

## 第9章 文件上传与PDF报表入门

- 理解DataURL的基本使用，实现DataURL的文件上传
- 完成基于七牛云的文件上传
- 理解 JasperReport生命周期
- 独立完成 JasperReport的入门案例

### 1 图片上传

#### 1.1 需求分析

● 详情 ×

---

登录账户设置   个人详情   岗位信息

---

工号：	<input type="text" value="111"/>	入职时间：	<input type="text" value="2018-11-04"/>
姓名：	<input type="text" value="zbz"/>	部门：	<input type="text" value="测试部"/>
手机：	<input type="text" value="13800000003"/>	聘用形式：	<input type="text" value="正式"/>
员工照片：			

如图所示，实现员工照片上传功能

#### 1.2 Data URL

##### 1.2.1 DataURL概述

所谓DataURL是指"data"类型的Url格式，是在RFC2397中提出的，目的是对于一些“小”的数据，可以在网页中直接嵌入，而不是从外部文件载入。

##### 1.2.2 Data URL入门

- 完整的DataURL语法：DataURL= data:**mediatype**;base64,<Base64编码的数据>。
- **mediatype**：表述传递的数据的MIME类型（text/html，image/png，image/jpg）

简单的说，data类型的Url大致有下面几种形式。

data:,<文本数据>

```
data:text/plain,<文本数据>

data:text/html,<html代码>

data:text/html;base64,<base64编码的html代码>

data:text/css,<css代码>

data:text/css;base64,<base64编码的css代码>

data:text/javascript,<javascript代码>

data:text/javascript;base64,<base64编码的javascript代码>

data:image/gif;base64,base64编码的gif图片数据

data:image/png;base64,base64编码的png图片数据

data:image/jpeg;base64,base64编码的jpeg图片数据

data:image/x-icon;base64,base64编码的icon图片数据
```

对于再程序开发中，使用最多的是基于DataURL的图片形式，接下来以图片形式的DataURL分析其原理和利弊

### 1.2.3 Data URL基本原理

[Data URL](#)给了我们一种很巧妙的将图片“嵌入”到HTML中的方法。跟传统的用 `img` 标记将服务器上的图片引用到页面中的方式不一样，在Data URL协议中，图片被转换成[base64](#)编码的字符串形式，并存储在URL中，冠以mime-type。

图片在网页中的使用方法通常是下面这种利用 `img` 标记的形式：

```

```

这种方式中，`img` 标记的 `src` 属性指定了一个远程服务器上的资源。当网页加载到浏览器中时，浏览器会针对每个外部资源都向服务器发送一次拉取资源请求，占用网络资源。大多数的浏览器都有一个并发请求数不能超过4个的限制。这意味着，如果一个网页里嵌入了过多的外部资源，这些请求会导致整个页面的加载延迟。而使用Data URL技术，图片数据以base64字符串格式嵌入到了页面中，与HTML成为一体，它的形式如下：



[illegible]



```
GDBG/C3q00nQMJoXyMy0SywYxr6Jpw+XGkiExbYw4tohq8VjJJ1nnmNoaM8ivz3CwH2sVaYcQ0B87Zn3k1vb11N
mhJ3ttY46XISrpW4HrUZ/z2qCjAMSQQW/vO4Pyq2ogQneP7uOvEANTjx8I+o9PFLipw0QH2hrzeLEbjjbIvvtN
x3CSHMPVtUy0hf6BOIfCPd4AdC6PeUj8+aR1A1Vrnme98JULYEYKFfgfl+yDFwrm8xfM3H0ekvcIjkhJX0LnODa
AMWIC0T4y8r1j+s7u+4qJQrmICafvkmPkVufE3wF/j4STGFbOoct43KrZ8pESN1O1V5PZGvDzjAM+2ANAGP6ew9P
X+DSYayaOs0zn1ZypU9n1Z0tWxDWxoFbOCndPsCY/YgwU/kJ0Rfr1uSBSiCa1ieYz868TsZJAWMdL6znSbsu8q
AF8qrvdNCPjOnvQS0j4NMBWHWCqXYCntuJDvMT3CKEwwz2bEaJkF9rqmshwZI6/VjEC1vSn6DeoQggHgt5c2tGO
9LeL3z8cA7/gCUJQPOQo38eT13wZ01m/zLMeemQxtK1KLHPgIz/QjI5yOw1Ms2wekBostsMOpfjUm2CGIz39awL
pE1ALrh3ZnFOY1Rxrs9Zxvhe27MuD6F6rySWxp7eL6roqebUVSC99L+e/6n7CJ91d5qw/n1B4G98EkyPewg4qZu
om0mHEEO/+BM8cpnx18kr0S2oGW/1TW8Fuq3nZGRMGGrz/yYLM/S3EmMiJNLAg6saQPKNWjXZUuX0DcaJKmNy6
3z0tXLSPDHyZFPDDPNOCVsyLSJPv+Y1r6C0g0jZaYmfo8ajyC5tKmaPC94L6WJIhCS3LLDbS70h3hnt29gXCQo+
Kwd61BNib7HQuWRWdQnprea+Z1iO/VG267zSKZ7cjAKeq6Ik4rM1PIHwq3+f7jMEVYPDupI2JxwOTvfiKgZGX1H
wwi8PCo4H3S1iwmQTG5w7n9C8JnWET5vgQ1ig7E37LkWL1t8NfeQ7FiD8Pi3F/WPTHOfTotoCuWDbE3zslEDZw6
kojwJ7J1gd43zCWzwzZbQwfdYzIZYqjLGOMY2UBHVGyq61JT1ZYQRDPy5o4ExvvtwTKNUZ3/A9+AVFCbcnJW8LJf
k96AknxNC44zQ6bI6IFjPftH5LptEz4TJsb6ZJY9kyAOv1YFT5jYAmQ7NyjHPYXAAXSMDLcq/5waLoHDhg1seL
DGYffsgA/L4Y1xeQETm19MinOHUdfjZGZ01v+pw4DtVgww5aBPZMwwD9ESMWRW8mvf/AFvgPoodVylIsYde08ng
EzqOerBKdy1SCwt1G5VZpNBWUEPCRG/GeqouXyFYpNRi9uThVVS0VvhJCCSZX17CDg4v1dv1mCG+8tsMOZx04Wh
DDPyBpsiLF33sLBHITDWM+7VMO9sbgQYezdN0en9QUcP70H2L+kY/oeqQgWTmHGduu7MaFQEmLRIWZZQBqCGBqL
VPzThmsi2Bc4DM9uIEkbsCNGAdgSVX1BGWInWGuD9e/Y+ueabDP/n/s108g1SnGAXssy4J4LVfA/fhTxmRA0mZ+
PhUCGZ0dwXzi22CGOTRHYURUpeZfM2LALCZNxWDG3axU0LmZ8j2aZ11QROGazQc3x1135Gwr07BnEz9j09jrfmx
WEe4r0ZvXjsfwM4zGO783e2w1FenPP7sLouCiNSQ+jk80Hq5yESu0N5Y3fMN2XMjv5D7agRzFP2UapB2vcps/A
M6ZLAE2wy5mHuYvjkaJe/9RhUfLMOH5GXUOgw23im47is80jG3wCd0+paJPjQSEpATT0IMTxmxkAx92ECsAywdw
3MZfwJDX702P5dPzPAXLgTvmw0kTWnezQvRWqcuMGLD0FwnHwdckoHry9QiR3qyI70NI393Ky4eCYsQ1hzzNYie
EXMIAQTmd/BfZYcfRWuP9vRWRhB8WEI/lDskiA7UIdz8LJYHo8HDxNNMce+Pxb1tww+eZQdCoe5iuYaqqvM4mG
wFR98RhvkA4jEx4CEP8AHK2KvM+YVr+B2m/bGmBecidkP16kUO/XqBgY4KxFHMx6vq+dvJCCIJtAHZgwbhENbbe
AvAtaw00PmwzOSFhw3PMQLWU+QV1KEmeJT3RpxbRogVrufw/UUC+OcxT1hRnCNtMBa2NwGOTneq3A6tvtZrXY76
5xAP+9I43iHuPUBc2xaAGit1p9OQXMDGw32q8tmgu+kvD1w547Db5At4A/pDVe5ckPvi849jVOYQ9toT+eDY0va
QSm8Hkctv+AcPWDVYJfBOxarwnWghZnJoR8W+GBaxRmkXAV5PWax92eaLzgc/5JGoitw1jwCDY1an3kdyMvv8v
cdwAdmYVgSAUDN1rwPRP1id9SVLN8q1r8nDV6J6HD1Bc1QKncw/DLio4aKrPiVZ5t8yAt90QrrUxjDF69NACva
K1RWY+Qi11Gj3s8K/xjcvpxKr0NuUEg60ILsTKNDPN0vg1N+StDfa/QdwGILrTSSEt8DWnga/xNY5U7mm6sI/8L
modu1ntt31MRCBvymB5G5CnQzQQ09iyvV4id9myVW9dvfj+CPsxIpnDkX6qire1c60jLA2Ue004PP04r7/TCdp
oXur6AuuyHR+6LDPXUYX9K0UjvZT/7v6qkPEWtXhN8NO+gp7Y6Q3vn0SV/F3Sk4ms1yNE/pMC8m9B53c3j3kb5d
Q/qNzaot4iGgNL/j2N23EE4ERY91EqdzTiepvb44SOCANZQHsCciuvW4jo6pQ7ssw78vNHghKa/Ri/Z1kKFruPO
1lgkaAt33EIQ6o/rEq1U6UQipwuhtN1/DxJfD/TiqCYv9s0z4LRYIoAwwI1Tn5ga08d8R15vynq29AeyCm0xGGb
J1aI6Nfk4XCaUWck89iZbTtV5U5C60P+/hQpmKJTU46+xxnEwrVkaE1YzpnWvQer30Gr1/N+1wv6jGfagxGYV0i
ti1QBj8p8j9LYoPz/EbmmSjKpSAkbx2yeeJ/XXMe9PchV3mRnWU1gldChC0o/j9SxjBGMEOKvTHS4Epu77tv8g0
iJMBSjet+MOPvRisInu1T4Hu3pJxzG3z5Ia1WPIUV0hj/xo7NEG87jkwxB/5GBjtnMw0RnHuG9W/LedRn8KBGh5
KX0vd71SNWLHEvS1dtov5q/wwuFyEuS4jHquoy0ZRV1v7CraVrcPSUL/2OUrVTuP1PLLEO3La2yaw8qAN9Ihf+T
shEBHZRH0L1d0pjcfVm6KXxyqQmS5na9E8X5Ii9NOR3ay2rgkQ51ya4d5DPwFoVfVJrM26sjr1eM9ksAI//7ey
ikv8RVyaz1Chk87LadGYvk0xqJOcn0dufljvHjJwZ0gpgmHTonoE9k5ps4TFhg8msqVn1Z1LV8j8BBgdJtf+jYX
034QAAAABJRUErkJggg==
```

