金格中间件边看边学系列

金格 PDF 应用方案

一、 前言

目前的信息化系统的应用里，对文档管理方面的功能要求渐渐丰富起来。不只是需要对文档的流转、打印进行控制，更希望对文档的归档和浏览也能便利的管理和应用。越来越多的单位开始使用PDF文档作为单位正式文档的副本，因为PDF文档比较安全，能在保护文档内容时提供良好的阅读效果和打印等功能。

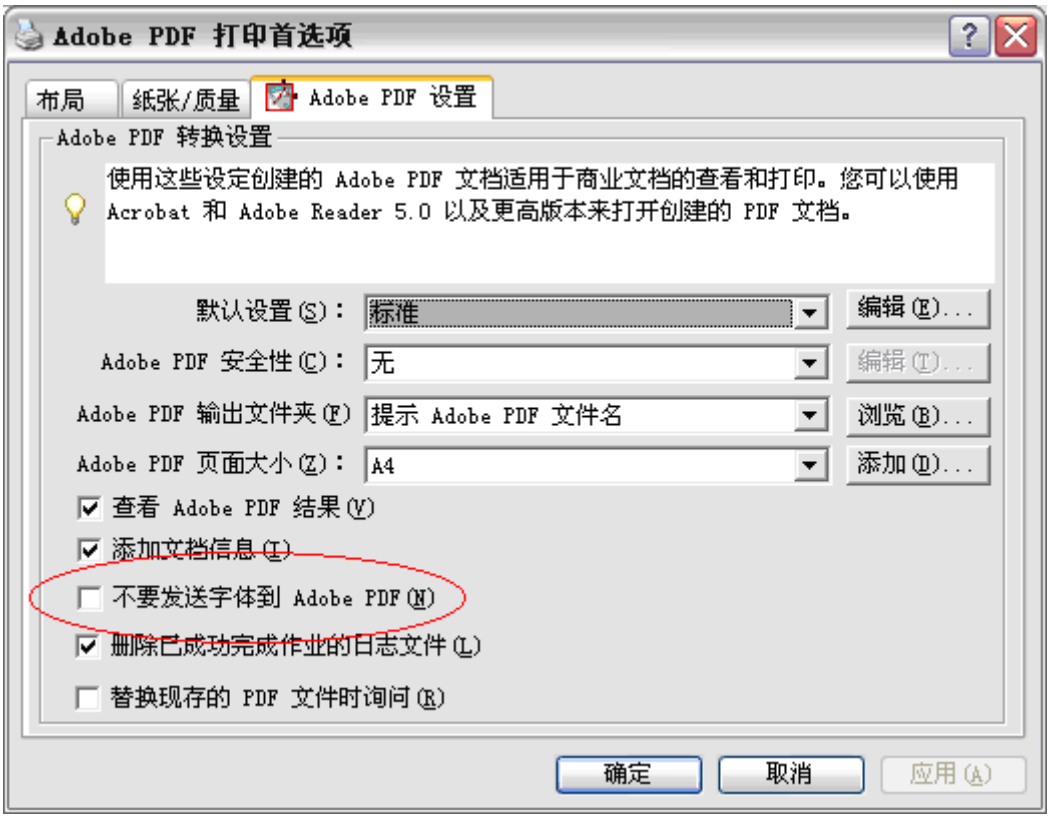
虽然在本地化应用是非常简单的，但是只有集成在信息化系统中才能统一和规范化的进行管理。现在，金格iWebOffice2006 V7.8版本中提供了转换PDF的功能，同时发布了能够在线浏览的iWebPDF中间件产品，这使得在信息系统中实现PDF文档的管理和控制成为了可能。是不是有些迫不及待想知道如何应用？现在让我们进入这篇文章的主题吧。

