

Interfaces (Contratos) da camada de regras de negócio:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.BLL.Contracts
{
    public interface IBaseBusiness<T> where T : class
    {
        void Cadastrar(T obj);
        void Atualizar(T obj);
        void Excluir(T obj);

        List<T> ConsultarTodos();
        T ConsultarPorId(int id);
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;

namespace Projeto.BLL.Contracts
{
    public interface IDependenteBusiness
        : IBaseBusiness<Dependente>
    {
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;

namespace Projeto.BLL.Contracts
{
    public interface IFuncaoBusiness
        : IBaseBusiness<Funcao>
    {
    }
}
```

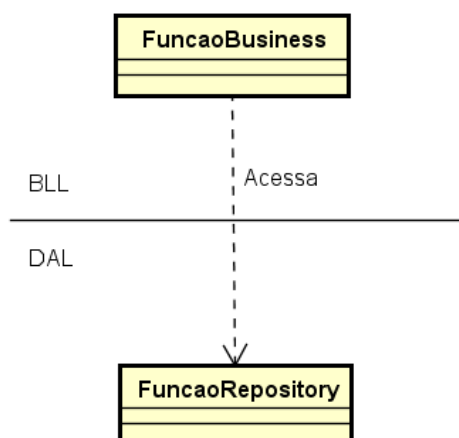
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;

namespace Projeto.BLL.Contracts
{
    public interface IFuncionarioBusiness
        : IBaseBusiness<Funcionario>
    {
    }
}
```

DIP - Principio de Inversão de Dependência

Princípio SOLID que define a seguinte regra: Camadas de alto nível não devem acessar classes baixo nível mas sim suas abstrações (interfaces)

- Errado:**



- Correto:**

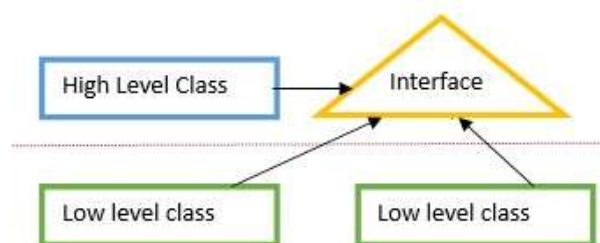
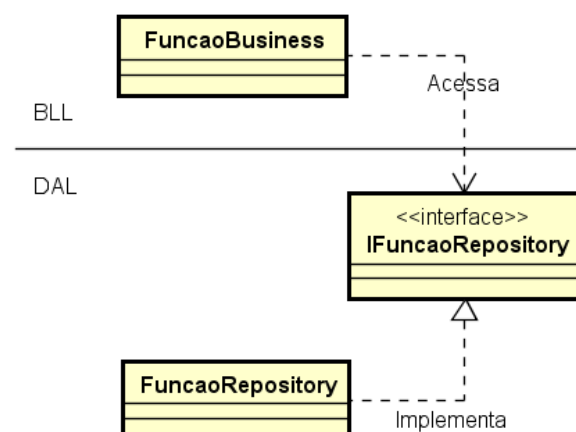


fig: Interface was defined by high level class

Criando as demais classes:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando
using Projeto.BLL.Contracts; //importando
using Projeto.DAL.Contracts; //importando

namespace Projeto.BLL.Business
{
    public class DependenteBusiness
        : BaseBusiness<Dependente>, IDependenteBusiness
    {
        //atributo
        private IDependenteRepository repository;

        //construtor para inicializar o atributo
        public DependenteBusiness(IDependenteRepository repository)
            : base(repository)
        {
            this.repository = repository;
        }
    }
}

using Projeto.BLL.Contracts;
using Projeto.DAL.Contracts;
using Projeto.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.BLL.Business
{
    public class FuncaoBusiness
        : BaseBusiness<Funcao>, IFuncaoBusiness
    {
        private IFuncaoRepository repository;

        public FuncaoBusiness(IFuncaoRepository repository)
            : base(repository)
        {
            this.repository = repository;
        }
    }
}
```

