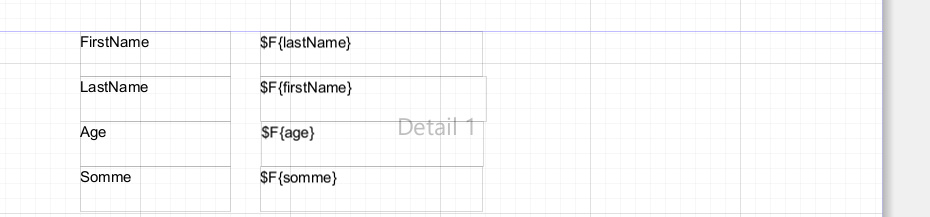
* **Subreport** - Un subreport este un report in interiorul alui report
* **Subreporting** – apelarea unui report din interiorul altui report
* **Nested report** = subreport
* **Nesting report** = parent report, main report
* Subreports ne permit sa refolosim acelasi design in oricate main reports
* Un subreport poate folosi un datasource de la main report, sau o poate crea pe a sa
* Putem trece parametri de la master report la subreport
* Un subreport poate returna o valoare catre un main report
* Un subreport poate avea si le subreports in el
* Un subreport mereu trebuie compilat inainte de a-l rula in reportul principal, de altfel nu vom vedea modificarile sale
* Subreports sunt folosite in reports folosin fisierele .jasper, deci care sunt deja compilate, nu folosim deci jrxml

**Creare**

* Cand adaugam un subreport element, va trebui fie sa importam un report existent ca sa fie subreport, fie alegem sa cream unul nou ce sa fie setat ca subreport
* Acum, cream parametrii. Totul depinde de ceea ce vrem sa facem. De ex, daca vrem sa afisam datele mai multor clienti, fiecare client avand un subraport a lui, vom face asa:

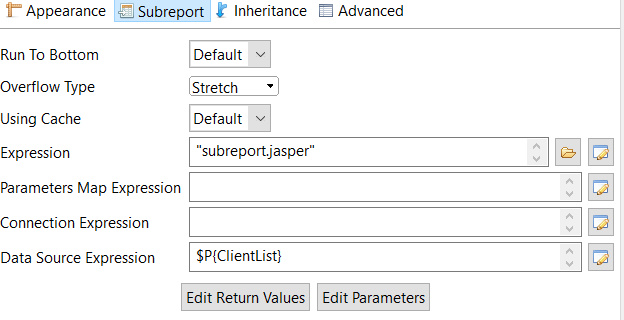
- Cream un parameter in main report, care sa fie sursa de clienti, adica un parametru de tip JRBeanCollectionDataSource

- Nu ne trebuie nici un parametru pentru fiecare client.

- In subreport adaugam fields pe care le va avea fiecare client. Nu e nevoie sa adaugam in subreport un parametru,asa cum $F{} daca nu are specificat vreun parametru, sunt cautate in DatSet a raportului, si main report i va trimite un dataset pe care il vom seta. Asa va arata subraportul

Si nu are vreun parametru Client, exact ca main report, ci va prelua fiecare obiect din DataSet global si il va folosi

- Acum, alegem subreportul din main report, si alegem parametrul care va fi sursa, lista cu obiecte





Si gata asta e tot. NU trebuie parametru pentru fiecare client sau ceva. JR va trimite oricum lista la subreport, in dataset a lui, si subreport va accesa acele fields pentru fiecare obiect din dataset

* Codul va fi asa:
* @SpringBootApplication  
  public class Main {  
   public static void main(String[] args) throws JRException {  
   SpringApplication.*run*(Main.class);  
    
   String path = "C:\\Users\\user\\Desktop\\demo\\src\\main\\resources\\templates\\";  
   Map<String, Object> map = new HashMap<>();  
   map.put("ClientList", new JRBeanCollectionDataSource(List.*of*(  
   new Customer("Mititiuc","Eduard",21,20000.0),  
   new Customer("Goncear","Alex",25,1000.0),  
   new Customer("Petru","Turcan",20,500.0)  
   )));  
    
   JasperReport report = JasperCompileManager.*compileReport*(path+"clients.jrxml");  
   JasperPrint print = JasperFillManager.*fillReport*(report,map, new JREmptyDataSource());  
   JasperExportManager.*exportReportToPdfFile*(print,path+"report.pdf");  
   }  
  }

