Fody

Против рутины

Константин Васильев, Очень Интересно

О себе

- .NET 4+ лет
- SQL
- Knockout, Angular 1/2
- .NET @ Очень Интересно

Fody

Средство для манипуляций над IL кодом ваших сборок, встраивается в билд-процесс

Позволяет упростить некоторые рутинные задачи

Ближайший аналог - PostSharp

Проекты, использующие Fody

- Catel
- Enums.NET
- NServiceBus
- Orchestra & all Orc.* components

Конфигурация

FodyWeavers.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<Weavers>
  <Cauldron.Interception />
  <Anotar.NLog />
  <PropertyChanged CheckForEquality='false'/>
</Weavers>
```

Конфигурация

FodyWeavers.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <Weavers>
3    <Cauldron.Interception />
4    <Anotar.NLog />
5    <PropertyChanged CheckForEquality='false'/>
6  </Weavers>
```

INotifyPropertyChanged

```
public class User : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    private string firstName;
    public string FirstName
    {
        get => firstName;
        set
        {
            firstName = value;
            OnPropertyChanged(nameof(FirstName));
            OnPropertyChanged(nameof(FullName));
        }
    }

    private string lastName;
    public string LastName
    {
        get => lastName;
        set
        {
            lastName = value;
            OnPropertyChanged(nameof(LastName));
            OnPropertyChanged(nameof(FullName));
        }
    }

    public string FullName => $"{FirstName} {LastName}";

    public virtual void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

INotifyPropertyChanged

```cs
public class User : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    private string firstName;
    public string FirstName
    {
        get => firstName;
        set
        {
            firstName = value;
            OnPropertyChanged(nameof(FirstName));
            OnPropertyChanged(nameof(FullName));
        }
    }

    private string lastName;
    public string LastName
    {
        get => lastName;
        set
        {
            lastName = value;
            OnPropertyChanged(nameof(LastName));
            OnPropertyChanged(nameof(FullName));
        }
    }

    public string FullName => $"{FirstName} {LastName}";

    public virtual void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

```cs
public class User : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    public string FirstName { get; set; }

    public string LastName { get; set; }

    public string FullName => $"{FirstName} {LastName}";
}
```

Equals

```
public class Point
{
    public int X { get; set; }

    public int Y { get; set; }

    private static bool EqualsInternal(Point left, Point right)
    {
        return left.X == right.X && left.Y == right.Y;
    }

    public override bool Equals(object right)
    {
        return !ReferenceEquals(null, right)
            && (ReferenceEquals(this, right)
                || GetType() == right.GetType()
                && EqualsInternal(this, (Point)right));
    }

    public override int GetHashCode()
    {
        return unchecked(X.GetHashCode() * 397 ^ Y.GetHashCode());
    }
}
```

Equals

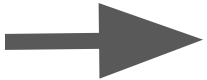
```
public class Point
{
    public int X { get; set; }

    public int Y { get; set; }

    private static bool EqualsInternal(Point left, Point right)
    {
        return left.X == right.X && left.Y == right.Y;
    }

    public override bool Equals(object right)
    {
        return !ReferenceEquals(null, right)
            && (ReferenceEquals(this, right)
                || GetType() == right.GetType()
                && EqualsInternal(this, (Point)right));
    }

    public override int GetHashCode()
    {
        return unchecked(X.GetHashCode() * 397 ^ Y.GetHashCode());
    }
}
```



```
[Equals]
public class Point
{
    public int X { get; set; }

    public int Y { get; set; }
}
```


if err != nil {
return nil, err
}
Strg

???

Fody

- .NET 4.5.2+
- .NET Core 1.0+

Fody

- .NET 4.5.2+
- .NET Core 1.0+

Не поддерживаются проекты:

- project.json
- xproj
- .csproj в старом формате

Fody

- .NET 4.5.2+
- .NET Core 1.0+

Не поддерживаются проекты:

- project.json
- xproj
- .csproj в старом формате

Не работает функция “Edit and continue”

Процесс билда с Fody

Fody добавляет дополнительные шаги в пайплайн MSBuild.
Когда билдится проект, Fody:

- Находит файлы сборки и pdb файлы

Процесс билда с Fody

Fody добавляет дополнительные шаги в пайплайн MSBuild.
Когда билдится проект, Fody:

- Находит файлы сборки и pdb файлы
- Парсит их с помощью Mono.Cecil

Процесс билда с Fody

Fody добавляет дополнительные шаги в пайплайн MSBuild.
Когда билдится проект, Fody:

- Находит файлы сборки и pdb файлы
- Парсит их с помощью Mono.Cecil
- По порядку выполняет все включённые плагины

Процесс билда с Fody

Fody добавляет дополнительные шаги в пайплайн MSBuild.
Когда билдится проект, Fody:

- Находит файлы сборки и pdb файлы
- Парсит их с помощью Mono.Cecil
- По порядку выполняет все включённые плагины
- Изменённая сборка сохраняется вместе с обновлённым pdb файлом

ToString

Автоматически генерирует метод ToString для классов

```
public class User
{
    public string FirstName { get; set; }

    public string LastName { get; set; }

    public override string ToString()
    {
        return string.Format(
            CultureInfo.InvariantCulture,
            "{{T: User, FirstName: {0}, LastName: {1}}}",
            this.FirstName,
            this.LastName);
    }
}
```

Obsolete

Предоставляет удобный интерфейс для использования атрибута Obsolete

```
[ObsoleteEx(
        Message = "Not secure",
        TreatAsErrorFromVersion = "2.0",
        RemoveInVersion = "4.0",
        ReplacementTypeOrMember = "UserService")]
public class UserHelper
{
    // ...
}
```

Obsolete

Предоставляет удобный интерфейс для использования атрибута Obsolete

```
[ObsoleteEx(
        Message = "Not secure",
        TreatAsErrorFromVersion = "2.0",
        RemoveInVersion = "4.0",
        ReplacementTypeOrMember = "UserService")]
public class UserHelper
{
    // ...
}
```

Obsolete

Предоставляет удобный интерфейс для использования атрибута Obsolete

```
[ObsoleteEx(
        Message = "Not secure",
        TreatAsErrorFromVersion = "2.0",
        RemoveInVersion = "4.0",
        ReplacementTypeOrMember = "UserService")]
public class UserHelper
{
    // ...
}
```

Undisposed

Помогает отслеживать не освобождённые из памяти объекты

Работает только с пользовательскими типами

Null

- Внезапные NullReferenceException

Null

- Внезапные NullReferenceException
- Компилятор беспомощен

NullGuard

Автоматически добавляет проверки на null

Работает для свойств и методов (входные и выходные значения)

В режиме Debug вместо исключений вызывает Debug.Assert

Anotar

Добавляет абстракцию над используемым логгером

Поддерживает Log4Net, NLog, Serilog, Catel, CommonLogging, LibLog, NServiceBus, Splat

 Anotar

Добавляет абстракцию над используемым логгером

Поддерживает Log4Net, NLog, Serilog, Catel, CommonLogging, LibLog, NServiceBus, Splat

Предоставляет:

- Статичные методы для логгирования
- Атрибуты для логгирования необработанных исключений в методе
- Статичные свойства для проверки, включено ли логгирование какого-либо уровня

Anotar

```
public class UserHelper
{
    public string GetName(User user)
    {
        LogTo.Debug("Getting Name");
        return "";
    }
}
```

Anotar

```
public class UserHelper
{
    public string GetName(User user)
    {
        LogTo.Debug("Getting Name");
        return "";
    }
}
```

```
public class UserHelper
{
    static Logger logger = LogManager.GetLogger("UserHelper");

    public string GetName(User user)
    {
        logger.Debug("Method: 'String GetName(User)'. Line: ~21. Getting Name");
        return "";
    }
}
```

Anotar

```
public class UserHelper
{
    public string GetName(User user)
    {
        LogTo.Debug("Getting Name");
        return "";
    }
}
```

```
public class UserHelper
{
    static Logger logger = LogManager.GetLogger("UserHelper");

    public string GetName(User user)
    {
        logger.Debug("Method: 'String GetName(User)'. Line: ~21. Getting Name");
        return "";
    }
}
```

QueryValidator

Валидирует SQL-запросы во время компиляции

```
        public IEnumerable<User> GetActive()
        {
            var sql = @"|>
Select  Id,
        FirstName,
        LastName
From    User
Where   IsDeleted = 0";
            return connection.Query<User>(sql);
        }
    }
}
```

