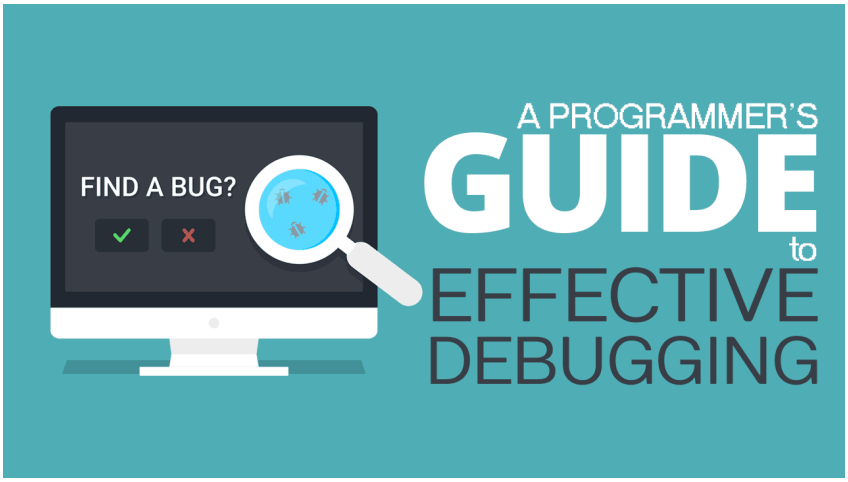
# General

## Developers guide to debugging

(click on the image below)




- When dealing with objects, always check that they exist and content/elements are available

```
public IEnumerable<exSample> GetSamples(SignifyHRDAL dbContext)
{
 var samples = SampleHelper.GetSampleList();

 if (samples != null && samples.Count > 0)
 {
  //Do something
 }

 return samples;
}
```

- Format dates according to system standards (use SignifyTypeExtensions)
- Enums are singular
- Method names make sense

- Use the var keyword instead of long namespace.object.names if the type that is returned is clear from the variable initialisation.

```
public void UpdateSample()
{
  //Bad
  bool isActive = false;
  //Good
  var isActive = false;

  //Bad
  string sampleDescription = "This is the new description";
  //Good
  var sampleDescription = "This is the new description";
}
```

- Removed the old author from the method. Tracking of who made the change can be done in Git.
- Use object initializer instead of empty constructor, when setting properties.

```
public class Cat
{
    // Auto-implemented properties.
    public int Age { get; set; }
    public string Name { get; set; }

    public Cat()
    {
    }

    public Cat(string name)
    {
        this.Name = name;
    }
}
```

- Ensure that null checks are performed and handled, where necessary.
- Remember to remove unnecessary code.
- Lambda expressions – variables should make sense. Abbreviations can be used as long as it makes sense.
- Is foreach used in preference to the for(int i...) construct?
    - Foreach

    - For
- Use Translation Resources instead of hard coded text, especially for enums and meta data annotations. Also ensure that the translation resources are generated correctly using the admin page. Find steps here.
- Always prefix an interface with the letter I, for example ITransaction, IAuditable, etc.
- When naming an interface, where possible use adjective phrases, for example IRunnable, IAuditable, IPersistable, IDisposable, IComparable, IEnumerable, however, nouns can also be used such as ITransaction, IHttpModule, etc are allowed when deemed necessary.
- Use singular form in naming Enums, unless the enum represents bit-wise flags, where plural names should rather be used.
- When using generic type parameters in a generic type based classes or methods, use descriptive names such as IDictionary<TKey, TValue>, and prefix all generic type parameters with the letter T.
- Only use the letter T as a generic type parameter if it self-explanatory and usually the only generic type parameter, for example ICollection<T>, IEnumerable<T>, etc.
- When extending from the following classes, add the base class name as a suffix:
    - System.EventArgs, e.g. System.RepeaterEventArgs
    - System.Exception, e.g. System.SQLException
    - System.Delegate

---

Revision #21
Created 17 September 2020 01:44:39 by Theuns Pretorius
Updated 5 October 2020 10:52:52