

# Использование XSLT для разработки сайтов (на ASP.NET)

Андрей Майоров, BYTE-force  
xor@byte-force.com  
twitter.com/xorets



<http://www.devconf.ru>

## Что нужно от языка шаблонов?

- Задача - показывать данные
  - В виде HTML
  - И не только в HTML
- Шаблон легко делается из HTML
- Общие фрагменты выносятся в отдельные файлы
- Работает быстро
- Кроссплатформенный и стандартный

**Все это – XSLT**

Уже готовый. С блэкджеком и ...



## Команды XSLT

stylesheet

transform

import

include

output

template

param

variable

copy

apply-imports

apply-templates

call-template

with-param

value-of

copy-of

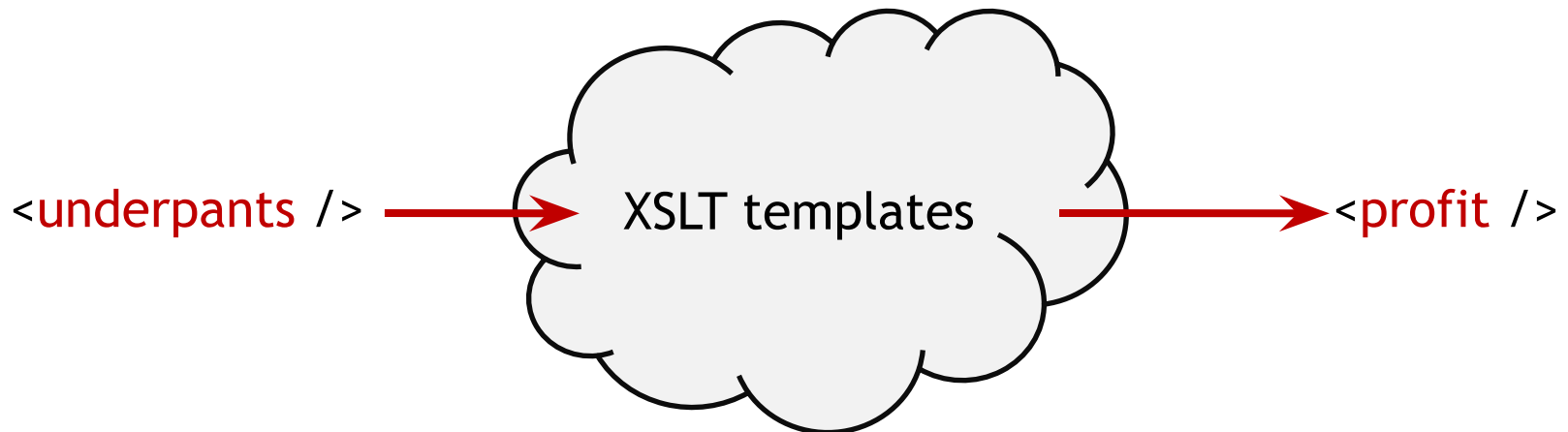
## XSLT ~ функциональный язык

- Декларативный язык: не 100% функциональный, но точно не императивный.
- Нельзя писать как в императивном - получится плохо.
- Правильный подход позволяет добиться невозможных в императивном языке вещей.