上面的base64字符串中你看不出任何跟图片相关的东西，但下面，我们将传统的 `img` 写法和现在的Data URL用法左右对比显示，你就能看出它们是完全一样的效果。但实际上它们是不一样的，它们一个是引用了外部资源，一个是使用了Data URL。

## 1.2.4 优缺点分析

### 1. 浏览器支持

几乎所有的现代浏览器都支持Data URL格式，包括火狐浏览器，谷歌浏览器，Safari浏览器，opera浏览器。IE8也支持，但有部分限制，IE9完全支持。

### 2. 数据容量

Base64编码的数据体积是原数据的体积4/3，也就是DataURL形式的图片会比二进制格式的图片体积大1/3。

### 3. 使用场景

DataURL形式的数据不会占用HTTP会话，所以再访问外部资源或当图片是在服务器端用程序动态生成时借用DataURL是一个不错的选择

## 1.3 Data URL实现用户头像上传

- 修改用户实体类，用户数据库表添加用户头像字段
- 使用基于Data URL的方式实现用户上传，实质是将前端上传的文件以Base64进行编码并且保存到数据库中。
- 用户controller中添加用户上传方法
- 用户service中添加上传文件处理的方法
- 在service中需要对文件进行base64编码，并且保存到数据库中

### (1) 在系统微服务的 UserController 中添加上传处理的方法

```
@RequestMapping(value="/user/upload/{id}")
public Result upload(@PathVariable String id,@RequestParam(name = "file")
MultipartFile file) throws Exception {
    String image = userService.uploadImage(id, file);
    return new Result(ResultCode.SUCCESS,image);
}
```

### (2) 在系统微服务的 UserService 中添加上传处理的方法

```
public String uploadImage(String id, MultipartFile file) throws Exception {
    //根据id查询用户
    User user = userDao.findById(id).get();
    //对上传文件进行Base64编码
    String s = Base64.encode(file.getBytes());
    //拼接DataURL数据头
    String dataUrl = new String("data:image/jpg;base64,"+s);
    user.setStaffPhoto(dataUrl);
    //保存图片信息
    userDao.save(user);
    return dataUrl;
}
```

## 2 七牛云存储

### 1.1 概述

[七牛云对象存储](#)服务提供高可靠、强安全、低成本、可扩展的非结构化数据的存储服务。它提供简单的 Web 服务接口，可以通过七牛开发者平台或客户端存储和检索任意数量的数据，支持“按使用付费”模式，可以通过调用 REST API 接口和 SDK开发工具包访问，下载协议采用HTTP 和 HTTPS 协议。方便程序员聚焦业务应用，而无需关注底层存储实现技术。

使用七牛云实现图片存储也比较简单只需要按照如下的步骤操作即可：

#### 1. 申请七牛云账号



2. 创建空间 Bucket
3. 上传文件
4. 请求获取图片

## 1.2 账户申请

(1) 进入七牛云官方网站注册开发者账户

七牛云是通过邮箱注册的，注册激活后就进行认证，认证后即可开通对象存储业务了

### 欢迎注册七牛云

注册账号

激活邮箱

完成注册

输入邮箱，作为登录账号

输入账号密码

请输入手机号码

输入手机验证码

☒ 语音 ☐ 短信

获取验证码

请选择用户类型

-- 现居省份 --

-- 现居城市 --

下一步

(2) 创建存储空间 Bucket

点击左侧菜单 **对象存储**，一开始我们需要新建一个存储空间来存放我们的图片资源。点击**新建存储空间**，设置一些需要的内容，然后在左侧的存储空间列表我们就可以看到新加的空间了。

新建存储空间

搜索存储空间

存储空间列表

暂无存储空间

你可以点击下方按钮或  
者上方新建按钮创建第  
一个存储空间

立即添加

存储空间名称

存储空间名称作为唯一的 Bucket 识别符，遇到冲突请更换名称。  
名称由 4 ~ 63 个字符组成，可包含 字母、数字、中划线。

存储区域

北美区域尚未支持自定义数据处理服务，一旦创建区域无法修改，请谨慎选择。

华东

华北

华南

北美

东南亚

访问控制

公开和私有仅对 Bucket 的读文件生效，修改、删除、写入等对 Bucket 的操作均需要拥有者的授权才能进行操作。

公开空间

私有空间

账号注册有些需要注意的点如下：

- 注册账号之后需要实名认证（个人/企业）
- 实名认证之后才可以创建存储空间
- 存储空间创建成功之后，找到[个人中心](#)获取accessKey，secretKey和存储空间名称就可以进行上传操作了

