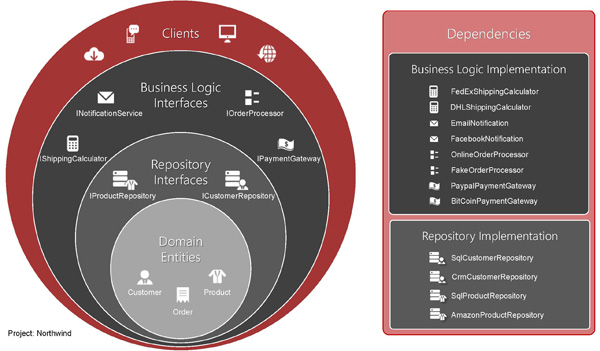
## SSW Data Onion

Integrate Entity Framework into your Onion Architecture Enterprise Application

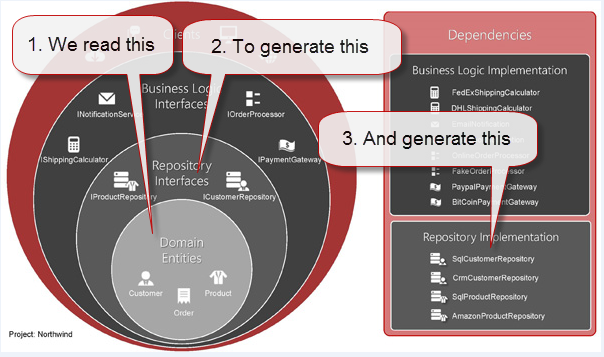


**Figure: The Layers of the onion architecture – as per ssw rule** [**http://rules.ssw.com.au/SoftwareDevelopment/RulesToBetterMVC/Pages/The-layers-of-the-onion-architecture.aspx**](http://rules.ssw.com.au/SoftwareDevelopment/RulesToBetterMVC/Pages/The-layers-of-the-onion-architecture.aspx)

At SSW, when building enterprise .net applications we follow the Onion Architecture as outlined above. This domain model centric approach encourages the development of maintainable solutions by integrating loosely coupled components via interfaces.

Adopting Onion architecture can lead to writing lots of extra code. SSW Data Onion makes this easy by generating an Entity Framework Code First implementation for persisting your Domain Entities.

In a nutshell:



1. SSW Data Onion contains a 3 .tt templates that reads domain classes from your appropriate domain assembly
2. We generate a repository interface for each domain class.
3. We generate a default EF Code first repository implementation for each domain class.
4. The DbContext also needs to reference every class so we also generate that.
5. We also have a set of core classes to support:
   1. Dependency injection of DBInitializers <http://www.entityframeworktutorial.net/code-first/database-initialization-strategy-in-code-first.aspx>
   2. Managing DbContext lifecycle in a unit of work pattern

## The packages

One core consideration for implementing this under the onion architecture was the ability to place all components in separate projects / assemblies. To allow this, SSW Data Onion is split across multiple Nuget packages. You can install these packages to separate projects (recommended) or you can install to one test project.