SOLVERA

10 Great Years

How to Bake a SharePoint Environment Without Getting GUI

CANADA'S BEST MANAGED COMPANIES

SOLVERA
10 Great Years

Things I hope you'll take away from this session

- An understanding of what you can do in SharePoint with PowerShell.
- Ideas of how to format your scripts for easy reuse
- Knowledge that if you want to build\modify a script like this you have to have a full picture of your environment.

SOLVERA
10 Great Years

What we'll cover today:

- What is PowerShell?
- Basic SharePoint PS Commands
- How the script was built
- The Config
- Overview of the script
- Publishing vs Non-Publishing Sites
- Limitations I found while building the script.
- Let's Bake us an Environment
- Questions and Comments

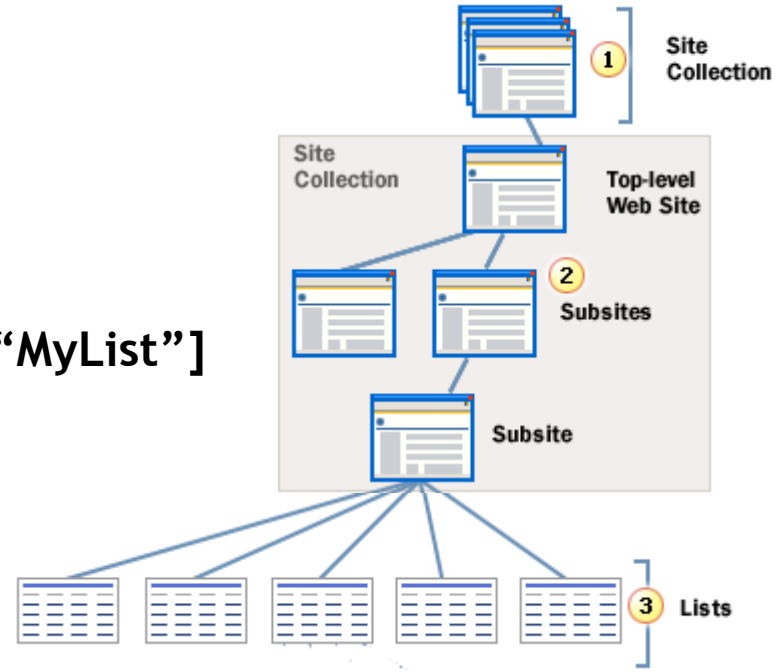
SOLVERA
10 Great Years

So… What's this PowerShell thing?

- Basically, PowerShell is an extendable command shell based scripting language.
- Is a Verb-Noun based language. Like Get-SPSite or Connect-ConfigurationDatabase
- It is based on the .NET framework so you can actually use a lot of the same methods and techniques you do in C#\VB.NET
- It can be used to gather data, enter data, modify data, configure systems and even install solutions.

Some basic SharePoint PowerShell Commands

- New-SPSite
- New-SPWeb
- $spSite = Get-SPSite
- $spWeb = Get-SPWeb
- $list = $spWeb.Lists[“MyList”]
- $spSite.Dispose();
- $spWeb.Dispose();



http://blogs.technet.com/b/stefan_gossner/archive/2008/12/05/disposing-spweb-and-spsite-objects.aspx

SOLVERA
10 Great Years

Why create the script?

- Initially was a task assigned to me for a project.
- Creating databases won't put those ugly GUIDs into the database names.
- AutoSPInstaller didn't do what I was looking for.
- I wanted something I could reference easy to use later.
- I'm a developer…it's what we do.

SOLVERA
10 Great Years

The format of the Script

- **Separated into main functionalities**
  - Main
  - Configuration
  - Build
  - Modify
  - Utility (General Methods)
- **A separate config xml.**

SOLVERA
10 Great Years

The Config File

- Like other programming languages, PowerShell can import and use an xml config file.
- With some exceptions, attributes are used instead of elements for item properties
- Broken up into sections for easy modification

SOLVERA
10 Great Years

How did I decide on the format?

- Argued with my fellow developer on the project for a while
- Did my own research (lots of boring highly technical documents out there arguing both sides)
- Finally, weeded out the basic rule from the documents that sat the best with me
  - …
  - …
  - Always use attributes unless you have to use elements

SOLVERA
10 Great Years

When to use elements

- If the element you are listing can be repeated using the same name
- Sequence is important.
- Ultimately the decision is yours.
- Personal experience: I found if I have a lot of similar items, using attribute based entries is easier to read.

