**在ABP中管理连接和事务**

ABP自动完成这些操作。

**约定的工作单元方法**

有些方法默认为工作单元方法：

* 所有[Mvc](http://www.cnblogs.com/kid1412/p/5996459.html" \t "http://www.cnblogs.com/kid1412/p/_blank)、[Web Api](http://www.cnblogs.com/kid1412/p/5996459.html" \t "http://www.cnblogs.com/kid1412/p/_blank)和[Asp.net Core Mvc](http://www.cnblogs.com/kid1412/p/5996459.html" \t "http://www.cnblogs.com/kid1412/p/_blank)控制器的Actions。
* 所有[应用服务](http://www.cnblogs.com/kid1412/p/5996459.html" \t "http://www.cnblogs.com/kid1412/p/_blank)方法。
* 所有[仓储](http://www.cnblogs.com/kid1412/p/5994961.html" \t "http://www.cnblogs.com/kid1412/p/_blank)方法。

**控制工作单元**

上面的方法里，工作单元是暗中工作的。web应用里大部分情况，我们都不用控制工作单元。如果你想在某些地方控制工作单元，你可以显式地使用它，有两种方法可以实现。

**UnitOfWork 特性**

优先推荐的方法是：使用UnitOfWork特性，例如：

**[UnitOfWork]**

public void CreatePerson(CreatePersonInput input)

{

var person = new Person { Name = input.Name, EmailAddress = input.EmailAddress };

\_personRepository.Insert(person);

\_statisticsRepository.IncrementPeopleCount();

}

**IUnitOfWorkManager**

第二种方法是：使用IUnitOfWorkManager.Begin(...)方法，如下所示：

public class MyService

{

public void CreatePerson(CreatePersonInput input)

{

using (var unitOfWork = **\_unitOfWorkManager.Begin()**)

{

}

}

}

**工作单元详情**

**禁用工作单元**

你如果想为约定的工作单元方法禁用工作单元，使用UnitOfWork特性的IsDisabled属性。例如：

**[UnitOfWork(IsDisabled = true)]**

public virtual void RemoveFriendship(RemoveFriendshipInput input)

{

\_friendshipRepository.Delete(input.Id);

}

**非事务性工作单元**

一个工作单元默认就是事务性的，因此ABP开始/提交/回滚数据库级别事务。在一些特殊情况下，事务可能会引起问题，因为它可能锁住数据库的一些行或表。

**[UnitOfWork(isTransactional: false)]**

public GetTasksOutput GetTasks(GetTasksInput input)

{

// 获取数据列表，不需要启动事物

var tasks = \_taskRepository.GetAllWithPeople(input.AssignedPersonId, input.State);

return new GetTasksOutput

{

Tasks = Mapper.Map<List<TaskDto>>(tasks)

};

}

**自动保存修改**

如果一个方法是工作单元，ABP会自动在这个方法的最后，保存所有修改。假设我们需要一个方法更新person的name：

**[UnitOfWork]**

public void UpdateName(UpdateNameInput input)

{

var person = \_personRepository.Get(input.PersonId);

**person.Name = input.NewName;**

}

这就是所有代码，name会被修改！

**IRepository.GetAll() 方法**

当你在一个仓储方法外调用GetAll()，它必须要有一个打开的数据库连接，因为它只是返回IQueryable。这是必须的，因为IQuery的延迟执行。

考虑如下示例：

**[UnitOfWork]**

public SearchPeopleOutput SearchPeople(SearchPeopleInput input)

{

//Get IQueryable<Person>

var query = **\_personRepository.GetAll();**

//Get paged result list

var people = query.ToList();

return new SearchPeopleOutput { People = Mapper.Map<List<PersonDto>>(people) };

}

SearchPeople方法必须是工作单元，因为IQueryable的ToList()方法在这个方法里被调用。并且当IQueryable.ToList()被执行时，数据库连接必须打开着。

大多数情况，在一个web应用里，使用GetAll方法是安全的，因为所以控制的Action默认都是工作单元并且数据库连接在整个请求里都是可用的。

**选项**

有些选项可以改变一个工作单元的行为。

public class SimpleTaskSystemCoreModule : AbpModule

{

public override void PreInitialize()

{

Configuration.UnitOfWork.IsolationLevel = IsolationLevel.ReadCommitted;

Configuration.UnitOfWork.Timeout = TimeSpan.FromMinutes(30);

}

//...other module methods

}