二、 PDF 应用

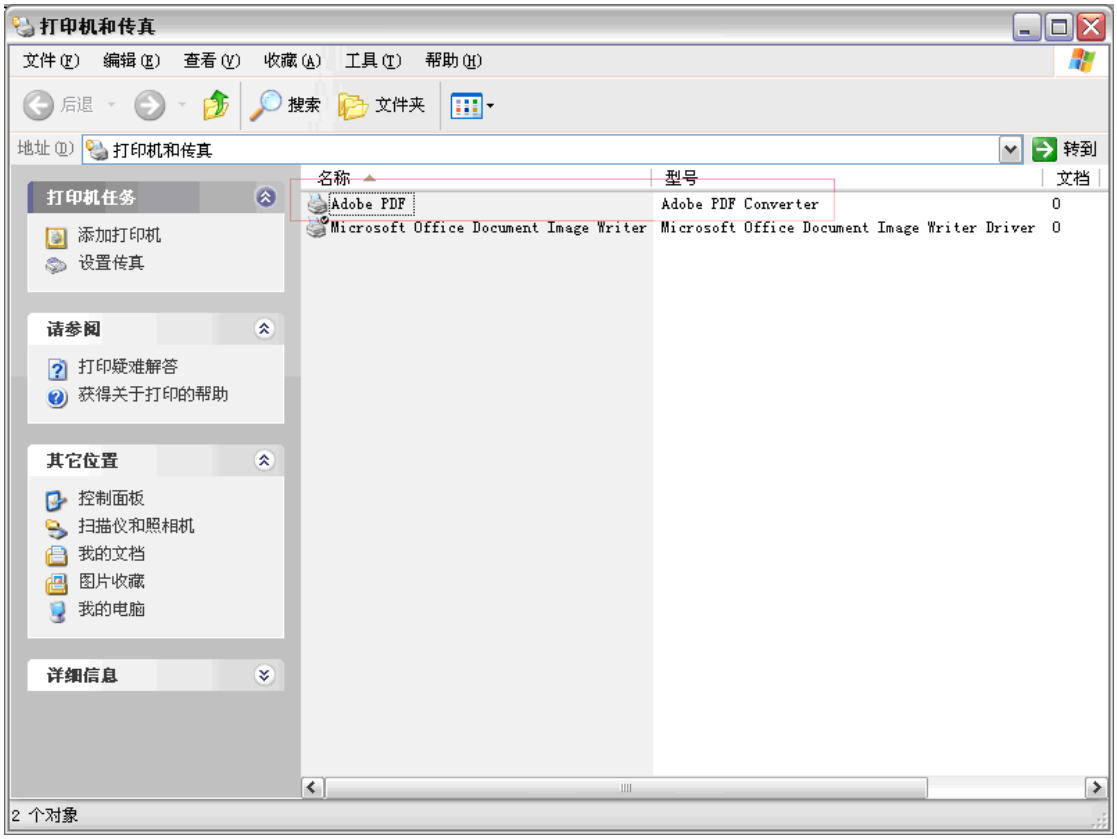
让我们假设这样一个应用：文档在系统中经过流转已经成文，现在需要将这个文档转换成PDF文档保存起来，同时提供在线浏览PDF的功能给系统的其他用户。理论上我们应用金格目前提供的iWebOffice2006中间件和iWebPDF中间件是可以实现上述功能的，现在我们看看具体如何实现。

2.1 iWebOffice2006 转换 PDF 文档

作为WORD文档流程的结束和PDF文档功能应用的开始，我们首先需要将WORD文档转换成PDF文档。这里需要用到金格iWebOffice2006v7.8版本中提供的一个新接口：WebSavePDF()，该接口将控件中的WORD文档转换成PDF文档并保存在服务器端。这个功能调用有一个条件，就是需要安装Adobe PDF虚拟打印机。我们翻开皮书白皮书，里面这样描述：必须首先在客户端安装Adobe PDF虚拟打印机（可在安装Adobe Acrobat PRO版本时自动安装），iWebOffice控件本身不提供该虚拟打印机。同时，请在PDF虚拟打印机的设置中将“不要发送字体到Adobe PDF”的勾去掉，如下图：



为了本次测试，我们安装了Adobe Acrobat 7.0 Professional软件（该软件为商业软件），该软件安装后会自动产生一个名称为“Adobe PDF”的打印机。



通过查阅技术白皮书我们了解到，WebSavePDF()接口调用后控件会自动将转换好的PDF

文档发送到服务器处理页面，对应的OPTION为“SAVEPDF”，其实是做了个文档保存的请求。好的，基于我们对iWebOffice2006控件的了解，我们可以开始动手了。我们在文档编辑页面增加一个转换的函数TestWebSavePDF()，里面调用WebSavePDF()接口，具体代码如下：

```
//-----编辑页面中调转换PDF文档的接口-----------------------------------------------//
function TestWebSavePDF(){
  try{
    if(webform.WebOffice.WebSavePDF()){
      alert("转换并保存成功");
    }
    else{
      alert("转换并保存失败");
    }
  }
  catch(e){
    alert(e.description);                    //出现异常时提示错误信息
  }
}
```

