Задание к модулю Custom serialization

Общее

В данном задании мы пытаемся сериализовать с помощью DataContractSerializer результат выборки различных данных из базы Northwind (для удобства можно использовать готовый solution Task, который представлен в материалах модуля).

Проблема, которую мы пытаемся решить, состоит в следующем – ОРМ библиотеки, использующие механизмы Lazy Loading (такие как Entity Framework или NHibernate), как правило делают это, используя автогенерируемые прокси-классы, о которых сериализатор ничего не знает.

В результате, мы получаем примерно такое сообщение:

System.Runtime.Serialization.SerializationException: Type 'System.Data.Entity.DynamicProxies.Category_C0BBA388964A08FA890D2324D05497A5E29614C817E8716A99529D02E4D2DC8A' with data contract name 'Category_C0BBA388964A08FA890D2324D05497A5E29614C817E8716A99529D02E4D2DC8A:http://schemas.datacontract.org/2004/07/System.Data.Entity.DynamicProxies' is not expected. Consider using a DataContractResolver if you are using DataContractSerializer or add any types not known statically to the list of known types - for example, by using the KnownTypeAttribute attribute or by adding them to the list of known types passed to the serializer.

Здесь возможны разные варианты решения:

- Отказ от ленивой загрузки.
  Если установить для EF настройку
  dbContext.Configuration.ProxyCreationEnabled = false;
  - то он перестанет использовать proxy-классы, но при этом все связанные сущности перестанут загружаться (например, у объекта Category есть свойство Products, которое, при отключении генерации Proxy будет содержать коллекцию с 0 элементов).
- Предварительное получение имен всех подобных Proxy классов и передача их в KnownTypes
- Доработка самого процесса сериализации

Примечание!!! Если вам более знаком другой ORM с поддержкой Lazy Loading (например, NHibernate), вы можете решать данную задачу с помощью него. Но прежде нужно убедиться, что проблема воспроизводится.

Задание 1.

В solution-е Tasks (который можно найти в материалах модуля) имеется тестовый класс SerializationSolutions, каждый метод которого пытается загрузить и сериализовать список сущностей того или иного типа. При этом:

- Часть методов использует генерацию proxy (и получают ошибку сериализации), а часть – нет (и не загружают связанные сущности)
- Часть сериализует с помощью DataContractSerializer, а часть – NetDataContractSerializer (он выбран потому, что лучше поддерживает такие методы сериализации как

ISerializationSurrogate и Serialization Callbacks – например, он позволяет передавать StreamingContext в callbacks, чего не позволяет DataContractSerializer)

Используя тот способ расширения процесса сериализации, который указан в имени метода, добейтесь корректной работы каждого метода, а именно:

1. Сериализация и десериализация работают без ошибок.
2. Сериализуется сами сущности, а не proxy
3. Сериализуется основная сущность (ту которую мы запрашиваем) и связанные с нею (но только на 1 уровень вложенности!)

Примечание!!! Для тех, кто мало работал с Entity Framework напоминаю, что если у сущности есть связи, но при этом отключен LazyLoading, поле со связанной сущностью можно проинициализировать так (например, для конкретной категории загрузить список продуктов):

```
var t = (dbContext as IObjectContextAdapter).ObjectContext;

t.LoadProperty(category, f => f.Products);
```

Только не забывайте, что DbContext должен быть тем же самым, в котором была получена сама категория!