

|  |  |
| --- | --- |
| **S.NO** | **Element and Description** |
| 1 | **Title**  Title contains the 'Title' of the report. It appears only once at the very beginning of the report, for example, "Tutorials Point Report." |
| 2 | **pageHeader**  PageHeader may contain date and time information and/or organization name. This appears at the top of each page. |
| 3 | **columnHeader**  ColumnHeader lists the names of those specific fields, which you want to display in the report, for example, "Author Name," "Starting Hour," "Finishing Hour," "Hours Worked," "Date," etc. |
| 4 | **Detail**  Detail is the part where entries of the specific fields (listed in columnHeader) are shown, for example "Manisha", "9:00", "18:00", "9", "10.02.2013." |
| 5 | **columnFooter**  ColumnFooter may display summation of any of the field, for example, "Total Hours Worked: "180." |
| 6 | **pageFooter**  PageFooter may contain page count information. It appears at the bottom of each page, for example, "1/23." |
| 7 | **Summary**  Summary contains information inferred from "detail" part, for example, after listing the number of hours, worked by each author, total hours worked by each author can be put in visual chart like pie chart, graph, etc. for better comparison. |

**Elemente**

* **Frame** – element care poate contine alte elemente, si sa creeze o bordura pentru a le face sa para ca ca un intreg. Este un container de elemente. Fiecare element e un node ce poate avea si el elemente in el

**Fields, parameters etc.**

* **Parameter: $P{}** – Simple input la report. Poate contine intre {} un string sau object din Java code al nostru. Parametrii vor fi inlocuiti cu date ce vin din program
* **Fields: $F{}** – este un field concret dintr-un obiect sau datasource object trimis la report. Fields pot contine mai multe entries, de ex column entries intr-un tabel
* **Varibl**e: **$V{}** – nu sunt date trimise la report. Ele sunt create in report pentru a pastra valori agregate
* **Internationalization $R{}** –

Putem adauga text impreuna cu parametri:

"Firstname:"+$P{firstName}

“” va fi static text, iar $P va fi dinamic

**Spring Boot**

1. **Trebuie dependenta:**

<dependency>

<groupId>net.sf.jasperreports</groupId>

<artifactId>jasperreports</artifactId>

<version>6.20.6</version>

</dependency>

1. **In templates, cream un fisier .jrxml ce va contine codul la report**
2. **Scriem codul**



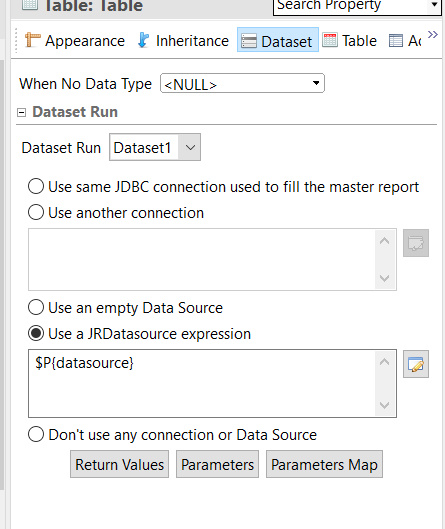
String path = "C:\\Users\\user\\Desktop\\demo\\src\\main\\resources\\templates\\testreport.jrxml";  
  
Map<String, Object> parameters = new HashMap<>();  
parameters.put("firstName", "Mititiuc");  
parameters.put("lastName", "Eduard");  
parameters.put("age", 21);  
parameters.put("dob","21/07/2020");  
  
JasperReport report = JasperCompileManager.*compileReport*(path);  
JasperPrint print = JasperFillManager.*fillReport*(report, parameters, new JREmptyDataSource());  
JasperExportManager.*exportReportToPdfFile*(print, "C:\\Users\\user\\Desktop\\demo\\src\\main\\resources\\static\\MyReport.pdf");

* In Map vom stoca parametrul ca key, si valoarea lui ca value. Numele trebuie sa fie identice cu parametrii din JRXML
* JasperReport – este raportul compilat, gata sa fie umplut cu date, dar inca nefinisat
* JasperPrint – raportul terminat, serializat, gata sa fie stocat in formate concrete, ca pdf
* JREmptyDataSource e datasource gol.

