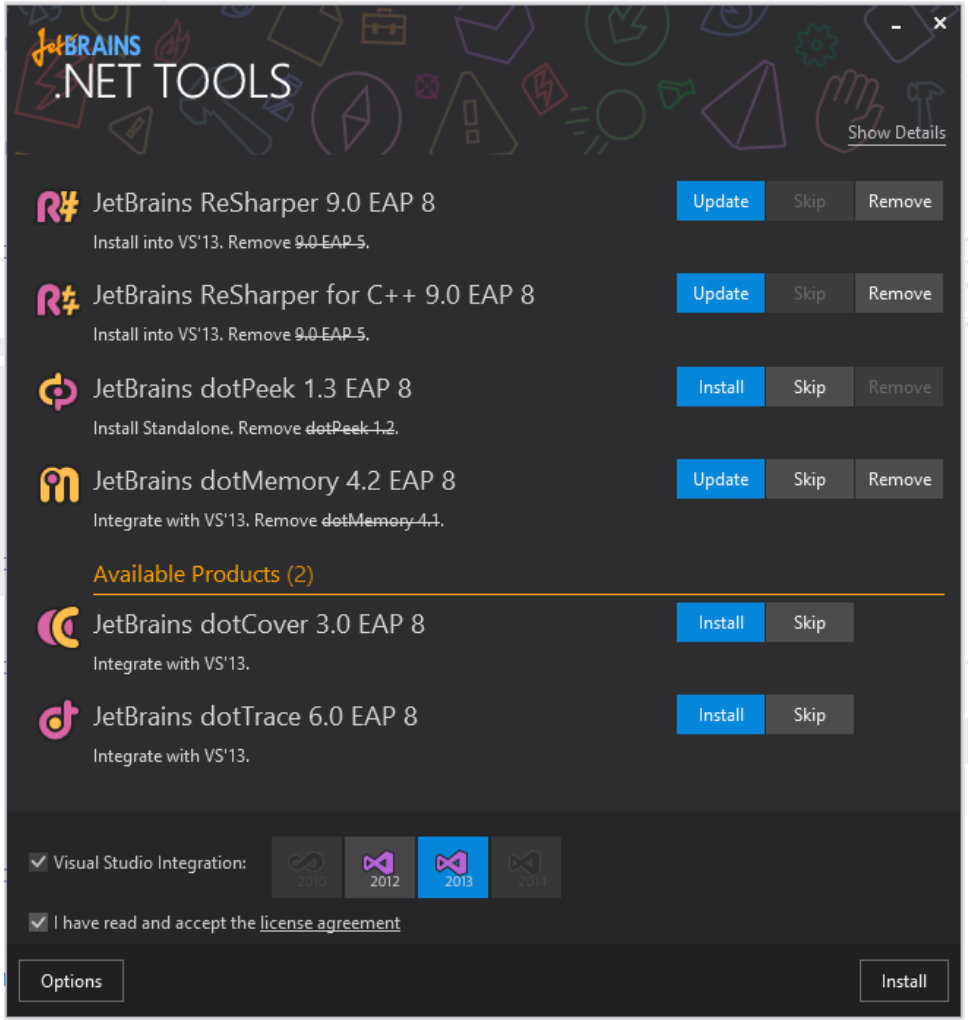
Почему только C#

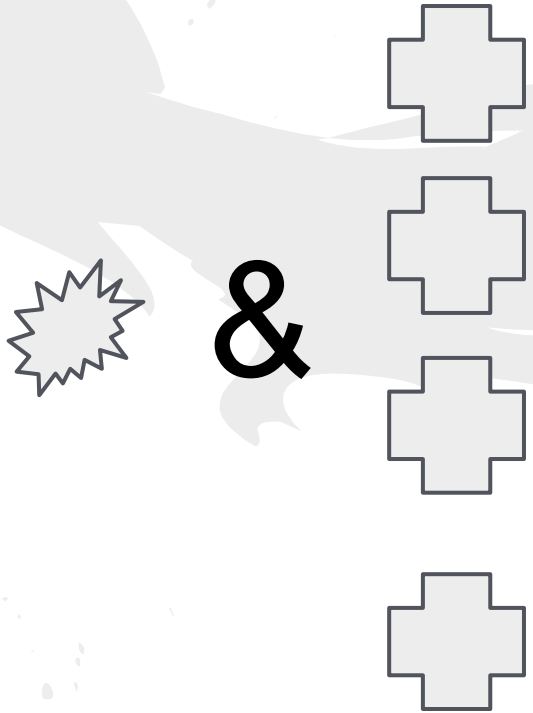
@serjic JetBrains

.Next

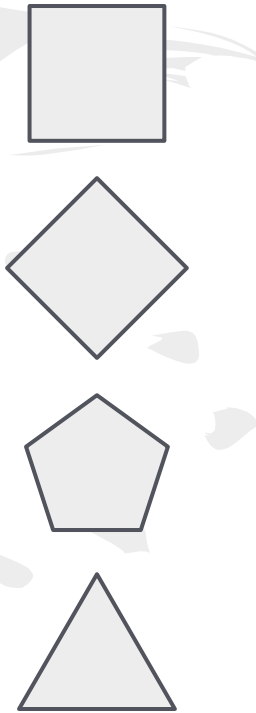
- .Net Core
- ASP.Net vNext
- Roslyn and C# 6
- Xamarin and Mono
- Visual Studio Community
- ...


JetBRAINS
.NET TOOLS
Show Details
JetBrains ReSharper 9.0 EAP 8
Update
Skip
Remove
Install into VS'13. Remove 9.0 EAP 5.
JetBrains ReSharper for C++ 9.0 EAP 8
Update
Skip
Remove
Install into VS'13. Remove 9.0 EAP 5.
JetBrains dotPeek 1.3 EAP 8
Install
Skip
Remove
Install Standalone. Remove dotPeek 1.2.
JetBrains dotMemory 4.2 EAP 8
Update
Skip
Remove
Integrate with VS'13. Remove dotMemory 4.1.
Available Products (2)
JetBrains dotCover 3.0 EAP 8
Install
Skip
Integrate with VS'13.
JetBrains dotTrace 6.0 EAP 8
Install
Skip
Integrate with VS'13.
Visual Studio Integration:
2010
2012
2013
2014
I have read and accept the license agreement
Options
Install

Declarative



Imperative



Declarative / Imperative

- Xml
- Structure
- What
- Pure
- DSL
- Good

- C
- Sequence
- How
- Stateful
- Framework
- Bad

Declarative / Imperative

- hard to predict
- hard to debug
- lack expressiveness

- hard to read
- error prone
- verbose

intuitive, design principles, observability, etc.

Xaml MsBuild kProj

```
<ResourceDictionary xmlns="h
    xmlns:x="http://schemas.
    xmlns:pPs="clr-namespace
  <DataTemplate DataType="{x
    <Grid HorizontalAlignmen
      <Grid.ColumnDefinition
        <ColumnDefinition Wi
        <ColumnDefinition Wi
      </Grid.ColumnDefinitio
