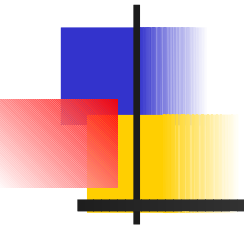


Annexe : DataSources Jasper

David THIBAU – 2009

david.thibau@gmail.com

Interface JRDataSource

- L'interface *JRDataSource* définit deux méthodes :
  - ***public boolean next()***
    Cette méthode doit retournée *true* si le curseur est positionné correctement sur l'enregistrement suivant. Chaque fois que le moteur invoque cette méthode. Les valeurs des champs sont remplis avec les valeurs de l'enregistrement courant et les expressions sont réévaluées.
    En conséquence, les entêtes d'un nouveau groupe peuvent être imprimées, un saut de page peut s'effectuer, ...
    Si cette méthode retourne *false*, les bas de groupe, bas de colonne, la section de dernière page et le résumé sont imprimés.
  - ***public Object getFieldValue(JRField jrField)***
    Cette méthode retourne la valeur du champ passé en paramètre pour l'enregistrement courant.

Ensemble de JavaBeans

- Un ensemble de JavaBeans implémente un *JRDataSource*
- Un JavaBean est une classe Java qui expose ses attributs avec des méthodes getter :
  `public <returnType> getXXX()`
  XXX est le nom du champ;
- Si le type de *XXX* est également un JavaBean. Il est possible d'utiliser la notation :
  XXX.YYY
- Le flag *Use Field description* permet d'utiliser la description du champ plutôt que son nom.
- La collection ou le tableau de JavaBeans doit être retourné par une méthode *static* d'une classe fournie (factory).

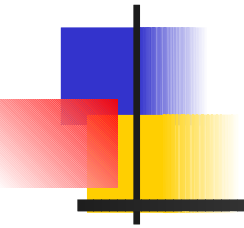
Exemple JavaBean

```
public class PersonBean {
  private String name = "";
  private int age = 0;

  public PersonBean(String name, int age){
    this.name = name;
    this.age = age;
  }

  public int getAge() {
    return age;
  }
  public String getName(){
    return name;
  }
}
```

Exemple Factory

```
public class SampleFactory {

  public static java.util.Collection generateCollection() {
    java.util.Vector collection = new java.util.Vector();
    collection.add(new PersonBean("Ted", 20) );
    collection.add(new PersonBean("Jack", 34) );
    collection.add(new PersonBean("Bob", 56) );
    collection.add(new PersonBean("Alice",12) );
    collection.add(new PersonBean("Robin",22) );
    collection.add(new PersonBean("Peter",28) );
    return collection;
  }
}
```

XML DataSource



- A la différence du format « table » voulu par Jasper, un document XML est organisé hiérarchiquement en forme d'arbre
- Une expression *XPath* est alors nécessaire pour extraire un ensemble de nœuds du document.
- L'expression *XPath* peut être fournie :
  - dans le rapport; dans ce cas il est possible d'utiliser les paramètres du rapport dans l'expression XPath
  - Au moment de la création de la datasource.
- Au moment de la création du datasource, il est également possible de spécifier des patterns Java pour convertir les chaînes de caractères XML en dates ou nombres

Champs XML

- La syntaxe *XPath* est également utilisée pour définir les champs. L'expression étant évaluée à partir du nœud courant. Elle se spécifie dans la description du champ.
- *IReport* propose un outil visuel pour la sélection des champs

Nom	Description	
id	@id	L'attribut id
lastname	lastname	La valeur du noeud-enfant lastname
categoryName	ancestor::category/@name	L'attribut name du noeud ancêtre category

Documents XML et sous-rapport

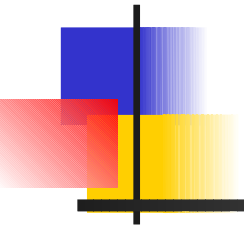
- L'organisation hiérarchique des documents est très adaptée à la notion de sous-rapport de JasperReport
- Une *JRXmlDataSource* expose 2 méthodes supplémentaires :
  - *public JRXmlDataSource dataSource(String selectExpression)*
  - *public JRXmlDataSource subDataSource(String selectExpression)*

La première méthode applique une expression *XPath* à partir du nœud racine du document XML. La seconde applique l'expression sur le nœud courant.

L'expression pour spécifier les données d'un sous rapport est alors :

```
((JRXmlDataSource)
$P{REPORT_DATA_SOURCE}).subDataSource(xExpression)
```

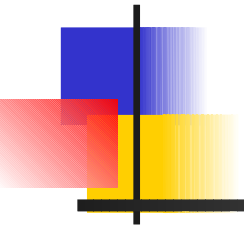
CSV DataSource



- Les fichiers CSV sont un format naturel pour une source de données Jasper.
- Les noms des champs sont :
  - Soit obtenus à partir de la première ligne du fichier
  - Soit spécifié un à un
- Il est possible également de spécifier le caractère délimiteur

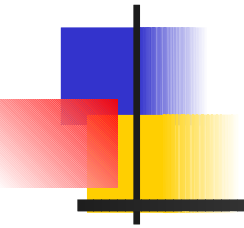
JREmptyDataSource

- JasperRepoprt fournit une datasource spéciale permettant d'effectuer des itérations sur des enregistrements dont la valeur de champ est *null*
- L'interface iReport permet de spécifier le nombre d'itérations voulues



Hibernate

- Avec Jasper, il est possible d'utiliser une requête HQL.
- La connexion hibernate (*hibernate.cfg.xml*) et les fichiers de mapping (**.hbm.xml*) doivent être dans le classpath.
- La requête est exprimée alors via HQL.
- En fonction du résultat retourné par la requête, les champs du rapport sont alors des entités Hibernate ou des objets simples Java.
- Il est possible de naviguer dans les objets entités avec la notation « . »



Datasources personnalisées

- Fournir une classe implémentant l'interface *JRDataSource*
- Fournir une factory ayant une méthode statique retournant la datasource personnalisée