在服务器端的处理页中，我们加入对应的保存代码，为了方便测试，我们将文档保存在服务器的目录中，PDF文档名称按RECORDID+".pdf"的方式来起名。

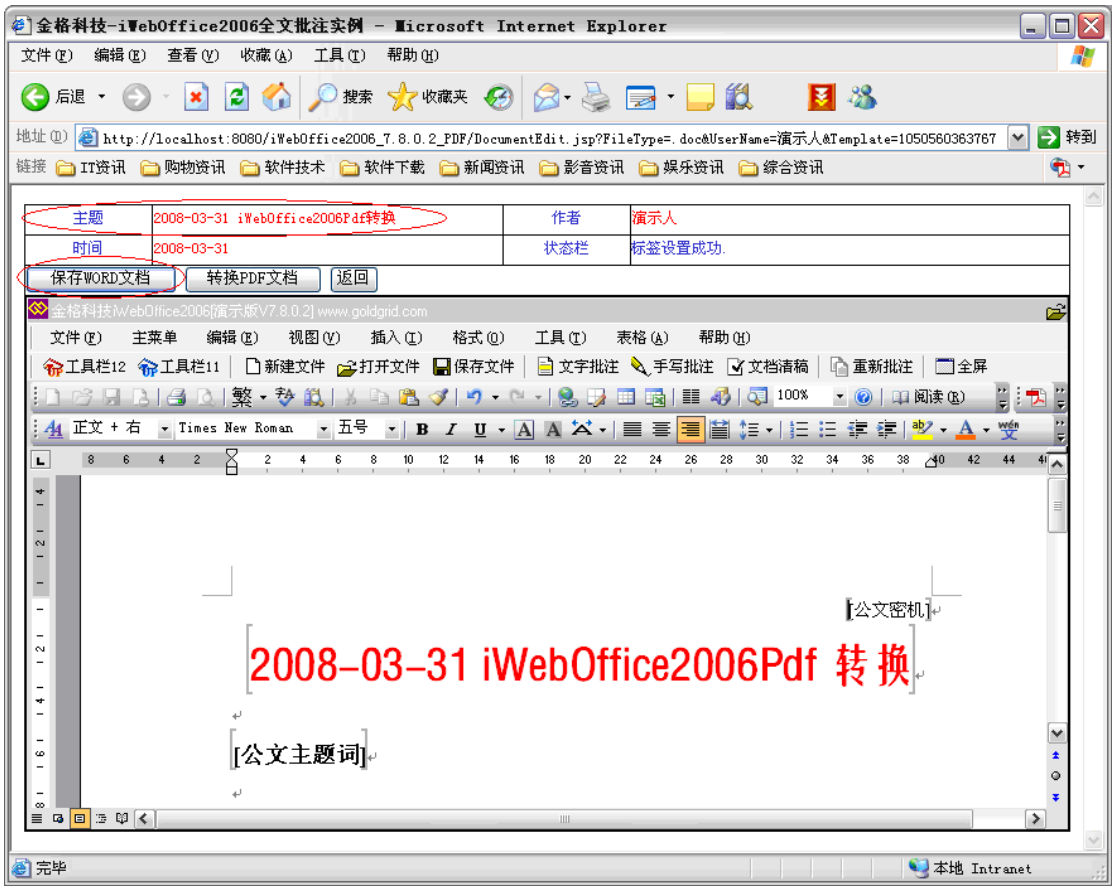
```
//-----服务器处理页保存PDF文档-----------------------------------------------------//
else if(mOption.equalsIgnoreCase("SAVEPDF")){  //下面的代码为保存 PDF 文件
  mRecordID=MsgObj.GetMsgByName("RECORDID");        //取得文档编号
  mFileName=MsgObj.GetMsgByName("FILENAME");        //取得文档名称
  MsgObj.MsgTextClear();                            //清除文本信息
  //保存文档内容到文件夹中
  if (MsgObj.MsgFileSave(mFilePath+"\\Document\\"+mRecordID+".pdf")){
    MsgObj.SetMsgByName("STATUS", "保存成功!");     //设置状态信息
    MsgObj.MsgError("");                            //清除错误信息
  }
  else {
    MsgObj.MsgError("保存失败!");                   //设置错误信息
  }
  MsgObj.MsgFileClear();                            //清除文档内容
}
```

下面我们图释一下应用的流程：

1、这是我们改造过的iWebOffice2006的示例，用的是JSP语言的版本。去掉了一些不必要的按钮，新增了一个“PDF浏览”的按钮，这里暂时未给这个按钮任何功能，该按钮的功能将在下一节详细说明。如：图1

