Бесплатная электронная книга

УЧУСЬ unity-container

Free unaffiliated eBook created from
Stack Overflow contributors.

#unity-container

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: unity-container

It is an unofficial and free unity-container ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official unity-container.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с контейнером-единством

замечания

Контейнер Unity (Unity) представляет собой легкий, расширяемый контейнер для инъекций зависимостей. Это облегчает создание слабосвязанных приложений и предоставляет разработчикам следующие преимущества: упрощенное создание объектов, особенно для иерархических структур объектов и зависимостей. [https://msdn.microsoft.com/en-us/library/ff647202.aspx]

Следует также упомянуть о любых крупных предметах в рамках единства-контейнера и ссылки на соответствующие темы. Поскольку Документация для контейнера единства является новой, вам может потребоваться создать начальные версии этих связанных тем.

Версии

Версия	Примечания к выпуску	Дата выхода
2.0.0	Unity 2	2011-05-05
2.1.0	Unity 2.1	2011-05-11
3.0.0	Unity 3	2013-04-26
3.5.0	Unity 3.5	2015-05-15
4.0.0	Unity передано сообществу OSS	2015-10-06

Examples

Монтаж

Чтобы начать работу, вам просто нужно установить пакет **Unity** nuget. Выполните следующую команду из консоли диспетчера пакетов:

```
PM> Install-Package Unity
```

Кроме того, вы можете использовать Visual Studio для установки Unity в конкретном проекте с помощью опции « *Управление пакетами NuGet для решения»* в меню «Инструменты»> «Диспетчер пакетов NuGet».

Привет, мир

```
interface IGreeter
{
    void Greet();
}

class Greeter : IGreeter
{
    public void Greet()
    {
        Console.WriteLine("Hello World");
    }
}

class SpanishGreeter : IGreeter
{
    public void Greet()
    {
        Console.WriteLine("Hola Mundo");
    }
}

class FrenchGreeter : IGreeter
{
    public void Greet()
    {
        Console.WriteLine("Bonjour le Monde");
    }
}

class Program
{
    static void Main(string[] args)
    {
        var container = new UnityContainer()
            .RegisterType<IGreeter, SpanishGreeter>("spanish")
            .RegisterType<IGreeter, FrenchGreeter>("french")
            .RegisterType<IGreeter, Greeter>();

        //Get default registration. Outputs "Hello World"
        var greeter = container.Resolve<IGreeter>();
        greeter.Greet();

        //Get specific named registration. Outputs "Hola Mundo"
        greeter = container.Resolve<IGreeter>("spanish");
        greeter.Greet();

        //Get all named registrations (excludes the default one)
        //Outputs "Hola Mundo" and "Bonjour le Monde"
        foreach (var g in container.ResolveAll<IGreeter>())
        {
            g.Greet();
        }

        Console.ReadLine();
    }
}
```

Инъекция конструктора

```
interface IService
{
    void ProcessRequest();
}

interface IRepository
{
    IEnumerable<string> GetData();
}

class HelloWorldRepository : IRepository
{
    public IEnumerable<string> GetData()
    {
        return new[] { "Hello", "World" };
    }
}

class HelloWorldService : IService
{
    private readonly IRepository repo;
    public HelloWorldService(IRepository repo)
    {
        this.repo = repo;
    }
    public void ProcessRequest()
    {
        Console.WriteLine(String.Join(" ", this.repo.GetData()));
    }
}

class Program
{
    static void Main(string[] args)
    {
        var container = new UnityContainer()
            .RegisterType<IRepository, HelloWorldRepository>()
            .RegisterType<IService, HelloWorldService>();

        //Unity automatically resolves constructor parameters that knows about.
        //It will return a HelloWorldService with a HelloWorldRepository
        var greeter = container.Resolve<IService>();
        //Outputs "Hello World"
        greeter.ProcessRequest();

        Console.ReadLine();
    }
}
```

Прочитайте Начало работы с контейнером-единством онлайн: https://riptutorial.com/ru/unity-container/topic/5292/начало-работы-с-контейнером-единством

глава 2: Unity WebAPI

Examples

Настройка Unity с помощью Web API.

1. Добавьте Unity в свой проект.

Если вы используете NuGet, вы можете использовать Unity-пакет . Запустите `Install-Package Unity` в консоли диспетчера пакетов. Это добавит библиотеку Unity (и ее зависимости) к вашему проекту.

2. Создайте реализацию `IDependencyResolver` .

Например:

```
public class UnityResolver : IDependencyResolver
{
    protected IUnityContainer Container;

    public UnityResolver(IUnityContainer container)
    {
        if (container == null)
        {
            throw new ArgumentNullException("container");
        }
        this.Container = container;
    }

    public object GetService(Type serviceType)
    {
        try
        {
            return Container.Resolve(serviceType);
        }
        catch (ResolutionFailedException)
        {
            return null;
        }
    }

    public IEnumerable<object> GetServices(Type serviceType)
    {
        try
        {
            return Container.ResolveAll(serviceType);
        }
        catch (ResolutionFailedException)
        {
            return new List<object>();
        }
    }

    public IDependencyScope BeginScope()
```

```
    {
        var child = Container.CreateChildContainer();
        return new UnityResolver(child);
    }

    public void Dispose()
    {
        Container.Dispose();
    }
}
```

3. Зарегистрируйте свой `IDependencyResolver` в своем `WebApiConfig`.

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        // Routes goes here..

        // Create your container.
        var container = new UnityContainer();

        // Do registrations here...

        // Assign your container.
        config.DependencyResolver = new UnityResolver(container);
    }
}
```

Прочитайте Unity WebAPI онлайн: https://riptutorial.com/ru/unity-container/topic/6396/unity-webapi

кредиты

S. No	Главы	Contributors
1	Начало работы с контейнером-единством	Clemens Tolboom, Community, Daniel J.G.
2	Unity WebAPI	smoksnes