C# Snippets 2013

http://msdn.microsoft.com/library

FTP Actions

Downloading files with FTP

http://msdn.microsoft.com/en-us/library/ms229711.aspx

```
using System;
using System.IO;
using System.Net;
using System.Text;

namespace Examples.System.Net
{
   public class WebRequestGetExample
   {
        public static void Main ()
        {
            // Get the object used to communicate with the server.
            FtpWebRequest request =
(FtpWebRequest)WebRequest.Create("ftp://www.contoso.com/test.htm");
            request.Method = WebRequestMethods.Ftp.DownloadFile;

            // This example assumes the FTP site uses anonymous logon.
            request.Credentials = new NetworkCredential
("anonymous","janeDoe@contoso.com");

            FtpWebResponse response = (FtpWebResponse)request.GetResponse();

            Stream responseStream = response.GetResponseStream();
            StreamReader reader = new StreamReader(responseStream);
            Console.WriteLine(reader.ReadToEnd());

            Console.WriteLine("Download Complete, status {0}",
response.StatusDescription);

            reader.Close();
            response.Close();
        }
   }
}
```

How to: Upload Files with FTP

http://msdn.microsoft.com/en-us/library/ms229715.aspx

```
using System;
using System.IO;
using System.Net;
using System.Text;

namespace Examples.System.Net
{
   public class WebRequestGetExample
```

```
   {
       public static void Main ()
       {
           // Get the object used to communicate with the server.
           FtpWebRequest request =
(FtpWebRequest)WebRequest.Create("ftp://www.contoso.com/test.htm");
           request.Method = WebRequestMethods.Ftp.UploadFile;

           // This example assumes the FTP site uses anonymous logon.
           request.Credentials = new NetworkCredential
("anonymous","janeDoe@contoso.com");

           // Copy the contents of the file to the request stream.
           StreamReader sourceStream = new StreamReader("testfile.txt");
           byte [] fileContents =
Encoding.UTF8.GetBytes(sourceStream.ReadToEnd());
           sourceStream.Close();
           request.ContentLength = fileContents.Length;

           Stream requestStream = request.GetRequestStream();
           requestStream.Write(fileContents, 0, fileContents.Length);
           requestStream.Close();

           FtpWebResponse response = (FtpWebResponse)request.GetResponse();

           Console.WriteLine("Upload File Complete, status {0}",
response.StatusDescription);

           response.Close();
           }
       }
   }
}
```

How to: List Directory Contents with FTP

http://msdn.microsoft.com/en-us/library/ms229716.aspx

```
using System;
using System.IO;
using System.Net;
using System.Text;

namespace Examples.System.Net
{
   public class WebRequestGetExample
   {
       public static void Main ()
       {
           // Get the object used to communicate with the server.
           FtpWebRequest request =
(FtpWebRequest)WebRequest.Create("ftp://www.contoso.com/");
           request.Method = WebRequestMethods.Ftp.ListDirectoryDetails;

           // This example assumes the FTP site uses anonymous logon.
           request.Credentials = new NetworkCredential
("anonymous","janeDoe@contoso.com");

           FtpWebResponse response = (FtpWebResponse)request.GetResponse();

           Stream responseStream = response.GetResponseStream();
           StreamReader reader = new StreamReader(responseStream);
           Console.WriteLine(reader.ReadToEnd());

           Console.WriteLine("Directory List Complete, status {0}",
response.StatusDescription);

           reader.Close();
           response.Close();
       }
   }
}
```

Downloading All Files in Directory FTP and C#

Example 1:

http://social.msdn.microsoft.com/Forums/en/ncl/thread/079fb811-3c55-4959-85c4-677e4b20bea3