## Основные преимущества XSLT

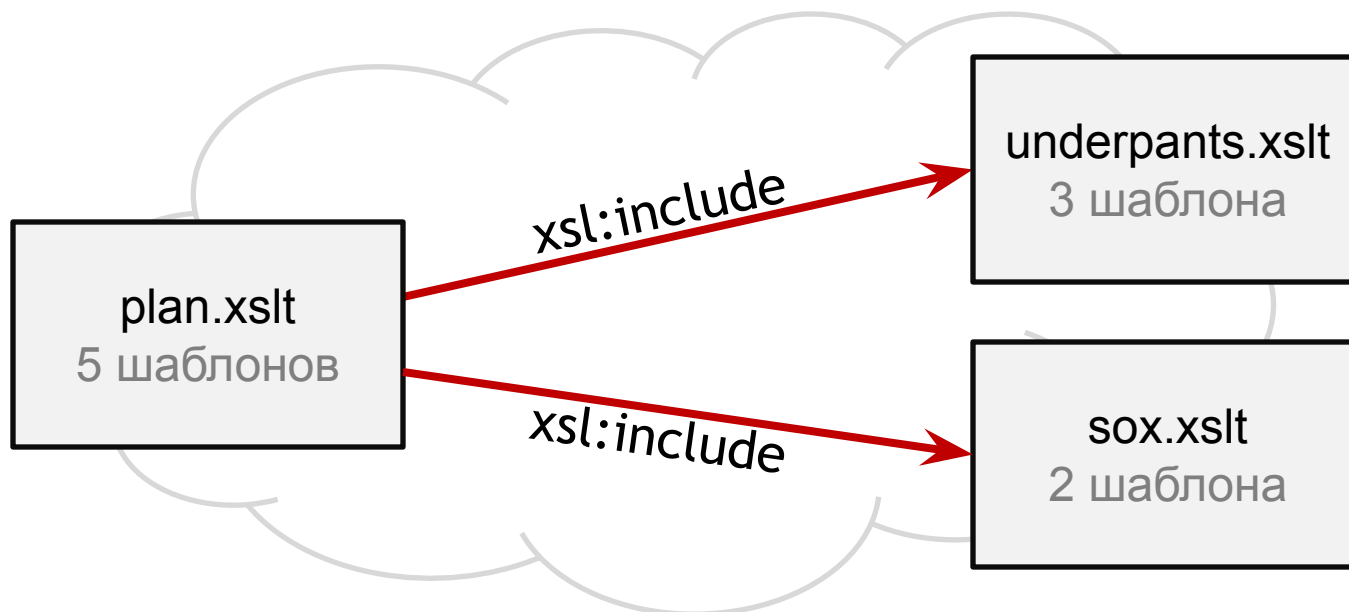
### Data driven

- Именно данные, находящиеся на конвейере обработки, управляют всем процессом.



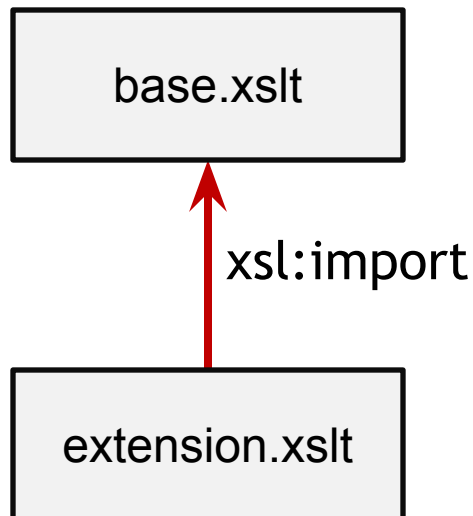
### Модульность

- Программа на XSLT состоит из независимых шаблонов, отвечающих на разные входные данные.



### Наследование

- Директива `import` позволяет устанавливать отношения, сходные с наследованием.





# Demo

## Закрепление материала

# xsl:import

- Используйте для наследования
- `<xsl:apply-imports>` - вызов базового шаблона
- `<xsl:apply-templates select="." mode="..." />`
  - template method, вызов в базовом шаблоне.
- `<xsl:call-template name="..." />`
  - вызывает шаблон, заданный последним.

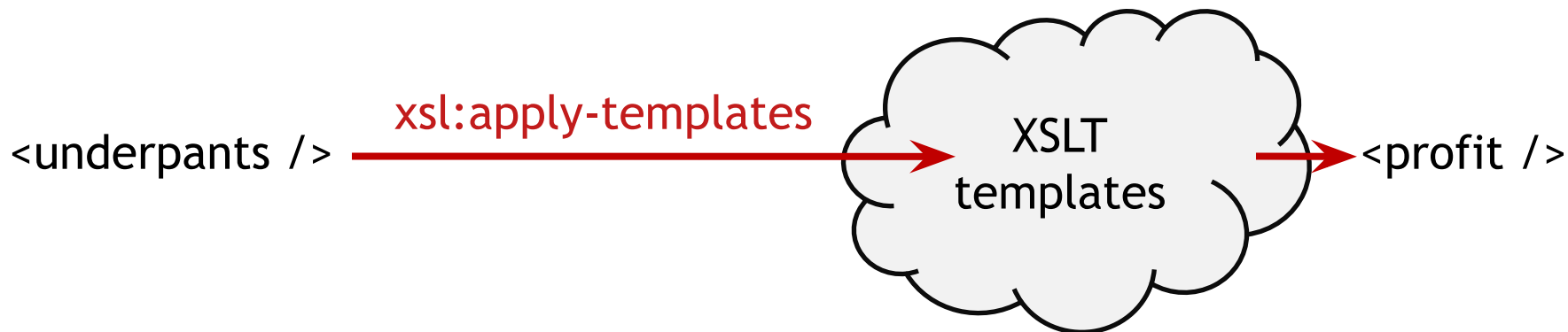
Закрепление материала

## **xsl:include**

- Используйте для подключения компонентов
- Просто вставка одного файла внутрь другого

# xsl:apply-templates

- Основной инструмент!
- Отдает узлы в обработку - позволяет делать волшебные преобразования.
- Использовать вместо: xsl:copy-of, xsl:for-each, xsl:choose.



