step2 9주차

- ajax의 데이터 순서가 어떻게 되죠?

: 컨트롤러로 보낼 때 data속성에 담아서 json 오브젝트 타입으로 보낸다.

그다음 누가 바꿔준다고 ? ajax전처리기가 쿼리스트링방식으로 바꿔준다.

결국에 컨트롤러에 떨어질땐 쿼리스트링으로 떨어지므로 다른것들을 사용할 수 있다.

우리가 메서드에 void 인데 ajax니까 보이드구나 라고 생각하지 말 것

내릴땐 response가 내려주고

ajax는 자기가 올리면 자기가 내려야하는 성격이 있다.

내릴때는 스트링타입으로 내려야 하는데 제이슨 오브젝트 스트링으로 내려야 한다.

서브밋은 jstlView가 받지만, ajax는 mapping jackson http message converter를 이용해서 받음

이전시간에 우리는 result라는 키에다가 SUCCESS라는 값을 담았고 , childList라는 키에다가

가져온 데이터를 Map에다가 담았음

원했던 대답은 ajax에 담을 때 json 오브젝트로 담고 컨트롤러에 올렸고 컨트롤러에서는 쿼리스트링으로 받았고, 받은것을 MVC 태울 때 그냥 String으로 태웠고 그래서 쿼리에 \_parameter로 받고 다시 리턴해서 가져와 최종적으로 화면에 내리는 작업을 해야하는데 우리가 내릴 때 Map을 하나 만들고 가져온 데이터를 Map에 담고 추가적으로 result라는 키에다가 SUCCESS라는 값을 담았고 화면에 내리기 위해서 ajax는 response로 내리니까 response에 get.write 해서 out.write 해야하는데 out.write에 넣어야 할 파라미터 타입이 String인데 우리는 결과값을 맵에 담았으니 이것을 스트링으로 바꾸는 작업을 해야 한다. >> 이제 json오브젝트 모양의 String으로 바꿀 것이다.

- 맵을 스트링으로 바꾸려는 작업을 할 것

: out.write(resultMap); 의 넣을 파라미터 타입이 String이기 때문에 바꾸는 것

이것을 제이슨 오브젝트 쉽게하는 방법은 Gson

Gson 을 디펜던시 추가해야 하기 때문에 pom.xml에 가서 추가해준다.

- 스프링 3버전에서는 mapping jackson http message converter 가 받고

스프링 4버전에서는 Gson message converter 가 받는다.

**-JsonUtil.*HashMapToJson*(resultMap);**

제이슨 오브젝트 형태를 띈 스트링으로 바꿔주는 것일 뿐

파라미터로 HashMap을 넘기면 Json 오브젝트 형태를 띈 String으로 바꿔준다

**스트링이 아니다** 착각하면 안됨 !

- 이제는 out.write에 들어갈 수 있는 형태가 되었으니 넣어준다.

String resultMapToJson = JsonUtil.HashMapToJson(resultMap);

out.write(resultMapToJson);

- 이제 화면에서 다시 돌려받아야 함

ajax의 success 속성의 funtion 에 들어가게 된다.

- success : function(data) {

alert(“resultMap : ” + data);

}

한글이 깨져서 나옴, 기존에 깨져서 나왔을때의 문제는 get방식으로 보낼 때 깨짐

post방식으로 보내면 안깨짐, 근데 우리가 지금 톰캣에 설정 해놓은상태인데 왜 깨지지?

저번주에 설명했는데 통신방법이 두개가 틀리기 때문에, submit방식은 톰캣에 직접적인 영향이 있고 ajax는 통신방법이 틀려서 톰캣과 연관이 없음, 이건 컨트롤러에서 response에 다국어 처리를 해줘야 된다. response.setCharacterEncoding("utf-8");

- 저번주에 우리한테 강력하게 얘기한 것,

: 항상 우리가 흔들리는 것, 약점, 덩어리로 가져와서 빼는 것

지금 덩어리로 가져왔음, 하나만 빼봐라

- 현재 이건 스트링으로 되어있어서 뺄 수 없음, 우리가 이전에 덩어리에서 뺄 때 두가지를 섞어서 썼었음 조건문과 반복문을 썼음. **덩어리를 for문을 통해서 분리하고 if를 이용해서 하나를 빼냈었음.**

컬랙션타입인 리스트, 배열의 경우에는 할수 있으나, 스트링은 할 수가 없다.