```
using Projeto.BLL.Contracts;
using Projeto.DAL.Contracts;
using Projeto.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.BLL.Business
{
    public class FuncionarioBusiness
        : BaseBusiness<Funcionario>, IFuncionarioBusiness
    {
        private IFuncionarioRepository repository;

        public FuncionarioBusiness(IFuncionarioRepository repository)
            : base(repository)
        {
            this.repository = repository;
        }
    }
}
```

Finalizando os controllers da API:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using AutoMapper; //importando
using Projeto.Entities; //importando
using Projeto.Services.Models; //importando
using Projeto.BLL.Contracts; //importando

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/Dependente")]
    public class DependenteController : ApiController
    {
        //atributo
        private IDependenteBusiness business;

        //construtor para inicializar o atributo
        public DependenteController(IDependenteBusiness business)
        {
            this.business = business;
        }

        [HttpPost] //requisições do tipo POST
        public HttpResponseMessage Post(DependenteCadastroViewModel model)
        {
            if(ModelState.IsValid)
            {
                try
                {

```

```

        //transferir os dados da model para entidade
        var dependente = Mapper.Map<Dependente>(model);
        business.Cadastrar(dependente);

        return Request.CreateResponse(HttpStatusCode.OK,
            $"Dependente {model.Nome}, cadastrado com sucesso.");
    }
    catch(Exception e)
    {
        //erro HTTP 500 -> INTERNAL SERVER ERROR
        return Request.CreateResponse
            (HttpStatusCode.InternalServerError,
            "Erro interno de servidor: " + e.Message);
    }
}
else
{
    //erro HTTP 400 -> BAD REQUEST
    return Request.CreateResponse(HttpStatusCode.BadRequest,
        "Ocorreram erros de validação.");
}
}

[HttpPut] //requisições do tipo PUT
public HttpResponseMessage Put(DependenteEdicaoViewModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            //transferir os dados da model para entidade
            var dependente = Mapper.Map<Dependente>(model);
            business.Atualizar(dependente);

            return Request.CreateResponse(HttpStatusCode.OK,
                $"Dependente {model.Nome}, atualizado com sucesso.");
        }
        catch (Exception e)
        {
            //erro HTTP 500 -> INTERNAL SERVER ERROR
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }
    else
    {
        //erro HTTP 400 -> BAD REQUEST
        return Request.CreateResponse(HttpStatusCode.BadRequest,
            "Ocorreram erros de validação.");
    }
}

[HttpDelete]
public HttpResponseMessage Delete(int id)
{
    try
    {
        var dependente = business.ConsultarPorId(id);
        business.Excluir(dependente);
    }
}