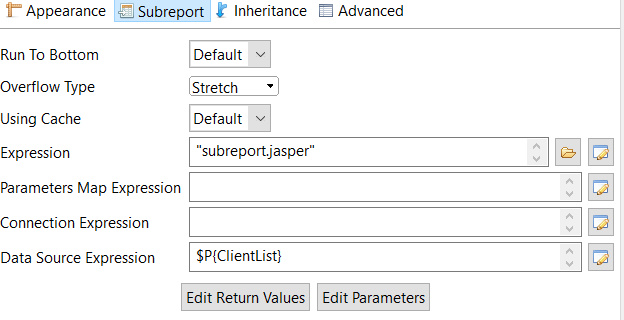
Si mare atentie! Main raport va fi in .jrxml, dar subrapoartele mereu vor fi in .jasper, deci subrapoartele necesita sa fie compilate

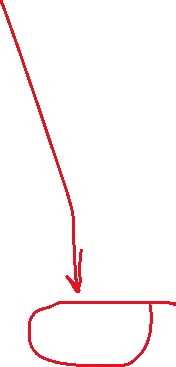
In .jrxml, avem grija sa punem, cand lucram cu spring, adresa la .jasper a rapoaartelor

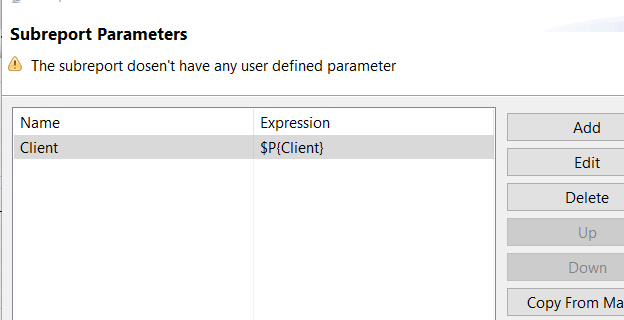
<subreportExpression><![CDATA["C:\\Users\\user\\Desktop\\demo\\src\\main\\resources\\templates\\subreport.jasper"]]></subreportExpression>

**Edit parameters**

**-** Putem sa apasam pe **Edit Parameters** si sa adaugam un parametru ce sa fie trimis catre subreport.

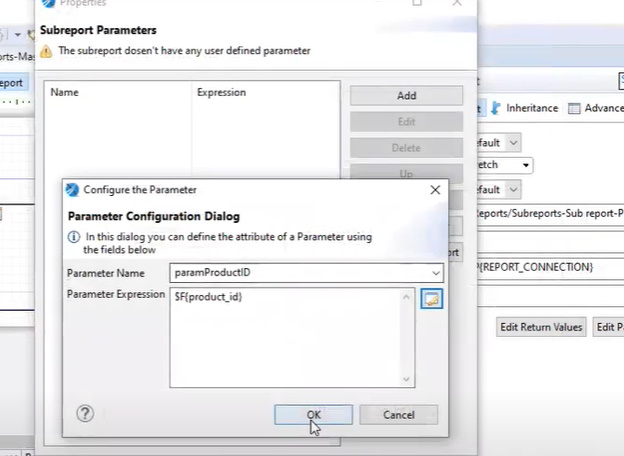






**Name** – numele parametrului din Subreport.. El nu trebuie sa existe in main report neaparat.

**Expression** – aici specificam anume ce se va trimite din main report in parametrul Client din subreport. Putem trimite orice, de ex un id al obiectului:

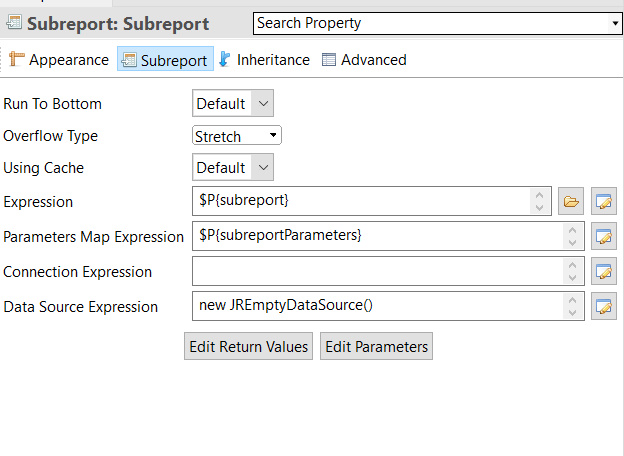


In imagine, trimitem parametrului cu numele paramProductID din subreport valoarea fieldului product\_id din obiect scos din lista. Asa deci se va trimite obiectul din lista in dataset global din subreport si se va seta si acest parametru inca. Mai sus, parametrul din subraport mereu va avea valoarea actuala a fieldului, in dependenta de ce obiect din datasource e next

