1. Konfigürasyon dosyasının yapısı;

Konfigürasyon dosyası C:\FreeORM\config klasörü altında conf.xml isimli dosyadır. Dosyanın içeriği şu şekilde olabilir;  
<?xml version="1.0" encoding="UTF-8"?>

<appsettings>

<current-culture>tr-TR</current-culture>

< dblogin >

<db-properties>

<properties name="sqlce" driver-type="sqlserverce" conn-str="">

<property key="Data Source" value="C:\FreeORM\Bookmarks\Bookmarks.sdf" />

<property key="Password" value="aWIyOXJiV0Z5YTNNPWJYbA==" />

</properties>

<properties name="main" driver-type="sqlserver" conn-str="">

<property key="Server" value=".\SQLEXPRESS" />

<property key="Initial Catalog" value="NorthWind" />

<property key="User Id" value="PWMyRQ==" />

<property key="Password" value="ek1USXpNVEk=" />

</properties>

</db-properties>

</dblogin>

<appconfig>

<row-count>20</row-count>

<log-setting>

<log-type>fl</log-type>

<log-path>C:\\FreeORM\\log\\log.txt</log-path>

</log-setting>

<picture-settings>

<picture-settings-folder>C:\\FreeORM\\picturedata\\</picture-settings-folder>

</picture-settings>

</appconfig>

</appsettings>

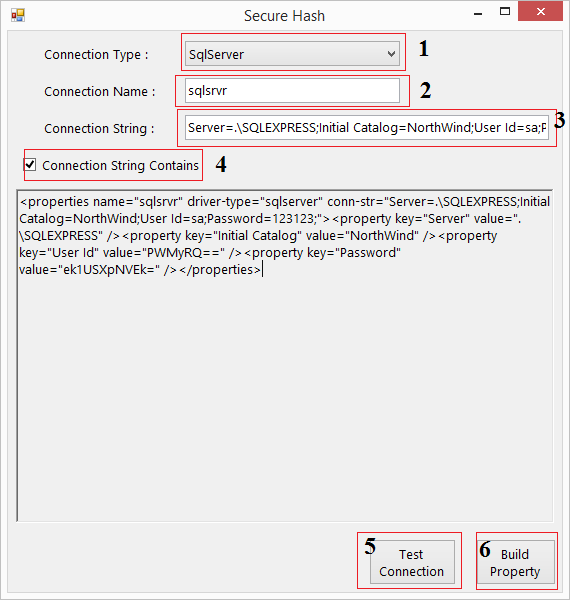
Bu dosyada appsettings\ dblogin\ db-properties\ properties nodu altında bulunan property nodları bağlantı metni(Connection String) özelliklerini oluşturur. Properties nodunda bağlantının ismi, bağlantı türü ve bağlantı metni(isteğe bağlıdır, test amaçlı olarak bağlantı metnini bu kısma şifrelenmeden de yazabilirsiniz. Property nodu, key-value prensibine bağlı olarak çalışır. Kullanıcı Id ve Şifre kısmı şifrelenmiştir.

1. Konfigürasyon dosyasında appsettings\ appconfig\ log-setting nodu altında hata kayıtları (logları)nın nasıl tutulacağı yer alır. Bu appsettings\ appconfig\ log-setting nodunun altında bulunan log-type nod değeri logların dosyaya ya da veritabanına mı kaydedileceği belirlenir.

Eğer bu nodun değeri db, dbase, database ise hata logları veritabanına kaydedilir, eğer nodun değeri fl, file veya başka bir değer ise hata logları dosyaya kaydeder.

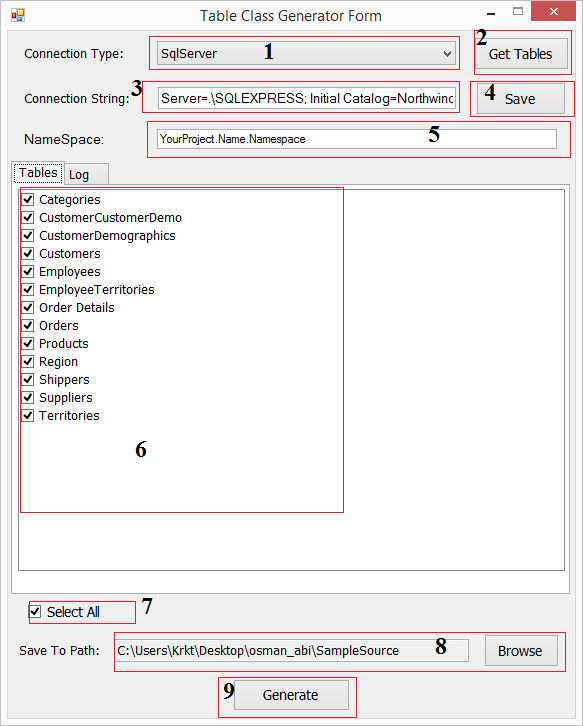
Bu appsettings\ appconfig\ log-setting nodunun altında bulunan log-path nodundaki değere göre hata logları dosyaya kaydedilecekse hangi dosya yolunun altındaki dosyaya kaydedileceği belirtilir.

