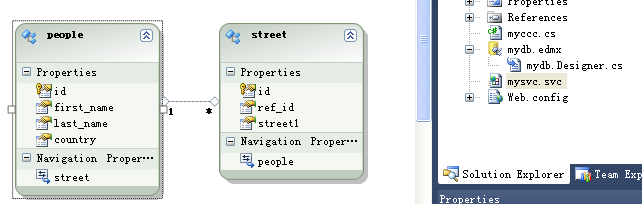
WCF 直接使用Entity Framework:

由于Entity Framework 的实体包含了关联与其他的一些约束规则。 通过WCF 发布会丢失这些关联与规则。

情况一： 直接使用Entity Framework 的 edmx

1. 定义 Data Model



1. 定义服务契约：

[ServiceContract]

[ServiceKnownType(typeof(people))]

[ServiceKnownType(typeof(street))]

public interface idb

{

[OperationContract]

List<people> getAll();

[OperationContract]

people getFirst();

[OperationContract]

people getByID(int id);

[OperationContract]

void addPeople(people a);

[OperationContract]

void upPeople(people a);

[OperationContract]

void delPeople(people a);

[OperationContract]

street getStreet();

}

1. 定义服务类：

namespace webEF1 {

[ServiceBehavior(ConcurrencyMode=ConcurrencyMode.Multiple, InstanceContextMode=InstanceContextMode.Single)]

public class myccc : idb

{

private mydb01 db;

public myccc() {

db = new mydb01();

}

public List<people> getAll() {

using (db = new mydb01()) {

return db.people;

}

}

public people getFirst() {

using (db = new mydb01()) {

return db.people.FirstOrDefault();

}

}

public people getByID(int i) {

using (db = new mydb01()) {

return db.people.Where(p => p.id == i).FirstOrDefault();

}

}

public void addPeople(people a) {

using (db = new mydb01()) {

db.people.AddObject(a);

db.SaveChanges();

}

}

public void upPeople(people a) {

using (db = new mydb01()) {

people np = db.people.Where(p => p.id == a.id).FirstOrDefault();

db.people.ApplyCurrentValues(a);

db.SaveChanges();

}

}

public void delPeople(people a) {

using (db = new mydb01()) {

db.people.DeleteObject(a);

db.SaveChanges();

}

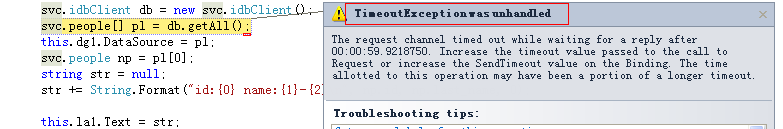
}

}

}

4）托管WCF 服务： <%@ ServiceHost Language="C#" Debug="true" Service="webEF1.myccc" %>

如果是使用默认的EF 设置， 在客户端访问则出现错误：



**原因**：

1）默认情况下，EF 为了支持它的一些高级特性(延迟加载等)，默认将自动生成代理类是设置为true，即

this.Configuration.ProxyCreationEnabled = true;

public partial class mydb01 : ObjectContext

{

#region Constructors

public mydb01() : base("name=mydb01", "mydb01")

{

this.ContextOptions.LazyLoadingEnabled = true;

OnContextCreated();

}

public mydb01(string connectionString) : base(connectionString, "mydb01")

{

this.ContextOptions.LazyLoadingEnabled = true;

OnContextCreated();

}

public mydb01(EntityConnection connection) : base(connection, "mydb01")

{

this.ContextOptions.LazyLoadingEnabled = true;

OnContextCreated();

}

#endregion end of Constructors

………

}

这样，如果我们的实体中包含其它实体的导航属性，则EF会自动的为这个实体生成代理类.

如上列子： people 包含有 street 的导航属性。 为 street 创建代理类。

出现如上错误是因为：

原因就是EF自动为street生成了代理类，WCF序列化的其实是EF生成的那个代理类，而不是我们自己定义的street，而代理类并没有标识这是一个可以序列化的实体。

**解决方法：**

**方法一** ： 禁用代理类： this.ContextOptions.ProxyCreationEnabled = false;

public partial class mydb01 : ObjectContext

{

#region Constructors

public mydb01() : base("name=mydb01", "mydb01")

{

this.ContextOptions.ProxyCreationEnabled = false;

this.ContextOptions.LazyLoadingEnabled = true;

OnContextCreated();

}

public mydb01(string connectionString) : base(connectionString, "mydb01")

