Separation of concerns

The main concept here is **Single Responsibility** - "a class/method must have only one reason to change". This deals specifically with cohesion.

```
//Bad
public exSample UpdateSample(int id, string newDescription, int productId)
{
    //Method purpose is to return a sample
    var sample = SampleHelper.GetSample(id);

    //sample description is updated
    sample.description = newDescription;

  //A new product is also added the sample and the VAT calculated
  var product = ProductHelper.GetProduct(productId);
    product.VAT = product.TotalAmount * 14 / 100;
  sample.Products.Add(product);

    return samples;
}

//Good
//Method to update the sample (serves one purpose)
public exSample UpdateSample(int id, string newDescription, int productId)
{
    //Method purpose is to return a sample
    var sample = SampleHelper.GetSample(id);

    sample.description = newDescription;

    return samples;
}

//Method to add a product to the sample and do the product calculations where necessary (serves one purpose)
private exSample AddSampleProduct(exSample sample, int productId)
{
```

```
    //A new product is also added the sample and the VAT calculated
    var product = ProductHelper.GetProduct(productId);
    product.VAT = product.TotalAmount * 14 / 100;
    sample.Products.Add(product);

    return sample;
}
```

https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles#single-responsibility

---

Revision #4
Created 17 September 2020 01:45:34 by Theuns Pretorius
Updated 5 October 2020 10:53:14