Creating a Windows Installer Package for your Web Application within **30 Minutes** ☺

Changelog

Version:	*Date:*	*Author:*	*Comments:*
0.1	*14.09.2007*	*Sebastian Glöckner*	*Initial version*
0.2	*19.10.2007*	*Bernhard Frank*	*Some minor updates*
0.3	*21.10.2007*	*Sebastian Glöckner*	*Added Windows Vista / Windows Server 2008 IUSR Support*
0.4	*22.10.2007*	*Bernhard Frank*	*Testing – Building Sample App – minor Comments*
0.5	*23.10.2007*	*Sebastian Glöckner*	*Some minor fixes*
0.6	*25.10.2007*	*Sebastian Glöckner*	*Added PHP and Perl Support*
0.7	*26.10.2007*	*Sebastian Glöckner*	*Added MySQL Support*
0.8	*29.10.2007*	*Sebastian Glöckner*	*Some minor fixes*
0.9	*29.10.2007*	*Sebastian Glöckner*	*Updated documentation*
1.0	*29.10.2007*	*Bernhard Frank*	*Minor updates*
2.0.5	*30.04.2008*	*Bernhard Frank*	• *Fixed some bugs (e.g. wrong url when installing in existing website)* • *Added logging and a few other properties* • *Extended scriptparse functionality to replace more @@properties@@* • *PHP/FastCGI support for IIS7 (vista)* • *Built Windows Installer for WAI* • *Aligned doc version with release version*
2.0.6	*03.06.2008*	*Bernhard Frank*	• *Fixed scripts (removed CRLF problem in createScriptConfig.vbs)* • *Added new Properties: checkIfIUSRisListedOnPHPUploadDir , WANTinstalllogFile* • *Updated this documentation (thanks Oliver)*
2.0.7	*10.06.2008*	*Bernhard Frank*	*Added new properties: checkMYSQL40 and SETREADONLY*
2.0.8	*02.10.2008*	*Bernhard Frank*	***added support for SQL Server 2008***: • *added more complexity to the sql password generation script to meet the password policy of sql server* • *changed wmi query to search for sql server to also find sql 2008* *minor changes to make PERL packages work better (needed this for AWSTATS)* *updated this doc (in section "XML Example…")*
2.0.9	*08.10.2008*	*Bernhard Frank*	*minor additions only regarding PERL*

2.0.10	*20.10.2008*	*Bernhard Frank*	*added support for ODBC 5.1 connector (implemented better query for detecting mysql odbc connector) - odbc driver now stored in Property "MYSQL_ODBC_DRIVER"*
2.0.11	*20.10.2008*	*Bernhard Frank*	*added additional template for x64 packages* *(note: to use the 64 template: rename "WebApplicationx64.tpl" to "WebApplication.tpl" before launching the generateconfig.bat)*
2.0.12	*21.10.2008*	*Johannes Michler/ Bernhard Frank*	*deleted x64 template - run "Buildx64Config.vbs" to automatically create a x64 bit package out of your *.wxs file*
2.0.13	*28.10.2008*	*Bernhard Frank*	*bugfixes:* • *GenerateSourceConfig.vbs now sets only sets folder permissions (no longer also on files) - files inherit the folders permissions* • *If you grant the Networkservice write permissions to a folder the IUSR gets also write permissions (required when working with php and fastcgi impersonation)* • *GenerateSourceConfig.vbs now creates valid fileIDs (e.g. typo3 has files with special chars WIX would not like)* *added Property IIS7REQUIREDMODULES this comma separated list gets checked when the your msi is run on IIS7* *updated this documentation*

Table Of Contents

Introduction

Goal:

This document will show you how you can build a Windows Installer package for your Web Application within half an hour - by only utilizing open source software.

The following technologies are supported:
Scripting: ASP.NET / PHP / Perl
Databases: Microsoft SQL Server / MySQL Server

The Idea:

As you have experienced yourself developing a web application isn't that easy and quite time consuming. You spend a lot of hours thinking about software design, security problems and of course how to improve end user's usability.

On the other hand a lot of your users or potential customers have only a limited knowledge about software development or webserver configuration options. For them it's rather difficult to e.g. "add the appropriate access rights" on a folder or to setup a database. It's really sad that a lot of people fail or get annoyed by just installing a web application and of course that isn't the best first impression they should get from your software either.

Seeing an ever growing market for dedicated server systems or virtual private servers (VPS, VSERVER: http://en.wikipedia.org/wiki/Vps) starting at low prices and running the Windows Server 2003 operating system it is time to rethink if there are other installing methods available.

If your users have access to a (virtual) dedicated server system they can utilize remote desktop connections to manage it. They can and are used to install software the way they've learned to do it on their desktop PCs by just clicking trough a nice little wizard.

The nice little wizard we'll create in this document will be capable of:

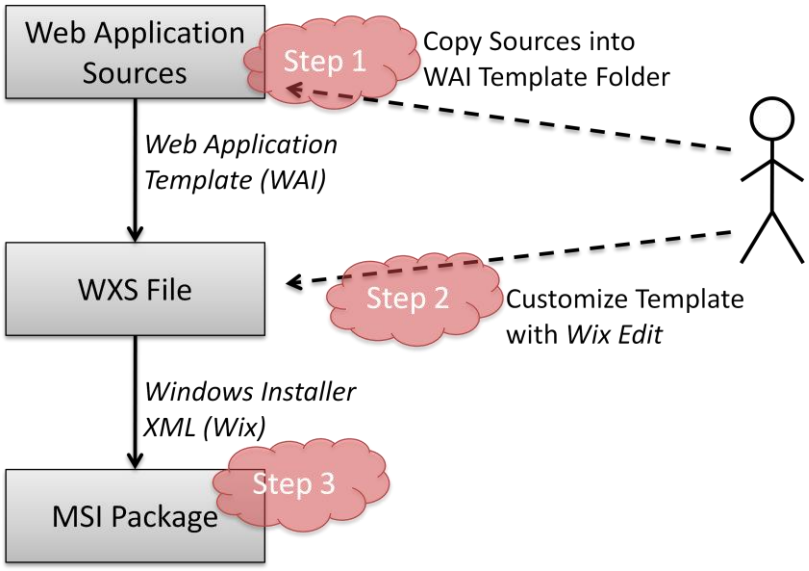
- *Creating a new website*
- *Adding a virtual directory to an existing website*
- *Creating an application pool and setting up a website/folder as ASP.NET / PHP / Perl application*
- *Creating a new Microsoft SQL or MySQL Database, Logins and Users*
- *Configuring your web application (e.g. web.config) and setting values collected in wizard mode*
- *A lot of user guidance, messages and validations*

Any feedback – good or bad – is highly appreciated… So please drop us a line. Tell us what application you've created an installer for or if you have experienced any problems with this documentation or the generated msi file itself. If you have any questions – we'll be glad to help (mailto:bfrank@microsoft.com).

