xml文件解析是软件设计中常见的功能需求，xml文件可以满足我们设计需要的数据格式，结构，同时xml文件解析也是一件复杂繁琐的事情，我们的项目中可能存在很多xml文件，但是我们希望把他们尽量设计在一起，或者有文件的关联规则，系统的xml文件输入尽量只有一个，或者几个，同时xml文件中涉及到程序与外界交互需要的大部分信息，但是这些信息的关联性往往不大，而且容易发生不同的变更。这时解析xml文件就不是一个类能完成得了，需要多个类进行协作，共同完成xml文件的读取。实现软件最大程度的可维护，可复用。易于理解。

这里我仅仅基于w3c-dom式的解析方式，设计一下xml文件解析的类库设计。

解析xml文件首先基于w3c-dom解析成document,也就是下面这段代码，但是每次解析都需要执行这行代码，但是执行时间长，效率低，我们要做到的就是复用document，复用解析xml文件的代码

DocumentBuilderFactory dfactory = DocumentBuilderFactory.*newInstance*();

Document document;

DocumentBuilder builder;

**try** {

builder = dfactory.newDocumentBuilder();

String classPath = Thread.**class**.getResource("/").getPath();

/\*\*

\* URLDecoder.decode(classPath,"UTF-8")对中文路径进行转码

\*/

document = builder.parse(**new** File(URLDecoder.*decode*(classPath,"UTF-8")+xmlPath));

XMLConfigResource resource = **new** XMLConfigResource();

resource.setDocument(document);

**this**.resource = resource;

} **catch** (ParserConfigurationException e){

**throw** **new** ResourceConfigException(ExcpMsg.*CANNOT\_CONFIG\_FILES*+xmlPath);

}

**catch** (SAXException e) {

**throw** **new** ResourceConfigException(ExcpMsg.*CANNOT\_CONFIG\_FILES*+xmlPath);

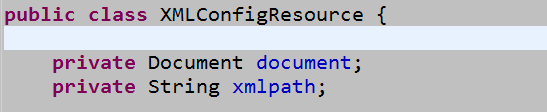
}**catch** (IOException e) {

**throw** **new** ResourceConfigException(ExcpMsg.*CANNOT\_CONFIG\_FILES*+xmlPath);

}

为此，我们设计了一个接口 XMLConfig

这个接口的唯一方法是返回xmlConfigResource类型对象，而XMLConfigResource类型封装了document类型,和文件所在位置。



xmlconfig只能规定，实现自己接口的一定要返回一个解析后的资源，这个资源中包含了解析后的Document。其他解析细节，xmlConfig无法规定。

同时针对解析xml的重复代码，可以设计抽象类实现xmlconfig接口，给具体的xml解析类提供复用。