## 1.3 入门案例

七牛对象存储将数据文件以资源的形式上传到空间中。可以创建一个或者多个空间，然后向每个空间中上传一个或多个文件。通过获取已上传文件的地址进行文件的分享和下载

### 1.3.1 搭建环境

```
<dependency>
  <groupId>com.qiniu</groupId>
  <artifactId>qiniu-java-sdk</artifactId>
  <version>[7.2.0, 7.2.99]</version>
</dependency>
```

### 1.3.2 文件上传

```
@Test
public void testUploadImage() {
    Configuration cfg = new Configuration(Zone.zone0());
    UploadManager uploadManager = new UploadManager(cfg);
    String accessKey = "COuODRVa7JLSuurzIvQSI_pEDceHDw3yGfJEmvwv";
    String secretKey = "3RWpTjB5Jxg3QosUFR4mxbHXJ5JR2m6AHQqYsSlr";
    String bucket = "test-bucket";
    String localFilePath = "C:\\Users\\ThinkPad\\Desktop\\ihrm\\day9\\资源\\照片\\001.png";
    //默认不指定key的情况下，以文件内容的hash值作为文件名
    String key = "test";
    Auth auth = Auth.create(accessKey, secretKey);
    String upToken = auth.uploadToken(bucket);
    try {
        Response response = uploadManager.put(localFilePath, key, upToken);
        //解析上传成功的结果
```





```
DefaultPutRet putRet = new Gson().fromJson(response.bodyString(),
DefaultPutRet.class);
System.out.println(response.bodyString());
} catch (QiniuException ex) {
    Response r = ex.response;
    System.err.println(r.toString());
    try {
        System.err.println(r.bodyString());
    } catch (QiniuException ex2) {
        //ignore
    }
}
}
```

### 1.3.3 断点续传

```
@Test
public void testUploadImage1() {
    Configuration cfg = new Configuration(Zone.zone0());
    String accessKey = "COuoDRVa7JLsuurzIvQSI_pEDceHDw3yGfJEmvwv";
    String secretKey = "3RwpTjB5Jxg3QosUFR4mxbHXJ5JR2m6AHQqYsSlr";
    String bucket = "test-bucket";
    String key = "test";
    Auth auth = Auth.create(accessKey, secretKey);
    String upToken = auth.uploadToken(bucket);

    String localFilePath = "C:\\Users\\ThinkPad\\Desktop\\ihrm\\day9\\资源\\test.xlsx";

    String localTempDir = Paths.get(System.getProperty("java.io.tmpdir"),
bucket).toString();
    System.out.println(localTempDir);
    try {
        //设置断点续传文件进度保存目录
        FileRecorder fileRecorder = new FileRecorder(localTempDir);
        UploadManager uploadManager = new UploadManager(cfg, fileRecorder);
        try {
            Response response = uploadManager.put(localFilePath, key, upToken);
            //解析上传成功的结果
            DefaultPutRet putRet = new Gson().fromJson(response.bodyString(),
DefaultPutRet.class);
            System.out.println(putRet.key);
            System.out.println(putRet.hash);
        } catch (QiniuException ex) {
            Response r = ex.response;
            System.err.println(r.toString());
            try {
                System.err.println(r.bodyString());
            } catch (QiniuException ex2) {
                //ignore
            }
        }
    }
}
```

```
    } catch (IOException ex) {  
        ex.printStackTrace();  
    }  
}
```

## 1.4 文件下载

对于公开空间，其访问的链接主要是将空间绑定的域名（可以是七牛空间的默认域名或者是绑定的自定义域名）拼接上空间里面的文件名即可访问，标准情况下需要在拼接链接之前，将文件名进行 `urlencode` 以兼容不同的字符。

## 1.5 七牛云实现用户头像上传

### （1）创建文件上传的工具类

```
public class QiniuUploadUtil {  
  
    private static final String accessKey = "COuoDRVa7JLsuurzIvQSI_pEDceHDw3yGfJEmvwv";  
    private static final String secretKey = "3RWpTjB5Jxg3QosUFR4mxbHXJ5JR2m6AHQqYsSlr";  
    private static final String bucket = "test-bucket";  
    private static final String prix = "http://pk9vj7em6.bkt.clouddn.com/";  
    private UploadManager manager;  
  
    public QiniuUploadUtil() {  
        //初始化基本配置  
        Configuration cfg = new Configuration(Zone.zone0());  
        //创建上传管理器  
        manager = new UploadManager(cfg);  
    }  
  
    public String upload(String imgName, byte[] bytes) {  
        Auth auth = Auth.create(accessKey, secretKey);  
        //构造覆盖上传token  
        String upToken = auth.uploadToken(bucket, imgName);  
        try {  
            Response response = manager.put(bytes, imgName, upToken);  
            DefaultPutRet putRet = new Gson().fromJson(response.bodyString(),  
DefaultPutRet.class);  
            //返回请求地址  
            return prix+putRet.key+"?t="+new Date().getTime();  
        } catch (Exception ex) {  
            ex.printStackTrace();  
        }  
        return null;  
    }  
}
```

### （2）使用七牛云实现用户头像上传

修改UserService方法

```
public String uploadImage(String id, MultipartFile file) throws Exception {  
    User user = userDao.findById(id).get();  
    String key = new QiniuUploadUtil().upload(user.getId(), file.getBytes());  
    if(key != null) {  
        user.setStaffPhoto(key);  
        userDao.save(user);  
    }  
    return key;  
}
```

## 3 PDF报表打印概述

### 3.1 概述

在企业级应用开发中，报表生成、报表打印下载是其重要的一个环节。在之前的课程中我们已经学习了报表中比较重要的一种：Excel报表。其实除了Excel报表之外，PDF报表也有广泛的应用场景，必须用户详细资料，用户简历等。接下来的课程，我们就来共同学习PDF报表

### 3.2 常见PDF报表的制作方式


目前市面上比较流行的制作PDF报表的工具如下：

1. iText PDF：iText是著名的开放项目，是用于生成PDF文档的一个java类库。通过iText不仅可以生成PDF或rtf的文档，而且可以将XML、Html文件转化为PDF文件。
2. Openoffice：openoffice是开源软件且能在windows和linux平台下运行，可以灵活的将word或者Excel转化为PDF文档。
3. Jasper Report：是一个强大、灵活的报表生成工具，能够展示丰富的页面内容，并将之转换成PDF

### 3.3 JasperReport框架的介绍

[Answers](#) | [Exchange](#) | [Docs](#) | [Wiki](#) | [Planet](#) | [Tracker](#)

Home » Exchange » JasperReports® Library



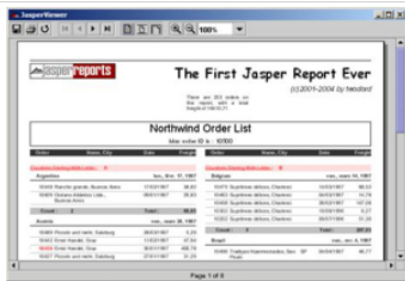
## JasperReports® Library

Open Source Java Reporting Library

[View](#) | [Resources](#) | [Answers](#) | [Tracker](#) | [Reviews](#) | [Releases](#)

★★★★★  
(2 reviews)

The JasperReports Library is the world's most popular open source reporting engine. It is entirely written in Java and it is able to use data coming from any kind of data source and produce pixel-perfect documents that can be viewed, printed or exported in a variety of document formats including HTML, PDF, Excel, OpenOffice and Word.



Quick Start

**What is JasperReports?**  
An introduction to the JasperReports Library and the capabilities it can bring to your Java applications.