Overview

The process of building your own installer package consists of 4 steps:

0. *Set up the working environment*
1. *Create an application template (*.wxs) for your web application (using GenerateConfig.bat)*
2. *Customize this application template (*.wxs) to fit your needs (using the WixEdit tool)*
3. *Generate a MSI package for your web application*



Step 0: Set up the working environment (Getting Started)

Before we can start you'll need the following utilities:

1. *WixEdit*
   Version used: 0.6.1762
   Filename: wixedit-0.6.1762.msi
   Note:Take the MSI package
   Url: http://wixedit.sourceforge.net/
2. *Windows Installer XML (Wix)*
   Version used: 3.0.2925.0
   Filename: wix-3.0.2925.0-binaries.zip
   Note: Take the binary zip
   Url: http://sourceforge.net/project/showfiles.php?group_id=105970&package_id=168888
3. *Web Application Template – What you have just downloaded*
   Version used: 2.0.6
   Filename: WAI-Installer.msi
   Note: Contains "Web Application Template" folder – which is our working directory
   URL: http://www.codeplex.com/wai

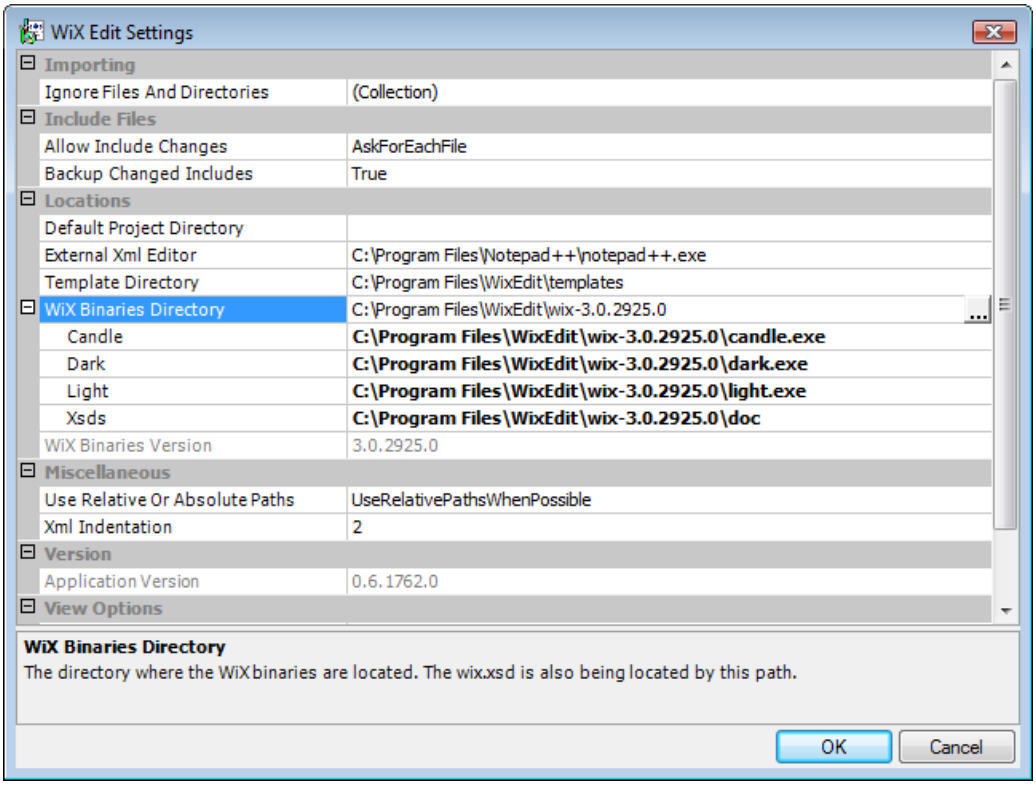
Putting it all together - steps:

1. *Install WixEdit with the provided installer package (e.g. "wixedit-0.6.1762.msi")*
2. *Extract "wix-3.0.2925.0-binaries.zip" into the WixEdit Folder like e.g.:*
   `C:\Program Files\WixEdit\wix-3.0.2925.0`
3. *Start WixEdit (*`Program Files -> WixEdit -> Binaries -> WixEdit.exe)`
4. *Tell WixEdit to use the new binaries because otherwise you'll encounter an bug with the ones shipped with the installation and the IIS configuration won't work:*

   `Tools -> Options -> WiX Binary Directory -> […]`

   Choose the new Folder you've extracted the binaries into, e.g.:

   `C:\Program Files\WixEdit\wix-3.0.2925.0`
   See the screenshot of Wix Edit:



Picture 1. The modified settings in Wix Edit pointing to the `wix-3.0.2925.0` folder

Optional: Change the '`External Xml Editor`' to an editor of your choice which can provide xml highlighting - for example notepad++ (See: http://notepad-plus.sourceforge.net/uk/site.htm)

5. *Run the WAI-Installer.msi – this installer extracts the web application template folder. Navigate to this folder as this is our working directory. After this you have successfully setup the working environment.*

Step 1: Create an application template (Preparations)

Now it's time to prepare your web application and to create an application template to work with:

1. *Go to the directory where the source files of the web application reside that you want to package into a MSI and copy all files into the web application template's "Sources" folder.*

2. *Execute GenerateConfig.bat*
   (It will just open a command shell and run the GenerateConfigSource.vbs file).

