13

Injeção de Dependências

Na aplicação web, todo acesso ao banco de dados é feito através dos DAOs, porém para construí-los, precisamos passar a sessão do NHibernate, o que torna o código muito repetitivo.

Para eliminar a repetição de código na criação dos objetos, utilizaremos uma biblioteca de injeção de dependências integrada com o Asp.Net MVC chamada `Ninject.MVC3`, que já está instalada no projeto (mas pode ser instalada através do NuGet).

O Ninject será o responsável por instanciar os nossos controllers, passando quaisquer dependências que possuam em seus construtores. Por exemplo, o `ProdutosController` depende do `ProdutosDAO`, então ao invés de instanciarmos o `ProdutosDAO` em cada um dos métodos do controller, vamos pedir uma instância do DAO como argumento do construtor do controller:

```
public class ProdutosController : Controller
{
    private ProdutosDAO dao;

    public ProdutosController(ProdutosDAO dao)
    {
        this.dao = dao;
    }
}
```

Com essa modificação, estamos passando a responsabilidade de criação do `ProdutosDAO` para o Ninject, é ele que deve conseguir uma instância do DAO para criar o controller, com isso conseguimos simplificar bastante o nosso código, porém para conseguir o `ProdutosDAO`, o Ninject precisa de um `ISession`, que é uma interface, portanto vamos configurá-lo para que ele saiba como recuperar uma instância da sessão.

Ao instalarmos o Ninject, ele coloca dentro da pasta `App_Start` uma nova classe chamada `NinjectWebCommon`, é nessa classe que colocaremos as configurações do Ninject. Na classe de configuração do plugin, temos um método chamado `RegisterServices`, onde ensinaremos o Ninject como instanciar a session. Esse método recebe um objeto que é responsável por registrar os componentes da aplicação, o `IKernel`:

```
public static void RegisterServices(IKernel kernel)
{
    // aqui dentro faremos o registro do ISession
}
```

Utilizamos o método `Bind` do kernel para registrarmos um novo componente:

```
kernel.Bind<ISession>()
```

Agora precisamos ensinar como instanciar o `ISession`, fazemos isso através do método `ToMethod`, esse método recebe uma função que devolve uma instância de `ISession`, vamos passar um lambda do c# que usa o método `AbreSession`:

```
kernel.Bind<ISession>().ToMethod(x => NHibernateHelper.AbreSession())
```

Mas queremos uma session por requisição web, ou seja, o tempo de vida da sessão é de uma request (Request Scope). Para associar o tempo de vida de uma sessão à requisição, utilizamos o método `InRequestScope` :

```
kernel.Bind<ISession>().ToMethod(
    x => NHibernateHelper.AbreSession()).InRequestScope();
```

O método `RegisterServices` com o registro da sessão fica da seguinte forma:

```
private static void RegisterServices(IKernel kernel)
{
    kernel.Bind<ISession>().ToMethod(
        x => NHibernateHelper.AbreSession()).InRequestScope();
}
```

E agora o código da aplicação não precisa mais se preocupar com a criação da session, deixaremos esse trabalho com o Ninject.

Configure o plugin do Ninject no projeto web, para isso registre o `ISession` do NHibernate dentro do método `RegisterServices` da classe `NinjectWebCommon` .

Modifique os contrutores dos seus controller para que eles recebam os DAOs ao invés de instânciá-los dentro das actions.

Responda

INSERIR CÓDIGO | FORMATAÇÃO

```
// Cole o código do ProdutosController aqui
```