

# Création d'états avec JasperReports et JSF

par Faisel Chabli ([Accueil](#))

Date de publication : 19/03/2008

Dernière mise à jour : 19/03/2008

Cet article a pour objectif de vous présenter une des façons pour éditer un état JasperReports depuis une application JSF.

---

I - Introduction.....	3
II - Génération du fichier source de l'état via iReport.....	4
2.1 - Présentation.....	4
2.2 - Téléchargement et installation.....	4
2.3 - Configuration de la source de données.....	4
2.4 - Configuration du répertoire de compilation.....	5
2.5 - Génération de l'état.....	5
III - Exécuter un état depuis JSF.....	9
3.1 - Configuration de l'environnement de travail.....	9
3.2 - Création du managedBean.....	9
3.3 - Configuration du managedBean dans le fichier faces-config.xml.....	10
3.4 - La création de la page JSF.....	11
3.5 - L'exécution du projet.....	11
IV - Conclusion.....	13
V - Remerciements.....	14

## I - Introduction

Cet article servira à présenter les taches de création d'un état JasperReports avec iReport ainsi que son utilisation depuis une application JSF.

## II - Génération du fichier source de l'état via iReport

### 2.1 - Présentation

Selon **définition** : Un état est tout simplement : un document présentant des informations organisées pour leur visualisation ou leur impression (pas de modification ou de mise à jour des infos possible). La liste des objets présents dans le stock d'une boutique est un exemple d'état. Syn: Rapport.

Jasper interprète un fichier source xml (.jrxml) qui définit le format de l'état, y injecte les données de l'utilisateur et génère des états dans divers formats (pdf, excel, etc.) iReport est une sorte de WYSIWYG qui sert à éditer le fichier jrxml.

Les outils utilisés pour fournir cet exemple sont : Eclipse/Tomcat5.5/MySQL5.0/iReport2.0.1.

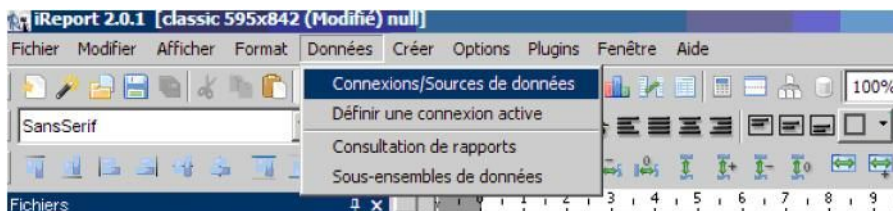
Notre base à exploiter est composée d'une seule table Person qui contient les champs suivant : id, nom, prenom, age. A partir de cette table on veut générer un état iReport pour le compiler et ensuite l'exécuter à partir d'une application JSF afin de l'exporter en PDF et Excel.

### 2.2 - Téléchargement et installation

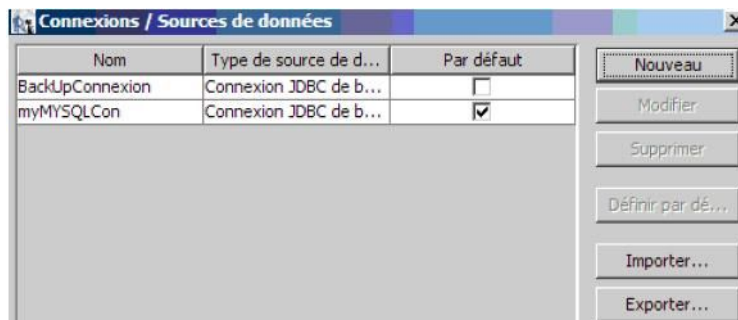
J'utilise iReport 2.0.1 pour la création des états, il peut être téléchargé à partir de [téléchargement de iReport](#). Après avoir téléchargé iReport 2.0.1 mettez le dans votre répertoire favoris et lancez le en cliquant sur **iReport.exe** sans aucune installation.

