

Configuration

In your *App.config* or *Web.config*:

- Ensure your connection string is added to the connection string section.
- Add the Active Record configuration section handler to the *configSections*.
- Add the Active Record configuration section, replacing the connection string name with your own.
- Web applications should set *isWeb=true* on the ActiveRecord section.

Initialization

```
ActiveRecordStarter.Initialize(typeof({a business object}).Assembly,
    ActiveRecordSectionHandler.Instance);
```

This call must be made before you access API functions. *Web Application* developers should consider doing this in *Application_Start* in *Global.asax*.

Web Application developers should also consider initializing session-per-request.

Configuration Tips

- Remember to *Allow Saving of Passwords* when you configure your connection.
- Consider excluding references you won't use immediately. *Ingredient.Unit* is useful, but the list of *Ingredients* that refer to a *Unit* just clutters your code.

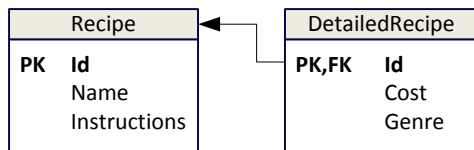
An ActiveRecord Friendly Database

Creating an ActiveRecord friendly database will smooth your development process and produce much easier to read code.

- Avoid using multiple primary key columns.
- Do not name columns the same as their table.
- Avoid prefixes and suffixes like *tbl*, *pk*, *fk* or *id*.
- Use simple names (*Description*, not *RecipeDesc*).
- Name foreign keys like columns.
- Avoid multiple-inheritance patterns.

Inheritance

In this example *DetailedRecipe* inherits *Recipe*. Both *Id* columns are declared as primary keys. The *DetailedRecipe.Id* is a foreign key to the *Recipe* table.



At runtime you can load a list of *Recipes* using the *Find* function. This list may contain some *DetailedRecipes*. You can determine this using your language's features such as the *is* operator.

```
if(someObject is DetailedRecipe) { ... }
```

Basic Persistence

Loading by primary key,

```
Recipe myRecipe = Recipe.Find(primaryKey);
```

Saving,

```
myRecipe.Save();
```

 Where you are not using Identity primary keys you will have to use *Create* for new records.

Deleting,

```
myRecipe.Delete();
```

Scopes

Remember to use your business objects from within a scope.

```
using(new SessionScope())
{
    ...
}
```

Transactions

To interact with your data layer using transactions, use the provided transaction scopes. These scopes can be nested.

```
TransactionScope transaction
    = new TransactionScope();
try
{
    {operations}
}
catch
{
    transaction.VoteRollBack();
    throw;
}
finally
{
    transaction.VoteCommit();
    transaction.Dispose();
}
```

Finding Records

By primary key,

```
Recipe myRecipe = Recipe.Find(primaryKey);
```

By property,

```
Recipe[] recipes = Recipe.FindByProperty("Name", {name});
```

All records of a type,

```
Recipe[] recipes = Recipe.FindAll();
```

Criteria

```
Recipe[] recipes = Recipe.FindAll({ICriterion, ...});
```

Criteria can be instantiated by static methods on the factory class *Expression*.

```
= Recipe.FindAll(Expression.Gt("Rating", 4));
= Recipe.FindAll(Expression.Between("Rating", 2, 5));
= Recipe.FindAll(Expression.Conjunction(
    Expression.In("Rating", 1, 2, 5),
    Expression.Eq("Public", true)
));
```

Additional Support

Get additional support, including guides and full API documentation,

<http://dunnchurchill.com/support/documentation/>

Get product updates and additional tools,

<http://dunnchurchill.com/support/downloads/>



diamondbinding