Закрепление материала

## Работа с переменными

- Можно только проинициализировать, менять значение нельзя.
- Цикл с увеличением счетчика - только через рекурсивный вызов.

## Домашняя работа

- XPath.
- Функции XSLT и XPath.
- EXSLT.
- Поддержка в разных платформах.

## MVC в веб-разработке

- Model-View-Controller.
- Отделяет разработку UI от разработки бизнес-логики.
- Члены команды работают параллельно:
  - Верстальщик делает интерфейс.
  - Серверный программист - контроллеры.

**В каком формате передавать данные между  
controller и view?**



## Передавать типизированный объект - неудобно

1. Типизированный объект - это код, его еще надо написать.
2. Наполнить объект данными - опять нужен код.



Кто его напишет, когда  
еще нет бизнес-логики?

## Храним промежуточные данные в XML

- Данные удобно прототипировать вручную. В простом текстовом редакторе.
- Не нужно ждать готовности контроллера. Данные для тестов уже есть в XML-документе.
- Этими данными можно тестировать и сам контроллер.

## Сделали прототип и пошли работать...



Верстальщик

```
<poem>  
  <title>Руслан и Людмила</title>  
  <rating>5</rating>  
  <description>...</description>  
</poem>
```



Программист


## XML в веб-приложениях

- В веб часть данных уже в HTML. Например, текст новостной статьи.
- Взяв XHTML, можем объединить все данные в один XML-документ:
  - Сильно структурированные — поля объектов.
  - Слабо структурированные — данные от пользователя.

## Объединение слабо и сильно типизированных данных

```

<poem id="156">
  <title>Руслан и Людмила</title>
  <rating>5</rating>
  <description>
    «Руслан и Людмила» —
    первая законченная поэма
    Александра Пушкина.
  </description>
</poem>
  
```

Поэм *			
	Column Name	Data Type	Allow Nulls
	poem_id	int	<input type="checkbox"/>
	title	nvarchar(256)	<input type="checkbox"/>
	rating	float	<input checked="" type="checkbox"/>
	description	xml	<input type="checkbox"/>
			<input type="checkbox"/>

## Критика XML в качестве модели

- Бизнес-логике неудобно работать с XML.
- Надо работать с типизированной моделью.
- Вручную преобразовывать объект в XML - мартышкин труд.
- XmlSerializer имеет свои ограничения.

## Возьмите XPathNavigator

- Позволяет работать с графом объектов, как будто это XML-документ.
- «Ленивый»
- Совместим с XmlSerializer.
- Расширяем.



## XsltView для ASP.NET MVC

- Нужен



## Чем плох MvcContrib.XsltViewEngine?

- Использование специального объекта XsltViewData - нельзя сменить view, не меняя контроллер.
- Не умеет преобразовывать модель в XML.

```
public class XsltView : IView
{
    private readonly MvpXsltTransform _transform;
    private readonly XmlResolver _resolver;

    public XsltView( MvpXsltTransform transform, XmlResolver resolver )
    {
        _transform = transform;
        _resolver = resolver;
    }

    public void Render( ViewContext viewContext, TextWriter writer )
    {
        var ctx = GetXPathContext();

        _transform.Transform(
            new XmlInput( ctx.CreateNavigator( viewContext.ViewData ), _resolver ),
            null,
            new XmlOutput( writer ) );
    }

    private static ObjectXPathContext GetXPathContext()...
}
}
```

# Формы в XSLT

## Сложности с формами

- Формы задаются в двух местах:
  - В XSLT задаем HTML-форму
  - В модели - поля данных, со статусами валидации, сообщениями и др.
- Возникает соблазн сделать «язык форм» на базе XML-модели.
  - Это сложное и громоздкое решение
- В принципе, те же проблемы у ASP.NET MVC

