электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

 # 09, сентябрь 2015

УДК 004.62

Использование скриптов PowerShell для обработки слабоструктурированных данных из Интернет-ресурсов

***Козлов М.В.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

***Рыбальченко М.А.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Остриков С.П., к.т.н, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
ostrikov@bmstu.ru

Введение

Сегодня наиболее острые проблемы управления информацией произрастают из организаций, полагающихся на большое число разнотипных, взаимосвязанных источников данных, но не имеющих возможности управлять этими пространствами данных удобным, интегрированным и обоснованным способом.

Взаимодействие с веб-сервисами, чаще всего, производится с использованием некоторой, специально написанной программы. Поэтому если требуется изменить какие-либо параметры, разработчик должен был изначально заложить такую возможность в код. Если этого не было сделано, придется переписывать код и перекомпилировать программу. *PowerShell* позволяет даже простым пользователям взаимодействовать с веб-сервисами [2]. Программисту достаточно единожды написать код обращения к веб-сервису, а в дальнейшем, внести изменения сможет любой пользователь, достаточно открыть скрипт в обычном текстовом редакторе.

PowerShell обладает всеми необходимыми возможностями для написания даже сложных программ. Имеется интегрированная среда сценариев – приложение, позволяющее записывать, выполнять и тестировать сценарии и модули в удобной среде.

В Университете работают и предоставляют информацию сторонним пользователям более 50 веб-сервисов[1]. Необходимо обеспечить простой и удобный доступ к данным

сервисам. На основе сервиса «Сессия» и «Приказы ректора по основной деятельности» будет рассмотрено взаимодействие с веб-сервисами и веб-страницами средствами *PowerShell*.

Результат обработки можно использовать для анализа, либо для хранилища данных.

Настройка *PowerShell* перед выполнением скриптов

У *PowerShell* имеется несколько режимов исполнения, которые определяют, какой тип кода разрешается выполнять[3]. Существует 5 различных режимов исполнения:

- *Restricted*: режим по умолчанию, не загружает файлы конфигурации и не выполняет скрипты.
- *AllSigned*: необходимо, чтобы все скрипты и файлы конфигурации были подписаны доверенным издателем, в том числе скрипты, подготовленные на локальном компьютере.
- *RemoteSigned*: требует, чтобы все скрипты и файлы конфигурации, загруженные из Интернета, были подписаны доверенным издателем.
- *Unrestricted*: загружает все файлы конфигурации и выполняет все скрипты. Если запущен неподписанный скрипт, который был загружен из Интернета, то программа просит ввести разрешение перед запуском.
- *Bypass*: ничего не блокируется, и никакие предупреждения и запросы не появляются.

Т.к. по умолчанию стоит режим *Restricted*, то при запуске скрипта появится ошибка, представленная на рисунке 1.

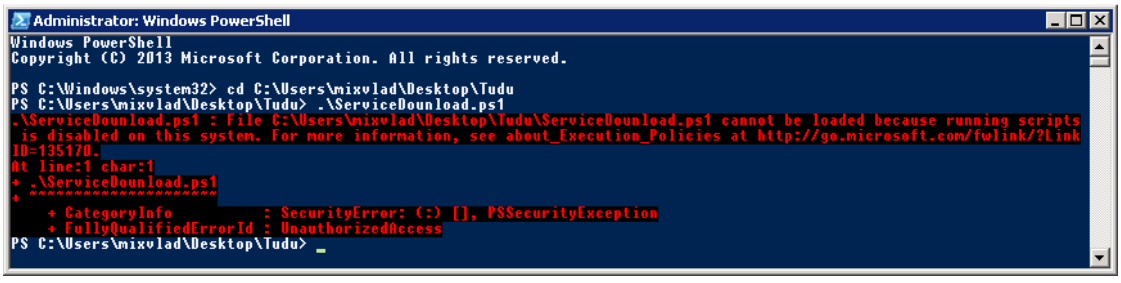

Рис. 1. Ошибка выполнения при Restricted режиме

Для того, чтобы изменить текущий режим исполнения, необходимо запустить *PowerShell* от имени администратора. Затем, выполнить команду *Set-ExecutionPolicy* с указанием требуемого режима исполнения:

Set-ExecutionPolicy Unrestricted

После того, как вы выполнили все необходимые вам скрипты, необходимо вернуть режим исполнения в состояние *Restricted*, чтобы в дальнейшем, обезопасить себя от выполнения вредоносных скриптов:

Set-ExecutionPolicy Restricted

Взаимодействия с *SOAP*-сервисом

Для обращения к веб-страницам или веб-сервисам, используется команда *Invoke-WebRequest*. Она позволяет отправлять *Http*, *Https*, *Ftp* или *File* запросы к веб-страницам или веб-сервисам. В результате, на выходе получается коллекция форм, картинок, ссылок и других *Html* элементов. Пример обращения к сервису, предоставляющему оценки студентов в сессии:

```
$url = "https://eu.bmstu.ru/modules/session/service/xml/student-marks/"
$request = Invoke-WebRequest -URI $url
$elements = $request.ParsedHtml.getElementsByTagName("collection")
```

Обработка самоподписанных сертификатов

Для того, чтобы обращаться к серверам с самоподписанными сертификатами, необходимо перед вызовом запроса добавить следующие указания[4]:

```
add-type @"
using System.Net;
using System.Security.Cryptography.X509Certificates;
public class TrustAllCertsPolicy : ICertificatePolicy {
public bool CheckValidationResult(
ServicePoint srvPoint, X509Certificate certificate,
WebRequest request, int certificateProblem) {
return true;}}"@
```

Изменение кодировки строки

Иногда, кодировка загружаемых данных определяется неправильно, или требуется сохранить данные в определенном формате, для этого подойдет следующая функция:

```
function ConvertTo-Encoding ([string]$From, [string]$To){
    Begin{
        $encFrom = [System.Text.Encoding]::GetEncoding($from)
        $encTo = [System.Text.Encoding]::GetEncoding($to)
```

```
    }
    Process{
        $bytes = $encTo.GetBytes($_)
        $bytes = [System.Text.Encoding]::Convert($encFrom, $encTo, $bytes)
        $encTo.GetString($bytes)
    }
}
```

Пример вызова описанной функции, для перекодирования строки из формата *windows*-1251 в формат *koi8-r*:

```
"Проверка" | ConvertTo-Encoding  "windows-1251" "koi8-r"
```

Взаимодействие с *REST*-сервисом

Для обращения к *REST*-сервисам, используется команда *Invoke-RestMethod*[6]. Результатом обращения будут сильно структурированные данные, что облегчает дальнейшее их использование. Для взаимодействия с серверами, которые имеют самоподписанный сертификат, в самом начале необходимо добавить строку:

```
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = { $true }
```

Указываем требуемый url и подставляем его в метод *Invoke-RestMethod*:

```
$url = "https://eu.bmstu.ru/modules/session/service/xml/student-marks/"
$request = Invoke-RestMethod -Uri $url
```

Взаимодействие с веб-страницей используя *XPath*

Опишем процесс взаимодействия с веб-страницей. Для начала, необходимо скопировать url интересующей страницы из браузера. Объявляем ее в качестве переменной:

```
$url = "http://ud.e-u.bmstu.ru/escado/officefree.nsf/W-A-2?OpenView&Start=1&
Count=250&Expand=1#1";
```

Подключаем библиотеку, которая позволит взаимодействовать с веб страницей и создаем элемент класса веб-страницы:

```
Add-Type -Path "C:\Users\mixvlad\Desktop\Tudu\HtmlAgilityPack.dll"
$webGraber = New-Object -TypeName HtmlAgilityPack.HtmlWeb
```

Класс *HtmlWeb* содержит метод *Load*, который позволяет указать в качестве параметра url или путь к файлу. Если на сайте имеются русские символы и кодировка по

умолчанию не *UTF8*, то сперва необходимо загрузить страницу на локальную машину с указанием правильной кодировки:

$client = New-Object System.Net.WebClient

[System.Text.Encoding]::GetEncoding('koi8-r').GetString(

[Byte[]]$client.DownloadData($url)) | out-file "C:\Users\mixvlad\Desktop\Tudu\temp.html"

Указываем в качестве параметра загруженный файл:

$webDoc = $webGraber.Load("C:\Users\mixvlad\Desktop\Tudu\temp.html")

Чтобы выбрать из страницы конкретный элемент, воспользуемся возможностями *HtmlAgilityPack*. Сделать это можно, используя язык запросов к элементам *XML – Xpath*. В браузере *Chrome*, *Xpath* можно определить, щелкнув правой кнопкой мыши по интересующему элементу, выделить его в списке элементов, правая кнопка мыши -> *Copy Xpath*. Указываем полученный *Xpath* в качестве параметра для метода *SelectNodes*:

$Thetable = $webDoc.DocumentNode.SelectNodes("/html/body/table[1]")

$trDatas = $Thetable.elements("tr")

Написание *PowerShell* скриптов

Для написания и отладки скриптов *PowerShell* подойдет утилита *Windows PowerShell ISE*[5]. Она значительно облегчает работу с кодом. Имеется подсветка синтаксиса, заполнение нажатием клавиши *TAB*, совместимость с Юникодом, контекстная справка, есть возможность устанавливать точки останова, отображается текущее значение переменных, можно выполнять пошаговую трассировку кода.