### 2.3 - Configuration de la source de données

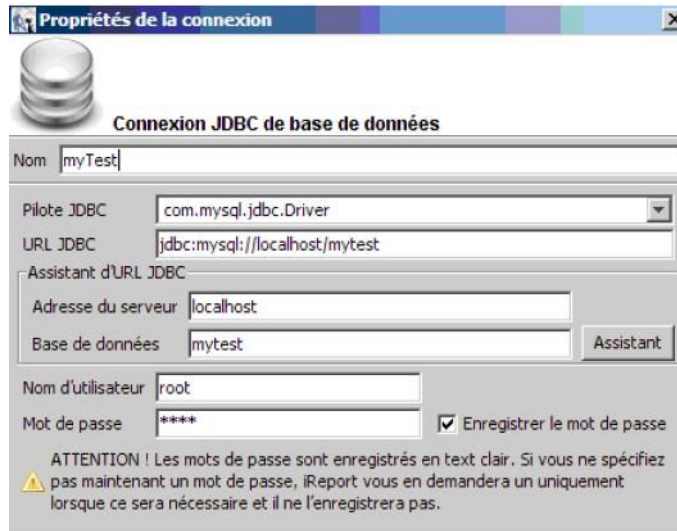
Avant de commencer à réaliser ses états, il va falloir configurer la source de données pour pouvoir se connecter à la base de données et éditer l'état sous iReport avant de passer son fichier compilé (le fichier JASPER) pour le déploiement. Cette source de données nous servira uniquement pour faire des tests d'exécution de notre état sous iReport. A noter que ce n'est pas cette source de données qui sera utilisée dans notre application JSF. Pour faire cette configuration, vous allez sur le menu : **Données-->Connexions/Sources de données**, cliquez dessus :



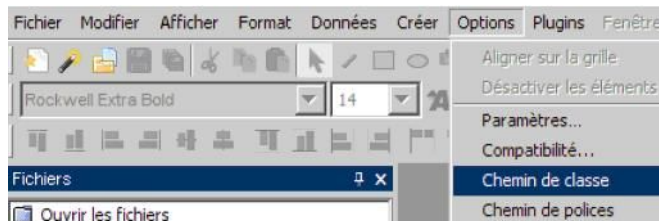
Une fenêtre s'ouvre pour configurer sa connexion au SGBD :



Cliquez sur **Nouveau**, choisissez **Connexion JDBC de base de données** et cliquez sur suivant et entrez vos paramètres de connexion comme suit :



**Pilote JDBC** : représente le Driver de la base de données à utiliser. L'outil iReport cherche par défaut le driver, sous format .JAR, dans le dossier lib situé dans le répertoire d'installation de iReport. Dans mon cas, le lib est sur C:\softwares\iReport-2.0.1\iReport-2.0.1\lib. Dans ce dossier on trouve bien le driver de MySQL, à savoir **mysql-connectorjava-3.1.11-bin.jar**. Si le driver de la base ne se trouve pas dans ce dossier, alors il va falloir le paramétrer en l'ajoutant au chemin de classe de iReport. Pour ajouter un driver, qui ne se trouve pas par défaut dans le lib de iReport, au chemin de classe il suffit d'aller sur **Options-->Chemin de classe** :



Après cliquer sur Ajouter JAR pour sélectionner le JAR du driver souhaité. **URL JDBC** : représente l'adresse de connexion à votre base de données.

**Adresse du serveur** : représente le nom ou l'adresse IP de la machine contenant la base de données à partir de laquelle vous souhaitez éditer vos états.

**Nom d'utilisateur/mot de passe** : c'est le nom d'utilisateur /mot de passe de la base de données.

Cliquez sur **Test** pour voir si la connexion a été bien effectuée à votre base de données, après faites **enregistrer**.