**Deploy the JasperReports Library**  
Download and deploy the JasperReports Library within your Java Application

**Deploying your first Report**  
Preparing and deploying your first report within your application.  
Coming Soon

**Report Execution and Exporting**  
Execute your report and export it into your desired format.  
Coming Soon

[Download JasperReports® Library](#)

[View Screenshots \(1\)](#)  
[Browse Source Code](#)

**Provider:** Jaspersoft Corporation

**Category:**

**License:** LGPL

**Issues for JasperReports® Library:**

All Issues: 402 open, 1787 total

Bug Reports: 263 open, 1247 total

Inappropriate Content?

Tags

JasperReports Library 6.2.0

Tag It

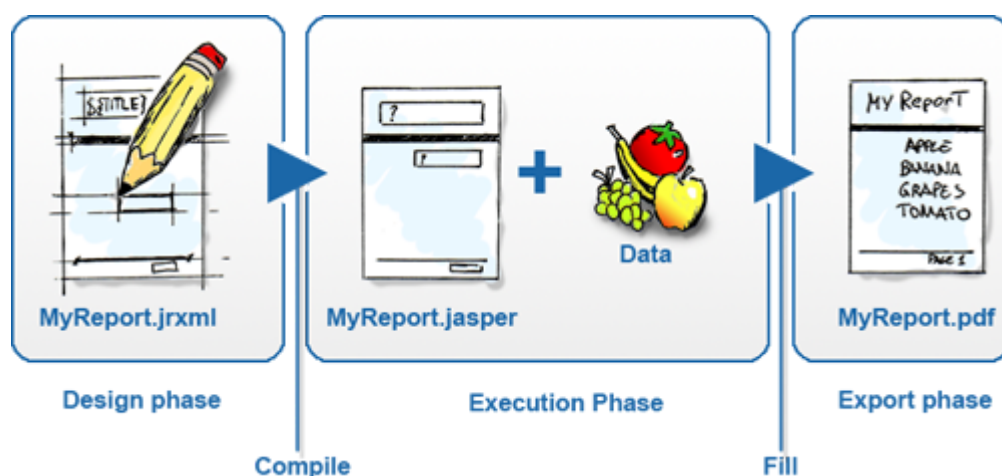
JasperReport是一个强大、灵活的报表生成工具，能够展示丰富的页面内容，并将之转换成PDF，HTML，或者XML格式。该库完全由Java写成，可以用于在各种Java应用程序，包括J2EE，Web应用程序中生成动态内容。只需要将JasperReport引入工程中即可完成PDF报表的编译、显示、输出等工作。

- 在开源的JAVA报表工具中，JASPER Report发展是比较好的，比一些商业的报表引擎做得还好，如支持了交叉报表、统计报表、图形报表，支持多种报表格式的输出，如PDF、RTF、XML、CSV、XHTML、TEXT、DOCX以及OpenOffice。
- 数据源支持更多，常用JDBC SQL查询、XML文件、CSV文件、HQL（Hibernate查询），HBase，JAVA集合等。还允许你定义自己的数据源，通过JASPER文件及数据源，JASPER就能生成最终用户想要的文档格式。

## 4 JasperReport的开发步骤

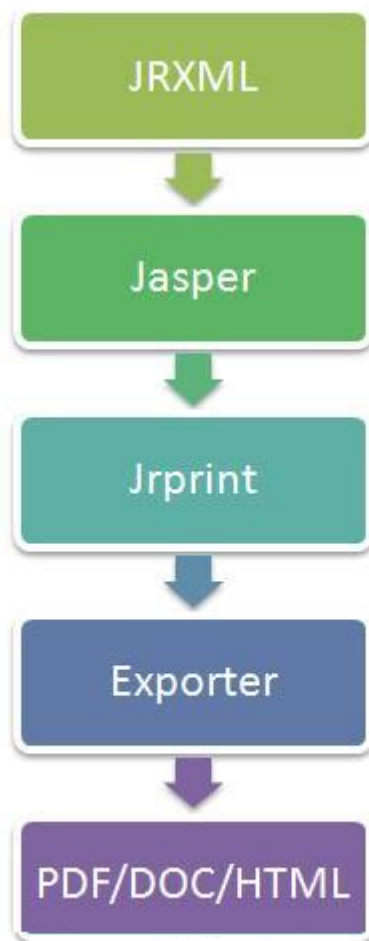
### 4.1 JasperReport生命周期

通常我们提到PDF报表的时候,浮现在脑海中的是最终的PDF文档文件。在JasperReports中，这只是报表生命周期的最后阶段。通过JasperReports生成PDF报表一共要经过三个阶段，我们称之为 JasperReport的生命周期，这三个阶段为：设计（Design）阶段、执行（Execution）阶段以及输出（Export）阶段，如下图所示：



1. 设计阶段（Design）：所谓的报表设计就是创建一些模板，模板包含了报表的布局与设计，包括执行计算的复杂公式、可选的从数据源获取数据的查询语句、以及其它的一些信息。模板设计完成之后，我们将模板保存为JRXML文件（JR代表JasperReports），其实就是一个XML文件。
2. 执行阶段（Execution）：使用以JRXML文件编译为可执行的二进制文件（即Jasper文件）结合数据进行执行，填充报表数据
3. 输出阶段（Export）：数据填充结束，可以指定输出为多种形式的报表

### 4.2 JasperReport原理简述



1. JRXML:报表填充模板，本质是一个XML。

JasperReport已经封装了一个dtd，只要按照规定的格式写这个xml文件，那么jasperReport就可以将其解析最终生成报表，但是jasperReport所解析的不是我们常见的.xml文件，而是.jrxml文件，其实跟xml是一样的，只是后缀不一样。

2. Jasper:由JRXML模板编译生成的二进制文件，用于代码填充数据。

解析完成后JasperReport就开始编译.jrxml文件，将其编译成.jasper文件，因为JasperReport只可以对.jasper文件进行填充数据和转换，这步操作就跟我们java中将java文件编译成class文件是一样的

3. Jrprint:当用数据填充完Jasper后生成的文件，用于输出报表。

这一步才是JasperReport的核心所在，它会根据你在xml里面写好的查询语句来查询指定是数据库，也可以控制在后台编写查询语句，参数，数据库。在报表填充完后，会再生成一个.jrprint格式的文件（读取jasper文件进行填充，然后生成一个jrprint文件）

4. Exporter:决定要输出的报表为何种格式，报表输出的管理类。

5. Jasperreport可以输出多种格式的报表文件，常见的有Html,PDF,xls等

## 4.3 开发流程概述

- 制作报表模板
- 模板编译
- 构造数据

- 填充模板数据

## 5 模板工具Jaspersoft Studio

### 5.1 概述

Jaspersoft Studio是JasperReports库和JasperReports服务器的基于Eclipse的报告设计器; 它可以作为Eclipse插件或作为独立的应用程序使用。Jaspersoft Studio允许您创建包含图表，图像，子报表，交叉表等的复杂布局。您可以通过JDBC，TableModels，JavaBeans，XML，Hibernate，大数据（如Hive），CSV，XML / AI以及自定义来源等各种来源访问数据，然后将报告发布为PDF，RTF，XML，XLS，CSV，HTML，XHTML，文本，DOCX或OpenOffice。

Jaspersoft Studio 是一个可视化的报表设计工具,使用该软件可以方便地对报表进行可视化的设计，设计结果为格式.jrxml 的 XML 文件，并且可以把.jrxml 文件编译成.jasper 格式文件方便 JasperReport 报表引擎解析、显示。