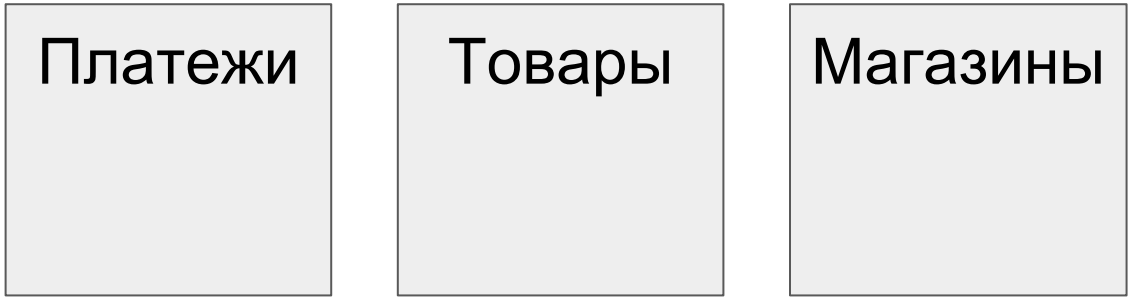
Аспектно-ориентированное программирование

- Проблема отделения сквозной функциональности

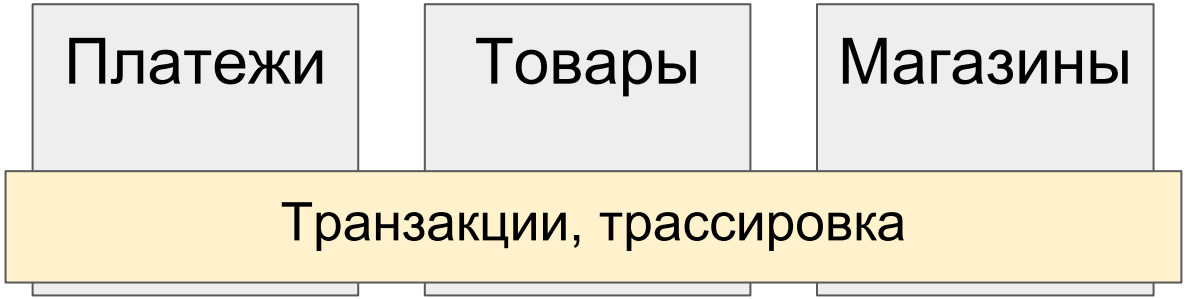
Аспектно-ориентированное программирование

- Проблема отделения сквозной функциональности
- Аспекты

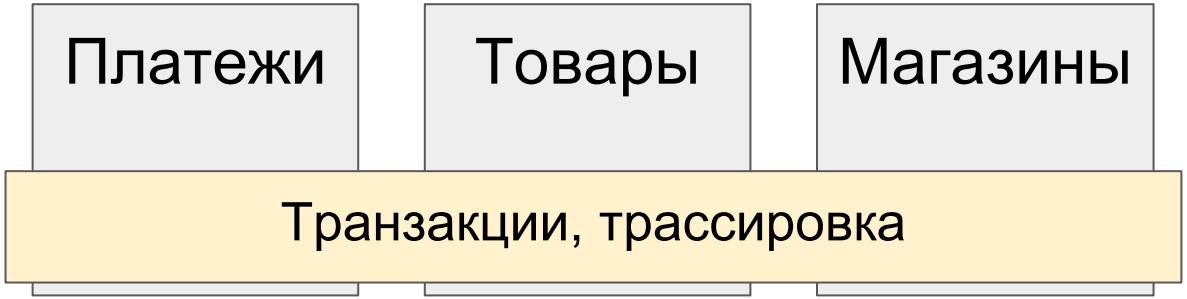
Без АОП



Без АОП



Без АОП


Платежи
Товары
Магазины
Транзакции, трассировка

АОП


Платежи
Товары
Магазины
Транзакции
Трассировка

Cauldron

Позволяет добавлять интерцепторы для методов/конструкторов/свойств

```
public class MyInterceptorAttribute : Attribute, IMethodInterceptor
{
    public void OnEnter(Type declaringType, object instance, MethodBase methodbase, object[] values)
    {
    }

    public void OnException(Exception e)
    {
    }

    public void OnExit()
    {
    }
}
```

Cauldron

```
public string GetName(User user)
{
    OnEnter(/*...*/);
    try
    {
        var result = $"{user.FirstName} {user.LastName}";
        OnExit();
        return result;
    }
    catch (Exception ex)
    {
        OnException(ex);
        throw;
    }
}
```