따라서 스트링을 json object로 바꿔야 함

success : **function**(data) {

**var** jObj = JSON.parse(data);

console.log("data : ",data);

console.log("JSON.parse(data) : ",jObj);

- Json.parse() 메서드는 제이슨 오브젝트 스트링을 다시 제이슨 오브젝트로 바꿔주는 메서드다.

위의 Json.parse() 메서드를 거치지 않은 데이터는 단순 문자열로 나오고

Json.parse() 로 바꾼 jOBj는 맵에 배열이 들어와 키=벨류 형식의 제이슨 오브젝트로 변환됨

ex) 0 : {prdCd : “P00002” , prdNm : “별점퍼”} 등등등 으로 컨솔창에 나옴

- 컨솔창에서 확인하면 childList를 현재 배열로 읽고있다.

배열의 가장 큰 특징은 인덱스로 읽힌다. 0으로 시작하는게 가장 큰 특징

- if 조건문에 result 가 SUCCESS 라면 즉 데이터가 잘 내려왔다면

메서드를 호출하면서 자식박스 배열을 파라미터로 넘겨준다. (두번째 셀렉트 박스에 해당하는 품목을 넘겨준다.)

select 라는 객체나 displayChildSelectBox라는 메서드가 없는데 껍데기를 만들어라 !

🡺 답은 다음페이지

**var** select = {

displayChildSelectBox : **function**(childList) {

}

- 두번째시간 –

- 이 메서드는 뭘하는 메서드? 두번째 셀렉트박스에 옵션을 넣을 것

**var** select = {

displayChildSelectBox : **function**(childList) {

**if** (childList.length > 0) {

$.each(childList,**function**(i, item) {

**var** optionStr = "<option>" + item.prdNm + "</option>"

$("#childSelectBox").append(optionStr);

})

} **else** {

$("#childSelectBox").append("<option value=''>없음</option>");

}

}

}

바로 넣으면되지 여기에 왜 메서드롤 또 호출하지 ?

가독성이 떨어지기 때문 !! 업무를 분리하는 것, 메서드의 역할은? 처리를, 기능을 담당함

ajax를 태우는거는 태우기만 하면 됨 결과물은 success안에 떨어지고, success는 컨트롤러에서 잘 태운 데이터를 화면단으로 가져오는 역할,

- 옵션을 넣을라면 우리가 가져온 덩어리가 필요하다. 그래서 파라미터로 덩어리를 넣어 준 것

- select.displayChildSelectBox(jObj.childList);

이걸보고 껍데기를 만들줄 알아야 한다 ..

- 현재 우리가 자식 데이터가 있을수도 있고 없을 수도 있다. 그럼 우리가 해야하는건 예외처리

가져온 값이 있는지 없는지를 적으면 됨, 만약에 가져온 데이터가 없으면

- append 메서드는

해당 태그를 타겟팅 한 것의 멘 마지막 자식태그에 붙는다. 프리팬드는 맨 위에 붙는다.

$("#childSelectBox").append("<option value=''>없음</option>");

이런 행위를 **동적으로 HTML을 생성했다**고 한다.

- $.each 가 반복문

( 첫번째 파라미터의 길이만큼 반복한다, 다음의 i는 인덱스 이기 때문에 우리가 최대 4번보니 3까지 증가한다, item은 키가 아니고 한 줄 {prdCd:”P00002”, prdNm:”별점퍼”}중괄호 하나가 다 item이다. )

forEach 에서의 var 과, each 에서의 item 을 동일하다고 생각해주시면 item.prdNm 의 사용법이 익숙할 것 // 따라서 item.prdNm은 “별점퍼” 가 되는 것 !!! 유의하자

얘는 반복을 실행할때마다 두번째 파라미터에 있는 익명함수

$.each(childList,**function**(i, item)를 실행한다.

길이가 3이면 익명함수를 3번 실행한다.