      <Grid.RowDefinitions>
        <RowDefinition Heigh
        <RowDefinition Heigh
      </Grid.RowDefinitions>
      <Button BorderThicknes
      <Button BorderThicknes
      <TextBlock HorizontalA
    </Grid>
  </DataTemplate>
</ResourceDictionary>
```

```
<?xml version="1.0" encoding="u
<Project ToolsVersion="12.0" De
  <Import Project="$(MSBuildExt
  <PropertyGroup>
    <Configuration Condition="
    <Platform Condition=" '$(Pl
    <ProjectGuid>{C328F091-85AC
    <OutputType>Library</Output
    <RootNamespace>MSBuildProfi
    <AssemblyName>MSBuildProfil
  </PropertyGroup>
  <PropertyGroup Condition=" '$
  <ItemGroup>
    <Reference Include="Microso
    <Reference Include="System"
    <Reference Include="System.
    <Reference Include="System.
    <Reference Include="System.
    <Reference Include="Microso
    <Reference Include="System.
    <Reference Include="System.
  </ItemGroup>
```

```
"webroot" : "wwwroot",
"exclude": "wwwroot/**/*.*",
"dependencies": {
    "EntityFramework.SqlServer": "7.0.0
    "Microsoft.AspNet.Mvc": "6.0.0-alph
    "Microsoft.AspNet.Identity.SqlServe
    "Microsoft.AspNet.Identity.Authenti
    "Microsoft.AspNet.Security.Cookies"
    "Microsoft.AspNet.Server.IIS": "1.0
    "Microsoft.AspNet.Server.WebListene
    "Microsoft.AspNet.StaticFiles": "1.
    "Microsoft.Framework.ConfigurationM
    "Microsoft.VisualStudio.Web.Browser
},
"commands": {
    /* Change the port number when you
    "web": "Microsoft.AspNet.Hosting --
},
"frameworks": {
    "aspnet50" : { },
    "aspnetcore50" : { }
}
```