### 5.2 安装配置

到JasperReport官网下载 <https://community.jaspersoft.com/community-download>

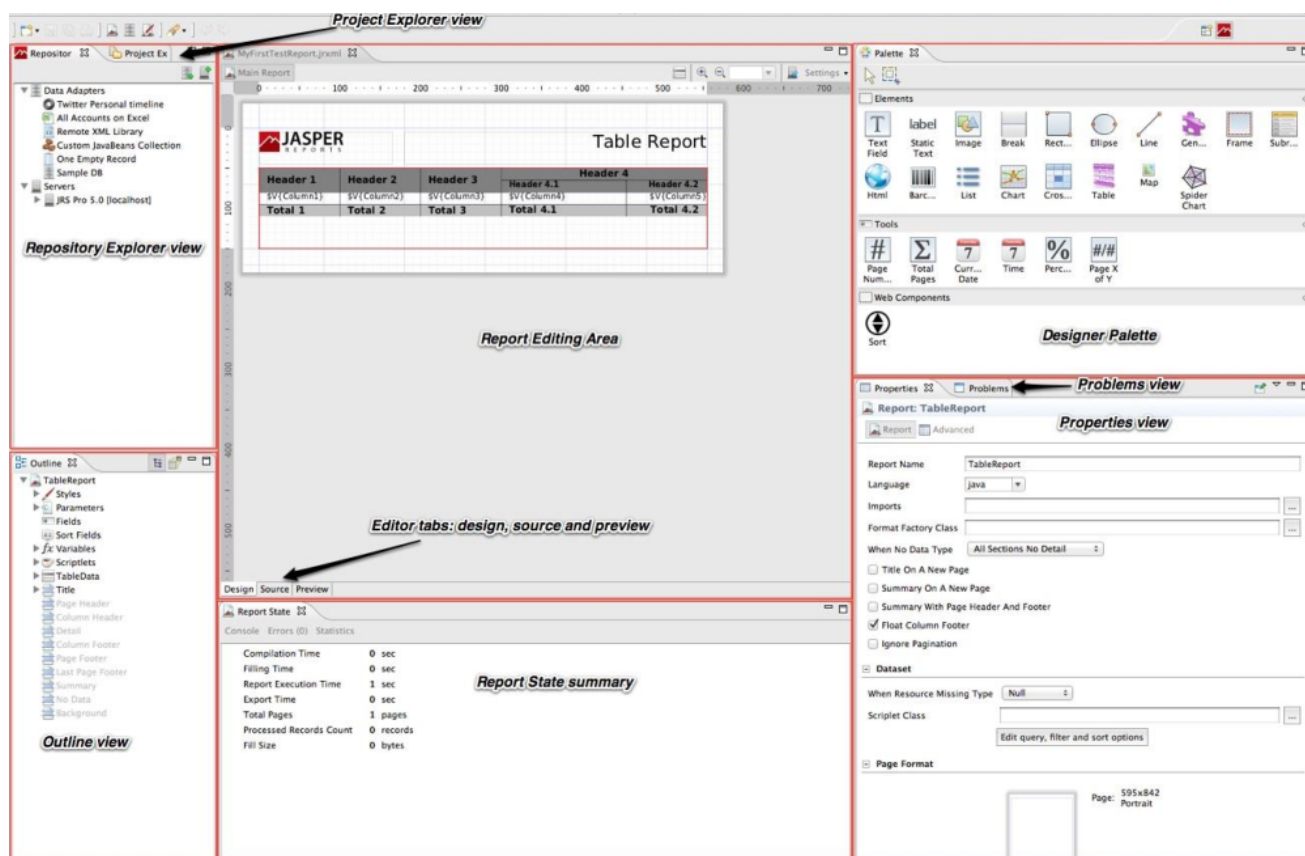
#### Community Editions

	JasperReports® Server CE	v7.1.0 5 May 2018	<a href="#">Download</a>
	JasperReports® Library CE	v6.7.0 9 Aug 2018	<a href="#">Download</a>
	Jaspersoft® Studio CE	v6.6.0 30 May 2018	<a href="#">Download</a>
	iReport Designer CE	v5.6.0 28 May 2014	<a href="#">Download</a>
	Jaspersoft® ETL CE	v5.6.2 14 Jun 2015	<a href="#">Download</a>

下载 Library Jar包（传统导入jar包工程需下载）和模板设计器Jaspersoft studio。并安装Jaspersoft studio，安装的过程比较简单，一直下一步直至安装成功即可。



## 5.3 面板介绍

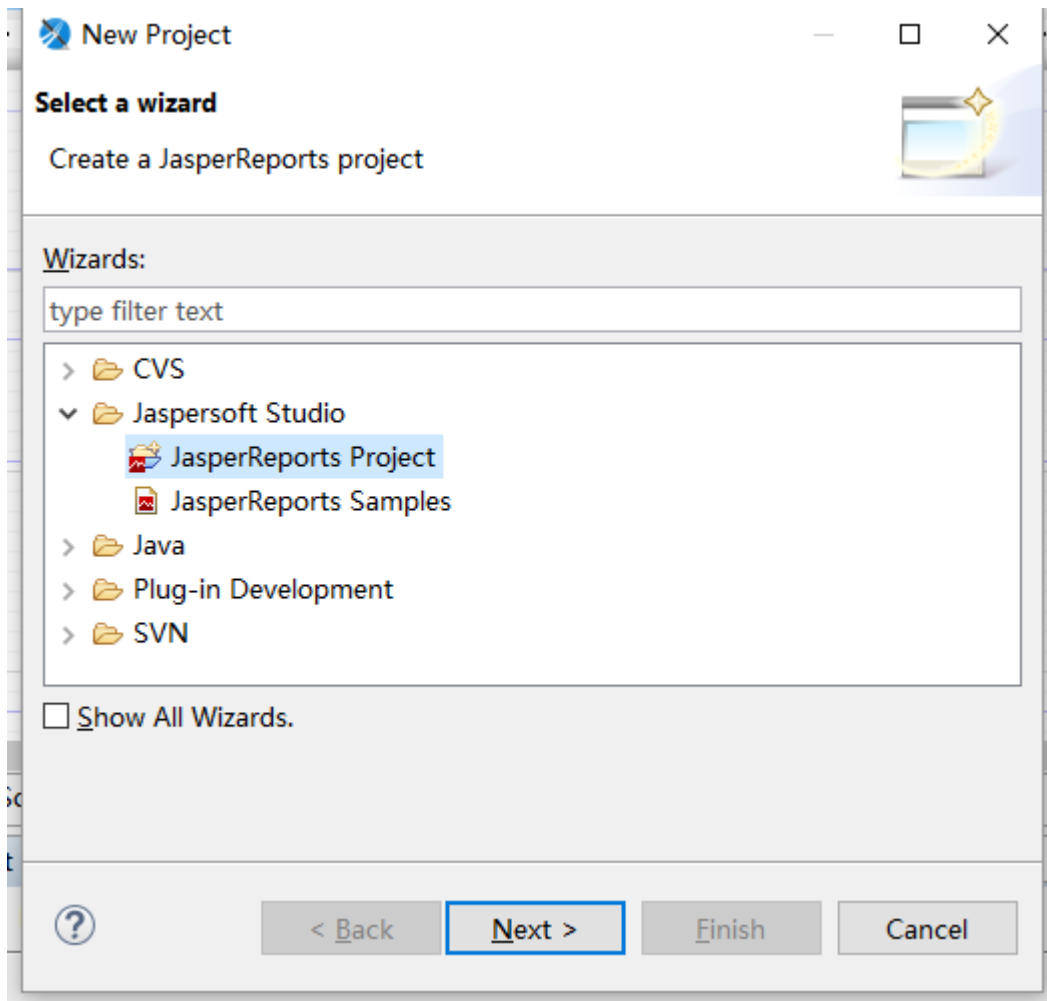


- Report editing area（主编辑区域）中，您直观地通过拖动，定位，对齐和通过 Designer palette（设计器调色板）对报表元素调整大小。JasperSoft Studio 有一个多标签编辑器，Design,Source 和 Preview：
  - Design tab：当你打开一个报告文件，它允许您以图形方式创建报表选中
  - Source tab：包含用于报表的 JRXML 源代码。
  - Preview tab：允许在选择数据源和输出格式后，运行报表预览。
- Repository Explorer view：包含 JasperServer 生成的连接和可用的数据适配器列表
- Project Explorer view：包含 JasperReports 的工程项目清单
- Outline view：在大纲视图中显示了一个树的形式的方式报告的完整结构。
- Properties view：通常是任何基于 Eclipse 的产品/插件的基础之一。它通常被填充与实际所选元素的属性的信息。这就是这样，当你从主设计区域（即：一个文本字段）选择一个报表元素或从大纲，视图显示了它的信息。其中一些属性可以是只读的，但大部分都是可编辑的，对其进行修改，通常会通知更改绘制的元素（如：元素的宽度或高度）。
- Problems view：显示的问题和错误，例如可以阻断报告的正确编译。
- Report state summary 提供了有关在报表编译/填充/执行统计用户有用的信息。错误会显示在这里