**if** (childList.length > 0) {

$.each(childList, **function**(i,item) {

optionStr = "<option>" + item.prdNm + "</option>"

$("#childSelectBox").append(optionStr);

})

한줄 안에 있는 prdNm을 가져오는데 이건 유저눈으로 보는 글자로 이거 전체체를 변수에 넣고 어팬드 하면 벨류에는 값이 안들어가니 value값 까지 넣어준다

$.each(childList, **function**(i,item) {

**var** optionStr = "<option value=" + item.prdCd + ">" +

**item.prdNm + "</option>"**

$("#childSelectBox").append(optionStr);

- 계속 값이 쌓이는데

$("#childSelectBox").children().remove();

여기서 children은 타겟의 자식태그들 전부를 의미하는 것

-------- @ModeAttribute 의 추가적인 기능 ----------

- @ModeAttribute을 보통 쓸 때 매개변수 안에 있었는데 이번에는

리퀘스트 매핑자리에 알리아스를 parentList 모델의 키의 명을 적으면 해당 매서드의 리턴을 바로 서비스를 붙여넣어 주면 서비스의 결과값이 모델에 이 "parentList"키 에 담긴다.

@ModelAttribute("parentList")

**public** List<EgovMap> parentList() **throws** Exception {

**return** selectBoxService.selectParentBoxList();

}

만약에 컨트롤러에 여러 개의 메서드가 있다고 생각하면 내가 기존에 있는 메서드위에

@ModeAttribute을 적고 리턴으로 해당 결과값을 받으면 parentList라는 키 안에 그 결과값이

들어가는데 모든 메서드에 있는 모델에 다 들어간다. 단 이 클래스에 있는 모든 메서드에 들어감

이걸 왜 가르켜주느냐 ?? 도큐먼트 레디에 많이 써넣는거 , 컨트롤러에 많이 써넣는거 아주 좋지 못함, 실무에가면 하나의 업무에 여러 개의 화면이 붙을 수 있다. a화면과 b화면에 동일한 라디오버튼 셀렉트박스가 있는데 이것을 컨트롤러에서 구현하려고하면 A라는 컨트롤러에 A라는 메서드와 B라는 메서드를 만들면 각각의 메서드를 만들게 되면서 여러 개의 줄이 생성되는데

위와 같은 방법으로 하면 컨트롤러가 짧아지게 됨 !! 공통화 할수 있음 !!!

그리고 모든 메서드보다 먼저 타기 때문에 모델어트리뷰트어노테이션을 먼저 찾기 때문에 먼저 실행된 다음 .do reuqestMapping 어노테이션이 실행되는 것

------ ajax 보충 ------

- 우리가 배운 ajax는 사회에서 쓰지 않는 ajax

: 지금부터가 쓰는 ajax, 왜냐하면 get방식으로 하니까 !!

data : {"param" : thisParam}

- 우리는 위에서 Json Object String을 Json Object로 바꿀 때

Json.parse() 를 써서 바꿨음, 안에 있는 값을 꺼내기 위해서

Json.stringify()은 Json Object를 Json Object String으로 바꿔주는 것 제이슨 파스와 정반대의 것

data : JSON.stringify(form),

완벽하게 문자열로 바뀐다 Object가 String으로 바뀐다

- 네트워크 탭에서 오류가 발생한 것을 볼 수 있음

: 제이슨 오브젝트로 보내야지만 올바르게 쿼리스트링으로 변환이 되는데, 우리가 제이슨오브젝트스트링으로 보내다 보니까 제대로 인코딩하지 못하고 “”를 %22%로 받고있음

심지어 보내는 타입을 적지 않으면 get으로 가서 발생하는 오류

이 엉망진창인 상황에서 type을 “post”타입으로 바꿔준다. – 다시 확인하면 네트워크탭에서 오류가 발생하지 않음을 확인 할 수 있다. body에 있는 그대로를 읽는게 post방식이기 때문에 오류가 생기지 않았음 따라서 우리가 보냈던 모양 그대로 간다.

{“param”:”B00002”}:

포스트방식으로 가서 잘 보이는건 맞는데 결국에는 지금 : 이 뒤에 붙은상태를 보면 문제가있다.

우리는 현재 param을 키로 B00002를 벨류로 썻는데 저 문자열 자체를 key로 인식한 것. 심지어 value는 없는 상태인 것

그럼 우리가 어떻게 해야하는가 ???

>> ajax전처리가 이딴식으로 읽고 있는건데 우리가 처리를 해줘야한다.