## 2.4 - Configuration du répertoire de compilation

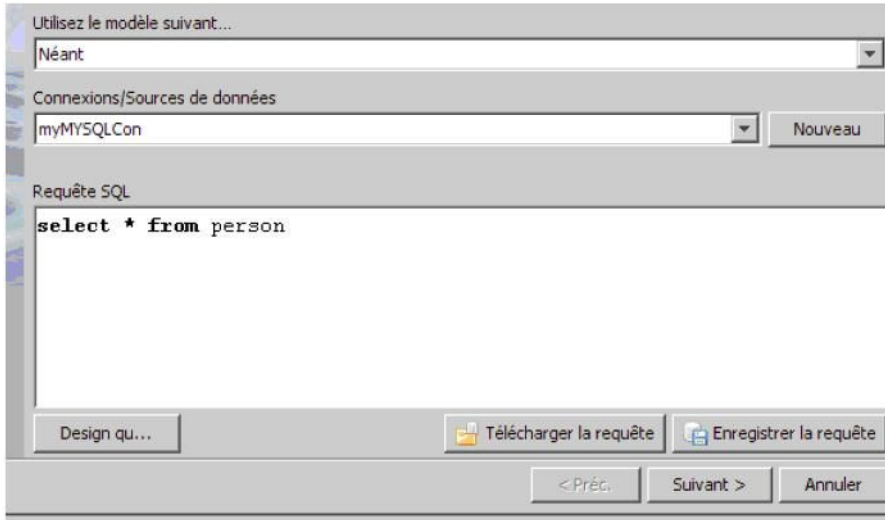
Le fichier résultant d'une compilation sera déposé par défaut dans la racine où est installé le répertoire de iReport. Il vaut mieux donc avoir un répertoire dédié aux fichiers compilés, un répertoire que vous pouvez configurer à partir de **Options-->Paramètres-->Onglet Compiler-->faites parcourir** sur le répertoire de compilation par défaut pour sélectionner votre répertoire qui contiendra dorénavant vos fichiers compilés, les JASPER.

## 2.5 - Génération de l'état

Maintenant que votre connexion a été configurée, on passe à la création de notre premier état. On va utiliser l'assistant de iReport pour créer nos états. Faites un click sur le bouton de l'assistant en haut à gauche :

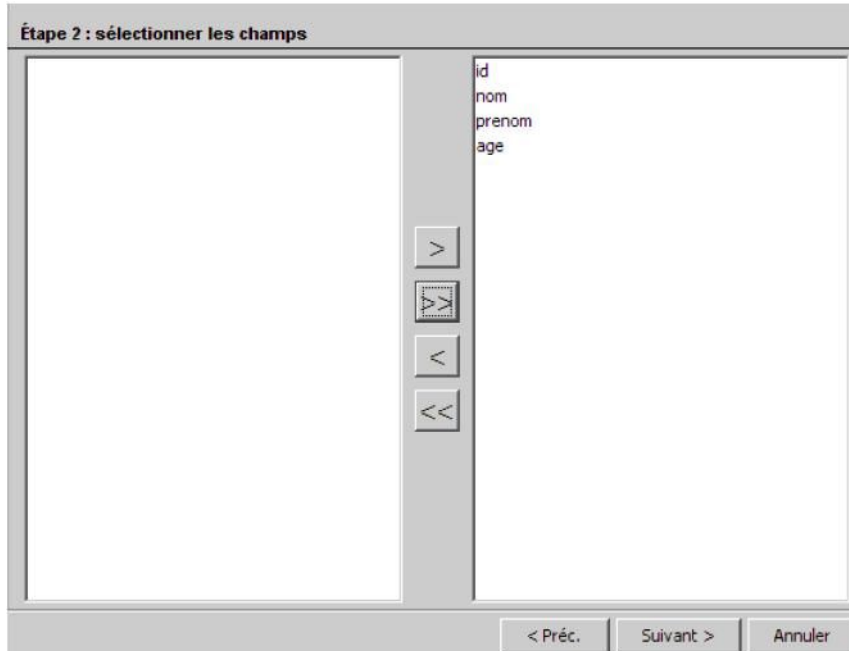


Dans la fenêtre qui s'ouvre saisissez votre requête SQL.

