uGoLive
Check Developers Guide

Umbraco // The Friendly CMS

Contents

1 Introduction 3

2 Getting Set Up 4

2.1 System Requirements 4

2.1.1 uGoLive for Umbraco 4.x 4

2.1.2 uGoLive for Umbraco 5.x 4

3 Creating a Check 5

3.1 Making your check Rectifiable 7

4 Deploying Your Checks 8

5 Alternative Base Classes 9

5.1 AbstractAppSettingCheck 9

5.2 AbstractConfigCheck 10

5.3 AbstractPathCheck 11

6 Useful Links 12

1 Introduction

uGoLive is an Umbraco dashboard control which allows you to run a series of checks on your website in order to prepair it for launch. **uGoLive** comes complete with the most fundamental checks to ensure your site is as secure as possible, however **uGoLive** is completely pluggable so can be easily be extended to include your own checks. In this way, you will be able to build up your own set of checks that you can test your website against each and every launch. No more forgetting whether you remembered to update a setting, run the check and **uGoLive** will tell you.

The aim of this guide is to detail how you would go about creating your very own **uGoLive** checks.

2 Getting Set Up

2.1 System Requirements

Before you get started, there are a number of things you will need:

2.1.1 uGoLive for Umbraco 4.x

1. .NET 3.5+
2. Visual Studio 2008/2010
3. Umbraco 4.5.x
4. The **uGoLive** DLL
   (Our.Umbraco.uGoLive.dll)
5. Any 3rd party library DLLs

2.1.2 uGoLive for Umbraco 5.x

6. .NET 4+
7. Visual Studio 2010
8. Umbraco 5.1.x
9. The **uGoLive** DLL
   (Our.Umbraco.uGoLive.dll)
10. Any 3rd party library DLLs

3 Creating a Check

Creating a check is actually very simple. Each check is a single class that extends the **AbstractCheck** class found in the **uGoLive** dll *(Our.Umbraco.uGoLive.Checks.AbstractCheck).*

Example:

```
1.  import Our.Umbraco.uGoLive.Checks;
2.
3.  public class MyCustomCheck : AbstractCheck
4.  {
5.      ...
6.  }
```

In addition to extending the **AbstractCheck** class, each check must be attributed with the **CheckAttribute** also found in the **uGoLive** dll *(Our.Umbraco.uGoLive.Attribution.CheckAttribute).*

Example:

```
1.  import Our.Umbraco.uGoLive.Checks;
2.  import Our.Umbraco.uGoLive.Attribution;
3.
4.  [Check("79584689-EC78-448E-9D88-57D72D388283", "My Custom Check",
    "My Custom Check Description", "My Custom Checks")]
5.  public class MyCustomCheck : AbstractCheck
6.  {
7.      ...
8.  }
```

The **CheckAttribute** has three required parameters, and one optional parameter as follows:

Member	Type	Description
ID	Guid	A unique ID for the check. This should be a string representation of a guid.
Name	String	A friendly name for the check which will be used as the checks label. **NB**: All checks are displayed in alphabetical order.
Description	String	A description for the check which will be used as the checks description field below the checks label. These should generally explain what issue the check aims to rectify.
Group (Optional)	String	The name of the group the check belongs to. This is used to visually group similar checks together. **NB**: All groups are displayed in alphabetical order.

Each check must then implement a required **Check** method returning a **CheckResult** object.

Example:

```
1.  import Our.Umbraco.uGoLive.Checks;
2.  import Our.Umbraco.uGoLive.Attribution;
3.
4.  [Check("79584689-EC78-448E-9D88-57D72D388283", "My Custom Check",
    "My Custom Check Description", "My Custom Checks")]
5.  public class MyCustomCheck : AbstractCheck
6.  {
7.      public CheckResult Check(){
8.          ...
9.      }
10. }
```

The **CheckResult** class is a simple class and has one required property, and one optional property as follow:

Member	Type	Description
Status	CheckResultStatus	The status determines whether the check has passed or failed, and can be a value of **Passed**, **Failed** or **Indeterminate**. Depending on the status, one of three icons will be displayed. A **tick**, a **cross** or a **warning symbol**. If a check has passed, and the check is a rectifiable check, then its associated rectify icon will be disabled. See Making your checks Rectifiable for more info.
Message (Optional)	String	A message to display next to the check status icon.

The **Check** method is where you put your code to perform the actual check. It is up to you how simple of complex your check will be. The only requirement is that you must return a **CheckResult** object as a response.

3.1 Making your check Rectifiable

Whilst everything so far is all you need to create a check, it is also possible to make your checks rectifiable. A rectifiable check is a check that is capable of automatically rectifying itself in the event of a **Failed** or **Indeterminate CheckResult** status.

To make a check rectifiable, you must override the base **CanRectify** boolean property returning a **true** value, and the base **Rectify** method returning a **RectifyResult** object.