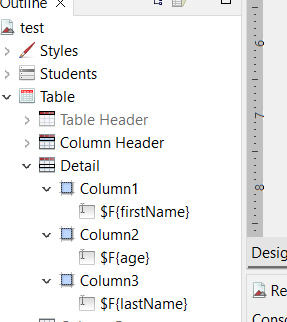
**Table**

* Fiecare tabel are un dataset atribuit
* Dataset tot contine fields, variables etc.
* pentru tabel, avem nevoie sa cream fields, nu parameters, caci un parametru poate contine o singura valoare, dar un fields se refera la un field a unui obiect concret:
* Pentru a adauga coloane, apasam de 2 ori pe tabel, ca sa se deschida intr-o noua file, Detail si acolo sunt coloanele. Selectam o coloana si adaugam un Text Field in ea

1. Cream un parametru global, dar ce sa fie de tip **JRBeanCollectionDataSource**
2. **Cream un datasource si fielduri in el**
3. **Cream un tabel si alegem datasource creat**
4. Ne ducem la tabel la dataset si alegem jos parametrul ca sursa de date



1. Adaugam static text pentru coloanele ce vor avea nume constante, si tragem field text in coloanele ce vor fi umplute cu date(Detail) si alegem field din data source pentru fiecare



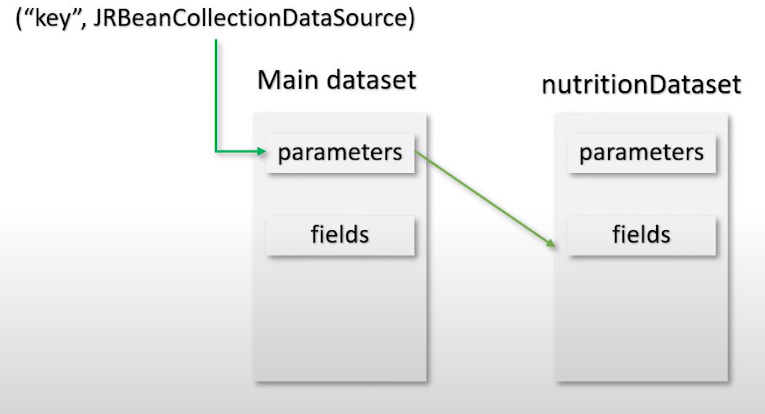
* Double click pe text field adaugat si adaugam in el field cu $F

**$F{nutritionName}**

Column Header – aici deobicei punem numele la coloane. Dam Create Row pentru a crea linia pentru el

Details – aici sunt deobicei randurile cu datele dinamice

* Acum, mai trebuie deja cumva sa setam ca tabelul sa preia datele din aplicatie
* Pentru asta, e necesar sa cream un parametru, dar nu in dataset, ci in reportul propriu zis, si acest parametru va fi de tip: **JRBeanCollectionDataSource**, caci un parametru simplu stocheaza o valoare, dar un asa parametru mai multe, deci e o colectie
* Acum ne ducem la tabel, dataset si selectam parametrul ce sa fie colectia de date venita de la aplicatie

****

* Acum in code vom crea o classa, care va fi ca un entity pentru tabel. Numele la fileduri trebuie sa fie indentice cu fields din dataset

@Data  
@AllArgsConstructor  
public class Nutrition {  
 private String nutritionName;  
 private Integer total;  
 private Integer goal;  
 private String metric;  
}

Cream o lista cu entittati

List<Nutrition> data = List.*of*(  
 new Nutrition("Bread",100,110,"g"),  
 new Nutrition("Milk",120,140,"g"),  
 new Nutrition("Cucumber",200,240,"g"),  
 new Nutrition("Pie",300,320,"g"),  
 new Nutrition("IceCream",500,580,"g")  
);  
JRBeanCollectionDataSource dataSource = new JRBeanCollectionDataSource(data);

si neaparat am creat si JRBeanCollectionDataSource care va prelua elementele din lista

Si adaugam acest object in parameters hash map

SpringApplication.*run*(DemoApplication.class, args);  
  
 String path = "C:\\Users\\user\\Desktop\\demo\\src\\main\\resources\\templates\\testreport.jrxml";  
  
 List<Nutrition> data = List.*of*(  
 new Nutrition("Bread",100,110,"g"),  
 new Nutrition("Milk",120,140,"g"),  
 new Nutrition("Cucumber",200,240,"g"),  
 new Nutrition("Pie",300,320,"g"),  
 new Nutrition("IceCream",500,580,"g")  
 );  
 JRBeanCollectionDataSource dataSource = new JRBeanCollectionDataSource(data);  
  