```



```
namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/Funcao")]
    public class FuncaoController : ApiController
    {
        //atributo
        private IFuncaoBusiness business;

        //construtor para inicializar o atributo
        public FuncaoController(IFuncaoBusiness business)
        {
            this.business = business;
        }

        [HttpPost]
        public HttpResponseMessage Post(FuncaoCadastroViewModel model)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    //transferir os dados da model para entidade
                    var funcao = Mapper.Map<Funcao>(model);
                    business.Cadastrar(funcao);

                    return Request.CreateResponse(HttpStatusCode.OK,
                        $"Função {model.Nome}, cadastrado com sucesso.");
                }
                catch (Exception e)
                {
                    //erro HTTP 500 -> INTERNAL SERVER ERROR
                    return Request.CreateResponse
                        (HttpStatusCode.InternalServerError,
                            "Erro interno de servidor: " + e.Message);
                }
            }
            else
            {
                //erro HTTP 400 -> BAD REQUEST
                return Request.CreateResponse(HttpStatusCode.BadRequest,
                    "Ocorreram erros de validação.");
            }
        }

        [HttpPut]
        public HttpResponseMessage Put(FuncaoEdicaoViewModel model)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    //transferir os dados da model para entidade
                    var funcao = Mapper.Map<Funcao>(model);
                    business.Atualizar(funcao);

                    return Request.CreateResponse(HttpStatusCode.OK,
                        $"Função {model.Nome}, atualizado com sucesso.");
                }
                catch (Exception e)
                {
                    //erro HTTP 500 -> INTERNAL SERVER ERROR
                }
            }
        }
    }
}
```

```

        return Request.CreateResponse
            (HttpStatusCode.InternalServerError,
             "Erro interno de servidor: " + e.Message);
    }
}
else
{
    //erro HTTP 400 -> BAD REQUEST
    return Request.CreateResponse(HttpStatusCode.BadRequest,
                                    "Ocorreram erros de validação.");
}
}

[HttpDelete]
public HttpResponseMessage Delete(int id)
{
    try
    {
        var funcao = business.ConsultarPorId(id);
        business.Excluir(funcao);

        return Request.CreateResponse(HttpStatusCode.OK,
                                        $"Função {funcao.Nome} excluído com sucesso.");
    }
    catch (Exception e)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
                                        "Erro interno de servidor: " + e.Message);
    }
}

[HttpGet]
public HttpResponseMessage GetAll()
{
    try
    {
        var funcoes = business.ConsultarTodos();
        var model = Mapper.Map<List<FuncaoConsultaViewModel>>(funcoes);

        return Request.CreateResponse(HttpStatusCode.OK, model);
    }
    catch (Exception e)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
                                        "Erro interno de servidor: " + e.Message);
    }
}

[HttpGet]
public HttpResponseMessage GetById(int id)
{
    try
    {
        var funcao = business.ConsultarPorId(id);
        var model = Mapper.Map<FuncaoConsultaViewModel>(funcao);

        return Request.CreateResponse(HttpStatusCode.OK, model);
    }
    catch (Exception e)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,

```



```

        "Erro interno de servidor: " + e.Message);
    }
}

-----

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using AutoMapper; //importando
using Projeto.Services.Models; //importando
using Projeto.Entities; //importando
using Projeto.BLL.Contracts;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/Funcionario")]
    public class FuncionarioController : ApiController
    {
        //atributo
        private IFuncionarioBusiness business;

        public FuncionarioController(IFuncionarioBusiness business)
        {
            this.business = business;
        }

        [HttpPost]
        public HttpResponseMessage Post(FuncionarioCadastroViewModel model)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    //transferir os dados da model para entidade
                    var funcionario = Mapper.Map<Funcionario>(model);
                    business.Cadastrar(funcionario);

                    return Request.CreateResponse(HttpStatusCode.OK,
                        $"Funcionario {model.Nome}, cadastrado com sucesso.");
                }
                catch (Exception e)
                {
                    //erro HTTP 500 -> INTERNAL SERVER ERROR
                    return Request.CreateResponse
                        (HttpStatusCode.InternalServerError,
                            "Erro interno de servidor: " + e.Message);
                }
            }
            else
            {
                //erro HTTP 400 -> BAD REQUEST
                return Request.CreateResponse(HttpStatusCode.BadRequest,
                    "Ocorreram erros de validação.");
            }
        }
    }
}

