**Import**

* Putem accesa clasele si obiectele din Package in JSP.
* Pentru asta, la inceput de JSP folosim:

<%@ **page** import="com.java.MyApp.Utility" %>

* Putem accesa in JSP toate librariile ce putem si in Java, si cu anotatia de sus si package ale noastre. Pentru a spune ca vrem sa accesam anume o valoare din Java sau Package importat, scriem:

**value=”<%=Clasa.atribut%>”**

De ex:

<**c:url** var="firstName" value="/customer/listSort">  
 <**c:param** name="fName" value="<%=Integer.*toString*(SortClass.*FIRST\_NAME*)%>"/>  
</**c:url**>

aici cream o variabila atribut ce va avea un parametru cu valoarea FIRST\_NAME din clasa SortClass. Am folosit si clasa Integer din libraria standart Java. Putem folosi orice folosim si in proiectul nostru din Java.

**required=false**

* @RequestParam default este @RequestParam(required=true).
* Totusi, s-ar putea sa nu avem nevoie ca el sa fie obligatoriu, caci daca un param nu e gasit, avem eroare.Deci folosim:

@RequestParam(required=false).

**Param conversii**

* In JSP, orice param trebuie mereu sa aiba valoare de tip String, de aceea daca ii oferim o variabila din program ce e de tip int, trebuie sa facem conversia in String ca mai sus:
* <**c:param** name="fName" value="<%=Integer.*toString*(SortClass.*FIRST\_NAME*)%>"/>
* In schimb, in Java, conversia din String se poate face automat in int, de aceea putem scrie
* @GetMapping("/listSort")  
  public String listSorted(@RequestParam(name = "sort", required = false)int sort,Model model) {  
   List<Customer> customers = customerService.sortCustomers(sort);  
    
   model.addAttribute("customers",customers);  
   return "list-customers";  
  }

parametrul local e de tip int, desi param e de tip String, dar Java face conversiile automat.