{

this.ContextOptions.ProxyCreationEnabled = false;

this.ContextOptions.LazyLoadingEnabled = true;

OnContextCreated();

}

public mydb01(EntityConnection connection) : base(connection, "mydb01")

{

this.ContextOptions.ProxyCreationEnabled = false;

this.ContextOptions.LazyLoadingEnabled = true;

OnContextCreated();

}

#endregion end of Constructors

**方法二：**不需要禁用代理类: 凡是有导航属性的查询，使用Include 来 eager load. 如果有多个导航实体， 都必须include.

this.ContextOptions.ProxyCreationEnabled = true;

public List<people> getAll()

{

using (db = new mydb01()) {

return db.people.Include("street").ToList();

}

}

public people getFirst()

{

using (db = new mydb01())

{

return db.people.Include("street").FirstOrDefault();

}

}

public people getByID(int i) {

using (db = new mydb01())

{

return db.people.Include("street").Where(p => p.id == i).FirstOrDefault();

}

}

**方法三**：反序列化： （试验过，好像不工作？？）

既然代理类是由实体序列化而来的，我们就可以在返回数据前将代理类序列化成我们所需要的实体。

反序列代码：

using System.Xml;

using System.Data.Objects;

using System.ServiceModel.Description;

using System.ServiceModel.Channels;

using System.ServiceModel.Dispatcher;

public class ProxyDataContractResolver : DataContractResolver

{

private XsdDataContractExporter \_exporter = new XsdDataContractExporter();

public override Type ResolveName(string typeName, string typeNamespace, Type declaredType,

DataContractResolver knownTypeResolver)

{

return knownTypeResolver.ResolveName(

typeName, typeNamespace, declaredType, null);

}

public override bool TryResolveType(Type dataContractType, Type declaredType,

DataContractResolver knownTypeResolver,

out XmlDictionaryString typeName,

out XmlDictionaryString typeNamespace)

{

Type nonProxyType = ObjectContext.GetObjectType(dataContractType);

if (nonProxyType != dataContractType)

{

// Type was a proxy type, so map the name to the non-proxy name

XmlQualifiedName qualifiedName = \_exporter.GetSchemaTypeName(nonProxyType);

XmlDictionary dictionary = new XmlDictionary(2);

typeName = new XmlDictionaryString(dictionary,

qualifiedName.Name, 0);

typeNamespace = new XmlDictionaryString(dictionary,

qualifiedName.Namespace, 1);

return true;

}

else

{

// Type was not a proxy type, so do the default

return knownTypeResolver.TryResolveType(

dataContractType,

declaredType,

null,

out typeName,

out typeNamespace);

}

}

}

public class ApplyProxyDataContractResolverAttribute : Attribute, IOperationBehavior

{

public void AddBindingParameters(OperationDescription description, BindingParameterCollection parameters)

{

}

public void ApplyClientBehavior(OperationDescription description, ClientOperation proxy)

{

DataContractSerializerOperationBehavior

dataContractSerializerOperationBehavior =

description.Behaviors.Find<DataContractSerializerOperationBehavior>();

dataContractSerializerOperationBehavior.DataContractResolver = new ProxyDataContractResolver();

}

public void ApplyDispatchBehavior(OperationDescription description, DispatchOperation dispatch)

{

DataContractSerializerOperationBehavior

dataContractSerializerOperationBehavior = description.Behaviors.Find<DataContractSerializerOperationBehavior>();

dataContractSerializerOperationBehavior.DataContractResolver = new ProxyDataContractResolver();

}

public void Validate(OperationDescription description)

{

}

}

#endregion second code

把自定义反序列的属性， 应用到服务契约的操作契约上：

[ServiceContract]

[ServiceKnownType(typeof(people))]

[ServiceKnownType(typeof(street))]

public interface idb

{

[OperationContract]

[ApplyProxyDataContractResolverAttribute]

List<people> getAll();

[OperationContract]

[ApplyProxyDataContractResolverAttribute]

people getFirst();

[OperationContract]

[ApplyProxyDataContractResolverAttribute]

people getByID(int id);

[OperationContract]

[ApplyProxyDataContractResolverAttribute]

void addPeople(people a);

[OperationContract]

[ApplyProxyDataContractResolverAttribute]

void upPeople(people a);

[OperationContract]

[ApplyProxyDataContractResolverAttribute]

void delPeople(people a);

[OperationContract]

[ApplyProxyDataContractResolverAttribute]

street getStreet();