## ASP.NET Web Forms:

- Странные
  - Очень громоздкая обработка запроса
  - Выдают избыточный, страшноватый маркап
- Понятные
  - Разработка веб-форм в VS очень проста и интуитивно понятна
- Хочется минимизировать страх, оставив только пользу

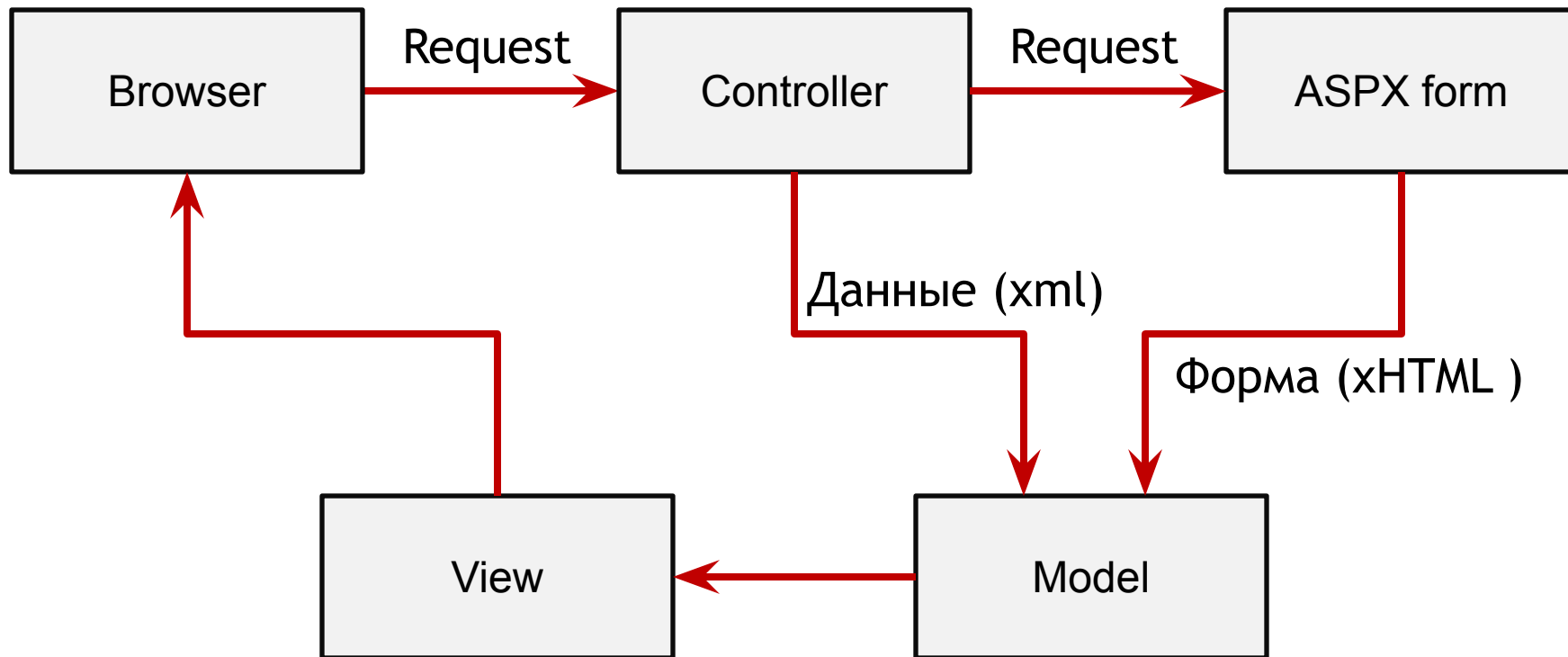
## Решение

- Совмещаем оба подхода.
- От веб-формы берем только поля ввода.
- Весь дизайн прикладываем при помощи XSLT.

## Решение подробнее

1. Веб-форма производит xHTML.
2. xHTML=XML. **Контроллер** вставляет форму в **модель**.
3. **View** копирует код формы наружу, в нужное место страницы.
4. Browser. Post back.
5. **Контроллер** передает пост-бэки в ASPX-файл.
6. goto 1.

