Entity Framework Add/Attach and Entity States

This topic will cover how to add and attach entities to a context and how Entity Framework processes these during SaveChanges. Entity Framework takes care of tracking the state of entities while they are connected to a context, but in disconnected or N-Tier scenarios you can let EF know what state your entities should be in. The techniques shown in this topic apply equally to models created with Code First and the EF Designer.

Entity states and SaveChanges

An entity can be in one of five states as defined by the EntityState enumeration. These states are:

- Added: the entity is being tracked by the context but does not yet exist in the database
- Unchanged: the entity is being tracked by the context and exists in the database, and its property values have not changed from the values in the database
- Modified: the entity is being tracked by the context and exists in the database, and some or all of its property values have been modified
- Deleted: the entity is being tracked by the context and exists in the database, but has been marked for deletion from the database the next time SaveChanges is called
- Detached: the entity is not being tracked by the context

SaveChanges does different things for entities in different states:

- Unchanged entities are not touched by SaveChanges. Updates are not sent to the database for entities in the Unchanged state.
- Added entities are inserted into the database and then become Unchanged when SaveChanges returns.
- Modified entities are updated in the database and then become Unchanged when SaveChanges returns.
- Deleted entities are deleted from the database and are then detached from the context.

The following examples show ways in which the state of an entity or an entity graph can be changed.

Adding a new entity to the context

A new entity can be added to the context by calling the Add method on DbSet. This puts the entity into the Added state, meaning that it will be inserted into the database the next time that SaveChanges is called. For example:

```
using (var context = new BloggingContext())
{
    var blog = new Blog { Name = "ADO.NET Blog" };
    context.Blogs.Add(blog);
    context.SaveChanges();
}
```

Another way to add a new entity to the context is to change its state to Added. For example:

```
using (var context = new BloggingContext())
{
    var blog = new Blog { Name = "ADO.NET Blog" };
    context.Entry(blog).State = EntityState.Added;
    context.SaveChanges();
}
```

Finally, you can add a new entity to the context by hooking it up to another entity that is already being tracked. This could be by adding the new entity to the collection navigation property of another entity or by setting a reference navigation property of another entity to point to the new entity. For example:

```
using (var context = new BloggingContext())
{
    // Add a new User by setting a reference from a tracked Blog
    var blog = context.Blogs.Find(1);
    blog.Owner = new User { UserName = "johndoe1987" };

    // Add a new Post by adding to the collection of a tracked Blog
    var blog = context.Blogs.Find(2);
    blog.Posts.Add(new Post { Name = "How to Add Entities" });

    context.SaveChanges();
}
```

Note that for all of these examples if the entity being added has references to other entities that are not yet tracked then these new entities will also be added to the context and will be inserted into the database the next time that SaveChanges is called.

Attaching an existing entity to the context

If you have an entity that you know already exists in the database but which is not currently being

tracked by the context then you can tell the context to track the entity using the Attach method on DbSet. The entity will be in the Unchanged state in the context. For example:

```
var existingBlog = new Blog { BlogId = 1, Name = "ADO.NET Blog" };

using (var context = new BloggingContext())
{
    context.Blogs.Attach(existingBlog);

    // Do some more work...

    context.SaveChanges();
}
```

Note that no changes will be made to the database if SaveChanges is called without doing any other manipulation of the attached entity. This is because the entity is in the Unchanged state.

Another way to attach an existing entity to the context is to change its state to Unchanged. For example:

```
var existingBlog = new Blog { BlogId = 1, Name = "ADO.NET Blog" };

using (var context = new BloggingContext())
{
    context.Entry(existingBlog).State = EntityState.Unchanged;

    // Do some more work...

    context.SaveChanges();
}
```

Note that for both of these examples if the entity being attached has references to other entities that are not yet tracked then these new entities will also attached to the context in the Unchanged state.

Attaching an existing but modified entity to the context

If you have an entity that you know already exists in the database but to which changes may have been made then you can tell the context to attach the entity and set its state to Modified. For example:

```
var existingBlog = new Blog { BlogId = 1, Name = "ADO.NET Blog" };
```

```
using (var context = new BloggingContext())
{
    context.Entry(existingBlog).State = EntityState.Modified;

    // Do some more work...

    context.SaveChanges();
}
```

When you change the state to Modified all the properties of the entity will be marked as modified and all the property values will be sent to the database when SaveChanges is called.

Note that if the entity being attached has references to other entities that are not yet tracked, then these new entities will attached to the context in the Unchanged state—they will not automatically be made Modified. If you have multiple entities that need to be marked Modified you should set the state for each of these entities individually.

Changing the state of a tracked entity

You can change the state of an entity that is already being tracked by setting the State property on its entry. For example:

```
var existingBlog = new Blog { BlogId = 1, Name = "ADO.NET Blog" };

using (var context = new BloggingContext())
{
    context.Blogs.Attach(existingBlog);
    context.Entry(existingBlog).State = EntityState.Unchanged;

    // Do some more work...

    context.SaveChanges();
}
```

Note that calling Add or Attach for an entity that is already tracked can also be used to change the entity state. For example, calling Attach for an entity that is currently in the Added state will change its state to Unchanged.

Insert or update pattern

A common pattern for some applications is to either Add an entity as new (resulting in a database insert) or Attach an entity as existing and mark it as modified (resulting in a database update) depending on the value of the primary key. For example, when using database generated integer primary keys it is common to treat an entity with a zero key as new and an entity with a non-zero key as existing. This pattern can be achieved by setting the entity state based on a check of the primary key value. For example:

```
public void InsertOrUpdate(Blog blog)
{
    using (var context = new BloggingContext())
    {
        context.Entry(blog).State = blog.BlogId == 0 ?
                                    EntityState.Added :
                                    EntityState.Modified;

        context.SaveChanges();
    }
}
```

Note that when you change the state to Modified all the properties of the entity will be marked as modified and all the property values will be sent to the database when SaveChanges is called.