3. *This VBScript file will now ask you a few questions:*
   a. *What name should the installer file have. (e.g. MyWebApplication)*

   b. *Which folders should be writeable for the web server or to be more precise for the "network service" user and the IUSR account. You should enter folder names relative to the Sources directory. If you have – for example - the following structure*

      …\Web Application Template\Sources\App_Data
      …\Web Application Template\Sources\App_Data\posts
      …\Web Application Template\App_Data\images

      you would type in App_Data or App_Data\posts to get these Folder and all files and subfolders beneath writeable for the end users web server.

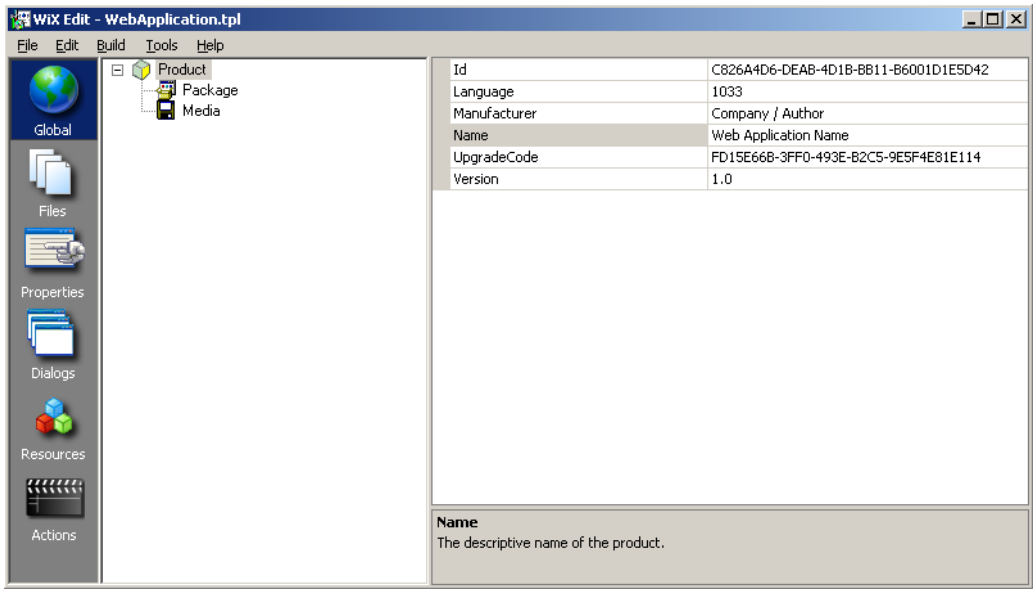
2. *When the VBScript is finished you'll find file a new .WXS File in the folder with the name you provided for the installer file*

3. *Open this file in WixEdit*

Please note that if you have a large web project the generated xml file is pretty big and it'll take some time until WixEdit has loaded and validated it

Step 2: Customize this application template (Customize)

A First Look Into WixEdit



*Picture 2. WixEdit's view on the *.wxs file generated in the previous steps*

WixEdit has 6 main categories to configure a Windows Installer Package:

1. ***Global***
   This section defines the basic attributes of your web application: like it's and your name for example, product version and description. You should take a look into the 'Product' and 'Package' Element and change the values.

2. ***Files***
   In this section all files, features and configuration settings are stored. Although WixEdit has it's own File/ Folder import function we've done it with the VBScript because the current WixEdit version has a small bug in this function adding a xml tag to the definition file which isn't supported anymore and you can get into serious trouble if you have some files with the same name in different folders. Another reason was that we wouldn't have gotten our installer package ready to go within 30 minutes. Web Applications normally have a lot more files and folders than desktop applications and you would have had to add permissions to every single file and folder and to map them to the corresponding feature. We'll come back to this section when we configure your web application after copying the files.

3. ***Properties***
   Here you have some configuration switches we'll discuss in the coming chapter.

4. ***Dialogs***
   This section lets you change the dialogs used by the installer so that you can add more fields or complete new dialogs.

5. ***Resources***

   Here you have all Resources used by the installer itself – for example the icons or images in the dialog boxes. All elements listed here come from the 'Scripts' and 'Binary' folders in the web application template folder. You should take a look into Binary\License.rtf (Has to be an RTF-File) or Binary\bannrbmp.bmp to have your own logo displayed during the installation and to Binary\myAppSQL.sql for SQL-Statements to create an initial database.

6. ***Actions***

   Nearly all elements listed in this section point to the scripts folder. These scripts are responsible for listing existing websites or checking provided database access data for example. All custom actions in this template are written with VBScript so that you can alter and review them with a simple editor.

About Dialogs, Options And Performed Checks

Dialogs during setup are for gathering required information from the user. What information you need and which dialogs you want to show to the user depends on your web application.

This Web Application Template comes with a set of predefined dialogs. You may want to influence the behavior of the dialogs like e.g. which dialogs should be displayed to the user and what system checks need to be done when installing your web application. This can be done by setting properties in the Properties category in WixEdit. Below is a list of the contained dialogs and the relevant properties.

Welcome dialog

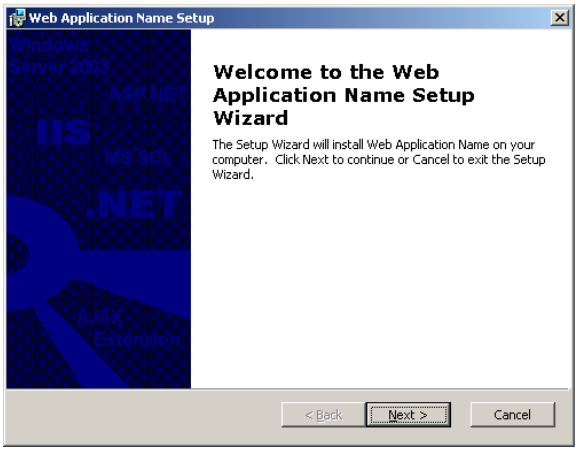
The first dialog.

Options:

- None

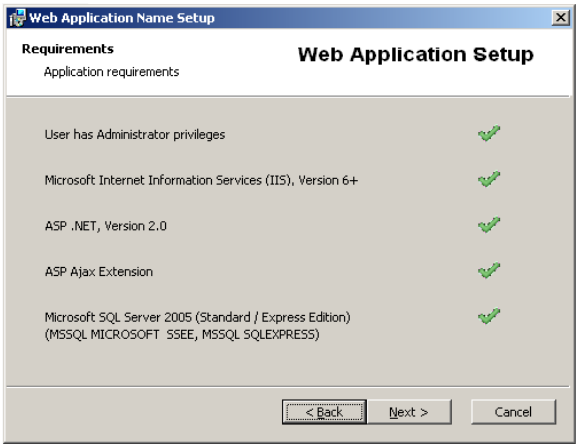
Checks:

- None



Requirements dialog

Gives the user feedback if the requirements for installing this web application are fulfilled. Which requirements are checked (and displayed) can be set by various properties.