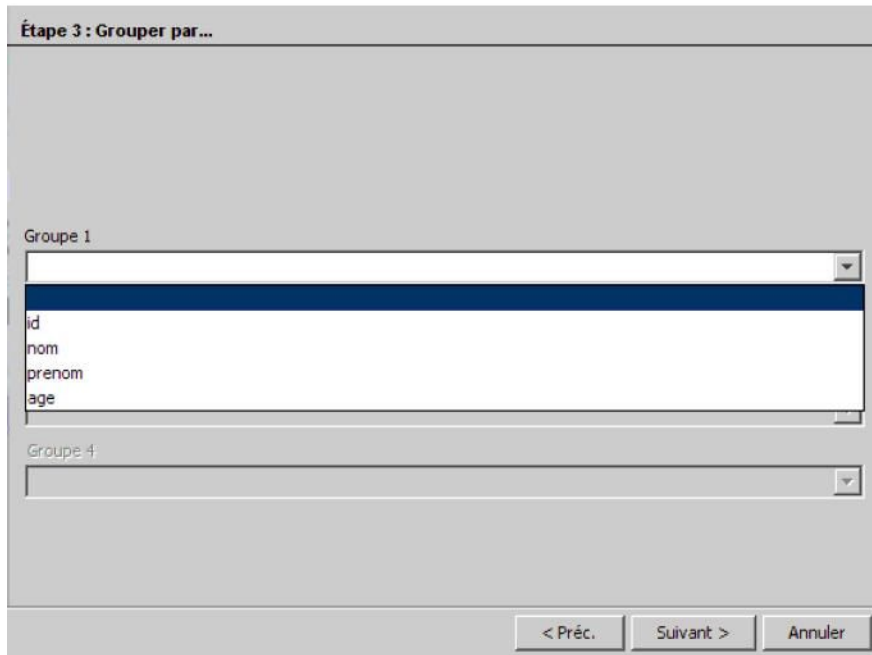


Vérifiez bien si votre connexion est bien sélectionnée dans **Connexions/Sources de données**.

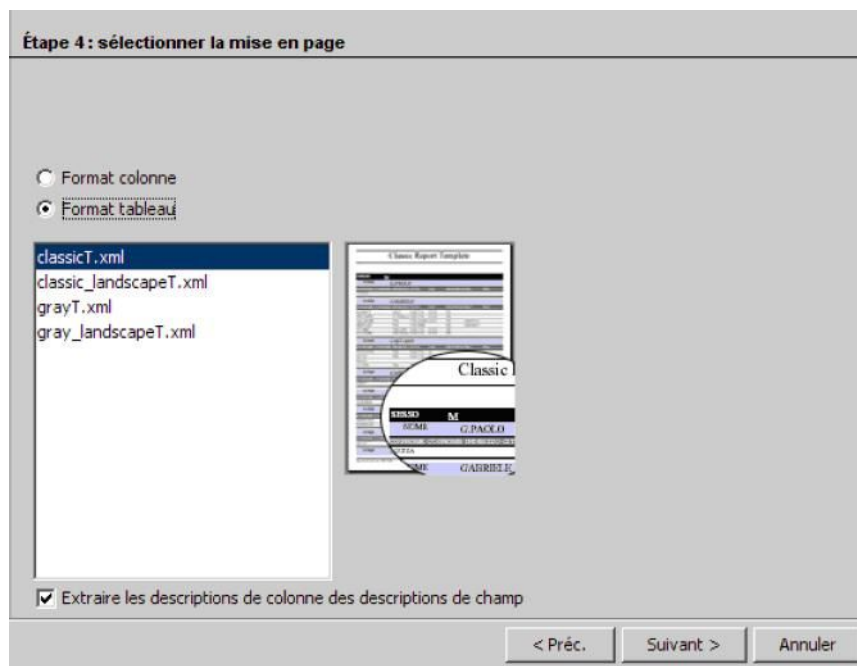
Cliquez sur **Suivant**> et sélectionnez les champs que vous souhaitez éditer, dans notre cas on sélectionnera tous les champs.



Cliquez sur **suivant**. L'écran suivant permet de faire des groupements, dans notre cas on n'en aura pas besoin. Cet écran se présente comme suit :

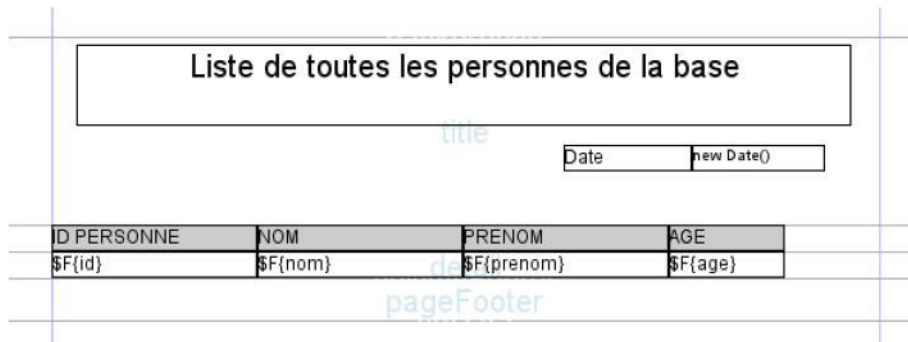


Cliquez donc sur **suivant** pour passer sur l'écran qui nous permettra de sélectionner **la mise en page** de notre état. Cet écran se présente comme suit :



Sélectionnez le **Format tableau (\*)**. Cliquez sur **Suivant** et après **Terminer**. Bravo votre premier état est créé sous iReport.