}

必须要将：this.ContextOptions.LazyLoadingEnabled = false; 否则出错。= false 不出错， 但是客户端也看不到导航属性的实体内容。 好像和理论上可以看到导航属性的记录不同？？？

public partial class mydb01 : ObjectContext

{

#region Constructors

public mydb01() : base("name=mydb01", "mydb01")

{

this.ContextOptions.LazyLoadingEnabled = false;

OnContextCreated();

}

public mydb01(string connectionString) : base(connectionString, "mydb01")

{

this.ContextOptions.LazyLoadingEnabled = false;

OnContextCreated();

}

public mydb01(EntityConnection connection) : base(connection, "mydb01")

{

this.ContextOptions.LazyLoadingEnabled = false;

OnContextCreated();

}

#endregion

**客户端的代码**

svc.idbClient db = new svc.idbClient();

svc.people[] pl = db.getAll();

this.dg1.DataSource = pl;

string str = null;

svc.people np = db.getFirst();

str += String.Format("id:{0} name:{1}-{2}\n", np.id, np.last\_name, np.street.Count() ); // 导航属性为空

this.la1.Text = str;

同样的道理： 我们将 EF 转化成 POCO

#region POCO 反序列化的属性定义：

public class ApplyDataContractResolverAttribute : Attribute, IOperationBehavior

{

public ApplyDataContractResolverAttribute()

{

}

public void AddBindingParameters(OperationDescription description, BindingParameterCollection parameters)

{

}

public void ApplyClientBehavior(OperationDescription description, System.ServiceModel.Dispatcher.ClientOperation proxy)

{

DataContractSerializerOperationBehavior dataContractSerializerOperationBehavior =

description.Behaviors.Find<DataContractSerializerOperationBehavior>();

dataContractSerializerOperationBehavior.DataContractResolver =

new ProxyDataContractResolver();

}

public void ApplyDispatchBehavior(OperationDescription description, System.ServiceModel.Dispatcher.DispatchOperation dispatch)

{

DataContractSerializerOperationBehavior dataContractSerializerOperationBehavior =

description.Behaviors.Find<DataContractSerializerOperationBehavior>();

dataContractSerializerOperationBehavior.DataContractResolver =

new ProxyDataContractResolver();

}

public void Validate(OperationDescription description)

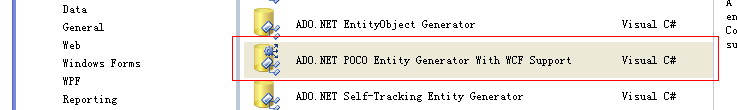
{

// Do validation.

}

}

#endregion POCO Attribute code



也是要将 this.ContextOptions.LazyLoadingEnabled = false;

如果想加载导航属性的实体集合， 需要在服务方法上使用 eager load. Include(“导航表”)

**客户端对实体的操作：** Add, Update, Del

插入操作：

public void addPeople(people a)

{

using (db = new mydb01())

{

db.people.AddObject(a);

db.SaveChanges();

}

}

svc.people newp = new svc.people

{

first\_name = "William Liu",

last\_name = "new",

country = "China"

};

db.addPeople(newp);

this.la1.Text = "id:" + newp.id; // id 并未和数据库同步更新， 仍然是 0

删除操作：

public void delPeople(people a)

{

using (db = new mydb01())

{

db.people.DeleteObject(a);

db.SaveChanges();

}

}

svc.people np2 = db.getByID(265);

db.delPeople(np2);

使用传进来的实体删除是不行的， 因为传入的 people a 是无状态的， 并不是实体集合里的一个记录。

所以删除应该改为使用 id 查找来删除：

public void delPeople(people a)

{

using (db = new mydb01())

{

people np = db.people.Where(p => p.id == a.id).SingleOrDefault();

if(np != null) db.people.DeleteObject( np );

db.SaveChanges();

}

}

修改操作：

public void upPeople(people a)

{

using (db = new mydb01())

{

people np = db.people.Where(p => p.id == a.id).FirstOrDefault(); //先定位很重要。没有此句则出错。

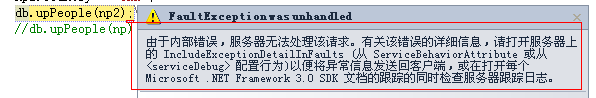
db.people.ApplyCurrentValues(a);

db.SaveChanges();

}

}

//更改， 必须先定位，否则出错：



svc.people np2 = db.getByID(270);

np2.first\_name = "change to wm";

np2.last\_name = "changto wm last";

np2.country = "UKK";

db.upPeople(np2);