Options (Properties):

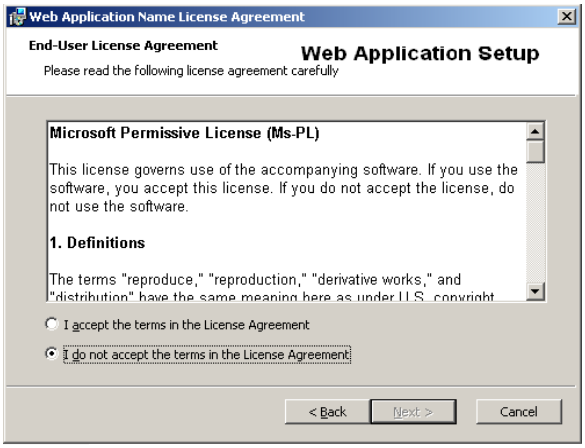
- SCRIPTLANGUAGE: ASP / PHP /PERL
- SCRIPTVERSIONMIN / SCRIPTVERSIONMAX
- PHPMODULES (e.g. odbc, mssql)
- CheckForAJAX = 0/1
- DATABASEENGINE: MSSQL / MySQL
- *CheckForSQLSERVER = 0/1*
- *IIS7REQUIREDMODULES (e.g. WMICompatibility,FastCgi)*

Checks:

- Privileged account
- IIS installed
- ASP.NET installed

License dialog

The license displayed here is taken from ..\Web Application Template\Binary\License.rtf file (needs to be RTF). Exchange with your license file.



Options (Properties):

- License text changeable by altering Binary\License.rtf (You have to keep the RTF-File Format)

Checks:

- The user has to accept the license in order to proceed

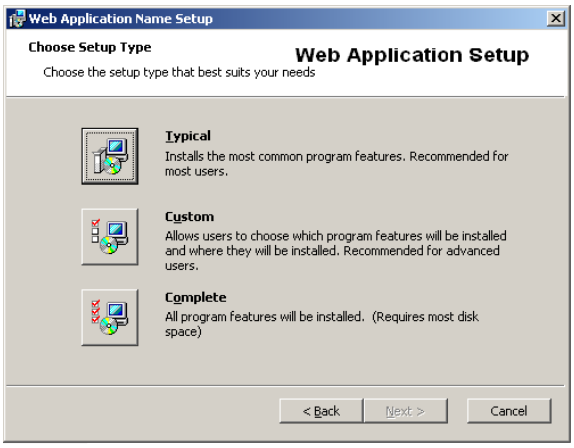
Setup type dialog

Useful if you want to give the user options of what parts to install.
Options (Properties):

- To enable this option use the ShowSetupTypeDlg = 0/1 property

Checks:

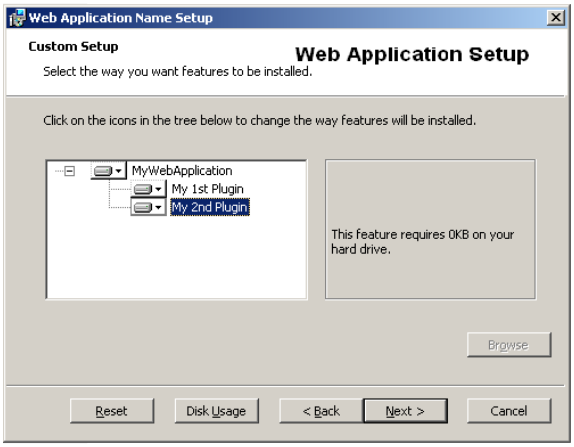
- None



Custom setup

Options (Properties):

- This dialog can only be reached through the setup type dialog -> 'Custom' so that this property has to be set to 1



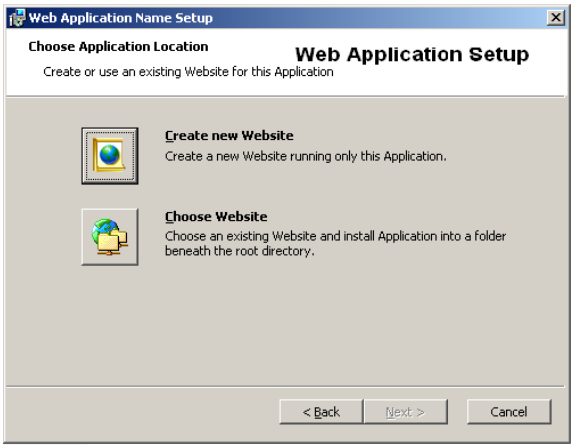
Target dialog

Options (Properties):

- If your application needs or should use a dedicated website for some reasons use the RequireNewWebsite = 1 property for this purpose

Checks:

- None



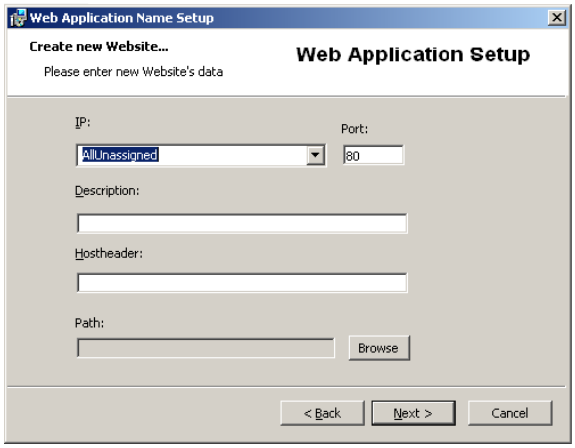
New website dialog

Options (Properties):

- No Options (Properties)

Checks:

- Valid port
- Valid description
- Valid hostheader
- Checking for conflicts (IP/Port /Hostheader) with existing websites
- Path
    - Questions if a new folder should be created
    - Warning if the chosen folder isn't empty
    - Warning if the chosen folder isn't beneath the IIS default website folder (e.g. C:\Inetpub)



