Степуро Е.Н. ©
Рязанский государственный радиотехнический университет

АВТОМАТИЗИРОВАННОЕ ДОБАВЛЕНИЕ ПЕЧАТНЫХ ШТАМПОВ ПРИ ПОМОЩИ ADOBE ACROBAT 6.0 БЛАГОДАРЯ ИСПОЛЬЗОВАНИЮ VISUAL BASIC (VBA) И JAVASCRIPT

В MS Word макрос представляет собой инструмент, который позволяет запоминать какие-либо определенные задачи и действия, а в дальнейшем их выполнять. Иногда на практике встречаются случаи, когда требуется определить удобный макрос для создания водяных знаков для печатных бланков. Первая мысль, которая может прийти в голову, это добавление графического объекта в верхний и нижний колонтитулы документа. Редактор MS Word может сделать это автоматически. Задача звучит достаточно просто. Но что если, во-первых, графический объект должен быть за текстом, а, во-вторых, предположим, что бланки должны быть только на первой странице. Их можно поместить в основной текст первой страницы, или изменить формат страницы так, чтобы получились разные верхний и нижний колонтитулы на первой и последующих страницах. Если необходимы печатные бланки на всех страницах, кроме первой, то можно использовать второй вариант. Но если получать документ из другого источника, как клиент или поставщик, то есть вероятность, что документ уже имеет разные колонтитулы в разных разделах. В итоге мы получаем слишком много условий.

Но в то же время Adobe Acrobat 6.0 имеет такую функцию, как размещение одного RDF-документа поверх другого. [1, 152] Это является отличным решением проблемы. Для правильной работы необходимо конвертировать документ MS Word в формат PDF и затем использовать Adobe Acrobat 6.0 для того, чтобы добавить печатный штамп в PDF. Единственное, что остается это описать инструкцию для пользователей. Повезет, если пользователь будет знаком с работой Adobe Acrobat 6.0 и редактора MS Word. А если он является новичком? Большинство пользователей не понимают, как работают эти программы, по какому алгоритму все происходит и почему происходит именно так, а не наоборот. Вместо этого они пытаются запомнить нужную им последовательность действий. Чтобы не обременять пользователя инструкциями «Конвертируйте документ MS Word в документ PDF. Затем объедините его с печатным штампом PDF, используя добавление водяных макросов, после чего распечатайте документ», стоит написать макрос, который максимально автоматизирует задачу.

В данной статье рассмотрим использование Visual Basic for Application (VBA) для решения данной задачи. На первый взгляд эта идея кажется трудной, но вполне разрешимой.

Для этого будем использовать Interapplication Communication for Acrobat (IAC). [4] На Windows термин «Interapplication» подразумевает OLE (Object Linking and Embedding), на MacOS он выступает в качестве AppleScripts. Объекты OLE могут быть описаны на языке Visual Basic. Существуют три версии IAC, каждая из которых соответствует одной из основных версий Adobe Acrobat: 4, 5 или 6. Но в них доступ к водяному знаку можно обеспечить только путем выполнения пункта меню, который имитирует нажатие кнопки мыши:

```
Dim app As Object
Set app = CreateObject("AcroExch.App")
app.MenuItemExecute( "COMP:AddBack" )
```

Метод MenuItemExecute() определен в справочнике IAC, в котором данный пункт меню описан в Acrobat Core API Reference.[4] Приведенный выше код открывает диалоговое окно, которое появляется при добавлении водяных знаков в Adobe Acrobat.

Также можно использовать JavaScript API для наложения двух PDF-страниц. Такая техника использует функции, которые доступны только через JavaScript, а не через IAC. Компания Adobe представила некий мост JavaScript – IAC с версии Acrobat 6.0. [3] Но документ, описывающий его работу, исключен из публичного доступа. Хотя в базе знаний Acrobat можно найти не задокументированный код на Visual Basic для метода PDDoc.GetJSObject(). Здесь JSObject является коллекцией объектов, находящихся в Acrobat JavaScript API. [2, 230] JSObject возвращает аргументы типа Variant и принимает почти все аргументы элементарных типов. Из этого можно сделать вывод, что мост Visual Basic/JavaScript должен быть вызван следующим образом:

```
Set app = CreateObject("AcroExch.App")

Set avDoc = app.GetActiveDoc ' get the logical doc
Set pdDoc = avDoc.GetPDDoc   ' get the physical doc
Set jso = pdDoc.GetJSObject  ' get the bridge
docs = jso.app.activeDocs    ' get array of active docs,
                             ' app is the JS handle to
Acrobat's Application top level object
For Each doc In docs         ' iterate docs
   ...
Next
```

На данном этапе есть все, что нужно для описания макроса. Таким образом, для реализации нашей задачи необходимо выполнить следующие шаги:

1. Преобразовать документ MS Word через Adobe PDF Printer.
2. Подождать, пока Acrobat откроет обработанный PDF-документ.
3. Получить JSObject из обработанного PDF-документа.
4. Добавить первую страницу другого PDF-документа в обработанный PDF-документ.
5. Включить добавленную страницу в шаблонную.
6. Инициализировать шаблонную страницу для всех обработанных PDF-документов, при помощи слияния содержимого шаблона с нормальным содержанием.
7. Удалить шаблонную страницу из обработанного PDF-документа.

Самым сложным является шаг 4, так как в JavaScript API нет метода GetActiveDoc(). Подобное по функционалу свойство JavaScript App.activeDocs возвращает массив всех открытых PDF-документов. Для того чтобы получить управление через JavaScript для обработанного PDF-документа, следует перебрать весь этот массив и сравнить имя каждого документа с именем документа полученного через метод GetActiveDoc().

Литература

1. Пономаренко, С. Adobe Acrobat 8. Формат PDF и печать: Самоучитель / С. Пономаренко. – Изд-во БХВ-Петербург, 2007. – 296с.;
2. Stefanov, S. Object-Oriented JavaScript / S. Stefanov // Packt Publishing, 2013. – 354 с.;
3. [Электронный ресурс] http://www.planetpdf.com/;
4. [Электронный ресурс] http://public.id/interapplication-communication-api-reference-ppt.htm.