Taobao Express – Produktverwaltung

# Inhalt

Contents

[Inhalt 1](#_Toc512287092)

[Einleitung 2](#_Toc512287093)

[Datenmodell 2](#_Toc512287094)

[ERM 2](#_Toc512287095)

[ERD 3](#_Toc512287096)

# Einleitung

TaobaoExpress ist ein mit ASP.NET MVC und **C# 7** entwickelter Produktekatalog, bei welchem man Produkte erstellen, editieren, verlinken, kommentieren und Herstellern zuweisen kann.

# Architektur

Die Applikation besteht aus drei verschiedenen Assemblies.

* TaobaoExpress.DataAccess
* TaobaoExpress.Services
* TaobaoExpress.Web

TaobaoExpress.DataAccess enthält dabei allen Code, welcher per Entity Framework direkt auf die Datenbank zugreift.

TaobaoExpress.Services enthält alle Services, welche in die Controller per Unity injected werden. Dieses Assembly enthält auch alle Business Logic und Design Patterns, welche den Zugriff auf die Datenbank abstrahieren. Sie enthält das imlementierte Repository, Unit Of Work und Business Rule Pattern.

# Business Rules

Jede Business Rule kann für einen bestimmten Typ, Basistyp oder Interface definiert werden. Die Aufrufreihenfolge der Business Rules ist von am wenigsten spezifischen Typen zu spezifischten Typen. Die Business Rules definieren PreSave und PostSave Methoden. Diese werden von der UnitOfWork beim Speichern jeweils aufgerufen. Das Ganze läuft dann in derselben Transaktion, sodass ein Fehler in der Business Rule auch alle Changes wieder reverten würde.

# Dependency Injection

Dependency Injection wurde mit Unity implementiert.

# Base Controller

Es gibt ausserdem auch einen Base Controller, welcher das Speichern von Entitäten vereinfacht und schon einige Dinge, wie Exception Handling wegnimmt.

# AuditLog

Im AuditLog werden alle Changes über alle Entities getrackt. Dieser wurde auch als Business Rule implementiert und ist auch ein guten Beispiel dafür, wieso ich das Business Rule Pattern sehr gut verwnden konnte. Er zeigt die Chnages auch im UI an und es war interessant zu sehen, wie man so etwas generisch implementieren kann.

# Concurrency Check

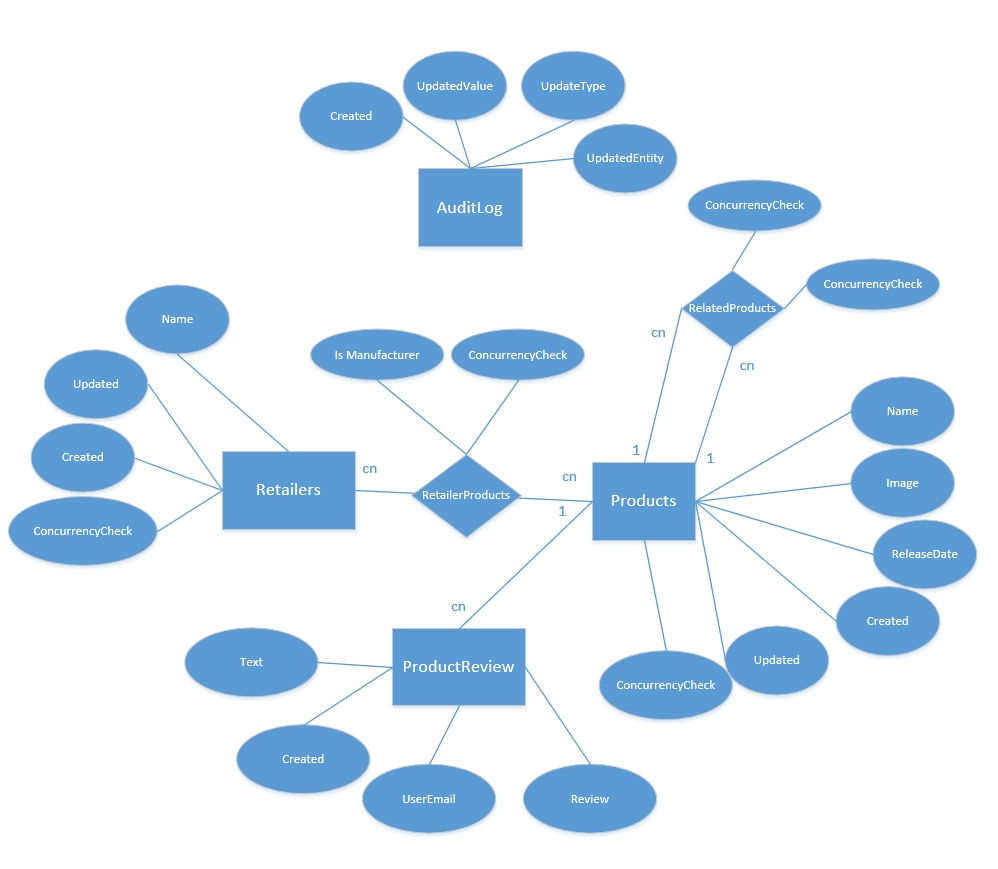
Concurrency Check wurde auch mit Optimistic Concurrency Check von Entity Framework implementiert.

