서 String이 되는 것은 확인 했고

자바에서 문자열을 배열로 바꿀 수 있는 것을 할 예정이다.

- 스플릿 메서드에 컴마를 기준으로 자른다, 따라서 3개가 잘리고

String[] arr = srt.split(“,”);

- foreach문

**for**(String a : arr) {

System.***out***.println(a);

}

// 우측에다 배열울 넣고 자동으로 왼쪽의 변수에 들어간다.

여기의 타입은 당연히 이 배열 안에 들어가있는 value의 타입이 된다.

- 배열은 일반적으로 16진수로 뜬다, 값의 주소를 찍는게 아니라 그 값이 있는 주소를 나오는 것

그것을 한번에 하고싶다 ? (Arrays.*toString*(arr)); 자바는 어레이.투스티링 을찍으면 한번에 하도록 도와준다

- 화면상에서는 뒤에 주소를 지우면 화면은 에러가발생했지만

자바 컨솔상에는 syso를 통해서 null값이 나옴,, 따라서 syso까지는 정상적으로 나온 것

**split**을 쓸 때 초급개발자가 실수 하는 경우는, null을 고려하지 않아서 발생한 에러

따라서 nullException이 발생한 것이고,

**항상 split을 쓸 때** 내가 나눌 변수가 비어 있는지 비어 있지 않은지 확인을 하고 스플릿을 써야 함으로 if구문을 추가한다. 이렇게 하면 주소창을 지워도 스플릿을 하지 않기 때문에 정상적으로 나올 것

- 스플릿을 하면서 키가 있으면 널이 들어가는게 아니라 빈문자열이 들어감, 그치만 스플릿은 널에대해서만 오류를 뱉고 빈문자열은 오류를 붙지 않음, 따라서 뒤에 주소를 지우고 화면을 띄웠을 때 정상적으로 나오는 이유는 index.jsp 에 미리 키를 넣어놨기 때문에

- 차트 숙제를 통해서 리스트와 배열에서는 c:out 등을 사용 할수 있다고 말했음

<c:forEach items="${arr}" var="arr">

alert("<c:out value="${arr}"/>");

</c:forEach>

-화면에서 올려야 할 값이 많다면 즉, 인풋히든이 1000개라면 컨트롤러에서는 해당 값을 빼기 위해서는 하나하나 다 쳐야 하는데 이건 옳지 못한 방법임으로 새로운 방법을 알려줄 예정

- request.getParameterNames();

이걸 쓰면 화면에서 올린 네임 객체를 한번에 다 가져 올 수 있음

타입없이 names라는 변수를 만들고 금방 만든 겟파라미터네임스를 넣으면 자동으로 리턴타입이생김

이게 바로 **열거형 인터페이스** 겟파라미터네임즈가 이 타입으로 만들어진 것이기 때문에 생김

해당 제네릭타입이 물음표인데 무엇을 넣을지 모르기 때문에 물음표로 되어있으나 정말 안좋은 코딩임으로 여기에 input hidden의 name타입인 String을 넣어준다.

열거형 인터페이스는 배열이랑 똑같다고 보면 되는데 데이터를 보내고 커서가 이동하고~ 반복하다가 보낼데이터가 없으면 끝이난다.

이 열거형 인터페이스는 키-벨류 형식이 아니고, 열거형 인터페이스의 안에는 벨류가 곧 네임으로 되어있음 왜냐하면 .getParameterNames 을 통해서 가져왔기 때문에

date = name 으로 되어있음

열거형 인터페이스는 포문보다는 와일문을 돌린다. 꼭꼭

names. hasMoreElements는 ‘안에 데이터가 있으면 가지고 오고 없으면 탈출’ 이라고 생각

names.nextElement는 커서가 있는 데이터에서 그 다음 커서로 넘겨주는 것이라고 생각

-체크박스는 input 태그중 name속성을 똑같이 한다. (체크박스는 곧 배열이다)

체크를 두개하고 눌렀지만 컨솔창에 value값 1 하나만 뜸

왜냐하면 getParameter는 같은 이름에 대한 값을 가져올 수 없음

그렇기 때문에 getParameValues 를 사용한다. 벨류즈는 하나의 name에 두개 이상의 값이 있으면 배열로 가져온다.

getParameValues를 사용하면 이전에 컴마 ‘ , ‘ 를 기준으로 스플릿하고 배열을 나눌필요 없이 한번에 배열로 만들어 낼 수 있다. 하지만 명심해야 할 것은 name의 속성값이 같고 value값이 여러개일 경우에 사용이 가능하다.

- 우리가 가져온 name과 value로는 키를 알 수가 없음

- 배열은 한 개의 변수에 여러값을 담기 위해 쓴건데

위에서 한 행위를 하게되면 값이 하나인 경우에도 배열을 갖게 되므로

if절을 통해 2개 이상이라면 배열을쓰고 하나라면 문자열로 갖게하도록 한다.

-지금까지 배운걸로 이런 것을 할수 있다.

벨류즈로 받아으면 변수에 1,3,5 가 들어가고

이전화면에서 찍은 밸류의 배열

---- 숙제 ----

1. 배열을 예를들어서 우리의 시작은 레프트 메뉴를 생성하고 레프트메뉴를 누르게되면 변수하나 선언하고 거기에다가 아무 배열을 넣고 그다음에 그것을 인풋히든 하나 만들어서 그 안에 넣음으로써 .val 을 연습하고 컨트롤러로 가져가서 split으로 나누고 나눈 것을 syso 찍고 확인한다음엔 다시 지우고, 그다음엔 그것을 모델안에 넣은다음에 화면으로 내려서 c:out 과 c:forEach 등을 이용해서 얼럿띄워라.

li 복사하는것 빼고 원테이크로 동영상 찍어서 처음부터 끝까지,

2. 오늘 우리는 팀 스터디에서 뭘 중점적으로 하느냐 : 열거형 인터페이스를 사용해가지고

오늘 새로운 메서드를 3가지 배웠는데, 이거에 대한 소스 이해를 중점적으로 하고

getParameter // getParameterNames // getParameterValues

while문을 쓰면서 돌린 내용//

split을 써서 배열로 만들기도 했고 //

getParameterValues 자체가 배열을 만들수 있으니 이걸로도 만들어보기도 했고 //

특히 request. 으로 시작하는 메서드들을 활용해가지고 화면에서 인풋태그가 몇 개가 됬든 와일문을 이용해서 키와 벨류를 뽑고 이프문을 써서 하나이상이면 스트링으로 넣고 두개 이상인 것은 배열을 넣고 했던 부분을 정리를 해서 낼 것.