**Select**

**Nu e o forma separata, ci face parte din form:form**

* In HTML folosim

<select>

<option>Val1 </option>

<option>Val2 </option>

</select>

* In MVC

<form:select path=””>

<form:option value=”Val1” label=”Val1”/>

<form:option value=”Val2” label=”Val2”/>

</form:select>

Path – numele metodei dupa set si get

Label – ceea ce se vede pe ecran

Value – ceea ce se trimite in cod

**Old form tags**

<form method="get" action="form">  
 <input type="checkbox" value="CSS" name="form">css</input>  
<input type="checkbox" value="Java" name="form">java</input>  
<input type="checkbox" value="HTML" name="form">html</input>  
<input type="checkbox" value="Python" name="form">python</input>  
<input type="checkbox" value="R" name="form">r</input>  
  
 <input type="submit" value="trimitere">  
</form>

O asa form va returna un param object ce va avea in el un Array String[] cu numele “form”, si deci vom avea nevoie de asa RequestMapping

Cand mai multe taguri identice au acelasi name, se creaza deci un String[], nu String simplu

@RequestMapping("/form")  
public String form(@RequestParam("form")String[] list){  
 for(String s : list){  
 System.*out*.println(s);  
 }  
 return "index";  
}

**Deci param poate contine nu doar String, ci si String[]**

**Select From a definted List**

* Putem crea si un obiect, cum ar fi Country, ce sa fie membru al unei clase, pe baza careia se face forma, si sa fie gen un HashMap, adica sa dam deodata formei select o lista definita.
* Pentru asta, cream in <form:form> </>

<**form:select** path="countries">  
 <**form:options** items="${student.country}"/>  
</**form:select**>

* Aici vom avea nevoie sa cream in Clasa corespunzatoare un membru de tip HashMap, si gen sa-l umplem cu valori, in Constructor de ex, si apoi sa cream o metoda getter si pentru hashMap, pe care o va folosi form:options
* **path** – variabila de tip String ce va stoca alegerea facuta
* **items** – hashmap cu itemele din select
* Key din HashMap va fi value

Value din HashMap va fi label

* **HashMap nu trebuie sa fie neaparat de tip HashMap<String, String>.**

**Ca value, trebuie sa fie mereu String, ca sa il putem afisa pe pagina, dar key, care e ceea ce se trimite de forma, nu se arata, poate fi de orice tip, si un obiect custom.**

**Utilizarea lui <util>**

* Utilizarea lui <context:property-paceholder location=”classpath:”/> e buna numai daca dorim sa extragem din property files cate o valoare corespunzatoare unui atribut. De ex, daca avem properties file:

Val1=text1

Val2=text2

Putem folosi @Value(“${Val1}”) pentru a obtine text1, dar problema e ca nu putem sa primim nicidecum toate datele deodata, si nici nu putem sa irnegistram valoarea Val1 ca key.

* Pentru a putea inregistra un property file ca Map, folosim:

<util:properties id=”nume” location=”classpath:/..”/>

<util:properties id="countries" location="classpath:/countries.properties"/>

Si

xmlns:util="http://www.springframework.org/schema/util"

* El ne ofera un id prin care ne putem referi la el, insa <context:property-placeholder> nu ne ofera optiunea de a-i seta un ID ca nume! In plus, asa putem avea mai multe properties files, caci ele au id la <util>
* Pentru a utiliza toate valorile sale ca un Map, folosim @Value(“#{ID}”)
* Folosim # IN LOC DE $

In Configuration facem asa:



**Radio Buttons**

* Tag: <form:radiobutton path=”” value=””/>
* Nu avem label, asa cum putem seta un text langa buton si singuri

Java <**form:radiobutton** path="programmingLanguage" value="Java"/>

* **Pentru a adauga mai multe butoane dintr-un Map, cream un Map in clasa obiectului, il setam cu @Value din properties:**

<**form:radiobutton*s*** items="${studentObj.programmingLanguageList}" path="programmingLanguage" />

**Folosim $, nu #, deoarece vom returna un Map, deci nu e nevoie sa se extraga mai multe valori, ceea ce se face cu #.**

* **Label nu avem si din cauza ca oricum la Map**

**Key din HashMap va fi value**

**Value din HashMap va fi label**

* Nu e nevoie de name, asa cum oricum vom putea alege doar unul din toate butoanele radio din forma
* Rezultatul va fi un String. Deci,

path="programmingLanguage"

acesta e Stringul caruia i se va trimite alegerea facuta

items="${studentObj.programmingLanguageList

Acesta e HashMap de unde se iau valorile afisate

**CheckBox**

* RadioButtons pot permite selectarea unui buton, dar checkBox a mai multor.
* Tag: <form:checkbox path=”” value=”” />

<**form:checkbox** path="os" value="Linux OS" label="Linux"/>  
<**form:checkbox** path="os" value="Mac OS" label="Mac"/>  
<**form:checkbox** path="os" value="Windows OS" label="Windows"/>  
<**form:checkbox** path="os" value="Androix OS" label="Androix"/>

* Insa, variabila din clasa care va pastra alegerile facute nu va mai fi un String, ci un Array String, adica String[], caci doar avem multichoise si forma va returna un String[] pentru checkBoxes
* Daca alegem sa folosim mai multe butoane, nu unul de tip form:checkboxes, folosim la path acelasi nume, caci oricum forma la trimitere, va crea un obiect de tip String[] pentru “os”
* Totusi, afisarea rezultatului pe o alta pagina web e mai problematica. Aici avem nevoie de un loop in JSP, si il importam cu:

<%@ **taglib** uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

Pentru ca el sa functioneze, punem dependenta:

<dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
</dependency>

Il folosim asa

<ul>  
 <**c:forEach** var="val" items="${studentObj.os}">  
 <li>${val}</li>  
 </**c:forEach**>  
</ul>

Unde var poate fi orice variabila dorim.

* Rezultatul returnat cu alegerile va fi de tip String[]

**CheckBoxes**

<**form:checkboxes** path="os" items="${studentObj.osList}"/>

* **Inapoi se va returna un String[] cu alegerile facute**

**Aceasta e membrul de tip String[] unde se vor pastra alegerile facute**

path="os"

Acesta e membrul HashMap de unde se iau obiectele de afisat

items="${studentObj.osList}

* **HashMap nu trebuie sa fie neaparat de tip HashMap<String, String>.**

**Ca value, trebuie sa fie mereu String, ca sa il putem afisa pe pagina, dar key, care e ceea ce se trimite de forma, nu se arata, poate fi de orice tip, si un obiect custom.**

**In asa caz, de ex daca HashMap e de tip HashMap<Teacher, String>, forma va returna Teacher[]**

**Atentie! Daca vom vrea sa recuperam de la HttpServletRequest o String[], metoda getParameter nu va mai merge, ci getParameterValues:**

@GetMapping("/processChoice")  
public String processChoice(HttpServletRequest servletRequest){  
 String[] countries = servletRequest.getParameterValues("select");  
 String[] language = servletRequest.getParameterValues("language");  
 String job = servletRequest.getParameter("job");  
   
 return "info";  
}