1. Bağlantı Metni (Connection String) oluşturma ve konfigürasyon dosyasına aktarma.



* 1. Bağlantı tipini belirtir.
  2. Bağlantı ismini belirtir.
  3. Bağlantı metnidir.
  4. Bağlantı metni oluşturulacak Xml de yer alacaksa seçilir, yer almayacaksa seçilmez.
  5. Bağlantıyı test etmek için kullanılır. “Valid Connaction” mesajı alınırsa bağlantı geçerlidir.
  6. Bağlantı metninin Xml halini oluşturur. Oluşturulan Xml parçası kopyalanıp, kofigürasyon dosyasında appsettings\ dblogin\ db-properties nodu altına yapıştırılır.

1. Veritabanından tabloları pojo classlarına aktarma, projeye dahil etme.



* 1. Bağlantı tipi seçilir.
  2. Bağlantı metnine göre veritabanından tabloları listeler.
  3. Bağlantı metninin girildiği kısım.
  4. Bağlantı metnini Class Generator’ın kendi xml dosyasına kaydeder.
  5. Dosyaların dahil edileceği projenin Namespace kısmı girilir.
  6. Tabloların listelendiği kısım.
  7. Çok sayıda tablonun olduğu veri tabanında az sayıda tablo için sınıf oluşturulacaksa “Select All” deaktif edilip sadece istenen tabloların sınıfları oluşturulur.
  8. Dosyaların kaydedileceği yolu belirtir ve belirler.
  9. Tablo-Sınıf dosyalarını oluşturur.

1. Veritabanına ekleme, güncelleme ve silme örnekleri, dosyaların oluşturulduğu klasörde Source/BO altında bulunan sınıf dosyalarında internal olarak mevcuttur.

internal int Insert()  
 {  
 try  
 {  
 using(OrdersDL \_ordersdlDL = new OrdersDL())  
 {  
 return \_ordersdlDL.Insert(this);  
 }  
 }  
 catch (Exception)  
 {  
 throw;  
 }  
 }

internal int InsertAndGetId()  
 {  
 try  
 {  
 using(OrdersDL \_ordersdlDL = new OrdersDL())  
 {  
 return \_ordersdlDL.InsertAndGetId(this);  
 }  
 }  
 catch (Exception)  
 {  
 throw;  
 }  
 }

internal int Update()  
 {  
 try  
 {  
 using(OrdersDL \_ordersdlDL = new OrdersDL())  
 {  
 return \_ordersdlDL.Update(this);  
 }  
 }  
 catch (Exception)  
 {  
 throw;  
 }  
 }

internal int Delete()  
 {  
 try  
 {  
 using(OrdersDL \_ordersdlDL = new OrdersDL())  
 {  
 return \_ordersdlDL.Delete(this);  
 }  
 }  
 catch (Exception)  
 {  
 throw;  
 }  
 }

1. Hata Loglama Net.FreeORM.ErrorHandling altında bulunan FreeLogger sınıfındaki LogMethod ve LogException(with overloads) metotları ile yapılır. Ayrıntı için bu sınıfın altındaki metotlara, Örnek kulanım için ise BoomarkStocker projesindeki kullanımlara bakılabilir.

Ör; FreeLogger.LogMethod(ex, this.Name, "FrmBookmark Ctor"); 🡪(Exception, className, methodName)

1. İşlem Kaydı için Net.FreeORM.TransactionManagement altında bulunan TransactionManager sınıfı kullanılır. Bu sınıfın yapıcı(constructor) metodunun parametreleri iki tanedir. Birisi veritabanına yapılacak işlem kaydının tutulacağı veritabanı için bağlantı ismi ve loglama sırasında oluşacak hataların handle edilip edilmeyeceği willHandleErrors ile belirtilir. Ayrıntılı bilgi için Log metotları incelenebilir.  
   Örnek olarak;

TransactionManager.LogInsert(userId, tableName, logObjectId);

Not: Hata Loglama ve İşlem Kaydı Loglama için nesnelere uygun veritabanı tabloları –hata dosyaya loglanacaksa gerek yok.- sınıflara uygun olarak oluşturulmalıdır.