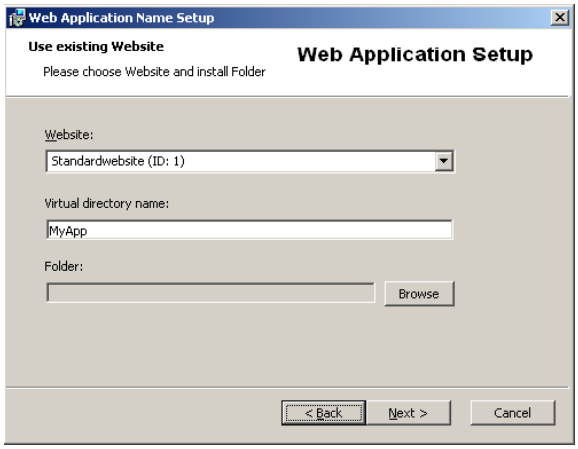
Existing website dialog

Options (Properties):

- Set the VIRTUALDIRECTORYNAME property for the proposed virtual directory name

Checks:

- Valid port
- Path
    - Questions if a new folder should be created
    - Warning if the chosen folder isn't empty
    - Warning if the chosen folder isn't beneath the chosen website



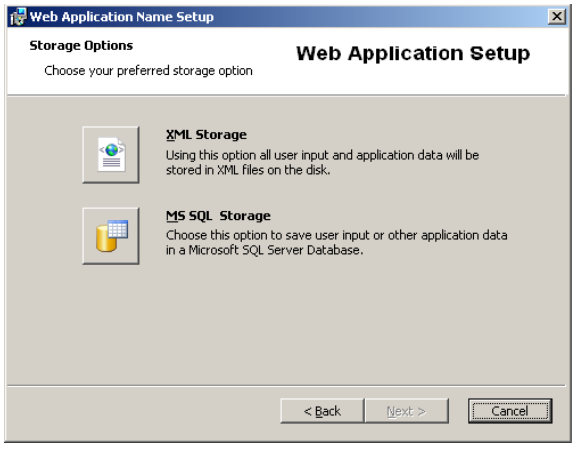
Storage Options (Properties) dialog

Options (Properties):

- To show this dialog set the ShowStorageOptions (Properties)Dlg = 0/1 property
- It won't show up if the RequireSQLDatabase property has been set to 1

Checks:

- None



Database setup I/II dialog

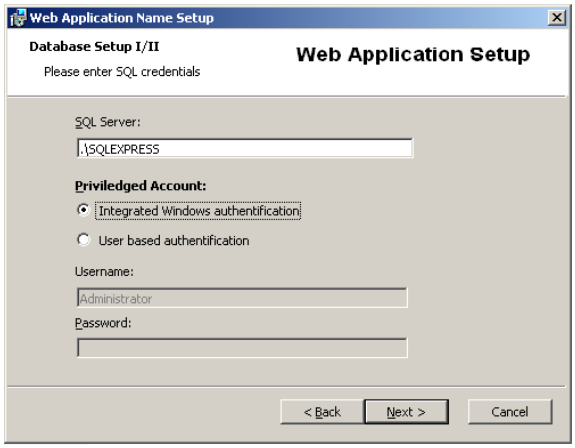
The user that is allowed to make changes to the database. (e.g. root for MySql)

Options (Properties):

- If your application needs an SQL database set RequireSQLDatabase property to 1
- If you have forced this dialog to show up the storage dialog won't be shown

Checks:

- Working connection
- User rights for creating a database, for creating a login and user
- Checking if the server is able to handle SQL user connects



(MSSQL Dialog – MySQL Dialog is similar)

Database setup II/II dialog

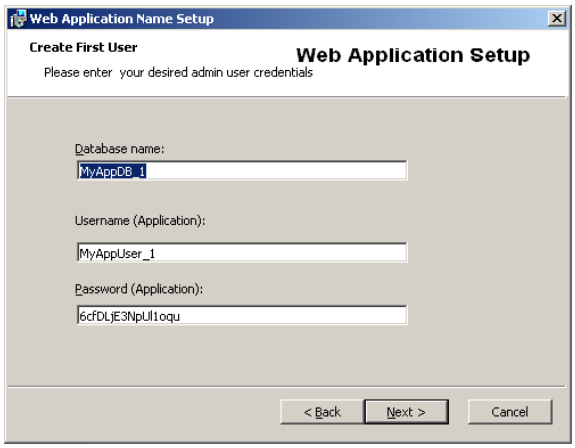
Used to create the user with which the web application connects to the database. (we do a prefill here.)

Options (Properties):

- Set the USRPrefix property to alter proposed username
- Set the DBPrefix property to alter proposed database name

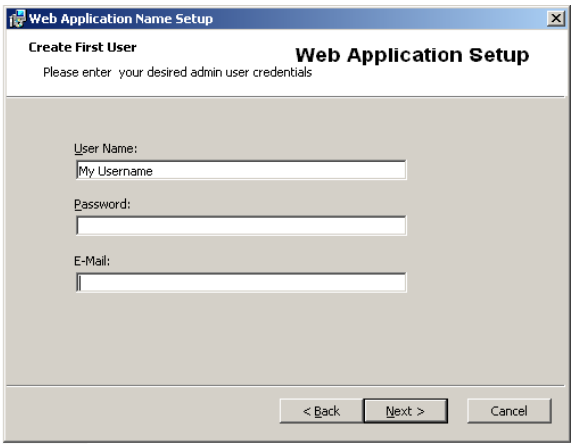
Checks:

- for existing databases and proposes a database name
- for existing users and proposes a username
- propose and validates password
- validates if the chosen database or username are already used



Create first user dialog

Used to create the (first) admin user of the web application (e.g. a CMS Web administrator)



Options (Properties):

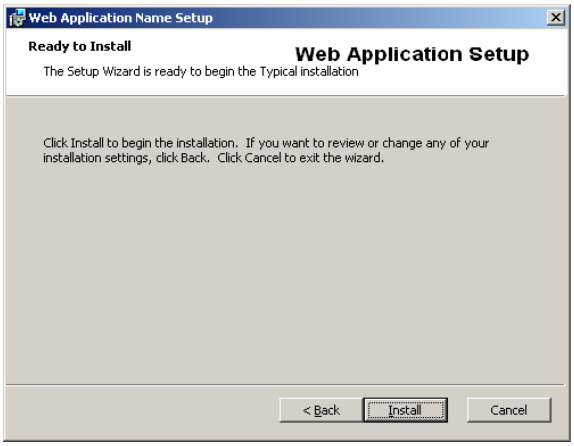
- To show this dialog set the ShowCreateFirstUserDlg property to 1
- If whitespaces are allowed for the username set the FirstUsernameWhitespaces property to 1

Checks:

- For valid username
- For valid password
- For valid email address

Verify ready dialog

Verify ready dialog. After this the installation starts.