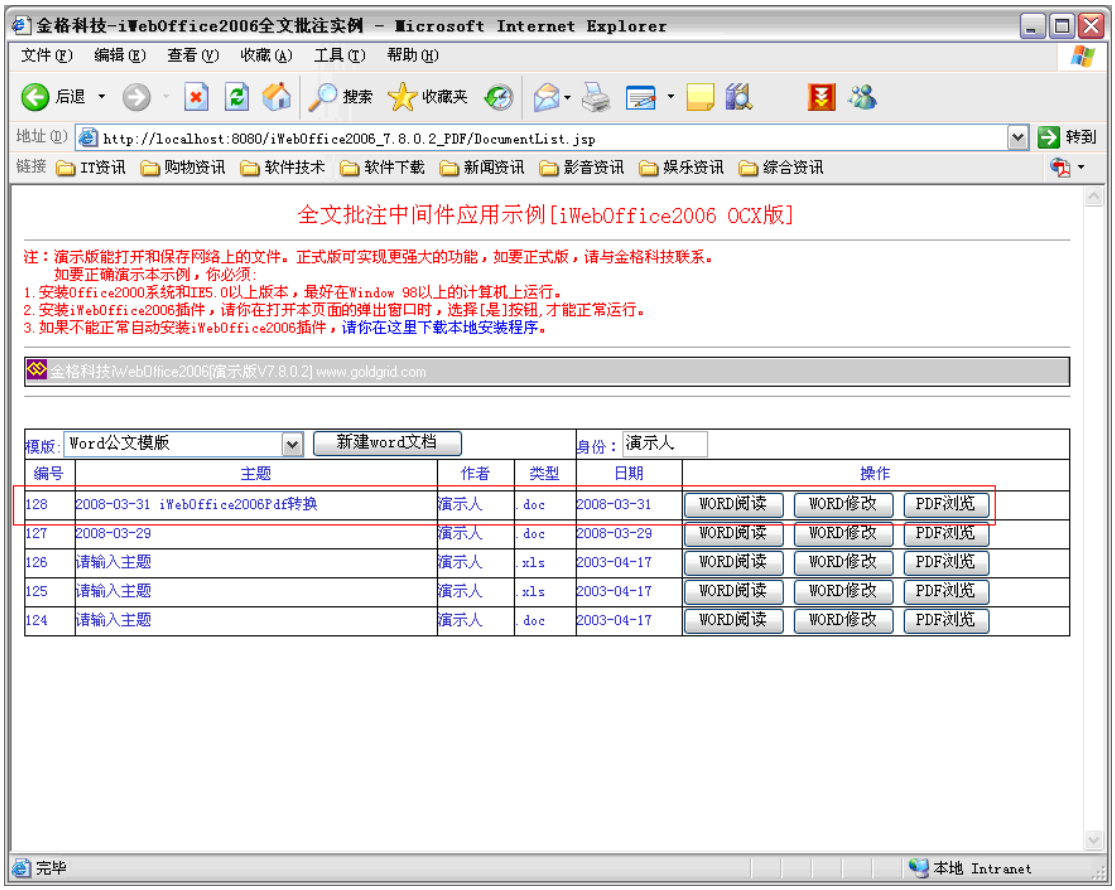
（图 1 列表页面）

2、新使用“新建WORD文档”的按钮新建一个文档，假设我们进入了WORD文档的流程。如：图2



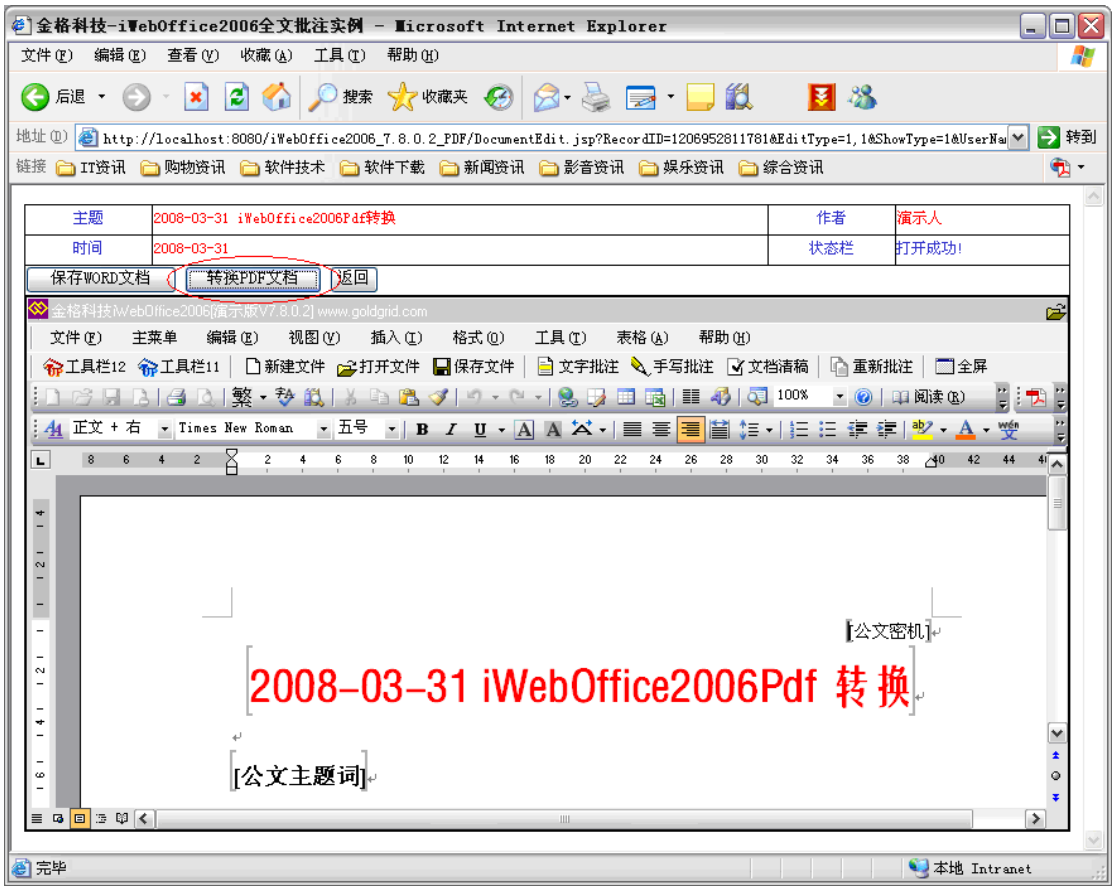
（图 2 编辑页面）

3、我们编辑这篇文档，假设这个文档的主题为“2008-03-31 iWebOffice2006Pdf转换”，然后我们保存它，假设我们在走WORD文档的流程。保存后我们可以看到这条记录，如：图3



（图 3 列表页面）

4、现在我们假设所有的WORD流程已经走完了，现在需要转换PDF文档。我们使用“WORD修改”按钮重新打开这篇文档。进入后我们点击“转换PDF文档”按钮来将这篇文档转换成PDF文档。如：图4