- 우리는 아까 화면에서 올렸을 때의 데이터 타입은 ajax json오브젝트 타입으로 보냈고 컨트롤러에서 최종적으로 받을땐 쿼리스트링으로 받았고 쿼리스트링으로 받은 것을 String으로 넣었고 String으로 받은 것을 mvc를 태웠고 받은 결과물과 또다른 SUCCESS라는 키를 맵에다 넣었고 맵에다 넣고 화면에 내릴려고 response를 사용하려고 보니 String으로 달라고 함, 그리고 그냥 스트링이 아니고 json오브젝트형태의 string으로 줘야 하는 상태임 !!!!!!!!!!!

- @RequestParam을 쓸 수 있는 조건이 두가지가 있다.

**1. 쿼리스트링으로 올라와야함 // 2. 화면에서 올린 name과 Param에서의 변수가 같아야함**

@RequestParam 을 지우는 이유는 쿼리스트링방식으로 올라오고 있지 않고

현재 키가 {"param" : thisParam}; 으로 되어 있기 때문에 삭제한다

지금 두가지 조건이 모두 만족하지 않기 때문에 삭제해야 한다 !!!!

- getParameterNames를 이용하면 화면에서 올린 모든 키를 가져올 수 있음

- Json Object String 타입의 {"param" : “B00002”}이 키로 들어가있음

위의 Json Object String 타입을

JsonUtil.JsonToMap() 메서드를 이용하면 우리가 원하는 키=벨류 형식의 맵으로 바꿀 수 있음.

Json Object String 타입인 paramKey를 파라미터로 보내면 맵으로 바뀐다.

**while** (name.hasMoreElements()) {

paramKey = name.nextElement();

System.***out***.println("paramaKey : " + paramKey);

}

JsonUtil.*JsonToMap*(paramKey);

- 위에 JsonToMap 메서드의 결과를 확인하면 {param=B0002}라는 결과를 알 수 있다

그렇게되면 param이라는 key의 value의 값을 뽑아 올 수 있게 된다. 그렇게 하기 위해서

먼제 Map에 담아주고 .get메서드를 통해 해당 키의 벨류 값을 가져온다.

Map<String,Object> resMap = JsonUtil.*JsonToMap*(reqParam);

param = (String) resMap.get("param");

이때 마지막에 (String)으로 타입을 정해주지 않으면 오류가 난다. 왜냐하면 Map을 만들 때 제네릭 타입을 <String, Object>로 지정해 놓은 상태이기 때문이다.

---------- @RequestBody String reqParam ------------

- 포스트방식으로 ajax를 태우는데 있어서 복잡하다. 스프링에서는 좋은 것을 제공하는데 그것을 할 예정, 그냥 알려주기 보다 이걸 왜 이렇게 하는지를 이해를 하고 넘어가야 하는 부분

- 포스트 방식은 바디에 데이터가 담겨지는 것

스프링중에는 @RequestBody이 있는데 바디에 있는 데이터를 가져오는 어노테이션이다. 즉 @RequestBody은 get방식에서 사용할 수 없음 get방식은 절대로 데이터가 바디로 들어가지 않기 때문이다.

기존에 request.getParameterNames를 통해 화면에서 보낸 name 가져오는 태그들을 주석처리

주석처리하는 이유는 name을 뽑을 필요가 없음 바로 데이터가 들어가기 때문에 !!

그러고 syso 로 reqParam을 찍으면

reqParam : %7B%22param%22%3A%22B00003%22%7D= 로 나오는 것을 알수 있다.

이를 디코딩을 할 것이다. 바디에 있는 데이터를 그대로 가져왔기 때문 이것을

paramKey = URLDecoder.*decode*(reqParam,"utf-8");

을 URLDecoder를 이용하여 변환한다.

변환하게되면 맨 끝에 ‘ = ‘ 이 있음을 확인 할 수 있다. {"param":"B00003"}=

substring 메서드를 이용하한다.

paramKey = paramKey.substring(0, paramKey.length()-1);

문자열을 자를 때 쓰는 것으로 첫번째 파라미터부터 두번째 파라미터까지 자르는 코드를 쓴다. 여기서는 0부터 마지막 글자의 전까지 자른다. length -1 은 ‘ =’바로 전까지 자른다는 것을 의미

해당 작업을 하면 정상적으로 구동되나 굳이 이렇게 할 이유는 없다.