Xaml

- .xaml → .g.cs + baml
- xaml использует C# типы
- C# использует Xaml типы
- Результат - 2 прохода компилятора
- Общее пространство символов

А могло быть по-другому!

- C# + точки расширения
- Типы и методы
- Xaml, C# и VB - части одного компилятора
- Фазы компиляции
- Декларативность

MsBuild

- Properties
- Items
- Tasks
- Imports
- Targets

MsBuild

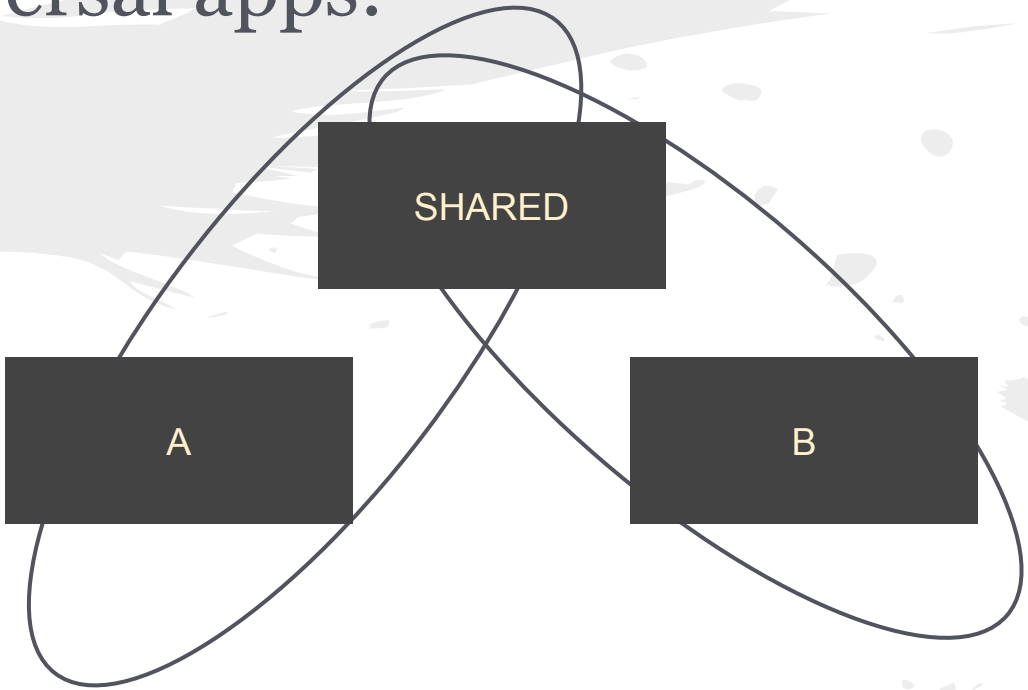
- Modules
- Cobol and procedures
- Shared state
- Build procedure for tasks
- Packages
- Complexity

CsHtml

```
<script>
  var t = $(@ToCss(“.myClass {color:red}”));
</script>
```

Code Sharing.

