iLove SharePoint - PowerWebPart 3.0

Version 0.1

Blog: <http://www.iLoveSharePoint.com>

Codeplex: <http://www.codeplex.com/iLoveSharePoint>

# Installation

## Prerequisites

* Microsoft SharePoint Services 3.0 SP1
* Microsoft PowerShell 1.0
* Microsoft .NET Framework 3.5

## Install and Deploy Solution

* Extract the ZIP-File
* Install the solution using setup.exe or ststadm.exe
* To impersonate the PowerWebPart to the current user, you have to change the “aspnet.config” in “%windir\Microsoft.NET\Framework\v2.0.50727\” as follows:

**<configuration>**

**<runtime>**

**<legacyImpersonationPolicy enabled=”false”/>**

**<alwaysFlowImpersonationPolicy enabled=”true”/>**

**</runtime>**

**</configuration>**

Read more: <http://blogs.msdn.com/powershell/archive/2007/09/10/impersonation-and-hosting-powershell.aspx>

* (Optional) To use the PowerWebParts AJAX (.NET 3.5) functionality, you have to modify the web.config of your web application as follows:

<httpHandlers>

...

<add verb="\*" path="\*\_AppService.axd" validate="false" type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />

<add verb="GET,HEAD" path="ScriptResource.axd" type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" validate="false" />

</httpHandlers>

## Central Administration Configurations

#### Script Signing

The scripts will be signed with a digital signature when a Farm Administrators applies the script. The key which is used can be configured in the Central Administration. A random key has been created during installation.

Application Settings:





#### PowerLibrary

You can configure a Document Library as a central script and DLL repository. Only farm administrators should have write access! Scripts stored there will not be signed.





# Users Guide

## Activate Feature

* Activate the “iLove SharePoint – PowerWebPart 3.0” feature on a site collection.



The activation adds the “PowerWebPart” to the web part gallery of the site collection

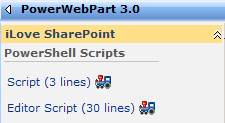
## Add “PowerWebPart” to a Web Part Page

* Navigate to a WebPart Page
* Choose “Site Actions” -> Edit Page
* Click on “Add a Web Part”
* Go to category “iLove SharePoint”, select PowerWebPart 3.0 and click add.

## Configure “PowerWebPart”

In the web part’s verb menu choose “Modify Shared Web Part”

### Script

**

*The “PowerShell Script” section is only visible to “Farm Administrators”!*

**Script**

Click on “**Script (… lines)**” to open the Simple Script Editor.

If you have PowerGUI ( <http://powergui.org> ) and iLove SharePoint PowerGUI Launcher ( <http://ilovesharepoint.codeplex.com/Release/ProjectReleases.aspx?ReleaseId=20019> ) installed you can edit the scripts in PowerGUI Script Editor. To do so click on the PowerGUI icon (only visible when PowerGUI Launcher is installed).

To get started delete the default script and enter:

function Render($writer)

{

$writer.Write("<h2>May the power be with you!</h2>")

}

Clicking “Apply” in the editor saves the script and closes the editor. To see your changes in the web part click “Apply” or “Okay” in the web part configuration panel.

## Script Reference

#### Predefined Variables

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| $this | iLoveSharePoint.WebControls.PowerControl  inherits [System.Web.UI.WebControls.WebParts.WebPart](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.webparts.webpart(VS.80).aspx) |  |
| $webpart | iLoveSharePoint.WebControls.PowerWebPart  inherits [System.Web.UI.WebControls.WebParts.WebPart](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.webparts.webpart(VS.80).aspx) | You can access the parameters as follows:  $webpart.Parameter1 -20 |
| $page | [System.Web.UI.Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) | ParentASP.NET Page |
| $isPostBack | [System.Boolean](http://msdn.microsoft.com/en-us/library/system.boolean.aspx) | Indicated if the request is a post back or not |
| $site | [Microsoft.SharePoint.SPSite](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.spsite.aspx) | SharePoint Site Collection of the context |
| $spContext | [Microsoft.SharePoint.SPContext](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.spcontext.aspx) | SharePoint Context |
| $httpContext | [System.Web.HttpConext](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.aspx) | ASP.NET HttpContext |
| $viewState | [System.Web.UI.StateBag](http://msdn.microsoft.com/en-us/library/system.web.ui.statebag.aspx) | ASP.NET View State |
| $scriptManager | [System.Web.UI.ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.aspx) | ASP.NET AJAX Script Manager |
| $web | [Microsoft.SharePoint.SPWeb](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.spweb.aspx) | SharePoint Web of the context |
| $list | [Microsoft.SharePoint.SPList](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.splist.aspx) | SharePoint List of the context |
| $item | [Microsoft.SharePoint.SPListItem](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.splistitem.aspx) | SharePoint List Item of the context |
| $webpart.parameter1 (..20) | [System.String](http://msdn.microsoft.com/en-us/library/system.string.aspx) | Parameter 1-20 (string) |