 Map<String, Object> parameters = new HashMap<>();  
 parameters.put("firstName", "Mititiuc");  
 parameters.put("lastName", "Eduard");  
 parameters.put("dob","21/07/2020");  
 parameters.put("age", 21);  
 parameters.put("datasource", dataSource);  
  
  
 JasperReport report = JasperCompileManager.*compileReport*(path);  
 JasperPrint print = JasperFillManager.*fillReport*(report, parameters, new JREmptyDataSource());  
 JasperExportManager.*exportReportToPdfFile*(print, "C:\\Users\\user\\Desktop\\demo\\src\\main\\resources\\static\\MyReport.pdf");  
}

**Mare atentie sa nu omiti EmptyDataSource!**

**Sorting Column in table**

* Click pe tabel, dataset si acolo avem Sort Fields
* Click dreapta, Create Sort Field

**Charts**

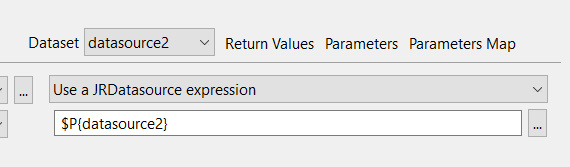
* Reportul are un Main dataset, acolo si se pastreaza Parameters si Fields globale

1. Cream un parametru ca **JRBeanCollectionDataSource**
2. Cream un dataset pentru un chart
3. Cream 2 fields:

Unul va fi key(numele la proprietate)

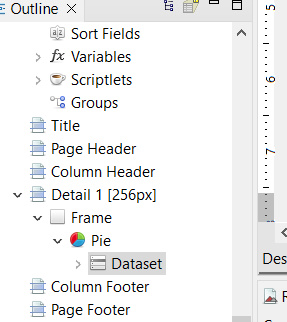
Altul va fi value(valoarea ce va fi luata pentru a face calcule in chart)

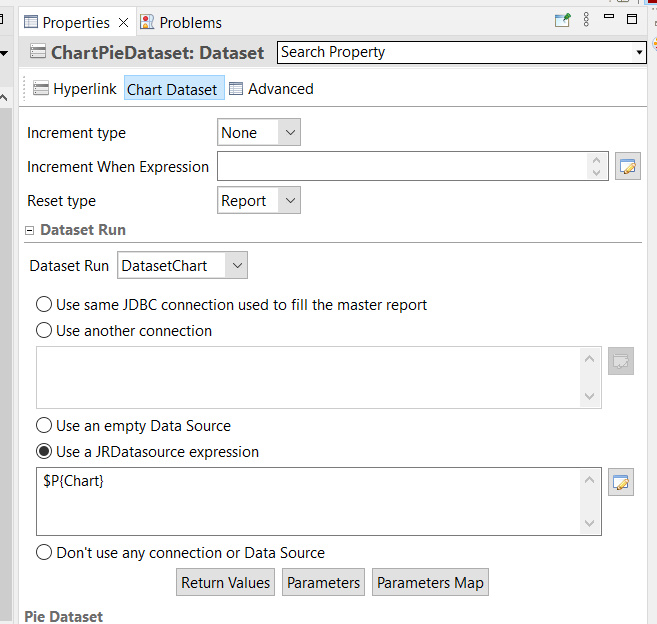
1. Cream chart si in optiuni:
2. Selectam dataset
3. apasam pe ... de la series si alegem key, si el se va pune automat
4. Alegem value si label si mare grija la label sa il facem string, folosind + “”. caci label mereu trebuei sa fie string, caci asta va fi ce afisam prorpiu zis ca valoare
5. Alegem parametrul de unde vin datele



1. Putem selecta max slices to show. Daca de ex selectam 5, vor fi afisate doar 5, insa daca de ex oferim 10, primle 4 cele mai mari vor fi afisate, si celelelate 6 vor fi puse toate in una mica.

6. Adaugam un parametru pentru datasourcece, daca nu e setat, ce va contine obiectele sursa la Filter Expression. Atentie unde intram:





**Accesarea la fieldurile unui parametru**

1. Cream un parametru, dar nu il setam ca Object! Intram in jrxml si il setam ca altceva, class necesar, caci doar asa vom putea

<parameter name="Param" class="com.example.demo.Student"/>

1. Adaugam text field si pur si simplu folosim $P{parametru}.getField() si gata, de ex:

$P{Param}.getFirstName()

1. Cream obiectul si il punem in Map si gata