```

```
}

[HttpPut]
public HttpResponseMessage Put(FuncionarioEdicaoViewModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            //transferir os dados da model para entidade
            var funcionario = Mapper.Map<Funcionario>(model);
            business.Atualizar(funcionario);

            return Request.CreateResponse(HttpStatusCode.OK,
                $"Funcionario {model.Nome}, atualizado com sucesso.");
        }
        catch (Exception e)
        {
            //erro HTTP 500 -> INTERNAL SERVER ERROR
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError,
                    "Erro interno de servidor: " + e.Message);
        }
    }
    else
    {
        //erro HTTP 400 -> BAD REQUEST
        return Request.CreateResponse(HttpStatusCode.BadRequest,
            "Ocorreram erros de validação.");
    }
}

[HttpDelete]
public HttpResponseMessage Delete(int id)
{
    try
    {
        var funcionario = business.ConsultarPorId(id);
        business.Excluir(funcionario);

        return Request.CreateResponse(HttpStatusCode.OK,
            $"Funcionário {funcionario.Nome} excluído com sucesso.");
    }
    catch (Exception e)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
            "Erro interno de servidor: " + e.Message);
    }
}

[HttpGet]
public HttpResponseMessage GetAll()
{
    try
    {
        var funcionarios = business.ConsultarTodos();
        var model = Mapper.Map<List<FuncionarioConsultaViewModel>>
            (funcionarios);

        return Request.CreateResponse(HttpStatusCode.OK, model);
    }
}
```

```

        catch (Exception e)
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }

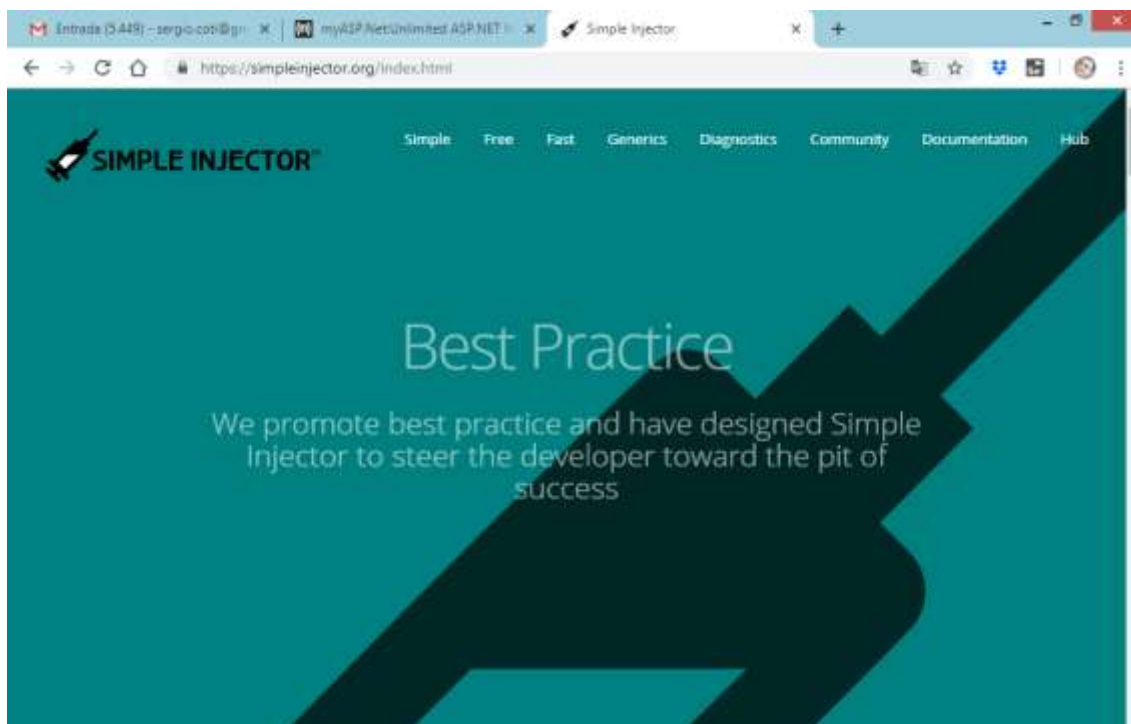
    [HttpGet]
    public HttpResponseMessage GetById(int id)
    {
        try
        {
            var funcionario = business.ConsultarPorId(id);
            var model = Mapper.Map<FuncionarioConsultaViewModel>
                (funcionario);

            return Request.CreateResponse(HttpStatusCode.OK, model);
        }
        catch (Exception e)
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }
}

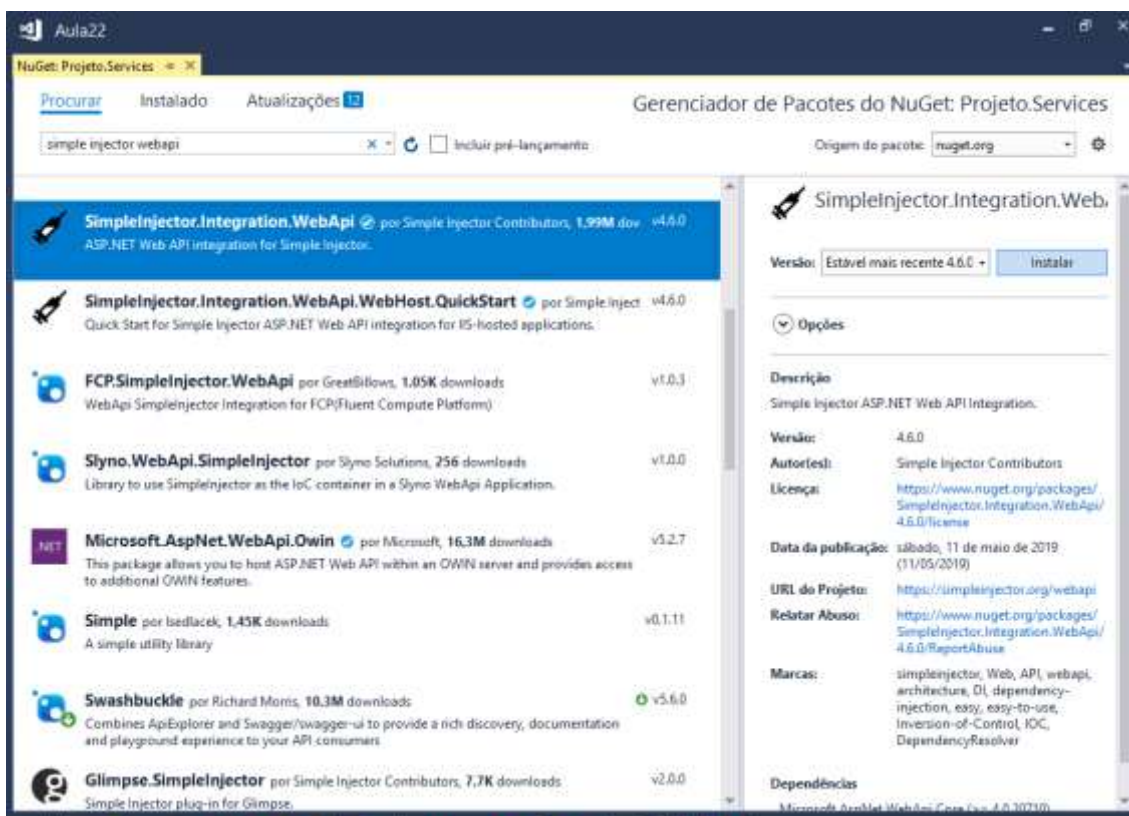
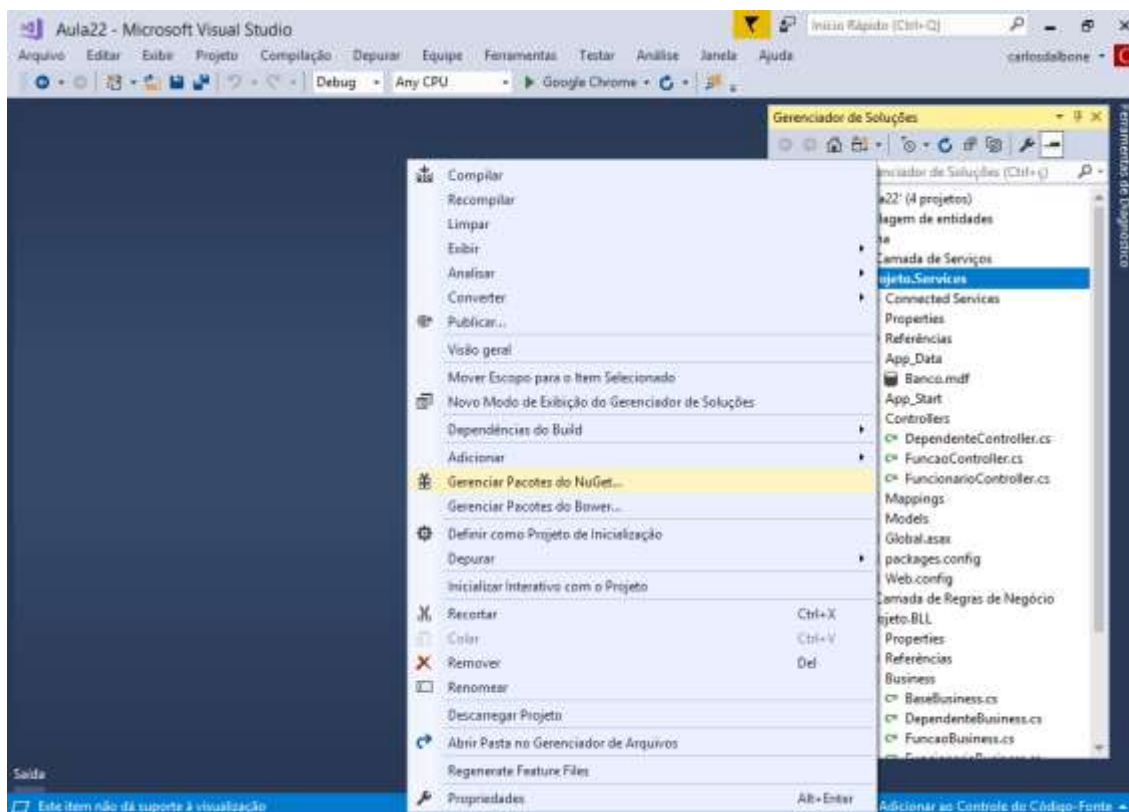
```