#### Predefined Functions

|  |  |  |
| --- | --- | --- |
| Functions | Parameters | Description |
| Get-SPSite | -webUrl |  |
| Get-SPWeb | -webUrl |  |
| Subscribe-Event | -object -event -callback | e.g. Subscribe-Event –object $button –event “Click” –callback “MyFunction” |
| Import-PowerModule | -name [-noCache] | Import a script from the PowerLibrary. Have to be configured in the central administration |
| Import-Assembly | -name [-noCache] | Import a .NET assembly from the PowerLibrary. Have to be configured in the central administration. |
| Query-Connections |  | Get data from webpart connections. Fires OnReceiveRow and OnReceiveTable functions |
| Init-Parameter | -name -defaultValue [-defaultOnEmpty] | Initialize the WebPart Parameters (e.g. $color = Init-Parameter “Parameter1” “Red” –DefaultOnEmpty) When DefaultOnEmpty is set and the parameter value is an empty string the default value will be returned. |
| Register-JavaScriptBlock | -name -script |  |
| Register-CSSBlock | -css |  |
| Register-JavaScriptInclude | -name -url |  |
| Register-CSSInclude | -url |  |
| Add-HtmlToHeader | -html |  |
| RunAs-System | -script -args | Impersonates the script in the script parameter to the application pool user. |

### 

### Example

########## Initialize ############

## declare global variables and functions

$textbox = **New-Object** System.Web.UI.WebControls.TextBox

$button = **New-Object** System.Web.UI.WebControls.Button

$label = **New-Object** System.Web.UI.WebControls.Label

############# Load ##############

## first time the OnLoad fires before CreateChildControls

function OnLoad

{

# Check if GET Request (first request).

if($isPostBack -eq $false)

{

$label.Text = 'GET request.'

}

}

####### 3. Create Controls ########

# create child controls

function CreateChildControls($controls)

{

$controls.Add($textbox)

$button.Text = 'Click Me'

Subscribe-Event $button 'Click' 'OnButtonClicked'

$controls.Add($button)

$controls.Add($label)

}

####### Events ########

# handle control events

# subscribe to an event with "Subscribe-Event($control, 'eventName','callback function name')"

function OnButtonClicked($sender, $args)

{

$label.Text = $textbox.Text

}

# fires when the AJAX timer has elapsed

# function OnAjaxRefresh{}

######## Connections #########

### Provider ####

# returns an object containing the properties you want to provide (for reflection only)

#function GetSchema{}

# returns an object matching the schema defined in the GetSchema function

# function SendRow{ return $null}

# returns a collection of objects matching the schema defined in GetSchema function

#function SendTable{ return $null }

### Consumer ####

# receives an object from a webpart row provider

#function OnReceiveRow($row, $schema)

#{

#}

# receives a collection of objects from a webpart table provider

#function OnReceiveTable($table, $schema){}

######## PreRender #########

## Is fired when the event stage is completed.

## Last chance to modify controls before rendering

## function OnPreRender{ }

######### Render #########

## render html

function Render($writer)

{

$writer.Write("<a href='http://iLoveSharePoint.com'>iLove SharePoint</a><br/>")

$textbox.RenderControl($writer)

$button.RenderControl($writer)

$writer.Write("<br/>")

$label.RenderControl($writer)

}

######## Error Handling #########

## render custom error message

#function OnError($exception, $writer){}

######### Unload #########

## clean up

#function OnUnload()

#{

#}