# Database erstellen

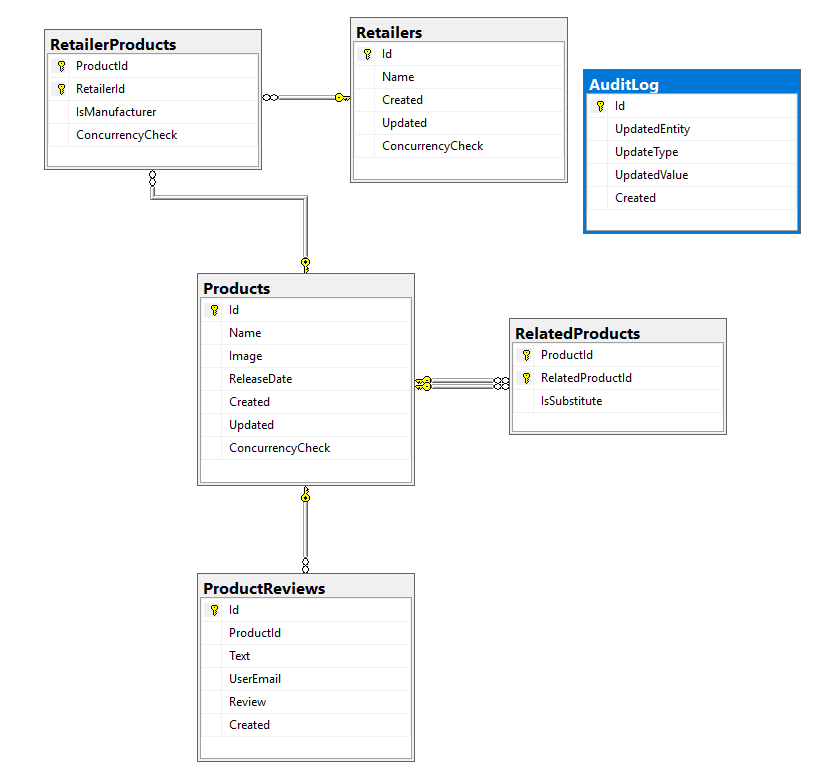
Im TaobaoExpress.DataAccess Projekt ist auch ein SQL File mit Namen **create\_database.sql**, welches das Projekt mit Demo-Datensätzen erstellt.

# Datenmodell

## ERM



## ERD



# Run

Um das Projekt auszuführen, muss Visual Studio 2017 und C#7 vorhanden sein. Danach muss ausserdem der Connection String angepasst werden. Vor der Ausführung muss jedoch die Datenbank mit dem «**create\_database.sql**» File erstellt werdem.

# Rückblick auf das Projekt

## Gut gelungene Teile

Ich denke gut gelungen ist mir das Business Rule Pattern. Schlussendlich hat mit die Einfachkeit und Effektivität dieser Business Rules sehr überzeugt. Beim nächsten Mal würde ich jedoch versuchen, etwas anderes als eine Business Rule Registry zu erstellen, da nun jeder Zugriff auf die Business Rules von der Registry abhängig ist.

## Schwierigkeiten

Schwierigkeiten gab es bei der N zu N Editierung von den Retailern und Produkten. Aber bei dieser Ansicht konnte ich am Schluss mit ASP.NET MVC ein Template für den Typ erstellen und dieses per JavaScript manuell entfernen oder hinzufügen. So konnte ich dann schlussendlich ein Array des Types an den Server im Formular senden. Da ich in der Web-Entwicklung schon länger nicht mehr mit Standard HTML Formularen gearbeitet habe und fast nur noch AJAX Requests mit Angular schreibe, hatte ich etwas Probleme mit dem Default Model Binder von ASP.NET MVC, welcher die Daten ja aus dem Formular liest.

Zusätzlich hatte ich etwas Probleme mit der XML-Serialisierung von Typen beim AuditLog. Das Problem war hauptsächlich das Lazy-Loading von Entity Framework, welches dynamische Proxy-Klassen erstellte. Auch dies konnte ich aber schlussendlich lösen. Ausserdem mussten auch Collections beim AuditLog vorerst mal ignoriert werden, da diese andere Entitäten sind und sowieso separat getrackt werden würden.