* In query, putem adauga si parametri sau fields

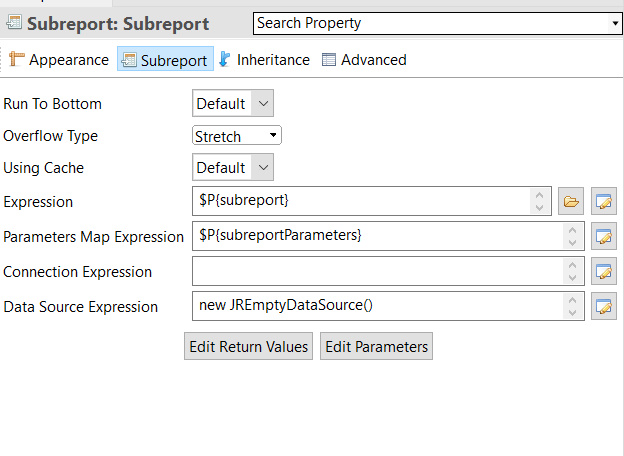
**Subreport advanced**

* Decat sa trimitem adresa fisierului .jasper a subreportului, am putea face si altfel, si anume sa cream in main report un parametru de tip JasperReport si anume el sa fie subreportul compilat, si deci vom crea un subreport in main report pe baza la acest parametru, il setam in Expression, aici:





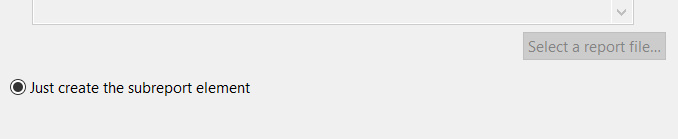
* Tototdata, mai putem si crea un parametru in main report de tip Map<String, Object> ce va contine parametrii la subraport, si il vom trimite la Parameters Map Expression:



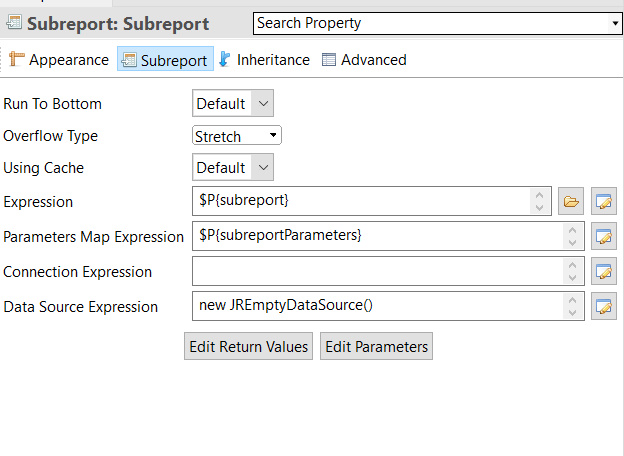


**Pasi:**

1. Cream in main report un parametru de tip JasperReport, ca el sa fie prorpiu zis subreportul compilat
2. Cream un parametru de tip Map, ca el sa contina parmetrii subreportului
3. Cream subreport si il adaugam in report main. Atentie sa nu alegem un report .jasper, vom trimite un parametru care deja e raportul, deci mare grija sa alegem asta cand cream subreport:

****

1. **Cream datasoruce in subreport, daca e vorba de tabel de ex. Si cream fields in el.**
2. In subreport, cream parametriu cu datele, de tip JRBeanCollectionDataSource
3. Adaugam subreport in report si mare grija sa cream si un empty data source:





Adaugam in templates, la java, ori oriunde, fisierele .jrxml pentru main raport si subreport:

String path = "C:\\Users\\user\\Desktop\\demo\\src\\main\\resources\\templates\\";  
Map<String, Object> reportParam = new HashMap<>();  
Map<String, Object> subreportParam = new HashMap<>();  
  
JasperReport jasperReport = JasperCompileManager.*compileReport*(path+"clients.jrxml");  
JasperReport subreport = JasperCompileManager.*compileReport*(path+"subreport.jrxml");  
  
reportParam.put("subreport", subreport);  
subreportParam.put("data", new JRBeanCollectionDataSource(List.*of*(  
 new Customer("Mititiuc","Eduard",21,20000.0),  
 new Customer("Goncear","Alex",25,1000.0),  
 new Customer("Petru","Turcan",20,500.0)  
  
)));  
reportParam.put("subreportParameters", subreportParam);  
  
JasperPrint print = JasperFillManager.*fillReport*(jasperReport, reportParam, new JREmptyDataSource());  
JasperExportManager.*exportReportToPdfFile*(print,path+"report.pdf");