Simple Injector (<https://simpleinjector.org>)

Framwork desenvolvido para .NET que permite utilizar no projeto o padrão denominado "Injeção de Dependência", ou seja, no nosso projeto iremos mapear quais classes deverão ser utilizadas para instanciar cada interface que os métodos construtores precisam receber.



Instalando o Simple Injector: Gerenciador de pacotes do NuGet



Global.asax

Classe de inicialização do projeto Asp.Net

Configurando o AutoMapper:

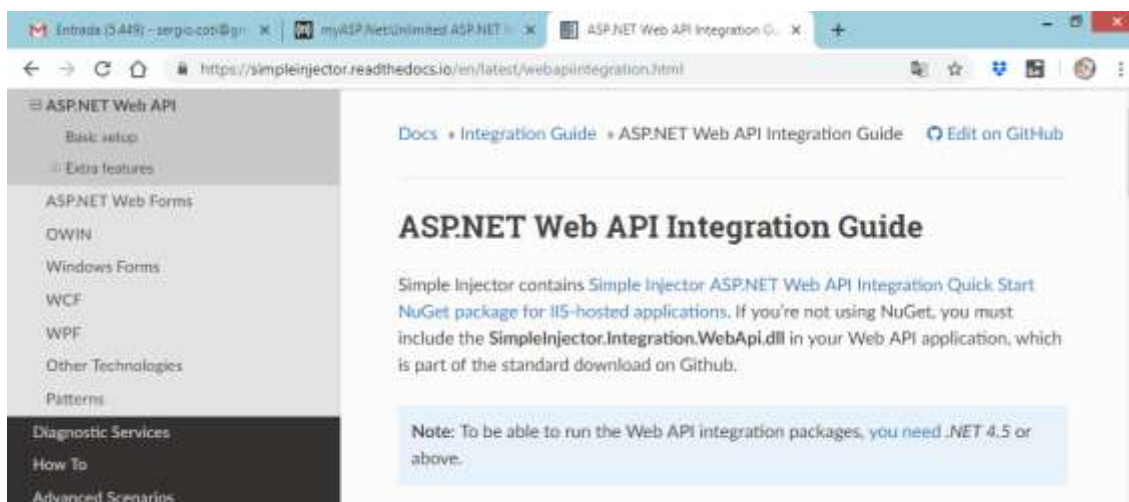
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Routing;
using AutoMapper;
using Projeto.Services.Mappings;

namespace Projeto.Services
{
    public class WebApiApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            GlobalConfiguration.Configure(WebApiConfig.Register);