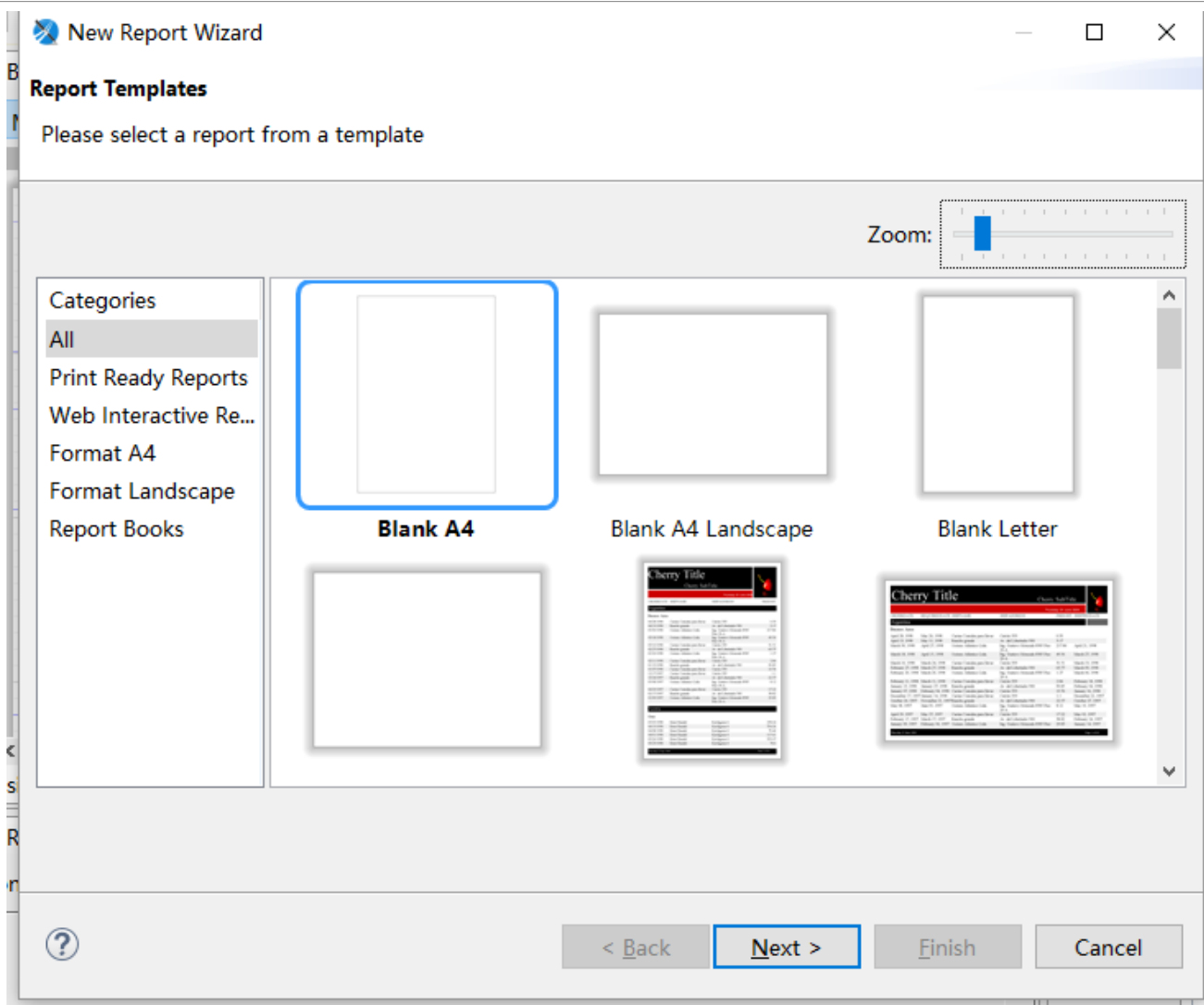
## 5.4 基本使用

### 5.4.1 模板制作

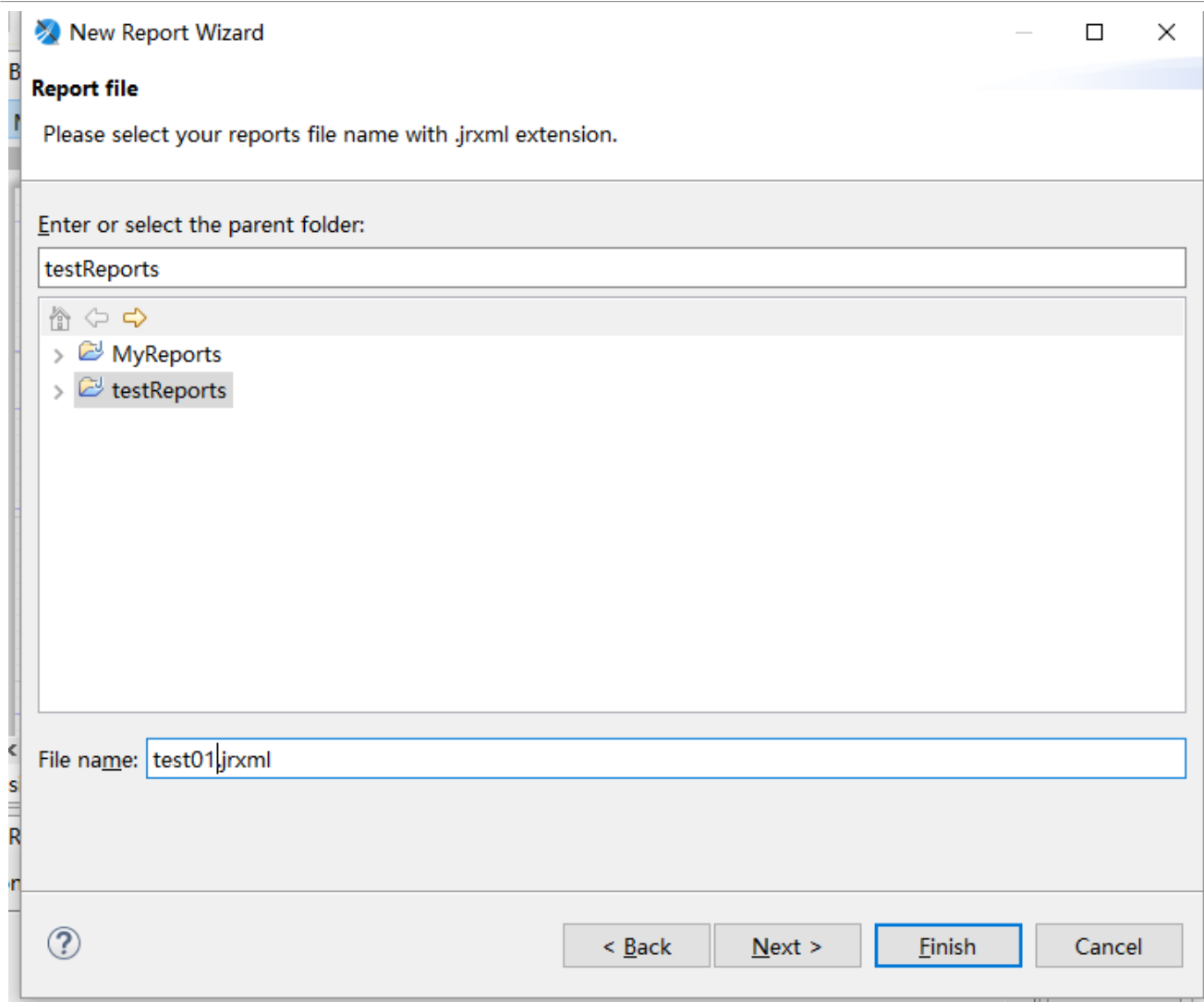
(1) 打开Jaspersoft Studio，新建一个project, 步骤：File -> New -> Project-> JasperReports Project