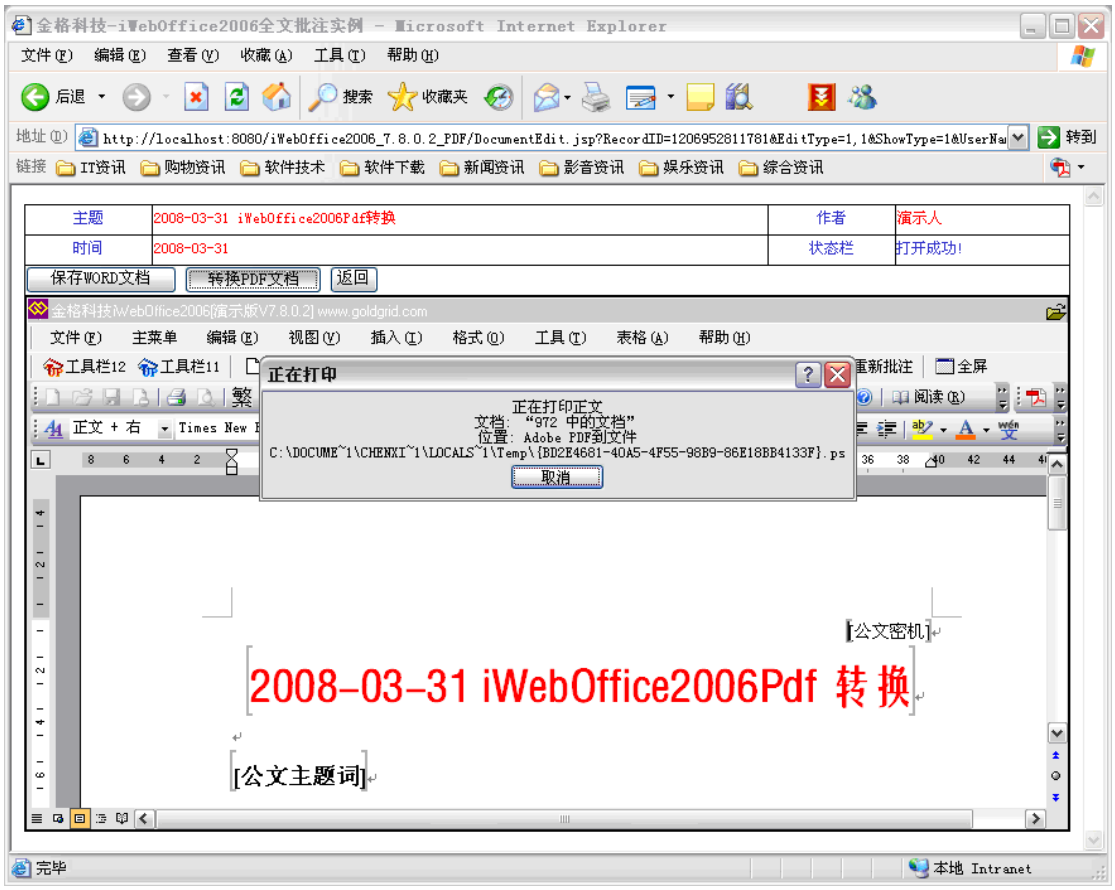


（图 4 编辑页面）

5、还记得我们按钮触发的代码吗？下面是按钮触发的转换并保存成PDF文档的代码。

```
//-----编辑页面中调转换PDF文档的接口-------------------------------------------------------//
function TestWebSavePDF(){
  try{
    if(webform.WebOffice.WebSavePDF()){
      alert("转换并保存成功");
    }
    else{
      alert("转换并保存失败");
    }
  }
  catch(e){
    alert(e.description);                    //出现异常时提示错误信息
  }
}
```