Vous pouvez ajuster les champs pour rendre les choses un peu agréables et belles à voir, pour moi j'utilise un affichage du style :



**Format tableau** : Dans ce format vous aurez les noms des champs en haut, les données correspondantes en bas en ligne. Les noms des champs s'affichent une seule fois sur l'état. Si vous choisissez le **Format colonne** au lieu du Format tableau alors les noms des champs s'afficheront en ligne et se répéteront pour chaque enregistrement de type :

id	1
nom	sdsd
prenom	sdsd
age	89
id	2
nom	FCH
prenom	FCH
age	89

Cet état est composé d'un titre en haut centré " Liste de toutes les personnes de la base ", de la date du système en haut à gauche, cette date est extraite via l'expression *new Date()*, ensuite une table avec une colonne pour chaque champ de la table.

Une fois votre état conçu vous pouvez l'exécuter pour voir un aperçu des données saisies dans la table Person. Nommez le fichier person.jrxml. Cette exécution va générer un fichier person.jasper qui sera déposé dans le répertoire précédemment configuré.

Après l'exécution de votre état, vérifiez si le fichier jasper a été bien créé dans le répertoire configuré précédemment dans la partie **2.2. Configuration du répertoire de compilation**. C'est ce fichier qu'on va utiliser par la suite dans notre application web.

## III - Exécuter un état depuis JSF

### 3.1 - Configuration de l'environnement de travail

- jasper-runtime.jar.
- jasperreports-1.2.8.jar
- mysql-connector-java-5.0.5-bin.jar
- itext-2.0.1.jar
- poi-2.0-final-20040126.jar

Ces JAR sont à mettre dans le *CLASSPATH* du projet en les ajoutant au chemin de génération.

### 3.2 - Création du managedBean

Dans le dossier src on va créer un package, par exemple *com.jaub.view* dans lequel on crée une classe *InitPageBean*.

Le corps de cette classe est constitué essentiellement de deux méthodes, l'une pour la visualisation excel, l'autre pour pdf.

Ces deux méthodes se présentent comme suit :

**Pour la visualisation PDF :**

#### Code1 - Visualisation PDF

```
public String viewReportPDF() throws SQLException, JRException, IOException {
    String reportId = "Person";
    Driver mDriver = new Driver();
    DriverManager.registerDriver(mDriver);
    Connection connection = DriverManager.getConnection(
        Constants.DRIVER_DATABASE_MYSQL,
        Constants.LOGIN_DATABASE_MYSQL,
        Constants.PASSWORD_DATABASE_MYSQL);
    File file = new File(Constants.PATH_OF_REPORTS_JASPER);
    JasperPrint jasperPrint = JasperFillManager.fillReport(
        new FileInputStream(new File(file, reportId + ".jasper")),
        null, connection);
    byte[] bytes = JasperExportManager.exportReportToPdf(jasperPrint);
    FacesContext context = FacesContext.getCurrentInstance();
    HttpServletResponse response = (HttpServletResponse) context
        .getExternalContext().getResponse();
    /******
    * Pour afficher une boîte de dialogue pour enregistrer le fichier sous
    * le nom rapport.pdf
    * *****/
    response.setHeader("Content-disposition",
        "attachment;filename=rapport.pdf");
    response.setContentLength(bytes.length);
    response.getOutputStream().write(bytes);
    response.setContentType("application/pdf");
    context.responseComplete();
    return null;
}
```

**Pour la visualisation Excel :**

#### Code2 - Visualisation EXCEL

```
public String viewReportExcel() throws SQLException, JRException, IOException {
    String reportId = "Person";
```