For being able to download all files from a FTP directory to a local folder, you will have to list all files in the remote directory and then download them one by one. You can use the following code to do the same:

```
      string[] files = GetFileList();
       foreach (string file in files)
       {
           Download(file);
       }

       public string[] GetFileList()
       {
           string[] downloadFiles;
           StringBuilder result = new StringBuilder();
           WebResponse response = null;
           StreamReader reader = null;
           try
           {
               FtpWebRequest reqFTP;
               reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" +
ftpServerIP + "/"));
               reqFTP.UseBinary = true;
               reqFTP.Credentials = new NetworkCredential(ftpUserID,
ftpPassword);
               reqFTP.Method = WebRequestMethods.Ftp.ListDirectory;
               reqFTP.Proxy = null;
               reqFTP.KeepAlive = false;
               reqFTP.UsePassive = false;
               response = reqFTP.GetResponse();
               reader = new StreamReader(response.GetResponseStream());
               string line = reader.ReadLine();
               while (line != null)
               {
                   result.Append(line);
                   result.Append("\n");
                   line = reader.ReadLine();
               }
               // to remove the trailing '\n'
               result.Remove(result.ToString().LastIndexOf('\n'), 1);
               return result.ToString().Split('\n');
           }
           catch (Exception ex)
           {
               if (reader != null)
               {
                   reader.Close();
               }
               if (response != null)
               {
                   response.Close();
               }
               downloadFiles = null;
               return downloadFiles;
           }
       }

       private void Download(string file)
       {
           try
           {
               string uri = "ftp://" + ftpServerIP + "/" + remoteDir + "/" +
file;
               Uri serverUri = new Uri(uri);
```

```
                if (serverUri.Scheme != Uri.UriSchemeFtp)
                {
                    return;
                }
                FtpWebRequest reqFTP;
                reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" +
ftpServerIP + "/" + remoteDir + "/" + file));
                reqFTP.Credentials = new NetworkCredential(ftpUserID,
ftpPassword);
                reqFTP.KeepAlive = false;
                reqFTP.Method = WebRequestMethods.Ftp.DownloadFile;

                reqFTP.UseBinary = true;
                reqFTP.Proxy = null;
                reqFTP.UsePassive = false;
                FtpWebResponse response = (FtpWebResponse)reqFTP.GetResponse();
                Stream responseStream = response.GetResponseStream();
                FileStream writeStream = new FileStream(localDestnDir + "\" +
file, FileMode.Create);
                int Length = 2048;
                Byte[] buffer = new Byte[Length];
                int bytesRead = responseStream.Read(buffer, 0, Length);

                while (bytesRead > 0)
                {
                    writeStream.Write(buffer, 0, bytesRead);
                    bytesRead = responseStream.Read(buffer, 0, Length);
                }
                writeStream.Close();
                response.Close();
            }
            catch (WebException wEx)
            {
                MessageBox.Show(wEx.Message, "Download Error");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Download Error");
            }
        }
```

Selenium Components

Wait Until Element Present

Using the solution provided by Mike Kwan may have an impact in overall testing performance, since the implicit wait will be used in all FindElement calls. Many times you'll want the FindElement to fail right away when an element is not present (you're testing for a malformed page, missing elements, etc.). With the implicit wait these operations would wait for the whole timeout to expire before throwing the exception.

I've written a little extension method to to IWebDriver that adds a timeout (in seconds) parameter to the FindElement() method. It's quite self-explanatory:

```
public static class WebDriverExtensions
{
    public static IWebElement FindElement(this IWebDriver driver, By by, int
timeoutInSeconds)
    {
        if (timeoutInSeconds > 0)
        {
            var wait = new WebDriverWait(driver,
TimeSpan.FromSeconds(timeoutInSeconds));
```

```
            return wait.Until(drv => drv.FindElement(by));
        }
        return driver.FindElement(by);
    }
}
```

I didn't cache the WebDriverWait object as its creation is very cheap, this extension may be used simultaneously for different WebDriver objects, and I only do optimizations when ultimately needed.

Usage is straight-forward:

```
var driver = new FirefoxDriver();
driver.Navigate().GoToUrl("http://localhost/mypage");
var btn = driver.FindElement(By.CssSelector("#login_button"));
btn.Click();
var employeeLabel = driver.FindElement(By.CssSelector("#VCC_VSL"), 10);
Assert.AreEqual("Employee", employeeLabel.Text);
driver.Close();
```

Wait Until Element Present: Multiple Elements

Here's a variation of @Loudenvier's solution that also works for getting multiple elements:

```
public static class WebDriverExtensions
{
    public static IWebElement FindElement(this IWebDriver driver, By by, int
timeoutInSeconds)
    {
        if (timeoutInSeconds > 0)
        {
            var wait = new WebDriverWait(driver,
TimeSpan.FromSeconds(timeoutInSeconds));
            return wait.Until(drv => drv.FindElement(by));
        }
        return driver.FindElement(by);
    }

    public static ReadOnlyCollection<IWebElement> FindElements(this IWebDriver
driver, By by, int timeoutInSeconds)
    {
        if (timeoutInSeconds > 0)
        {
            var wait = new WebDriverWait(driver,
TimeSpan.FromSeconds(timeoutInSeconds));
            return wait.Until(drv => (drv.FindElements(by).Count > 0) ?
drv.FindElements(by) : null);
        }
        return driver.FindElements(by);
    }
}
```

Selenium WebDriver: Explicit Waits

http://seleniumhq.org/docs/04_webdriver_advanced.jsp

```
IWebDriver driver = new FirefoxDriver();
driver.Url = "http://somedomain/url_that_delays_loading";
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
IWebElement myDynamicElement = wait.Until<IWebElement>((d) =>
    {
        return d.FindElement(By.Id("someDynamicElement"));
    });
```

This waits up to 10 seconds before throwing a TimeoutException or if it finds the element will return it in 0 - 10 seconds. WebDriverWait by default calls the ExpectedCondition every 500

milliseconds until it returns successfully. A successful return is for ExpectedCondition type is Boolean return true or not null return value for all other ExpectedCondition types.

This example is also functionally equivalent to the first Implicit Waits example.

Selenium WebDriver Implicit Waits

```
WebDriver driver = new FirefoxDriver();
driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(10));
driver.Url = "http://somedomain/url_that_delays_loading";
IWebElement myDynamicElement = driver.FindElement(By.Id("someDynamicElement"));
```