Интерфейс программы представлен на рисунке 2.

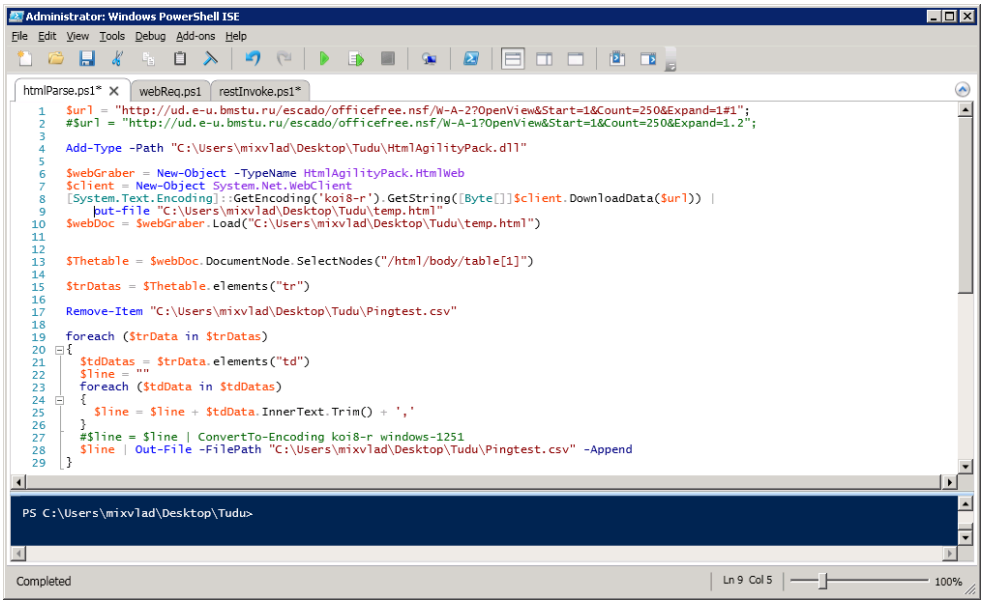

Рис. 2. Интерфейс программы *Windows PowerShell ISE*

Выводы

Были рассмотрены варианты взаимодействия с *Soap* и *Rest* сервисами. Описаны 2 варианта взаимодействия с веб-страницей. Разобраны способы перекодирования символов и способы обращение к серверам с самоподписанными сертификатами. Описаны преимущества использования утилиты для написания и отладки скриптов *PowerShell – Windows PowerShell ISE*.

Организация взаимодействия с веб-сервисами средствами *PowerShell* позволяет, даже неквалифицированным пользователям, выполнять дальнейшее сопровождение существующих скриптов, без необходимости установки на машину специальных инструментов.

Список литературы

1. Козлов М.В. Модель информационной среды университета на основе единого репозитория сервисов // Молодежный научно-технический вестник. Электрон. журн. 2014. № 01. Режим доступа: http://sntbul.bmstu.ru/doc/681173.html (дата обращения 01.05.2015).
2. Попов А. Введение в Windows PowerShell. Санкт-Петербург: БХВ-Петербург, 2009. 452 с.
3. Payette B. Windows PowerShell in Action, Second Edition. NY: Manning Publications, 2011. 1016 p.
4. Holmes L. Windows PowerShell Cookbook: The Complete Guide to Scripting Microsoft's Command Shell. Sebastopol: O'Reilly Media, 2013. 1036 p.
5. Вальковский С. Windows PowerShell. Режим доступа: http://www.windowsfaq.ru/content/view/707/94/ (дата обращения 05.12.2015).
6. Hill K. Effective Windows PowerShell. Available at: http://rkeithhill.wordpress.com/2009/03/08/effective-windows-powershell-the-free-ebook/, accessed 05.12.2015.