

Nell'articolo di oggi vedremo come generare il nostro report in pdf partendo dal modello generato con Ireport.

Il primo passo è quello di generare il nostro report a partire dal formato jrxml. Per fare questo JasperReports ci mette a disposizione la classe JasperCompileManager

```
JasperReport jasperReport = JasperCompileManager.compileReport("reportFromXml.jrxml");
```

Tale classe permette di compilare il report e ottenere un file .jasper, lo stesso file che otteniamo quando generiamo l'anteprima con Ireport.

A questo punto resta da riempire il report compilato con i dati che provengono da un datasource. Questo meccanismo viene realizzato dalla classe JasperFillManager

```
JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, new HashMap(), ds );
```

La classe JasperFillManager mette a disposizione il metodo fillReport che permette di riempire il nostro report tramite una serie di parametri che vengono forniti tramite un oggetto di tipo Map e un datasource di tipo JRDataSource.

L'interfaccia JRDataSource è un'interfaccia che prevede la definizione di 2 metodi: il metodo next che permette di scorrere il datasource e il metodo getField che permette di recuperare la proprietà del bean del datasource.

Nel nostro esempio ho definito una semplice implementazione dell'interfaccia che permette di scorrere un elenco di libri e accedere ad alcune sue proprietà.

```
JRDataSource ds = new JRDataSource() {
```

```
Collection<Libro> beanCollection = (Collection<Libro>) LibroFactory.getBeanCollection();
private Iterator<Libro> iterator = beanCollection.iterator();
private Libro libro = null; int counter = 0;
@Override
public boolean next() throws JRException {

    if (counter>=beanCollection.size())    return false;    libro = iterator.next();    counter++;
return true;
}

@Override
public Object getFieldValue(JRField arg0) throws JRException {
    if (arg0.getName().equals("autore"))
        return libro.getAutore();
    if (arg0.getName().equals("id"))
        return libro.getId();
    if (arg0.getName().equals("titolo"))
        return libro.getTitolo();
    if (arg0.getName().equals("genere"))
        return libro.getGenere();
    return null;
}
};
```

Il metodo next permette di scorrere il nostro datasource. Ritorna true se è presente un altro elemento all'interno del datasource, mentre il metodo getFieldValue permette di accedere alla singola proprietà del record.

A questo punto tramite la classe JasperExportManager possiamo estrarre il nostro report nel formato desiderato.

```
JasperExportManager.exportReportToPdfFile(jasperPrint, "c://Simple_Report.pdf");
```

[Qui](#) trovate le classi per compilare il progetto

[Prossimo](#)