运行时的效果如：图5

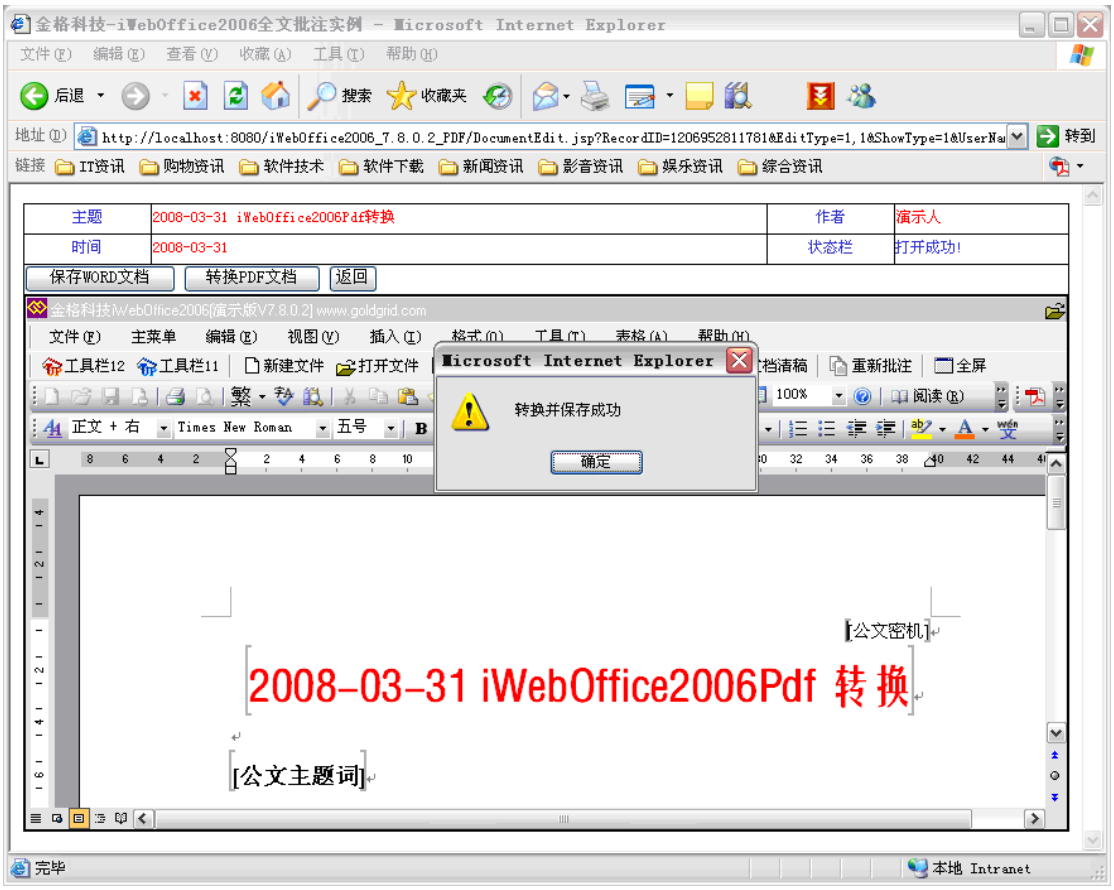


（图 5 转换 PDF 文档中）

6、转换完成后会自动提交文档数据到后台处理页面中，这时看看我们的处理代码：

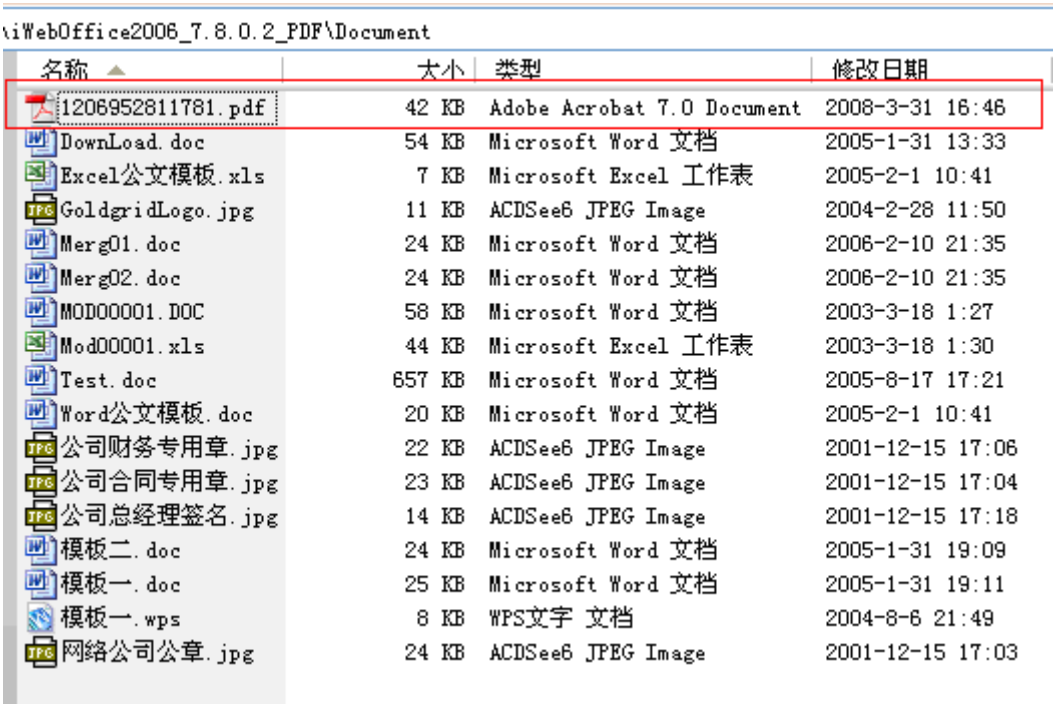
```
//-----服务器处理页保存PDF文档-------------------------------------------------------------//
else if(mOption.equalsIgnoreCase("SAVEPDF")){  //下面的代码为保存 PDF 文件
  mRecordID=MsgObj.GetMsgByName("RECORDID");        //取得文档编号
  mFileName=MsgObj.GetMsgByName("FILENAME");        //取得文档名称
  MsgObj.MsgTextClear();                            //清除文本信息
  //保存文档内容到文件夹中
  if (MsgObj.MsgFileSave(mFilePath+"\\Document\\"+mRecordID+".pdf")){
    MsgObj.SetMsgByName("STATUS", "保存成功!");     //设置状态信息
    MsgObj.MsgError("");                            //清除错误信息
  }
  else {
    MsgObj.MsgError("保存失败!");                   //设置错误信息
  }
  MsgObj.MsgFileClear();                            //清除文档内容
}
```