왜 이렇게 나오느냐? ajax 전처리기가 우리가 원하는대로 인코딩을 하지 못하기 때문에 우리가 원하는 값을 가져오기 위해서 디코딩하는 작업을 한 것, 따라서 전처리기가 우리가 원하는대로 인코딩 하지 못할 뿐 제대로 했던 것, ajax 가 가지고 있는 디폴트 컨텐츠 타입이 x-www~로 되어 있기 때문에 원하는대로 못갔음

따라서 스크립트에서 content-type을 json으로 지정해주면 제대로 나온다

data : JSON.stringify(form),

contentType : "application/Json",

**Json.stringify() 하면, 컨텐트타입을 정해준다는 것을 같이 묶어서 외울 것 !!**

**우리가원하는 Json타입을 지정해준다 !!**

**------ 참고 ------**

jObj.result != "SUCCESS" ||

select.displayChildSelectBox(jObj.childList);

- 간편한 조건문인데 최근에 표준으로 바뀐 조건문

하이픈을 기준으로 좌측이 false면 우측을 실행한다

**------ 참고 ------**

- 이제 오류처리에 대해서 배울 것

컴파일 오류 : 내 손가락이 낸 오류, 차라리 좋음 왜냐면 빨간색의 오류를 확인 할 수 있고 그것을 고치면 정상이 됨,

실행 오류 : 손가락이낸 오류가 아닌 것, 내가 for문을 돌리는데 i가 0이고, i가 0보다 작거나 같으면 무한반복 하면서 서버가 뻗는데, 이건 eclipse에서도 오류가 확인이 안되는데 이게 실행오류 앞으로 실행오류에 대해서 공부할 예정,, 우리가 흔히 말하는 버그

- 시스템오류는 ex) 서버에 메모리공간이 모자르다, 따라서 우리가 신경쓸 필요가 없음, 우리가 제어할수 있는 영역이 아니니

예외(exception) : 우리가 제어할 수 있는 오류로 앞으로 공부 할 것

**- 가져온 7줄을 맵에 담에서**

**{1=한큐, 2=두큐 …, 7=뿌꾸뿌꾸}로 출력해라**

: Map을 하나 만들고 맵에다가 .put 메서드를 통해서 넣어준다.

해당 리스트의 한줄을 빼고, 그줄의 “seqNo”를 키로

그리고 해당하는 원하는 컬럼인 userName을 value로 넣어준다.

**for** (**int** i=0; i< welcomeWebList.size(); i++) {

exceptionMap.put((String) welcomeWebList.get(i).get("seqNo"),

welcomeWebList.get(i).getValue(1));

}

System.***out***.println("exceptionMap : " + exceptionMap);

getvalue(1) 은 두번째 컬럼의 키를 가져온다. 인덱스상 0부터 시작하기 때문에

**- 담은 맵의 데이터를 배열로 똑같이 출력해라**

: 배열을 선언하고 맵의 개수 만큼 포문을 돌리면서

키는 맵의 것을 가져오고 value는 배열로 가져온다.

맵에 있는 밸류를 어떻게 배열에 넣을지

String[] arrStr = **new** String[exceptionMap.size()];

**for** (**int** i=0; i<exceptionMap.size(); i++) {

arrStr[i] = (String) exceptionMap.get(String.*valueOf*(i+1));

System.***out***.println((String) exceptionMap.get(String.*valueOf*(i+1)));

}

String.valueOf( i+1) 는 get은 맵에서 무언가를 뽑아올 때 쓰는 건데, 파라미터에서

String. 은 스트링으로 바꾸는데, 바꾸기 위해서 valueOf를 사용한 것이고 왜 i+1을 했지??

>> 지금 값을 가져올건데 한큐 두큐 세큐 네큐를 지금 배열에 넣으라고 했는데,

그 해당하는 한큐 두쿠 세큐가 들어 있는 키는 지금 첫번째 키가 아닌 두번째 키에 있음

seqNo는 첫번째 키 // userName은 두번째 키

i를 그냥 (i)로 하여 배열에 넣었다면 seqNo가 배열에 들어갔을 것 !!