Comparison of two methods

- My config:

```
<WebParts>
  <WebPart Name="Poll" SourceList="Web Part Gallery" SiteCollectionName="IntranetRoot" DestinationUrl="Pages/WebpartTest.aspx"
           SolutionName="EmployeePollWebPart" ZoneName="TopLeftRow" ZoneIndex="1">
  </WebPart>
  <WebPart Name="Feedback" SourceList="Web Part Gallery" SiteCollectionName="IntranetRoot" DestinationUrl="Pages/WebpartTest.aspx"
           SolutionName="FeedbackWebPart" ZoneName="RightColumn" ZoneIndex="1">
  </WebPart>
</WebParts>
```

- Element based:
  - Had to move it to the next page it was too big

SOLVERA
10 Great Years

Element based xml config

```
<WebParts>
  <WebPart>
    <Name>Poll</Name>
    <SourceList>Web Part Gallery</SourceList>
    <SiteCollectionName>IntranetRoot</SiteCollectionName>
    <DestinationUrl>Pages/WebpartTest.aspx</DestinationUrl>
    <SolutionName>EmployeePollWebPart</SolutionName>
    <ZoneName>TopLeftRow</ZoneName>
    <ZoneIndex>1</ZoneIndex>
  </WebPart>
  <WebPart>
    <Name>Feedback</Name>
    <SourceList>Web Part Gallery</SourceList>
    <SiteCollectionName>IntranetRoot</SiteCollectionName>
    <DestinationUrl>Pages/WebpartTest.aspx</DestinationUrl>
    <SolutionName>FeedbackWebPart</SolutionName>
    <ZoneName>RightColumn</ZoneName>
    <ZoneIndex>1</ZoneIndex>
  </WebPart>
</WebParts>
```

SOLVERA
10 Great Years

What about reading element or attribute in code?

- Attribute based:

```
$buildConfig = [xml](Get-Content $configXMLPath)

try
{
    foreach($webPart in $buildConfig.Config.Webparts.Webpart)
    {
        $siteColURL = getSiteColURL($webPart.GetAttribute("SiteCollectionName"))

        addWebPartToPage $siteColURL `
                         $webPart.GetAttribute("DestinationUrl") `
                         $webPart.GetAttribute("Name") `
                         $webPart.GetAttribute("SourceList") `
                         $webPart.GetAttribute("SolutionName") `
                         $webPart.GetAttribute("ZoneName") `
                         $webPart.GetAttribute("ZoneIndex")

    }
}
```

SOLVERA
10 Great Years

What about reading element or attribute in code?

- Element based:

```
$buildConfig = [xml](Get-Content $configXMLPath)

try
{
    $rootNode = $buildConfig.get_DocumentElement()
    $webParts = $rootNode.Config.Webparts.Webpart

    foreach($webPart in $webParts.ChildNodes)
    {
        $siteColURL = getSiteColURL($webPart.SiteCollectionName)

        addWebPartToPage $siteColURL `
                         $webPart.DestinationUrl `
                         $webPart.Name `
                         $webPart.SourceList `
                         $webPart.SolutionName `
                         $webPart.ZoneName `
                         $webPart.ZoneIndex `
    }
}
```

SOLVERA
10 Great Years

The setup

- **Including files**
  - . .\SPBuildFunctions.ps1
- **Get the config (xml)**
  - $runDirectory = Resolve-Path .\
  - $configXMLPath = Join-Path -Path $runDirectory -ChildPath "BuildSLGAIntranetConfig.xml“
  - $buildConfig = [xml](Get-Content $configXMLPath)

SOLVERA
10 Great Years

The setup

- Make sure we can work in SharePoint

```
if((Get-PSSnapin -Name Microsoft.SharePoint.PowerShell -ErrorAction SilentlyContinue) -eq $null)
{
    Add-PSSnapin Microsoft.SharePoint.PowerShell
}
```

SOLVERA
10 Great Years

Non-SharePoint specific code

- if([string]::Compare($buildConfig.Config.AppSettings.CreateContentDB,"True",$true) -eq 0)
- foreach($siteCollection in $buildConfig.Config.SiteCollections.SiteCollection)

SOLVERA
10 Great Years

Non-SharePoint specific code

- Write-Progress -Id $parentId -Activity "Running master script" -Status "Creating SubSites" -PercentComplete 16.7
- [gc]::Collect()

SOLVERA
10 Great Years

The SharePoint Code: Configuration Building Blocks

- New-SPConfigurationDatabase -DatabaseServer $DBServer -DatabaseName $DBName -FarmCredentials $credentials -Passphrase $passPhrase
- or - Connect-SPConfigurationDatabase
- Install-SPHelpCollection -All
- Initialize-SPResourceSecurity

SOLVERA
10 Great Years

The SharePoint Code: Configuration Building Blocks

- Install-SPService
- Install-SPFeature -AllExistingFeatures
- Install-SPApplicationContent
- New-SPCentralAdministration -Port $portNumber - WindowsAuthProvider "NTLM"

SOLVERA
10 Great Years

The SharePoint Code: Content Building Blocks

- **New-SPContentDatabase** –DatabaseServer $DBServer -Name $DBName -WebApplication $WebApp

- **New-SPWebApplication** –ApplicationPool $AppPoolName -Name $WebAppName –DatabaseServer $DBServer –DatabaseName $DBName -HostHeader $HostHeader –Port $PortNumber

SOLVERA
10 Great Years

The SharePoint Code: Content Building Blocks

- New-SPSite $siteURL -OwnerAlias $siteAdmin -Template $siteTemplate
  -Description $siteDescription -ContentDatabase $siteDB -Confirm:$false

- New-SPWeb -Url $webURL -Name $webName -Template $webTemplate
  -Description $webDescription -AssignmentCollection $spAssignment

  -AddToQuickLaunch -AddToTopNav -UseParentTopNav

SOLVERA
10 Great Years

Building with Publishing sites

- Often extra steps are required if a site, page or list is created in a publishing site
- In enterprise environments you are almost guaranteed to be working with publishing sites at one point or another.