保存操作结束后会得到提示信息，如：图6



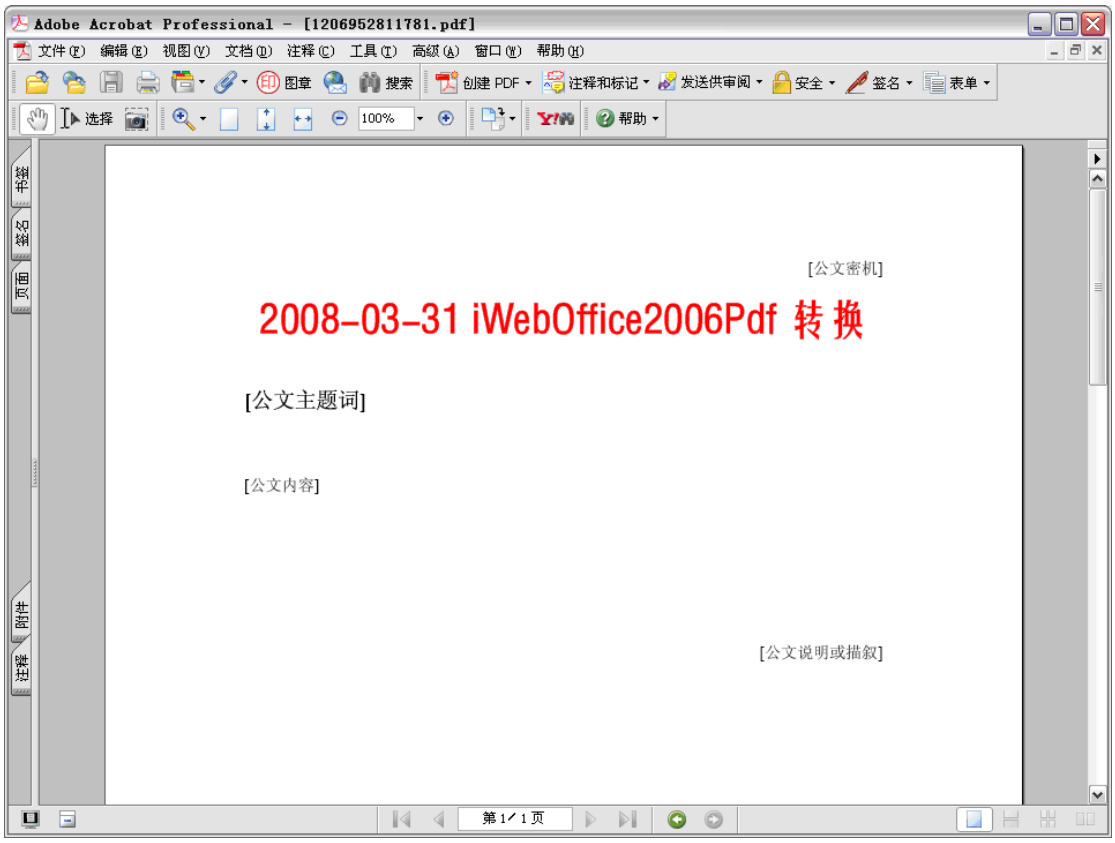
（图 6 转换 PDF 文档完成）

7、我们找到项目的Document目录，以RecordID内容为名称的PDF文件已经在里面了。如：图7



（图 7 保存在服务器中的 PDF 文档）

8、用PDF软件打开看看，是不是我们WORD文档中的内容。如：图8



（图 8 打开服务器中保存的 PDF 文档）

至此，转换和保存PDF文档的工作就完成了，整个操作非常简单快捷。当WORD文档内容比较多，文档比较大时速度是会等比增加的，总体速度和PDF软件自己转换速度差不多。

2.2 iWebPDF 控件浏览 PDF 文档

文档生成了，可以归档工作自然不会困难。我们现在来实现后续的功能：归档的PDF文档提供给用户浏览。

通过仔细阅读iWebPDF的中间件白皮书，我们发现该控件和iWebOffice2006的工作原理极其相似，开发接口的命名规范也很类似。而且控件提供的功能虽然并不复杂，但是确都非常实用。我们决定将这个控件和我们之前做的示例结合起来，来实现控制PDF文档浏览的功能。

现在需要用到之前新增的“PDF浏览”按钮了，我们为它的onClick事件增加以下代码：

```
//-----“PDF浏览”按钮-------------------------------------------------------------------//
<input type=button onClick="javascript:location.href='PdfView.jsp?RecordID=<%=RecordID%>
&UserName='+username.value;" value="PDF浏览">
```

代码的功能是打开PDF浏览的页面。PDF浏览的页面我们是从iWebPDF中间件的示例包中取得的编辑页面（DocumentEdit.jsp），并做了一些修改去掉本次不需要的内容。同时我们