## Решение в картинках



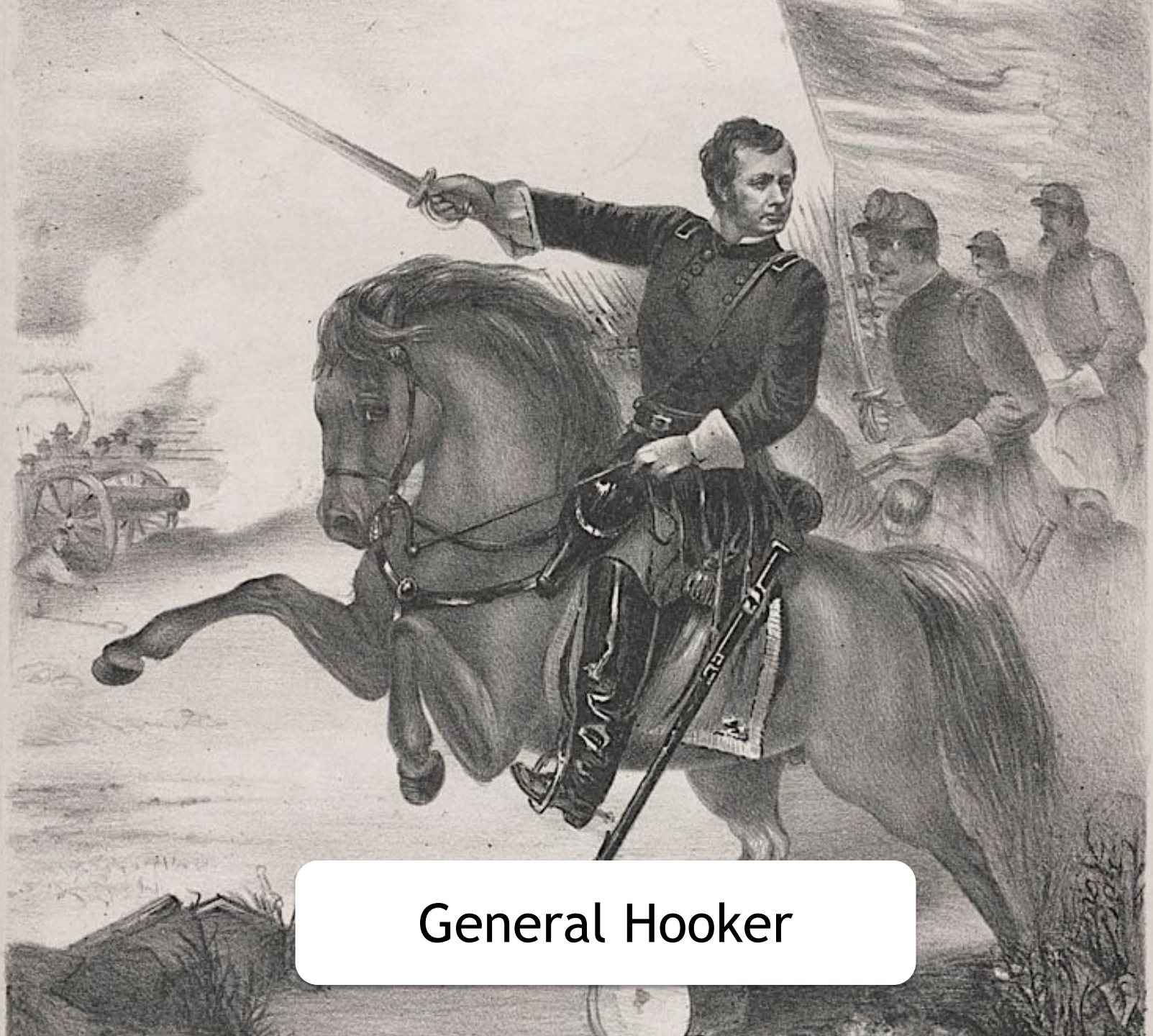


## Технические сложности

- `HttpServerUtility.Execute` не дает доступа к полям формы.
- Берем код `Execute` рефлексором.
- Некоторые нужные части `HttpContext` и `HttpResponse` закрыты.  
Используем reflection:
  - `HttpContext.SetCurrentHandler`
  - `HttpContext.RestoreCurrentHandler`
  - `HttpContext.SwitchWriter`

## Итоги

1. XSLT - мощный язык шаблонов
2. XML удобен в качестве модели в MVC
3. ObjectXPathNavigator позволяет типизированные модели
4. Примирает с веб-формами



General Hooker

**Спасибо за внимание!**

Андрей Майоров, BYTE-force

[xor@byte-force.com](mailto:xor@byte-force.com)

[twitter.com/xorets](https://twitter.com/xorets)

[blogs.byte-force.com/xor](http://blogs.byte-force.com/xor)