Options (Properties):

- None

Checks:

- None

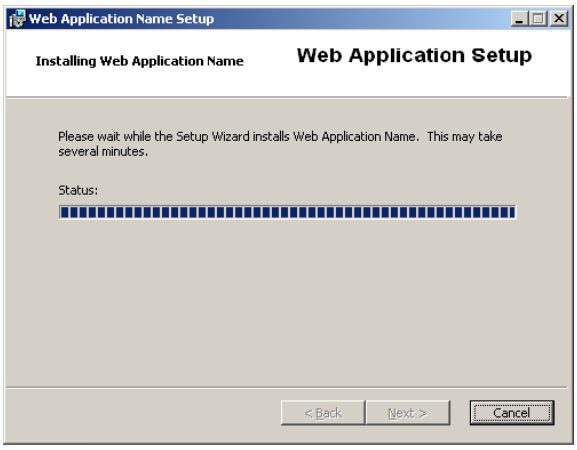
Progress dialog

Options (Properties):

- None

Checks:

- None



Exit dialog

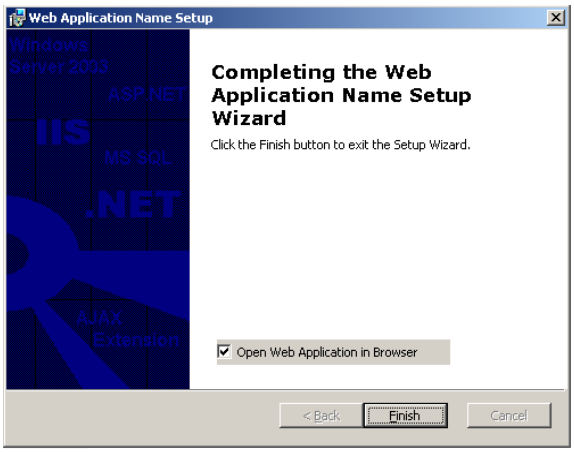
On the last dialog you have the possibility to navigate the user automatically to the just installed web application using the standard browser.

Options (Properties):

- To show the web launch option set the ShowWebsiteLaunchOption property to 1
- If the checkbox should be checked by default set the openurl property to yes
- If you want to route the user into a specific post setup directory change the STARTUPDIR property

Checks:

- None



Options (Properties) Overview

Properties change the way the installer behaves during install, e.g. what is being checked before your web application installs, which dialogs to show to the user. Here is the list of all the properties you can set. Use WixEdit -> click on properties (left) to modify the properties.

Property Name:	Purpose:	Values:
CheckForAJAX	(ASP.NET) checks if ASP.net AJAX extensions (V1) are installed. 1 : check is done 0: no check Note: if check is done and fails – user will be informed and cannot continue with installation.	0/1
CheckForSQLSERVER	1 : search for SQL Server 0: no check	0/1
checkMYSQL40	(MySQL only) 1 :makes sure that sql-mode="MYSQL40" (my.ini) is present 0: no check (default) Note: does a "select @@sql_mode" to verify that version 4 is used – few web apps require this (e.g. Typo3).	0/1
checkForFullTextSearch	(MSSQL Server only) 1 : checks if MSSQL Server instance running is capable to do full text search 0: no check	0/1
ShowSetupTypeDlg	1 : show this dialog 0: omit dialog	0/1
VIRTUALDIRECTORYNAME	Name of the virtual mapping directory	String (W3C url naming constraints)
RequireNewWebsite	If set to 1 – the web application can be installed only into a new website. 1 : shows only this dialog 0: shows also this dialog	0/1
WEBAPPLICATIONNAME	Used in IIS as web application name. (e.g. MyWebApplication)	String (MS IIS application naming constraints)
WEBAPPLICATIONPOOLNAME	Websites installed using this template are always associated with a new separate application pool (serving one w3wp.exe) Used in IIS as application pool	String (MS IIS application pool naming

	name *(e.g. MyWebApplicationPool)*	*constraints)*
ShowStorageOptions (Properties)Dlg	*1 : show this dialog* *0: omit dialog*	*0/1*
RequireSQLDatabase	*Set this property to 1 if your web application requires a SQL Database* *1 : show this dialog* *0: omit dialog*	*0/1*
DBPrefix	*Prefix for free database name proposal* *(e.g. MyWebApplicationDB_)*	*String (MS SQL Database naming constraints)*
USRPrefix	*Prefix for free database user name proposal* *(e.g. MyWebApplicationDBUser_)*	*String (MS SQL user/login naming constraints)*
ShowCreateFirstUserDlg	*1 : show this dialog* *0: omit dialog*	*0/1*
ShowWebsiteLaunchOption	*Show / hide launch checkbox for web-based post setup*	*0/1*
openurl	*Do you want the checkbox for launching the website on the last dialog.*	*yes/no*
STARTUPDIR	*Default startup directory for web-based post*	*String*
SCRIPTLANGUAGE	*Either ASP or PHP or Perl – determines which checks are performed and how the website / application is configured*	*Enum (ASP,PHP,PERL)*
SCRIPTVERSIONMIN/MAX	*For ASP.NET 2.0 for example:* *Min: 2.0.50727.1* *Max: 2.0.65535.65535*	*String*
DATABASEENGINE	*Checks, dialogs and setup procedures differ for MS SQL and MySQL databases*	*ENUM (MSSQL, MYSQL)*
MYSQLPrivileges	*Arguments for GRANT* *(e.g.: ALL or INSERT,SELECT,DELETE,UPDATE,ALTER)*	*String*
PHPMODULES	*Modules compiled in or added through php.ini file.* *Example:* *mssql for php_mssql.dll* *Use a comma separated list to check for more than one*	*String*