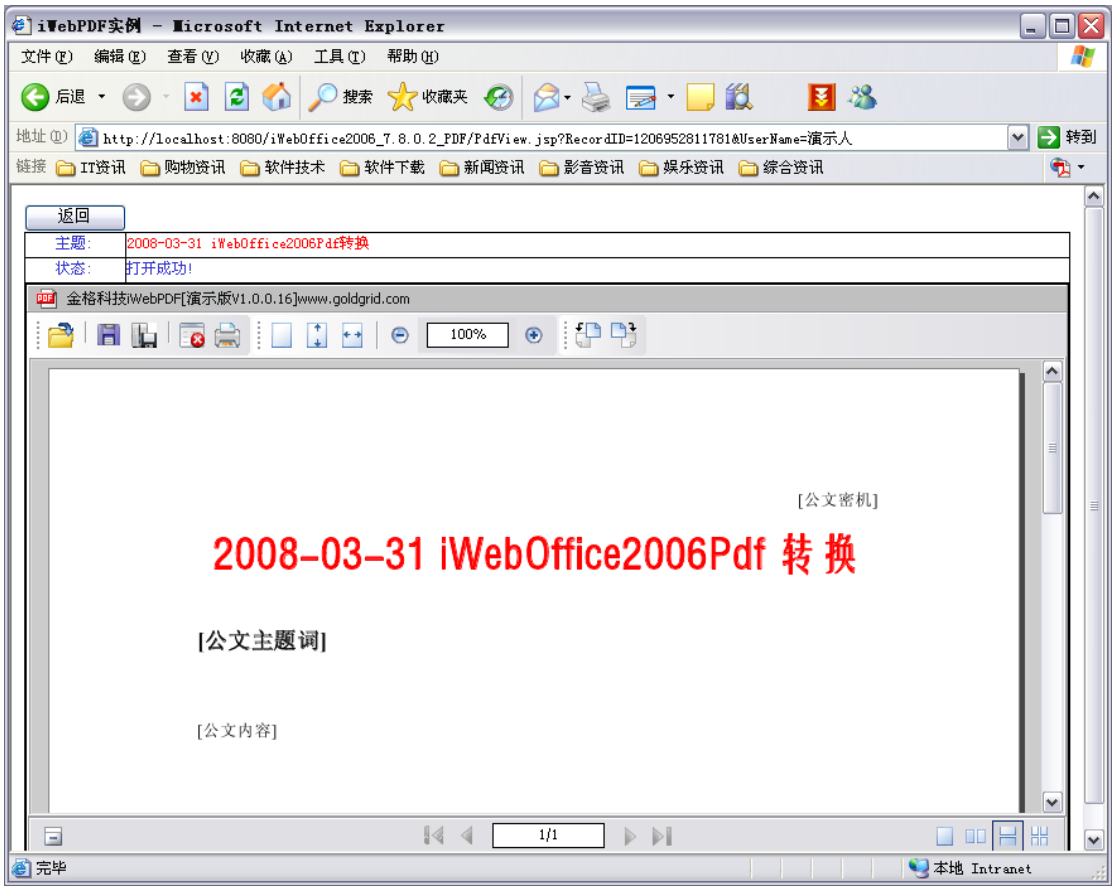
直接将iWebPDF.js、iWebPDF.cab、PDFServer.jsp三个文件复制过来放在我们本地的测试项目中。为了能读取服务器文件夹中的PDF文档，我们需要修改一下PDFServer.jsp中打开文档的部分代码，将原来从数据库读取文档数据的代码修改为从服务器上读取文档，路径按之前iWebOffice2006保存时的RECORDID+".pdf"的方式来。

```
//-----PDFServer.jsp中读取PDF文档的操作代码-----------------------------------------//
if (mOption.equalsIgnoreCase("LOADFILE")) {//下面的代码为打开服务器数据库里的文件
  mRecordID = MsgObj.GetMsgByName("RECORDID");          //取得文档编号
  mFileName = MsgObj.GetMsgByName("FILENAME");          //取得文档名称
  mFileType = MsgObj.GetMsgByName("FILETYPE");          //取得文档类型
  MsgObj.MsgTextClear();                                //清除文本信息
  //下面的内容为从文件夹调入文档
  if (MsgObj.MsgFileLoad(mFilePath+"\\Document\\"+mRecordID+".pdf")){
    MsgObj.SetMsgByName("STATUS", "打开成功!");         //设置状态信息
    MsgObj.MsgError("");                                //清除错误信息
  }
  else {
    MsgObj.MsgError("打开失败!");                       //设置错误信息
  }
}
```

全部完成后我们来看一下实际应用时的效果。按之前的流程，文档所有的流程已经走完，并且也成本好了 PDF 文档作为归档或提供用户浏览。现在我们假设这里的“PDF 浏览”按钮就是提供给用户的浏览已经归档的 PDF 文档功能。如：图 9

（图 9 准备浏览 PDF 文档）

打开浏览页面，iWebPDF 控件从服务器上读取 PDF 文档并显示出来。



（图 10 浏览中的 PDF 文档）

这里说一下，iWebPDF 控件可以在没有安装 PDF 浏览器的机器上实现 PDF 文档的在线阅读、打印和下载，从而免去了客户端都需要安装 PDF 浏览器的繁琐操作。这对于一个大型系统的应用来说带来的便利是非常可观的。

至此我们的一系列文档操作就完成了。当然，真正的系统中文档流转、控制和管理不会如此的简单，本文的目的是希望抛砖引玉，给更多的朋友介绍这种简单、便捷但非常实用的功能如何应用。

2.3 更多扩展应用的可能

在 iWebOffice 中如何控制 WORD 文档我们就不赘述了，相信很多人比我了解的还清楚。而关于 PDF 的应用扩展相应很多人用的并不多，当然很大程度是因为之前没有多少可实用的应用方案。目前应用 iWebPDF 则可以实现之前很多不好实现的功能应用，比如：没有安装 PDF 软件的客户机器浏览 PDF 文档，进行打印、下载控制，进行远程交互等功能。

（图 11 这些按钮都是可以通过接口控制的）

三、 总结

以上我们针对金格最新发布的 iWebOffice2006 v7.8 和 iWebPDF 两个中间产品关于 PDF 文档的应用做了个研究。基于两者的功能和应用方式，至少目前已经为客户提供了一种可以应用 PDF 文档管理的解决方案。相信随着应用客户的增多和产品自身的完善，使用金格的 PDF 应用方案将是一个很好、很强大的选择！

全文完。

江西金格产品服务部·陈湘凌

2008 年 3 月