一种基于.NET的对象/关系映射框架的研究与应用

Research and Application about an Object/Relation Mapping Framework based on .NET

甘 敏 王小玲 （中南大学信息科学与工程学院 长沙 410083）

摘要：持久化问题是面向对象开发和企业应用开发需要面临的问题，而关系型数据库是目前保存数据的有效手段，因此在对象层和关系数据库之间建立一种好的机制，可以更有效的改进开发的效率和质量。PDO就是专门面向分布式而设计的一种O/R（对象/关系）映射框架，它适用于Microsoft.Net环境。本文分析了基于"容器"的O/R持久化技术在分布式应用中的缺点，介绍了PDO的结构，并把它集成到一个快速开发平台中进行了测试和应用。

关键词：O/R映射 分布式 "容器"持久化技术

1 引言

在业务程序应用层和关系数据库之间增加一个持久层，实现对象和关系数据库之间的映射已成为Web应用中不可缺少的一部分。目前的主流O/R实现技术大概可以分为两类，以J2EE的EntityBean组件为代表的粗粒度对象，以及JDO/Hibernate和Microsoft ObjectSpace细粒度对象方式。这些方案都使用了数据对象"容器"来提供持久化对象的自动存储、Lazy Loading、Caching等特性。J2EE使用应用服务器（Application Server）作为实体组件"容器"，而对于JDO/Hibernate和ObjectSpace来说并不存在一个服务器来提供一个数据对象"容器"，而是在进程中创建一个依赖于进程的数据对象容器。使用容器的好处是可以管理数据对象的状态，比如根据对象属性是否修改决定是否保存，自动创建或删除或关联，提供数据对象"Cache"能力。当在分布式应用中，"容器"方式也带来复杂性的影响，并且有效性也受到限制。

PDO（Persistent Data Object的简称）是基于Microsoft .Net Framework的O/R映射框架，与其它映射框架不同，PDO是专门面向分布式应用而设计的。PDO使用Client/Server模式来管理数据对象。这种模式的重要特点就是假定使用者在一个分布式环境中，DataObjectManager并不管理持久化对象的状态，也就是说不存在一个"容器"来保存所有管理器创建的对象，而是根据客户端的请求进行对象的获取、创建、更新、删除等操作。

2 "容器"持久化技术在分布式应用中的不足

在容器方式中，数据对象总是依赖于创建它的容器，在分布式应用中，客户端总是与容器不在同一进程/服务器中，因此客户无法直接访问容器中数据对象，常规的方式是定义专门用于传输的DTO（Data Transfer Object，数据传输对象也被称之为Value Object）对象。这需要设计在服务端使用的数据对象和用于传递的DTO类，它们的属性基本上完全一致，这既增加了工作量也增加了维护的难度。另外一个不利的地方是客户端使用DTO对象并没有状态管理能力，容器并不能利用DTO对象进行更新处理，仍然要求开发者管理需要修改的属性以及关联对象，状态的管理被局限仅仅在服务期端使用。

在持久化容器中，对于关联对象和大数据量的属性，如图片或者二进制文件（BLOB字段），通常在需要的时候才提取，可以有效减少不必要的装载时间，提高系统的效率，这项特性称之为Lazy Loading或On Demand Loaidng。但在分布式环境中，Lazy Loading是一项难以利用的特性，容器的对象并不能传递给客户，通常通过DTO进行传递，而一旦要形成DTO，就已经决定是否需要装载某项属性，因此对于需要载进程间传递数据，Lazy Loading的价值就很有限。

3 PDO 的设计方案

PDO 被设计为直接支持分布式数据访问的轻量级的 O/R 框架，不需要定义 DTO，数据对象可以自由的传递到客户端。同时因为对象状态由客户端管理，客户端应用对数据对象属性的修改可以传递给对象管理器，对象管理器根据客户端的修改决定更新的属性和关联。