- Universal apps:



Issues

- Common symbol space
- Tooling
- Imperative in xml
- Complexity
- Observability
- Debug
- Extensibility

R# Experience

```
<action Text=”Foo”
Handler=”Bar”>

[Handler(“Bar”)]
class Foo: IAction {}
```

```
[Action(...)]
class Foo :
IAction,
IInsertAfter<X,Y>
{
}
```

R# Experience

Xml Configuration:

```
<Assembly Name=”X”
Configurations =
“VSSince10”/>
```

Zones configuration:

```
[ZoneMarker]
class Zone :
IRequire<ISinceVs10>
{
}
```

R# Experience

```
static class Build {
    [BuildStep]
    DotPeekSingleExe[] PackDotPeek(Package[] packages) {
      ....
    }
}
```

R# Experience

```
[SettingsKey]
class IntelliSenseSettings {
    [SettingsEntry(“Code completion enabled”)]
    bool CodeCompletionEnabled;
}
```

R# Experience

- Metadata model
  - Metadata layer
  - No assembly load
  - Advances features in IOC container
  - Build time
  - Runtime type bindings on demand

Code is data

- Refactoring
- Build transformation
- Roslyn
- Reflection
- PostSharp

meta C#

- String literals
  - No semantic checks
  - No support in tooling
  - No interop with type system
  - Expressive!

meta C#

- Attributes
  - Primitives, constants, arrays, typeof
  - Metadata access
  - Runtime access
  -

meta C#

- Interfaces
  - Generics
  - Inheritance
- Classes
  - + Constructors

meta C#

- Hierarchy and IOC

```
[ActionGroup("ReSharper.Edit", ActionGroupIns
public class EditMenu : IAction
{
  public EditMenu(
    EditIntellisenseGroup intellisenseGroup,
    Separator sep,
    GenerateGroup generateGroup,
    EditTemplatesGroup templatesGroup,
    Separator sep1,
    EditTextualGroup textualGroup,
    RearrangeCodeGroup rearrangeCode,
    WebToolsMenuGroup webTools,
    Separator sep2,
    EditOthersGroup othersGroup
   )
  {
  }
}
```

```
[Action("CleanupCode", "Cleanup Code…", VsShortcuts = new[] {"Cor
public class CodeCleanupAction : CodeCleanupActionBase,
  IInsertFirst<ToolsMenu>,
  IInsertAfter<EditorContextMenuOthersActionGroup, InspectMenu>,
  IInsertLast<IntoSolutionItemGroup_Modify>
{
  protected override CodeCleanupProfile GetProfile(CodeCleanupFil
  {
    return SelectProfileDialog(collector, false);
  }
}
```

meta C#

- Fields

```
[SettingsKey(typeof(CodeInspectionSettings), "Highlighting settings")]
public class HighlightingSettings
{
  [SettingsEntry(AnalysisScope.VISIBLE_FILES, "Analysis mode")]
  public AnalysisScope AnalysisEnabled;

  [SettingsEntry(false, "Identifier highlighting enabled")]
  public bool IdentifierHighlightingEnabled;

  [SettingsEntry(true, "Color usage highlighting enabled")]
  public bool ColorUsageHighlightingEnabled;

  [SettingsEntry(true, "Show analysis options in action list")]
  public bool ShowAnalysisOptionsInActionList;

  [SettingsEntry(true, "Show import popup")]
  public bool ShowImportPopup;
```

meta C#

- Fields
  - = Type + Name
  - C# 6 - nameof(expr)
  - Expression trees

```
protected override SettingsScalarEntry GetSettingsEntryInternal(ISettingsStore store)
{
  return store.Schema.GetScalarEntry((CSharpFilterStateSettingsKey key) => key.PrivateSymbols);
}
```

Build

- Access metadata at build time
- Build for build steps
- Consume packages
- Build packages
- Transfer build state
- Use build artifacts at runtime
- Composability

R# experience

- Build bootstrap
  - Compile build sources
  - Per-package build scirpt
  - Limited set of APIes
  - Prepare solution for editing
  - Pass build settings (configuration)
  - Substitute binary dependencies

R# experience

- Compile is simple
  - < 10 custom targets
  - 1 custom import in project files
  - MsBuild

R# experience

- Build graph

```
[BuildStep]
public static ArtifactB[] TransformToB(ArtifactA[] inputA)
{
  return inputA.Select(x => new ArtifactB(x)).ToArray();
}

[BuildStep]
public static ArtifactA IHaveA()
{
  return new ArtifactA();
}
```

CI configuration

- TeamCity
  - 28 configurations
  - 50 build agents
  - 20-30 branches a day
  - Builds exchange serializable artifacts
  - Build requests artifact type.

Takeaway

- Declarative is not xml
- Keep extensions imperative
- Apply Solid to declarative
- Make declarative simple
- Never introduce shared state
-

Thanks.