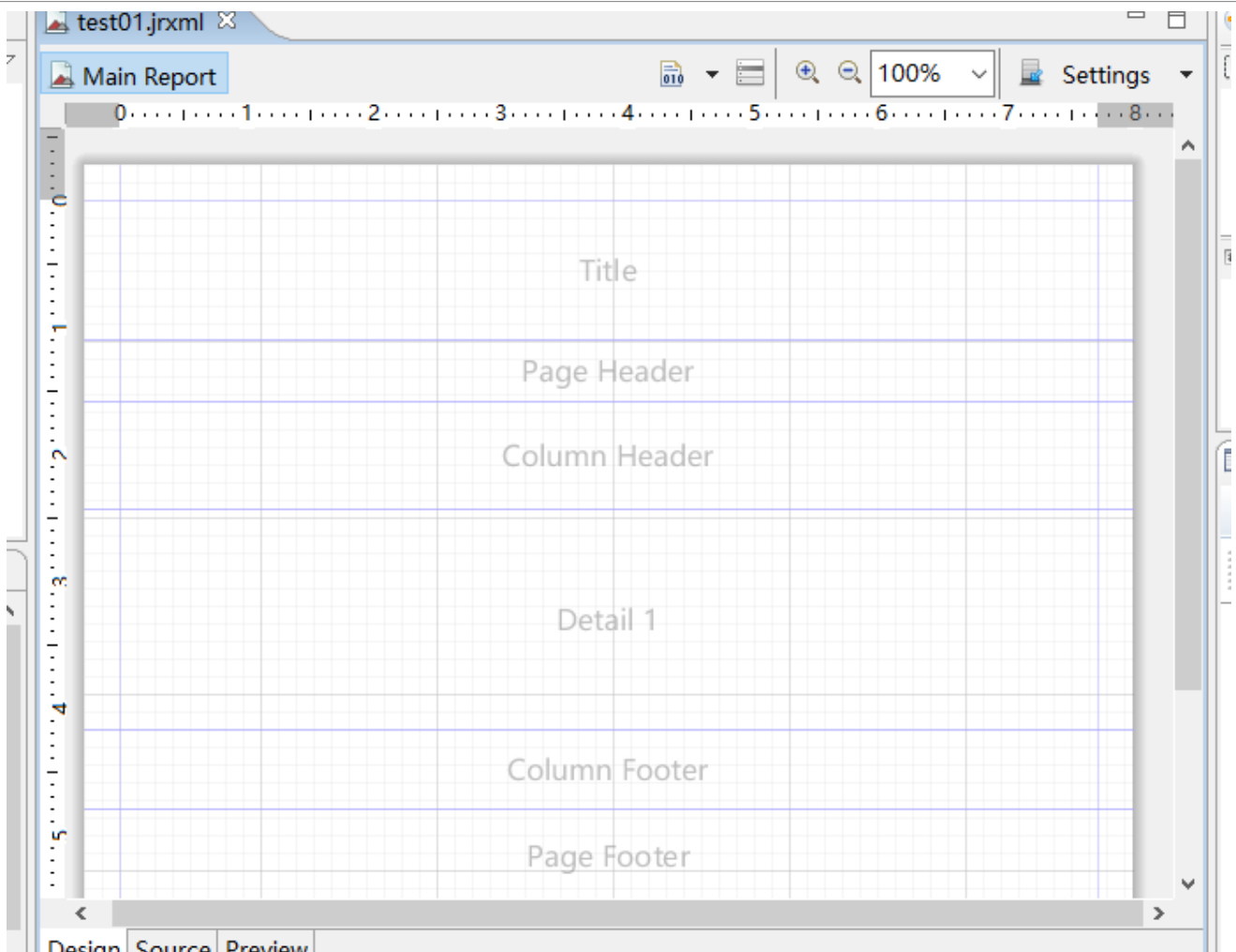
(2) 新建一个Jasper Report模板，在 Studio的左下方Project Explorer 找到刚才新建的Project (我这里新建的是 DemoReport),步骤：项目右键 -> New -> Jasper Report



( 3 ) 选择 Blank A4 (A4纸大小的模板) , 然后 Next 命名为DemoReport1.jrxml.



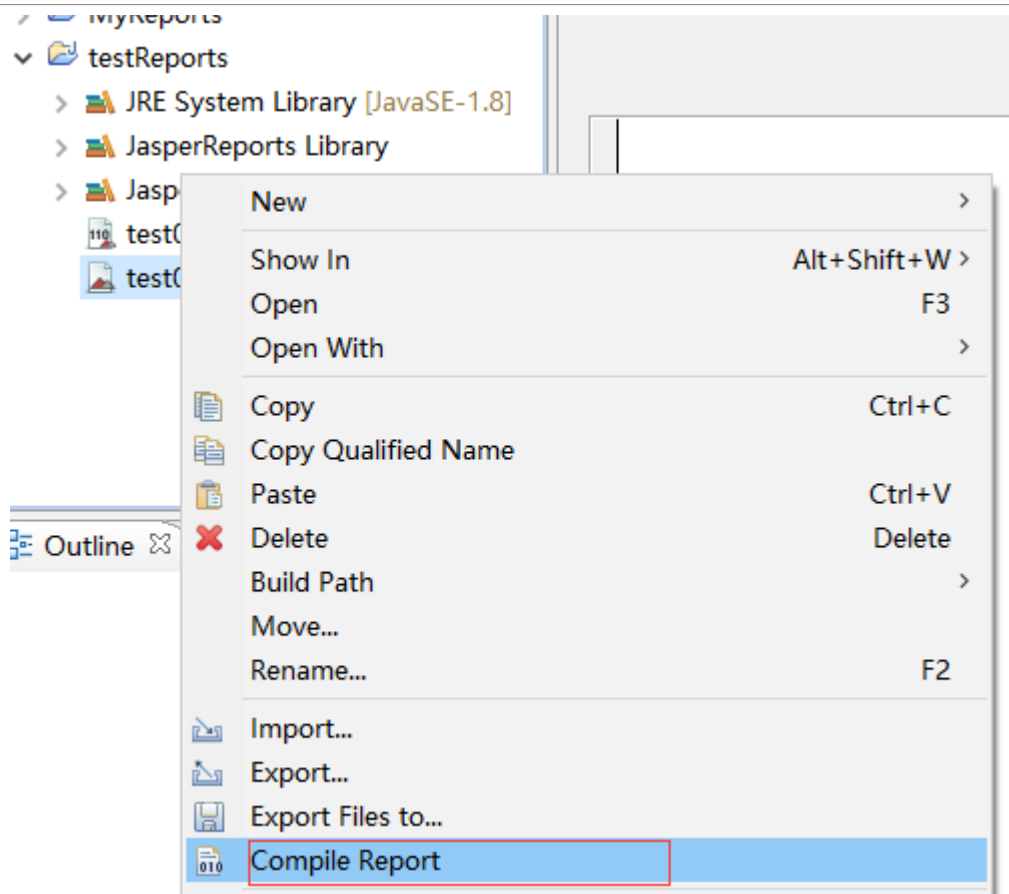
如图所示，报表模板被垂直的分层，每一个部分都是一个Band,每一个Band的特点不同：