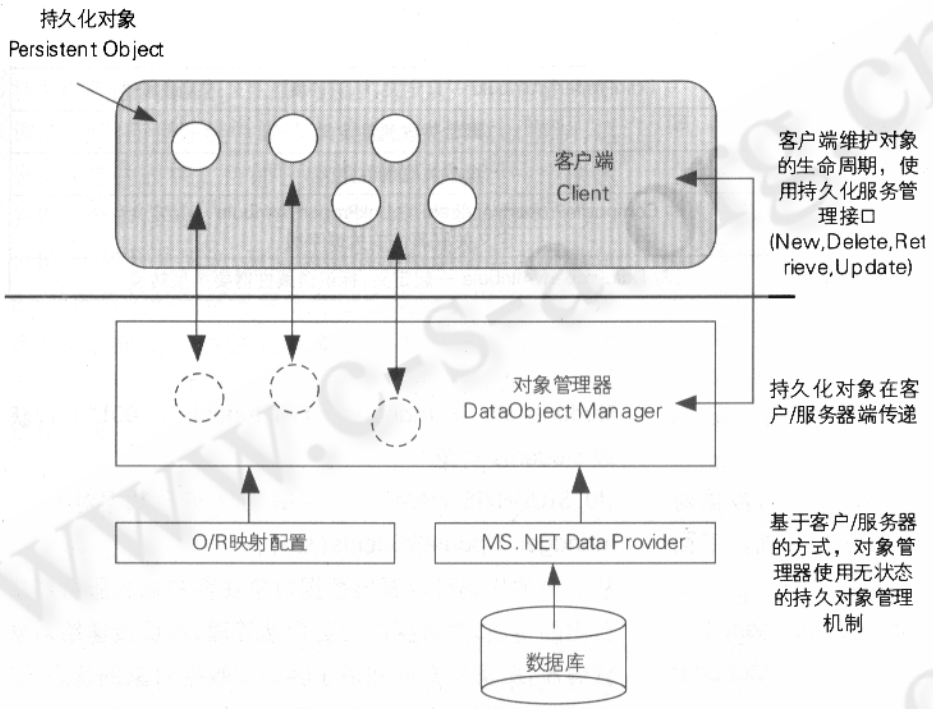

图 1　PDO Architecture：Client/Server Mode O/R Mapping Framework

PDO 并不提供用于缓存数据对象的“容器”，因此也不提供传统意义上的 Lazy Loading 功能，PDO 提供称之为 DelayLoading/LoadProperty 来实现按需访问特定的属性，数据对象的属性可以被定义为 DelayLoading，客户端可以根据需要决定是否装载 DelayLoading 属性。

4 PDO 的结构

PDO 由三部分组成：

- 映射定义。定义 O/R 映射属性。
- 数据对象管理器（DataObjectManager）。负责持久化对象的获取、创建、更新、删除，不管理持久化对象的状态。
- 数据访问层。利用 Microsoft. Net Data Provider 接口读写底层数据。

PDO 的体系结构如图 1。

4.1 定义持久化数据对象

PDO 支持的 O/R 映射属性如表 1。

4.2 基于客户端的状态管理

基于容器的 O/R 映射方式一般使用容器来管理数据对象状态，这种方式的好处在于开发者不需要在 DataObject 类上编写维护状态的代码，但它的缺点是不能离开容器管理器，一旦传递到另一个进程空间，状态管理将不再有效。对于分布式应用需要数据在服务器端和客户端传递的应用来说，采用容器管理状态，客户端对数据的修改将无法检测到，在 Update 时只能更新所有的信息。

PDO 并不存在这样的容器，同时也不假定数据对象存在于当前进程中，数据对象自身提供了关于状态的信息，并且这种信息可以在服务器和客户端之间保持一致。这种能力是怎么获取的呢？在 PDO 中，数据对象实现了 IDataObject 接口，此接口首先检查对象属性是否已被修改或已装载，然后返回数据对象的状态。IDataObject 继承了 IPersistable，IDataObject 实现需要在 PersistComplete 中修改对象的状态，数据对象管理器总是在 CRUD 操作成功之后调用 IPersistable 接口的 PersistComplete 方法。

```
/// <summary>
/// 持久化数据对象接口。
/// </summary>
public interface IDataObject : IPersistable{
/// <summary>
/// 检查对象属性是否已被修改。
/// </summary>
/// <param name="property">属性名</param>
/// <returns>如果已被修改返回 true，否则返回
```

false。如果对象状态为 New，则始终返回 true </returns>
bool IsDirty(string property);