### Code2 - Visualisation EXCEL

```

Driver mDriver = new Driver();
DriverManager.registerDriver(mDriver);
Connection connection = DriverManager.getConnection(
    Constants.DRIVER_DATABASE_MYSQL,
    Constants.LOGIN_DATABASE_MYSQL,
    Constants.PASSWORD_DATABASE_MYSQL);
File file = new File(Constants.PATH_OF_REPORTS_JASPER);
JasperPrint jasperPrint = JasperFillManager.fillReport(
    new FileInputStream(new File(file, reportId + ".jasper")),
    null, connection);
FacesContext context = FacesContext.getCurrentInstance();
HttpServletResponse response = (HttpServletResponse) context
    .getExternalContext().getResponse();
if (jasperPrint != null) {
    ByteArrayOutputStream xlsReport = new ByteArrayOutputStream();
    JRXlsExporter exporter = new JRXlsExporter();
    exporter.setParameter(JRXlsExporterParameter.JASPER_PRINT,
        jasperPrint);
    exporter.setParameter(JRXlsExporterParameter.OUTPUT_STREAM,
        xlsReport);
    exporter.exportReport();
    // Send response
    byte[] bytes = xlsReport.toByteArray();
    /*****
    * Pour afficher une boîte de dialogue pour enregistrer le fichier
    * sous le nom rapport.xls
    *****/
    response.setHeader("Content-disposition",
        "attachment;filename=rapport.xls");
    response.setContentType("application/vnd.ms-excel");
    response.setContentLength(bytes.length);
    response.getOutputStream().write(bytes, 0, bytes.length);
    response.getOutputStream().flush();
    response.getOutputStream().close();
} else {
    Writer writer = response.getWriter();
    writer.write("Aucun rapport à afficher");
    response.setContentType("text/HTML");
}
return null;
}

```

La partie accès à la base de données peut être externalisée dans une autre classe mais comme je l'ai cité au début de cet article, ce n'en est plus l'objectif.

Dans la ligne :

#### Remarque

```

JasperPrint jasperPrint = JasperFillManager.fillReport(
    new FileInputStream(new File(file, reportId + ".jasper")),
    null, connection);

```

le 3ème paramètre de *JasperFillManager.fillReport* est mis à *null*. Ce paramètre permet de passer des paramètres à l'état. Dans notre cas on veut éditer tous les éléments de la table Person, donc on n'a pas de paramètres à passer.

### 3.3 - Configuration du managedBean dans le fichier faces-config.xml

Dans le faces-config.xml on ajoute le bout de code suivant :

#### Code3- faces-config.xml

**Code3- faces-config.xml**

```
<managed-bean>
  <managed-bean-name>InitPageBean</managed-bean-name>
  <managed-bean-class>
    com.jaub.view.InitPageBean
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

**3.4 - La création de la page JSF**

Notre page JSF se présente, tout simplement, comme suit :

**Code4 - init.jsp**

```
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root version="1.2" xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:jsp="http://java.sun.com/JSP/Page">
  <jsp:directive.page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" />
  <f:view>
  <html>
  <head>
  <title>Test iReport With JSF</title>
  </head>
  <body style="background-color: #fff4db">
  <h:form id="reportForm" target="report">
  <h:commandButton id="pdfButton" value="Visualiser PDF"
    styleClass="buttonStyle" action="#{InitPageBean.viewReportPDF}" />
  <h:commandButton id="excelButton" value="Visualiser EXCEL"
    styleClass="buttonStyle" action="#{InitPageBean.viewReportExcel}" />
  </h:form>
  </body>
  </html>
  </f:view>
</jsp:root>
```

On y met deux boutons, l'un pour la visualisation pdf, l'autre pour la visualisation Excel. Chaque bouton a sa propre action. Lors d'un click sur le bouton " Visualiser PDF " c'est la méthode viewReportPDF du managedBean InitPageBean qui sera appelée, lors d'un click sur la bouton " Visualiser Excel " c'est la méthode viewReportExcel qui en sera appelée.

**3.5 - L'exécution du projet**

Après avoir tout configuré, lancez le projet JSF et cliquez sur les boutons pour faire vos tests.

Résultat :

## Liste de toutes les personnes de la base

Date 28/01/08 12:17

ID PERSONNE	NOM	PRENOM	AGE
1	FCH	FCH	26
1	MCH	MCH	26
1	AAI	AAI	26

## IV - Conclusion

Cet article a objectif de vous expliquer une des façons pour éditer un état JasperReports depuis une application JSF. Je me suis contenté d'afficher tous les enregistrements d'une table, dans un prochain article je vous expliquerai comment on peut passer des paramètres à notre état pour afficher juste les éléments dont on a besoin.

## V - Remerciements

Je tiens à remercier infiniment **djo.mos** pour ses retours et ses conseils. Je tiens à remercier aussi **ArHacKnIdE** pour ses corrections orthographiques.