	extension: *odbc, mssql, gettext, mysql, mysqli*	
SCRIPTPARSE	*List of files to parsed for strings like e.g. @@SQLUSERUSERNAME@@. If strings like this are found they are replaced with a variable collected in wizard mode – see below for detailed explanation* *Use a comma separated list for several files. All files should be relative to the source folder. (e.g. config.file, lib/another.file)*	*String*
SETREADONLY	*List of files to whose file attribute should be set to Read-only (e.g. oscommerce likes "config.php" file to be set read-only)* *Use a comma separated list for several files. All files should be relative to the source folder. (e.g. config.file, lib/another.file)*	*String*
MD5PREFIX	*Prefix added before user's password in md5() function* *Alternatively give it the value "randomSalt" for a random salt value – e.g. osCommerce needs this.*	*String*
DBPORT	*Default MySQL database port* *E.g.: 3306*	*String*
checkIfIUSRisListedOnPHPUploadDir	*Some PHP applications require the IIS Anonymous User to have List Folder / Read permission for the php upload directory (e.g. 4images). When setting this value to 1 ->IUSR permissions for the upload directory are checked and a warning is displayed to the user if the permissions do not fit (default is 0)*	*0/1*
WANTinstalllogFile	*Set this value to 1 if you want a log file with the name "install.log" to be created during the install. This file contains short information about the things that have been done during the install (default is 1).* *Note: Don't set this to 1 if your web application already has a file named install.log in the web applications directory it'll probably get overwritten.*	*0/1*
IISVERSION	*If the installer is run on Vista or Windows Server 2008 this Property will contain the string "7"*	*e.g."7" on IIS7* *empty string on IIS6*

IIS7REQUIREDMODULES	*If the installer is run on Vista or Windows Server 2008 this Property will be checked automatically if the required modules are installed (use a comma separated list)* *this registry hive is checked: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\InetStp\Components* *http://learn.iis.net/page.aspx/135/discover-installed-components/* *Note: You should check at least for WMICompatibility as the installer needs it by itself*	*e.g. WMICompatibility, FastCgi, CGI, ASPNET, BasicAuthentication, WindowsAuthentication*

Configuring Your Web Application With Collected User Data

What we've done so far: We've included all files and added the right permissions if necessary and we've altered some properties so that we get the desired windows installer dialog sequence. During this install sequence we get a lot of user data we might want to use for modifying a web application's config file(s) during setup.

Here is a list of the user input we could use:

Name:	***Description:***
WEBSITEIP	*IP of the website, either someting like 192.168.0.1 or ‚*' for all unassigned*
WEBSITEPORT	*Port of the website, e.g. 80*
WEBSITEDESCRIPTION	*Description of the website as listed in IIS configuration*
WEBSITEHOSTHEADER	*Hostheader of the website, e.g. www.domain.tld or empty*
TARGETDIR	*Installation directory*
VIRTUALDIRECTORYNAME	*Name of the virtual directory mapping*
SQLDATABASE	*Configured SQL database name*
DBHOST	*Database hostname*
SQLUSERUSERNAME	*Configured SQL user name*
SQLUSERPASSWORD	*Configured SQL user password*

USERUSERNAME	*First user name*
USERPASSWORD	*First user name's password*
USERMD5PASSWORD	*First user name's password md5 encrypted*
USEREMAIL	*First user name's email*
URLHOST	*Hostname of the Homepage (e.g. http://www.codeplex.com)*
URLDIR	*Directory of the page after the hostname (e.g. subfolder/folder)*
DBHOSTANDPORT	*Will combine DBHOST and DBPORT (DBHOST:DBPORT) if DBHOST is an IP address or only localhost if DBHOST is localhost*

XML Example (e.g.: web.config)

This is an example of how to use WIX to set a connection string in a web.config file during setup

(Also valid for other XML files)

Structure of the web.config file:

```
<configuration>
.
.
.
  <connectionStrings>
    <add name="SiteSqlServer" connectionString="Data Source=.\SQLExpress;Integrated
Security=True;User Instance=True;AttachDBFilename=|DataDirectory|Database.mdf;"
providerName="System.Data.SqlClient" />
.
.
.
```

We want to change the value of connectionString. Wix gives us the possibility to navigate to an xml element through an xpath expression to get to ours the xpath query would be:

```
/configuration/connectionStrings/add[@name="SiteSqlServer"]
```

1. *Which values do we want to add:*

Property	Meaning	Example
DBHOST	*Server and Instance name of your sql server (see Database setup I/II dialog)*	*".\sqlexpress"*
SQLDATABASE	*Name of the web application database (see Database setup I/II dialog)*	*"MyWebapplicationDB_1"*
SQLUSERUSERNAME	*The user with which your web application connects to the sql server (see Database setup II/II dialog)*	*"MyWebapplicationUser"*
SQLUSERPASSWORD	*The users password*	*"0815blabla"*

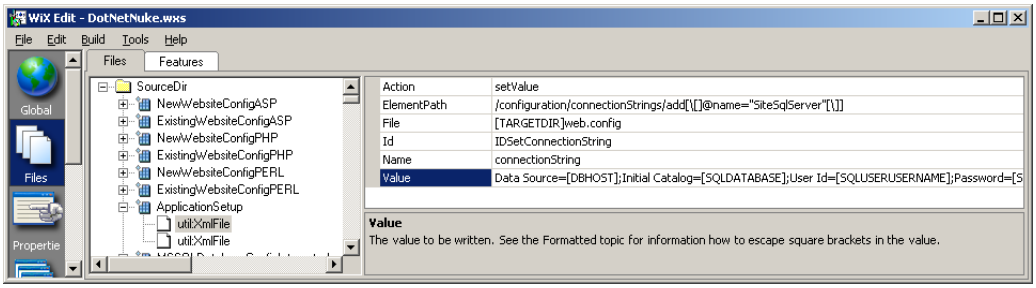
2. *The connection we enter in WixEdit should look like:*

```
Data Source=[DBHOST];Initial Catalog=[SQLDATABASE];User
Id=[SQLUSERUSERNAME];Password=[SQLUSERPASSWORD];
```