Example:

```
1.  import Our.Umbraco.uGoLive.Checks;
2.  import Our.Umbraco.uGoLive.Attribution;
3.
4.  [Check("79584689-EC78-448E-9D88-57D72D388283", "My Custom Check",
    "My Custom Check Description", "My Custom Checks")]
5.  public class MyCustomCheck : AbstractCheck
6.  {
7.     public CheckResult Check(){
8.          ...
9.     }
10.
11.     public RectifyResult Rectify(){
12.         ...
13.     }
14.
15.     public bool CanRectify { get { return true; } }
16. }
```

The **RectifyResult** class is a simple class and has one required property, and one optional property as follow:

Member	Type	Description
Status	RectifyResultStatus	The status determines whether the rectify attempt was successful or not, and can be a value of **Success** or **Failed**. Depending on the status, one of two icons will be displayed. A **tick** or a **cross**.
Message (Optional)	String	A message to display next to the rectify status icon.

The **Rectify** method is where you put your code to perform the actual rectification. It is up to you how you implement this method. The only requirement is that you must return a **RectifyResult** object as a response.

4 Deploying Your Checks

Deploying your checks is a very simple process and is just a case of deploying your compiled dll to the **bin** directory of your website. After your application has restarted, any visit to the **uGoLive** dashboard will automatically populate the **uGoLive** plugins internal checks collection with all check classes found in any of the dlls within the **bin** directory.

NB: When creating checks for use in Umbraco 5, please make sure to add the AssemblyContainsPlugins attribute to your AssemblyInfo.cs file to ensure your checks are picked up.

5 Alternative Base Classes

So far we have discussed how to create checks by extending the **AbstractCheck** class. In addition to this base class, **uGoLive** also ships with a number of helper abstract classes for common types of checks.

5.1 AbstractAppSettingCheck

The **AbstractAppSettingCheck** is a base class for checking the existence or none existence of a single **AppSetting**.

The **AbstractAppSettingCheck** has three required properties and three optional properties as follows:

Member	Type	Description
Key	String	The AppSetting key to check.
Value	String	The value to compare the AppSetting against.
ValueComparisonType	ValueComparisonType	A **ValueComparisonType** enum identifying the type of comparison to perform. Can be one of **ShouldEqual** or **ShouldNotEqual**.
CheckPassedMessage (Optional)	String	A message to display should the check pass. If omitted, will display a generic message.
CheckFailedMessage (Optional)	String	A message to display should the check fail. If omitted, will display a generic message.
RectifySuccessMessage (Optional)	String	A message to display should the rectify attempt be successfull. If omitted, will display a generic message.

If the **ValueComparisonType** is of type **ShouldEqual**, then the **AbstractAppSettingCheck** can also perform a rectify to update the **AppSetting** should the check fail.

5.2 AbstractConfigCheck

The **AbstractConfigCheck** is a base class for checking the existence or none existence of a node or attribute value within an **XML** config file.

The **AbstractConfigCheck** has four required properties and three optional properties as follows:

Member	Type	Description
FilePath	String	The file path of the config file. Should be in the format **"~/folder/file.config"**
XPath	String	The XPath to the node / attribute to compare.
Value	String	The value to compare against.
ValueComparisonType	ValueComparisonType	A **ValueComparisonType** enum identifying the type of comparison to perform. Can be one of **ShouldEqual** or **ShouldNotEqual**.
CheckPassedMessage (Optional)	String	A message to display should the check pass. If omitted, will display a generic message.
CheckFailedMessage (Optional)	String	A message to display should the check fail. If omitted, will display a generic message.
RectifySuccessMessage (Optional)	String	A message to display should the rectify attempt be successfull. If omitted, will display a generic message.

If the **ValueComparisonType** is of type **ShouldEqual**, then the **AbstractConfigCheck** can also perform a rectify to update the node / attribute should the check fail.

5.3 AbstractPathCheck

The **AbstractPathCheck** is a base class for checking the existence or none existence of a file or directory path on the web server.

The **AbstractPathCheck** has four required properties and three optional properties as follows:

Member	Type	Description
Path	String	The file / directory path to compare against. Should be in the format **"~/folder/file.config"**
PathComparisonType	PathComparisonType	A **PathComparisonType** enum identifying the type of comparison to perform. Can be one of **ShouldExist** or **ShouldNotExist**.
CheckPassedMessage (Optional)	String	A message to display should the check pass. If omitted, will display a generic message.
CheckFailedMessage (Optional)	String	A message to display should the check fail. If omitted, will display a generic message.
RectifySuccessMessage (Optional)	String	A message to display should the rectify attempt be successfull. If omitted, will display a generic message.
RectifyFailedMessage (Optional)	String	A message to display should the rectify attempt fail. If omitted, will display a generic message.

If the **PathComparisonType** is of type **ShouldNotExist**, then the **AbstractPatchCheck** can also perform a rectify to remove the file / directory should the check fail.

6 Useful Links

- **uGoLive Source**
  https://bitbucket.org/mattbrailsford/ugolive

- **uGoLive Our Umbraco Project Page**
  http://our.umbraco.org/projects/backoffice-extensions/ugolive

- **Blog**
  http://blog.mattbrailsford.com

- **Twitter Hash Tag**
  #uGoLive