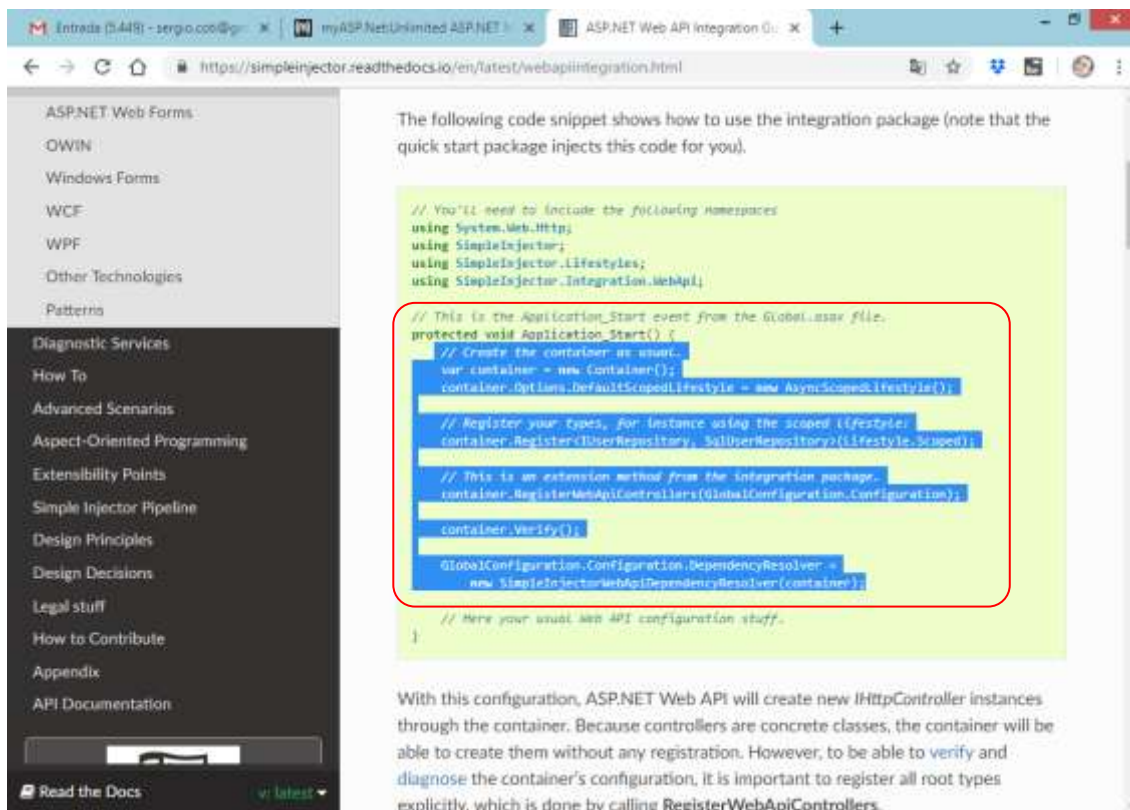
            //configuração do AutoMapper
            Mapper.Initialize(cfg =>
            {
                cfg.AddProfile<EntityToViewModelMap>();
                cfg.AddProfile<ViewModelToEntityMap>();
            });
        }
    }
}
```

Nesta classe iremos utilizar o método **Application_Start** para configurar o SimpleInjector no momento da inicialização do projeto.

<https://simpleinjector.readthedocs.io/en/latest/webapiintegration.html>



Copie o código abaixo:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Routing;
using AutoMapper;
using Projeto.Services.Mappings;
using SimpleInjector;
using SimpleInjector.Lifestyles;
using Projeto.DAL.Contracts;
using Projeto.DAL.Repositories;

using Projeto.BLL.Business;
using Projeto.BLL.Contracts;
using SimpleInjector.Integration.WebApi;

namespace Projeto.Services
{
    public class WebApiApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            GlobalConfiguration.Configure(WebApiConfig.Register);

            //configuração do AutoMapper
            Mapper.Initialize(cfg =>
            {
                cfg.AddProfile<EntityToViewModelMap>();
                cfg.AddProfile<ViewModelToEntityMap>();
            });
        }
    }
}
```

```
});

// Create the container as usual.
var container = new Container();