SOLVERA
10 Great Years

Building with Publishing sites

- Have the ability to check if working in a publishing site.
  - if(Get-SPFeature -Web $webURL | Where {$_.DisplayName -eq "Publishing"})
- If a site is a publishing site, then now we have to check-in the page or list and also approve the publishing of the item.

SOLVERA
10 Great Years

Creating new pages

- To add pages we first have to access the site collection and\or the subsite\web the page is to be added to.
- Once we have the site and web we have to then create publishing objects.
  - $pbSite = New-Object Microsoft.SharePoint.Publishing.PublishingSite($site)
  - $pbWeb = [Microsoft.SharePoint.Publishing.PublishingWeb]::GetPublishingWeb($web)

SOLVERA
10 Great Years

Creating new pages

- Next we build the new page by setting a layout for it to use and then creating the page using the layout.
  - $pageLayout = $pbSite.GetPageLayouts($true) | Where { $_.Title -eq $pageLayoutName}
  - $page=$pbWeb.AddPublishingPage($pageURL, $pageLayout)

SOLVERA
10 Great Years

Page is created, so we're done right?

- Only if NOT a publishing site.
- Can determine if publishing site.
  - if(Get-SPFeature -Web $webURL | Where {$_.DisplayName -eq "Publishing"})
- Declare the page as a file in PowerShell
  - $file = $web.GetFile($page.URI.ToString())

SOLVERA
10 Great Years

Page is created, so we're done right?

- Check the file in and publish it.
  - if($spFile.Level -eq [Microsoft.SharePoint.SPFileLevel]::Checkout)
  - $spFile.CheckIn("Checked in via PowerShell Script")
  - $spPage.ListItem.File.Publish("Published via PowerShell Script")

SOLVERA
10 Great Years

Approve your submission and clean up.

- Not all publishing sites have moderation enabled to check first.
  - if($spPage.ListItem.ListItems.List.EnableModeration)
- Approve the page publish
  - $spPage.ListItem.File.Approve("Approved via PowerShell Script")
- Clean up after yourself
  - $web.Dispose()
  - $site.Dispose()

SOLVERA
10 Great Years

Creating a non-publishing page

- Much simpler than publishing pages due to lack of options available
- First set the library you are going to use as our root folder
  - $rootFolder = $spWeb.Lists[$libraryName].RootFolder;
- Need a file object from the root folder to create the page
  - $files = $rootFolder.Files;
- Finally, create the page:
  - $newPage = $files.Add($rootFolder.ServerRelativeUrl + "/" + "$pgName", [Microsoft.SharePoint.SPTemplateFileType]::WikiPage);
- Done!! Wasn't that easy?

SOLVERA
10 Great Years

Adding Lists to a site

- Every list has a template. Determine the template first
  - $listTemplate = $web.ListTemplates | where {$_.Name -eq $listTemplateName}
- Using the template create the list
  - $web.Lists.Add($listName, $listDescription, $listTemplate) | Out-Null
- If your list requires custom columns you can add after the list is created
  - $newField = $list.Fields.Add($columnName, [string]$columnType, $true)

Special Columns

- Can add nearly any column type to the list
- Choice Column
  - First build list of choices
    - $choiceList = New-Object System.Collections.Specialized.StringCollection
    - foreach($choice in $column.Choices.Choice)
      {
      $choiceList.Add($choice)
      }
  - Add the column with choices.
    - $list.Fields.Add($columnName, [Microsoft.SharePoint.SPFieldType]::Choice, $isMandatory, $false, $choiceList) | Out-Null

SOLVERA
10 Great Years

Special Columns Cont.

- **Lookup Lists**

- **Need to know source list and column.**
  - $newColumn = $list.Fields.AddLookup($columnName, $sourceList.id, $isMandatory)
  - $lookupField = $updatingList.Fields.GetFieldByInternalName($newColumn)
  - $lookupField.LookupField = $sourceList.Fields[$columnName].InternalName
  - $lookupField.Update()

Publishing Sites vs Non-Publishing Sites

- **Publishing sites can be both more complex and easier at the same time**
  - Have to declare a web and then re-cast it as a publishing web.
  - Publishing sites usually require extra admin (ex. approving pages and publishing them).
  - The ability to access the layout gallery in publishing sites makes it easier to rollout different page types.

SOLVERA
10 Great Years

Limitations of PowerShell Script

- Webparts had to be added after the main script was complete.
- Default groups were not added when a site collection was provisioned. Had to create a second method to create the default groups.