When the installer runs the [PROPERTY] gets filled with the actual value

Please note that all variables which should be parsed have to be encapsulated by square brackets (e.g.: [SQLDATABASE])

a. *Adding it as configuration command in wixedit*



b. *Go to 'Files'*

c. *Select the ApplicationSetup Component -> Select 'New' -> Select 'utilXmlFile'*

d. *Select the utilXmlFile Element*

i. *Set **'Action'** to 'setValue'*

ii. *Set **'ElementPath'** to:*
`/configuration/connectionStrings/add[\[]@name="SiteSqlServer"[\]]`
Please note that we need to escape the right and left square bracket in WixWdit.

iii. *The **'File'** should always be relative to the target dir as you can see in the screenshot: [TARGETDIR]web.config or [TARGETDIR]App_Data/somefile.xml*

iv. ***'Id'** is the Windows Installer Identifier for this element you can choose as you like*

v. ***'Name' defines the xml attribute we want to change**. You'll have to add this element using right click in the left window and choose 'New' -> 'Name'. Set the **'Name'** to `connectionStrings`*

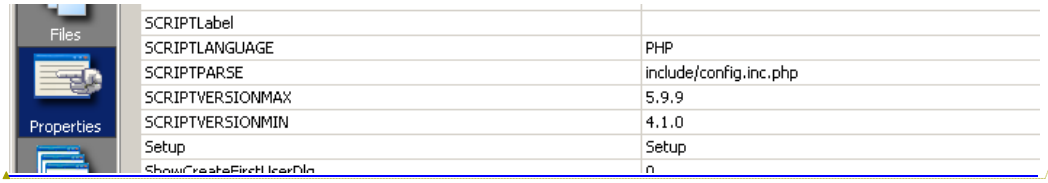
vi. *Modify the **'Value'** element to hold your connection string. Set **'Value'** to*

```
Data Source=[DBHOST];Initial Catalog=[SQLDATABASE];User
Id=[SQLUSERUSERNAME];Password=[SQLUSERPASSWORD];
```

Unstructed File Example (e.g.: config.php)

The Windows Installer technology has built in features to alter .INI or XML-Files which are great but won't help us if we want exchange some variables in a .PHP-File (or other ini files) for example. For thiscase you'll find a SCRIPTPARSE property in the template. Add a comma delimited string with all files into that property which need to be altered. This could be something like config.php, lib/settings.php, admin/settings/something.inc.php… I guess you get the idea… Specify the files relative to the "Source" folder.

Sample: If I would like to modify e.g. a file named "config.inc.php". This file lies in ..\Web Application Template\Sources\include I give the SCRIPTPARSE property the value "include\config.inc.php".



Now in the config.inc.php" file I escape the parameters that I would like to be filled:

Old:

```
…
$database_host = "put-you-host-in-here"
$database_user = "your-username"
…
```

New:

```
…
$database_host = "@@DBHOST@@"
$database_user = "@@SQLUSERUSERNAME@@"
…
```

Every string in these files which equals @@ a varibable from the "colleteded user data" list and an ending @@ is changed into the value collected during installation.

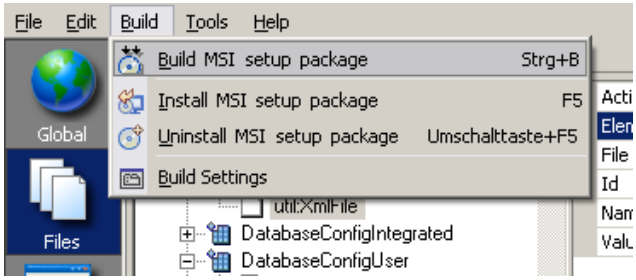
***Formatiert:** Schriftart: Calibri, 10 pt, Nicht Fett, Nicht unterstrichen, Schriftartfarbe: Automatisch*

SQL Database Setup

Add create queries for tables and initial data inserts in Binary/MyApp.sql. This SQL script is executed against the database during setup (Database and User configuration is done by WIX).

You can also use @@PARAM@@ variables there which will be parsed before execution (see example above).

Step 3: Generate a MSI package for your web application (Build)



Before we can build the MSI Package we take a look at

```
Build -> Build Settings
```

and check if the compiler Options (Properties) (for light.exe and candle.exe) are correct:

Options (Properties) for Candle.exe:

```
"<projectfile>" -ss -out "<projectname>.wixobj" <extensions>
```

Options for Light.exe

```
"<projectname>.wixobj" -cultures:en-us -out "<projectname>.msi" <extensions>
```

Now we'll build the MSI Package using

```
Build -> Build MSI setup package
```

and find a new .msi file in our web application template folder. Congratulations!

Creating an Installer for x64

Why can't I run the installer on 64bit windows?

Some checks that are done in the beginning to gather data, e.g. IIS7 required modules, PHP and Fastcgi Settings some of these settings are in registry (HKEY_LOCAL_MACHINE\SOFTWARE\) or on the file system (e.g. in system32). If you would run the x86 msi on a x64 OS these paths get masqueraded (e.g. HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node or SysWOW64) by the OS , i.e. the check would fail (because it is looking at the wrong place).

The cure: You need to run those checks in 64bit mode.

The process:

1. *Create your installer as before for the x86 platform (e.g. MyWebapplication.msi)*
2. *Test it - if it works fine, run the x64 config script generator against it, e.g.:*

```
Buildx64Config.vbs "MyWebapplication.wxs" "C:\Program Files\Windows Installer XML v3\bin"
```

Whereas the usage is:

```
Buildx64Config.vbs [*.wxs file] [path to WIX binaries(candle.exe & light.exe)]")
```

Getting Further Information

As already mentioned WiX is capable of doing a lot of things and if you want to extend or change the functions in the provided template you should check out the following resources:

Websites:

- ***Web application installer***
  http://www.codeplex.com/wai

- *WiX tutorial*
  http://www.tramontana.co.hu/wix/

- *Windows Installer Reference*
  http://msdn2.microsoft.com/en-us/library/Aa372860.aspx

Mailing lists:

- *WiX-Users / Windows Installer XML (WiX) Toolset*
  http://www.mail-archive.com/wix-users@lists.sourceforge.net/