- Title(标题)：只在整个报表的第一页的最上端显示。只在第一页显示，其他页面均不显示。
- Page Header(页头)：在整个报表中每一页都会显示。在第一页中，出现的位置在 Title Band的下面。在除了第一页的其他页面中Page Header 的内容均在页面的最上端显示。
- Page Footer(页脚)：在整个报表中每一页都会显示。显示在页面的最下端。一般用来显示页码。
- Detail 1(详细)：报表内容，每一页都会显示。
- Column Header(列头)：Detail中打印的是一张表的话，这Column Header就是表中列的列头。
- Column Footer(列脚)：Detail中打印的是一张表的话，这Column Footer就是表中列的列脚。
- Summary(统计)：表格的合计段，出现在整个报表的最后一页中，在Detail 1 Band后面。主要是用来做报表的合计显示。

## 5.4.2 编译模板

右键单机模板文件 -> compile Report 对模板进行编译，生成.jasper文件



### 5.4.3 整合工程

#### (1) 新建SpringBoot工程引入坐标

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.5.RELEASE</version>
  <relativePath/>
</parent>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>net.sf.jasperreports</groupId>
    <artifactId>jasperreports</artifactId>
    <version>6.5.0</version>
  </dependency>
  <dependency>
    <groupId>org.olap4j</groupId>
```





```
<artifactId>olap4j</artifactId>
<version>1.2.0</version>
</dependency>
<dependency>
<groupId>com.lowagie</groupId>
<artifactId>itext</artifactId>
<version>2.1.7</version>
</dependency>
<dependency>
<groupId>org.apache.poi</groupId>
<artifactId>poi</artifactId>
<version>4.0.1</version>
</dependency>
<dependency>
<groupId>org.apache.poi</groupId>
<artifactId>poi-ooxml</artifactId>
<version>4.0.1</version>
</dependency>
<dependency>
<groupId>org.apache.poi</groupId>
<artifactId>poi-ooxml-schemas</artifactId>
<version>4.0.1</version>
</dependency>
</dependencies>
```

## (2) 引入配置文件

```
server:
  port: 8181
spring:
  application:
    name: jasper-demo #指定服务名
  resources:
    static-locations: classpath:/templates/
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/ihrm?useUnicode=true&characterEncoding=utf8
    username: root
    password: 111111
```

## (3) 创建启动类

```
@SpringBootApplication(scanBasePackages = "cn.itcast")
public class JasperApplication {
    public static void main(String[] args) {
        SpringApplication.run(JasperApplication.class, args);
    }
}
```

## (4) 导入生成的.jasper文件

## (5) 创建测试controller

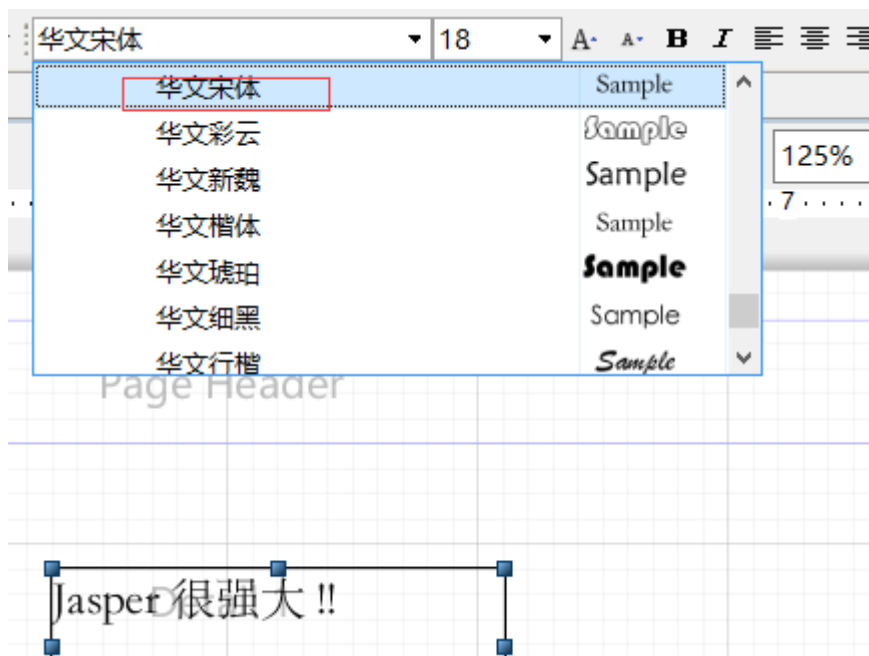


```
@RestController
public class JasperController {

    @GetMapping("/testJasper")
    public void createHtml(HttpServletResponse response, HttpServletRequest
request)throws Exception{
        //引入jasper文件。由JRXML模板编译生成的二进制文件，用于代码填充数据
        Resource resource = new ClassPathResource("templates/test01.jasper");
        //加载jasper文件创建inputStream
        FileInputStream isRef = new FileInputStream(resource.getFile());
        ServletOutputStream sosRef = response.getOutputStream();
        try {
            //创建JasperPrint对象
            JasperPrint jasperPrint = JasperFillManager.fillReport(isRef, new HashMap<>
(),new JREmptyDataSource());
            //写入pdf数据
            JasperExportManager.exportReportToPdfStream(jasperPrint,sosRef);
        } finally {
            sosRef.flush();
            sosRef.close();
        }
    }
}
```

## 5.4.4 中文处理

(1) 设计阶段需要指定中文样式



(2) 通过手动指定中文字体的形式解决中文不现实

- 添加properties文件：



```
net.sf.jasperreports.extension.registry.factory.simple.font.families=net.sf.jasperreports.engine.fonts.SimpleFontExtensionsRegistryFactory
net.sf.jasperreports.extension.simple.font.families.lobstertwo=stsong/fonts.xml
```

- 指定中文配置文件fonts.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<fontFamilies>

    <!--<fontFamily name="Lobster Two">-->
    <!--<normal>lobstertwo/LobsterTwo-Regular.otf</normal>-->
    <!--<bold>lobstertwo/LobsterTwo-Bold.otf</bold>-->
    <!--<italic>lobstertwo/LobsterTwo-Italic.otf</italic>-->
    <!--<boldItalic>lobstertwo/LobsterTwo-BoldItalic.otf</boldItalic>-->
    <!--<pdfEncoding>Identity-H</pdfEncoding>-->
    <!--<pdfEmbedded>true</pdfEmbedded>-->
    <!--<!-->
    <!--<exportFonts>-->
    <!--<export key="net.sf.jasperreports.html">'Lobster Two', 'Times New Roman',
Times, serif</export>-->
    <!--</exportFonts>-->
    <!-->-->
    <!--</fontFamily>-->
    <fontFamily name="华文宋体">
        <normal>stsong/stsong.TTF</normal>
        <bold>stsong/stsong.TTF</bold>
        <italic>stsong/stsong.TTF</italic>
        <boldItalic>stsong/stsong.TTF</boldItalic>
        <pdfEncoding>Identity-H</pdfEncoding>
        <pdfEmbedded>true</pdfEmbedded>
        <exportFonts>
            <export key="net.sf.jasperreports.html">'华文宋体', Arial, Helvetica, sans-
serif</export>
            <export key="net.sf.jasperreports.xhtml">'华文宋体', Arial, Helvetica, sans-
serif</export>
        </exportFonts>
        <!--
        <locales>
            <locale>en_US</locale>
            <locale>de_DE</locale>
        </locales>
        -->
    </fontFamily>
</fontFamilies>
```

- 引入字体库stsong.TTF