Продвинутое использование NuGet и MSBuild

Игорь Лабутин, 13.04.2017
ilabutin@gmail.com

О себе

- 16 лет в разработке ПО
  - C/C++, Linux, QNX, Embedded
  - Последние 9 лет – .NET (C#)
- Интересы
  - Сети, протоколы обмена данными
  - Проблемы производительности
  - Сборка проектов и удобство разработчиков

О докладе

- NuGet
  - Создание
  - Использование
- MSBuild – написание собственного кода
  - Как
  - Зачем?!

- Всё это для VS 2015 & NuGet 3.5

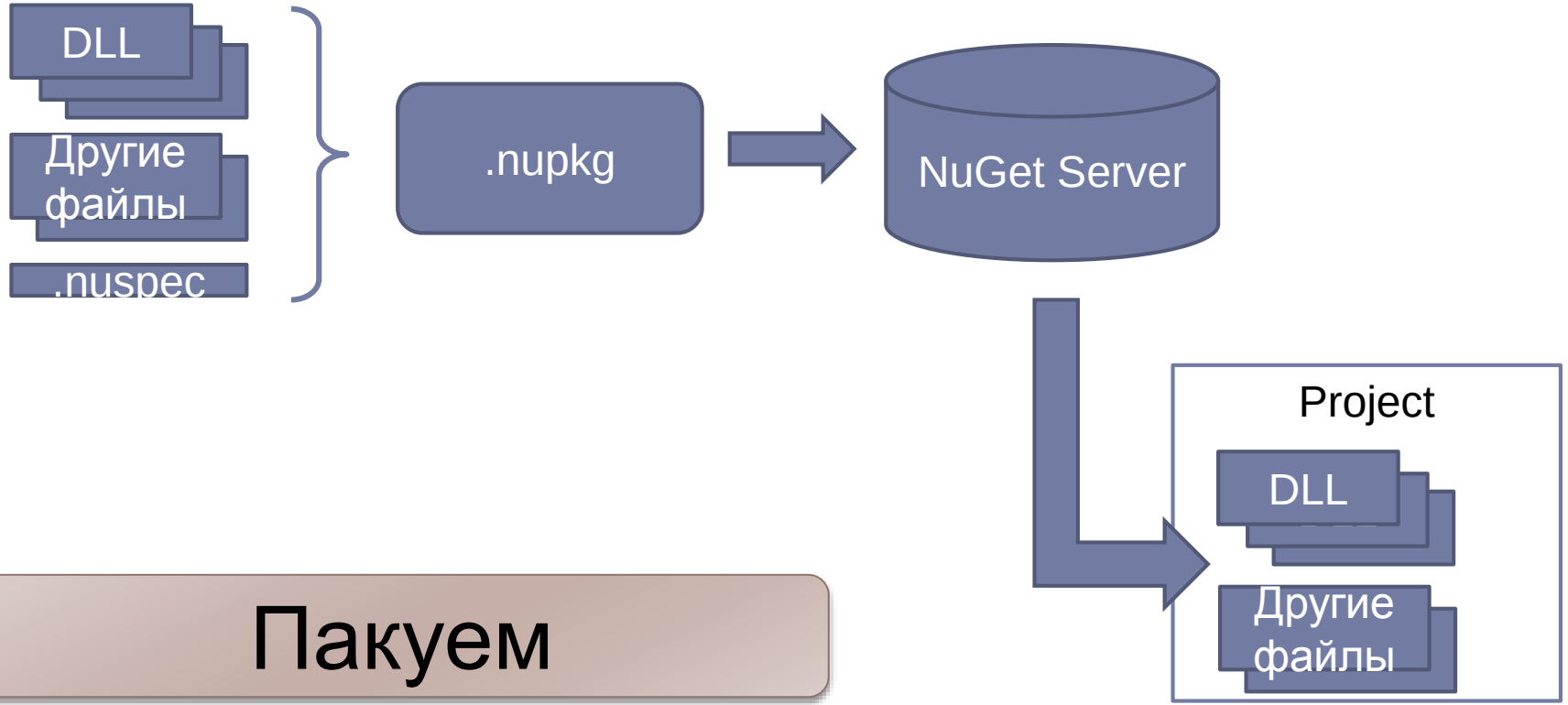
О примере

- Кратко
  - Читаем XML
  - Проверяем соответствие схеме (XSD)
  - Пишем в консоль

Посмотрим код

- Детали
  - Генерация XSD->C#
  - Много ручной работы ☹
  - Но можно автоматизировать (PreBuild)!

Полезное – в NuGet



Результат

- Код чтения XML – в NuGet
  - Но в основном проекте все еще нужно использовать PreBuild Action
- VS не всегда корректно понимает необходимость сборки
  - `DisableFastUpToDateCheck` помогает, но тогда сборка происходит чаще чем хотелось

Помечтаем?

- ☺ Код чтения XML – в NuGet
- ☺ NuGet пакет можно использовать из VS UI
- ☹ Нет необходимости менять .csproj вручную
- ☹ Конвертацию XSD в CS настраивается из VS UI
- ☹ Сборка корректно учитывает состояние файлов

Долой ручной труд

- Создаем свой собственный MSBuild Target: `TransformXsd2Cs`
  - Надо как-то знать список XSD файлов
- Добавляем в NuGet пакет
  - `build\SuperSerializer.targets`
- Но как его оттуда вызывать…

Вызываем Target

- SuperSerializer.targets импортируется автоматически
- Для вызова можно использовать
  - `CallTarget`
  - `DependsOnTargets`
  - `BeforeTarget/AfterTarget`
- Стараться использовать `DependsOnTargets`
- MSBuild для .csproj позволяет встраиваться почти везде

```
<Target
    Name="Build" DependsOnTargets="$(BuildDependsOn)">
```