/// <summary>
/// 数据对象状态。
/// </summary>

表 1 PDO O/R Map Attribute

Map Attribute	Target	Description
DataObjectAttribute	Class	定义类是数据对象类
DataPropertyAttribute	Field, Property	定义一个持久化属性
DelayLoadingAttribute	Field, Property	与 DataPropertyAttribute 一起定义，标识该属性是延迟装载属性
LinkPropertyAttribute	Field, Property	聚合关联属性定义
CompositePropertyAttribute	Field, Property	组成关联属性定义
LinkKeyAttribute	Field, Property	与 CompositePropertyAttribute 或 LinkPropertyAttribute 一起定义，定义关联属性的关联字段
ValueConverterAttribute	Field, Property	与 DataPropertyAttribute 一起定义，标识该属性需要类型转换

```
ObjectState ObjectState { get; }
..... //省略了延迟装载接口
}
```

对象状态可以在客户端和服务器端传递，数据对象管理器利用 IsDirty 来判断属性是否需要更新。下面的代码演示了一个更新的应用。

```
public class StudentsManager : ServiceComponent {
DataObjectManager dom = new DataObjectManager();
// 服务器端的查询 Students 接口
public Students GetStudents(string id) {
STUDENTS stu = dom.GetObject(typeof(Students), id) as Students;
return stu;
}
//服务器端的更新 Customer 接口
public void UpdateStudents(Students stu){
//保存更新，管理器将会利用客户端修改的状态决定需要更新的属性
dom.UpdateObject(stu);
}
.....
}
// 客户端代码
StudentsManager manager = new StudentsManager();
Students stu = manager.GetStudents("001");//获取 students 对象
stu.STUDENTS_NAME = "甘敏";// 修改数据对象
manager.UpdateStudents(stu);
```

从上面的代码可以看出数据对象在客户端和服务器端自由的传递，对象状态由客户端管理，然后传递给对象管理器，服务器端就知道了客户端数据对象的状态，这样就不必更新所有的信息了。比如，在 Students 类中有 SID，SNO，SNAME，COLLEGE，CLASS，PHOTO 等属性，而我们只修改了 SNAME 属性，那么在更新时就只需更新 SNAME 属性。

4.3 分布式应用中更新关联集合

在分布式数据访问中，每一次的数据传递均为远程访问，总是尽可能减少传递的数据，一些采用本地更新的方式并不适用于远程调用方式。被更新的对象的所有信息都将被传递到服务器，比如，如果 Grades 集合有 100 条记录，但只增加了 1 条记录，要更新 STUDENTS 对象，使用 UpdateObject 方式要传递 99 条无用的信息，对于性能将会是比较大的影响。使用 DataObjectManager 的 UpdateLinks 接口，可以只传递需要更新的项，UpdateLinks 接受 IChangedLinkList 类型参数，可以通过 ILinkListPersistable 返回。

4.4 支持的数据连接方式

PDO 集成了.Net Framework 库支持的 3 种数据访

问方式:OleDb,ODBC,SqlClient。PDO 还提供了其它数据访问方式的扩展支持,可以自定义 ADO. Net 兼容的数据访问方式。

5 结束语

在设计应用开发框架时,持久层已是不可缺少的一部分,它是 web 应用的一个末端。这里通常是程序最容易失控的地方,开发者总是低估构建他们自己的持久框架的挑战性。系统内部的持久层不但需要大量调试时间,而且还经常缺少功能使之变得难以控制,这是持久层的通病。目前已经把 PDO 集成到一个快速开发平台中,经过了一定的测试,系统的稳定性良好。但是,正如前面所说,系统内部的持久层需要大量调试时间,而目前的测试还只是停留在实验室里面,这还远远不足。在以后的研究中要把它投入到实际的应用中,因为只有在实践中才可以发现更多的问题,得到改进和完善。

参考文献

1 marshine. PDO Developer Guide [EB/OL]. http://marshine. nease. net/downloads/PDO _ Developer _ Guide. pdf. 2004 – 2 – 6.

2 Rick Hightower. Object – relation mapping without the container [EB/ON]. http://www – 128. ibm. com/developerworks/library/j – hibern/index. html. 2005. 12. 14.

3 孙卫琴,精通 Hibernate: Java 对象持久化技术详解[M],北京 电子工业出版社,2005。

4 车敦仁、周立柱,关系数据库与面向对象数据库的集成[J],软件学报,1996;(11)。

5 Michael S,Dorothy M. 对象 – 关系数据库管理系统——下一个大浪潮[M],杨冬青、唐世渭等译,北京 北京大学出版社,1997。

6 李也白、范春晓、面向对象数据库[M],北京 高等教