Собственный плагин

```
public class ModuleWeaver : BaseModuleWeaver
{
    public override void Execute()
    {
        var writeLineMethod = typeof(Trace).GetMethod(nameof(Trace.WriteLine), new[] { typeof(string) });
        var methodRef = ModuleDefinition.ImportReference(writeLineMethod);

        foreach (var type in ModuleDefinition.GetTypes().Where(t => t.IsClass))
        {
            var defaultConstructor = type
                .GetConstructors()
                .FirstOrDefault(c => c.Parameters.Count == 0);
            if (defaultConstructor == null)
                continue;

            var instructions = defaultConstructor.Body.Instructions;
            var message = $"Instance of {type.Name} was created";
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Ldstr, message));
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Call, methodRef));
        }
    }

    //...
}
```

Собственный плагин

```
public class ModuleWeaver : BaseModuleWeaver
{
    public override void Execute()
    {
        var writeLineMethod = typeof(Trace).GetMethod(nameof(Trace.WriteLine), new[] { typeof(string) });
        var methodRef = ModuleDefinition.ImportReference(writeLineMethod);

        foreach (var type in ModuleDefinition.GetTypes().Where(t => t.IsClass))
        {
            var defaultConstructor = type
                .GetConstructors()
                .FirstOrDefault(c => c.Parameters.Count == 0);
            if (defaultConstructor == null)
                continue;

            var instructions = defaultConstructor.Body.Instructions;
            var message = $"Instance of {type.Name} was created";
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Ldstr, message));
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Call, methodRef));
        }
    }

    //...
}
```

Собственный плагин

```
public class ModuleWeaver : BaseModuleWeaver
{
    public override void Execute()
    {
        var writeLineMethod = typeof(Trace).GetMethod(nameof(Trace.WriteLine), new[] { typeof(string) });
        var methodRef = ModuleDefinition.ImportReference(writeLineMethod);

        foreach (var type in ModuleDefinition.GetTypes().Where(t => t.IsClass))
        {
            var defaultConstructor = type
                .GetConstructors()
                .FirstOrDefault(c => c.Parameters.Count == 0);
            if (defaultConstructor == null)
                continue;

            var instructions = defaultConstructor.Body.Instructions;
            var message = $"Instance of {type.Name} was created";
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Ldstr, message));
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Call, methodRef));
        }
    }

    //...
}
```

Собственный плагин

```
public class ModuleWeaver : BaseModuleWeaver
{
    public override void Execute()
    {
        var writeLineMethod = typeof(Trace).GetMethod(nameof(Trace.WriteLine), new[] { typeof(string) });
        var methodRef = ModuleDefinition.ImportReference(writeLineMethod);

        foreach (var type in ModuleDefinition.GetTypes().Where(t => t.IsClass))
        {
            var defaultConstructor = type
                .GetConstructors()
                .FirstOrDefault(c => c.Parameters.Count == 0);
            if (defaultConstructor == null)
                continue;

            var instructions = defaultConstructor.Body.Instructions;
            var message = $"Instance of {type.Name} was created";
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Ldstr, message));
            instructions.Insert(instructions.Count - 1, Instruction.Create(OpCodes.Call, methodRef));
        }
    }

    //...
}
```

Плюсы

+ Позволяет существенно упростить некоторые рутинные задачи

Плюсы

+ Позволяет существенно упростить некоторые рутинные задачи
+ Легко встраивается в проект

Плюсы

+ Позволяет существенно упростить некоторые рутинные задачи
+ Легко встраивается в проект
+ Можно писать собственные плагины

Минусы

- Очень неочевидная вещь

Минусы

- Очень неочевидная вещь
- Усложняет процесс дебага

Минусы

- Очень неочевидная вещь
- Усложняет процесс дебага
- Некоторые плагины нужно допиливать

Минусы

- Очень неочевидная вещь
- Усложняет процесс дебага
- Некоторые плагины нужно допиливать
- Писать с нуля плагин - не просто

Ссылочки

https://github.com/Fody/Fody

http://www.mono-project.com/Cecil/

https://habrahabr.ru/post/309462/

https://www.postsharp.net/

Спасибки

- vk.com/ko_vasilev
- github.com/ko-vasilev
- oppa.kostya.bko@gmail.com