Применяем

Результат

- ☺ Код чтения XML – в NuGet
- ☺ NuGet пакет можно использовать из VS UI
- 😐 Нет необходимости менять .csproj вручную
- ☹ Конвертацию XSD в CS настраивается из VS UI
- ☹ Сборка корректно учитывает состояние файлов

Настраиваем сборку

- MSBuild умеет понимать надо ли собирать проект
  - Но ему надо чуть-чуть подсказать
- Задаем `Inputs/Outputs` атрибуты у `TransformXsd2Cs`

```
<Target
  Name="TransformXsd2Cs“
  Inputs="@(Xsd2Cs)“
  Outputs="@(Xsd2Cs->'%(RootDir)%(Directory)%(Filename).cs')">
```

Всё в GUI

- Сообщаем VS что есть новый build action

```
<ItemGroup>
  <AvailableItemName Include="Xsd2Cs" />
</ItemGroup>
```

- Теперь можно не редактировать .csproj

Результат

Результат

- ☺ Код чтения XML – в NuGet
- ☺ NuGet пакет можно использовать из VS UI
- ☺ Нет необходимости менять .csproj вручную
- ☺ Конвертацию XSD в CS настраивается из VS UI
- ☺ Сборка корректно учитывает состояние файлов

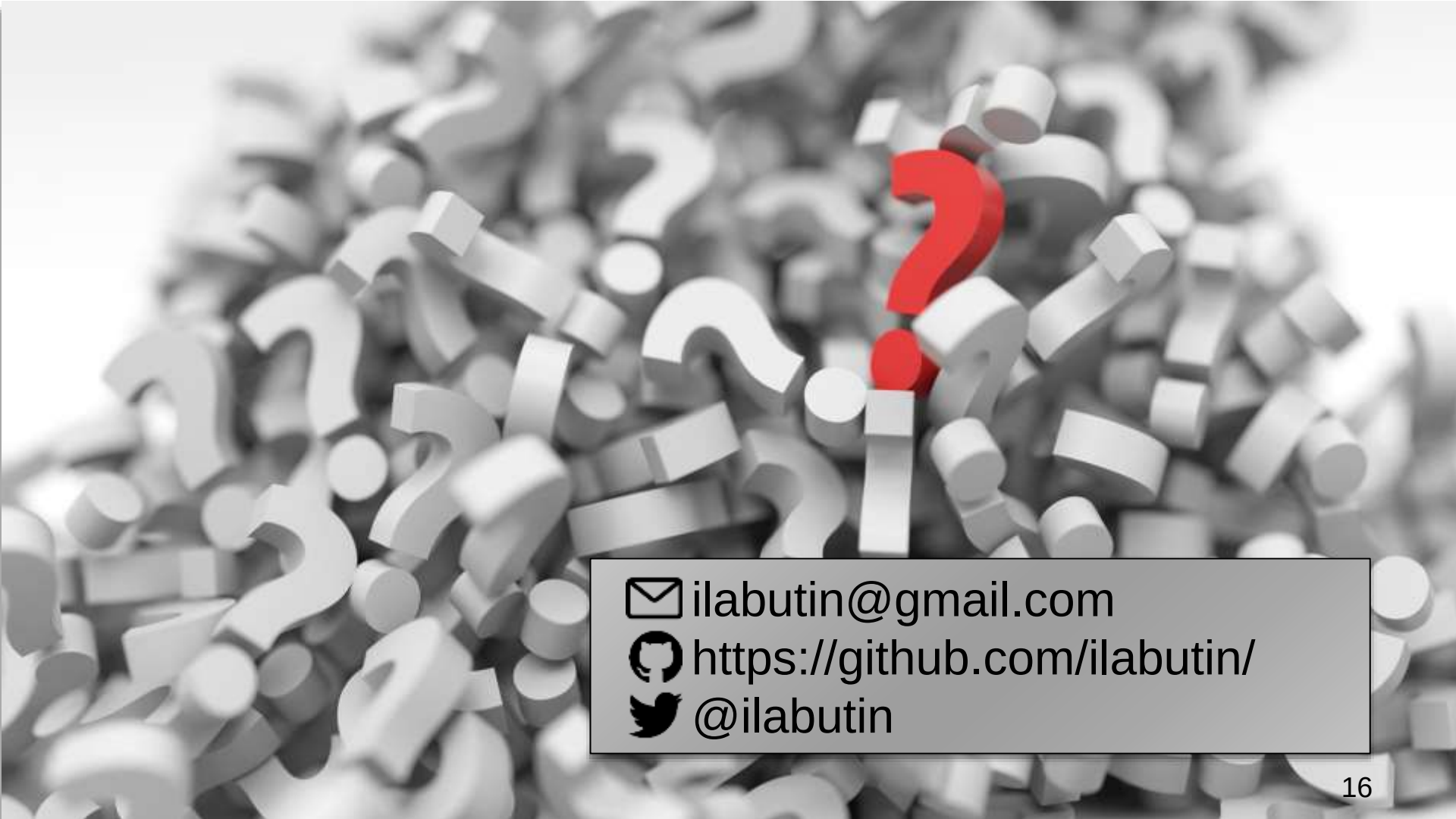
Последние штрихи

- Добавить readme.txt
- Без powershell не обойтись
  - Добавляем build\install.ps1 для модификации .csproj

Результат

Что дальше?

- Visual Studio 2017
  - Новый формат .csproj
  - Новый NuGet 4.0
  - Нет поддержки install.ps1
- Кроссплатформенность
  - .NET Standard



ilabutin@gmail.com
https://github.com/ilabutin/
@ilabutin