container.Options.DefaultScopedLifestyle
    = new AsyncScopedLifestyle();

// Register your types, for instance using the scoped lifestyle:
container.Register<IDependenteRepository,
    DependenteRepository>(Lifestyle.Scoped);

container.Register<IFuncaoRepository,
    FuncaoRepository>(Lifestyle.Scoped);

container.Register<IFuncionarioRepository,
    FuncionarioRepository>(Lifestyle.Scoped);

container.Register<IDependenteBusiness,
    DependenteBusiness>(Lifestyle.Scoped);

container.Register<IFuncaoBusiness,
    FuncaoBusiness>(Lifestyle.Scoped);

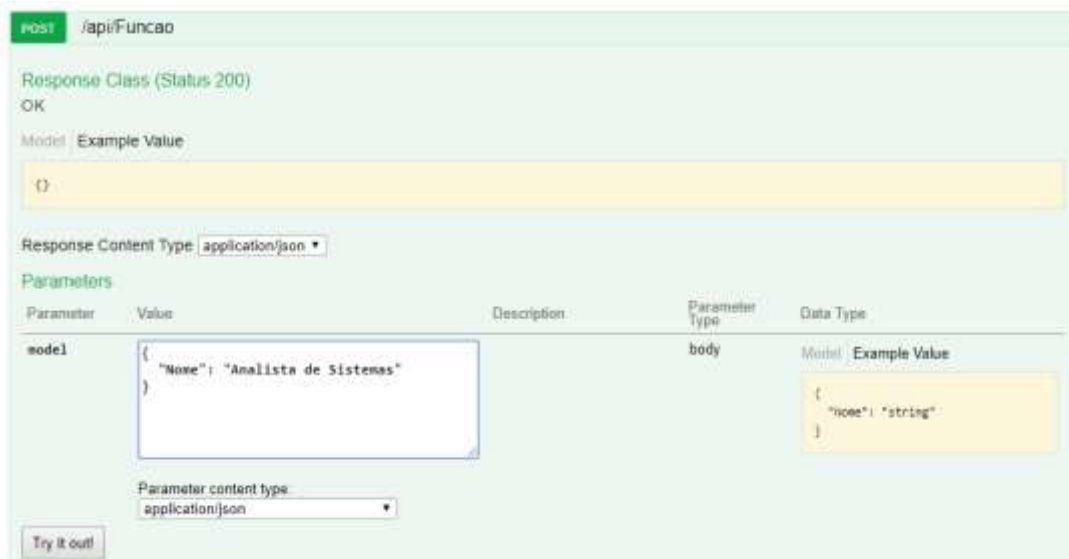
container.Register<IFuncionarioBusiness,
    FuncionarioBusiness>(Lifestyle.Scoped);

// This is an extension method from the integration package.
container.RegisterWebApiControllers
    (GlobalConfiguration.Configuration);

container.Verify();

GlobalConfiguration.Configuration.DependencyResolver =
    new SimpleInjectorWebApiDependencyResolver(container);
}
}
```

Executando:



POST /api/Funcao

Response Class (Status: 200)
OK

Model: Example Value

```
{
  "Nome": "Analista de Sistemas"
}
```

Response Content Type: application/json

Parameter	Value	Description	Parameter Type	Data Type
model	{ "Nome": "Analista de Sistemas" }		body	Model: Example Value { "Nome": "string" }

Parameter content type: application/json

Try it out!

