实体是DDD一个核心的概念。Eric Evans是这么描述的：“一个对象根本上不是按它的特性定义的，而是按一个线程的连续性和身份来定义”。所以实体有一个id属性存入数据库中。一个实体通常映射成关系型数据库的一个表。

**实体类**

在ABP里，实体从Entity类上继承，示例代码如下：

public class Person : Entity

{

public virtual string Name { get; set; }

public virtual DateTime CreationTime { get; set; }

public Person()

{

CreationTime = DateTime.Now;

}

}

Entity类定义了一个Id属性，它是这个实体的主键。所以所有的实体主键名都相同，都是Id。

Id（主键）的类型是可改的，默认是int（Int32）。如果你想把Id定义成其它类型，你应该显式声明它，如下所示：

public class Person : Entity<long>

{

public virtual string Name { get; set; }

public virtual DateTime CreationTime { get; set; }

public Person()

{

CreationTime = DateTime.Now;

}

}

Entity类重写了equality操作符（==），用它可以非常容易地检查两个实体是否相等（它们的Id是否相等）

**审计**

IHasCreationTime为一个实体的“创建时间”信息采用通用的属性，在一个实体插入到数据库前，ABP自动为实现了该接口的实体，设置CreationTime属性为当前时间。

public interface IHasCreationTime

{

DateTime CreationTime { get; set; }

}

Person类改写成实现IHasCreationTime接口，如下所示：

public class Person : Entity<long>, IHasCreationTime

{

public virtual string Name { get; set; }

public virtual DateTime CreationTime { get; set; }

public Person()

{

CreationTime = DateTime.Now;

}

}

ICreationAudited通过添加CreatorUserId扩展了IhasCreationTime：

public interface ICreationAudited : IHasCreationTime

{

long? CreatorUserId { get; set; }

}

也有一个类似的“修改”接口

public interface IHasModificationTime

{

DateTime? LastModificationTime { get; set; }

}

public interface IModificationAudited : IHasModificationTime

{

long? LastModifierUserId { get; set; }

}

当更新一个实体时，ABP也自动设置这些属性。你只需要为你的类定义它们就可以。

如果你想实现所有审计属性，你可以直接实现IAudited接口：

public interface IAudited : ICreationAudited, IModificationAudited

{

}

**软删除**

软删除是一个通用的模式，它把一个实体标记为“已删除”代替从数据库直接删除。例如，你不想把一个User从数据库硬删除，因为它可能与其它表有关联，ISoftDelete接口就是出于这种目的：

public interface ISoftDelete

{

bool IsDeleted { get; set; }

}

如果你使用软删除，当软删除一个实体时，你可能也会想保存是谁删除和什么时候删除，你可以实现IDeletionAudited接口，如下所示：

public interface IDeletionAudited : ISoftDelete

{

long? DeleterUserId { get; set; }

DateTime? DeletionTime { get; set; }

}

**IEntity 接口**

实质上，Entity类实现了IEntity接口（且Entity<TPrimaryKey>实现了IEntity<TPrimaryKey>）。如果你不想从Entity类继承，你可以直接实现这些接口，这些接口对于其它实体类也是适用的，但是这不是推荐的方式，除非你有一个好的理由。