



T4

John M. Miller  
Perpetual Data Systems

[jmiller@pdata.com](mailto:jmiller@pdata.com)

---

# Agenda

- What T4 is not
- What T4 is
- What T4 needs
- What are the T4 primitives
- What are the T4 resources and references

# What T4 is not

- Another movie starring the Govinator
- Story-Based Concept Album
- A Nazi genocide program
- A British TV Channel
- An abbreviation for the hormone Thyroxine

# What is T4

- **T**ext **T**emplating **T**ransformation **T**oolkit
  - <http://t4toolbox.codeplex.com/>
- Primarily used for code generation
  - Can generate any type of text file

# What else do you need?

- Visual Studio 2005, 2008 or 2010
- T4 Editor optional

# works

The Template is used to generate code which, when compiled and run, generates the output

Domain Specific Language  
ASP.NET like syntax

```
<#@ template language="C#" #>  
Hello  
<# Write("World!"); #>
```

Generation  
Transforms the template into .NET code

```
public class GeneratedTextTransformation:  
    Microsoft.VisualStudio.TextTemplating.TextTransformation  
{  
    public override string TransformText()  
    {  
        this.Write("Hello\n");  
        Write("World");  
        return this.TransformText();  
    }  
}
```

Compilation turns the template into a .NET Assembly

Running the compiled template generates the output file

```
Hello  
World!
```

Output can be any type of text file

# DSL Primitives

- Text Blocks
- Code Blocks
- Processing Directives

# Text Blocks

- Non-programmatic text in a text template.
  - You can write a text block directly to the output file of the text transformation.
  - You can add it by simply typing the text in the text template.
- You can add a text block to the generated transformation class as a Write statement.

# Demo

- Text Blocks
  - Generating a HelloWorld application

# <#@ template #> directive

- Specifies the characteristics of the generated transformation class
  - Code language
  - Class inheritance
  - Culture
  - Debug

# Demo

- Template Directive
  - Debug = true
- SET %TEMP%
- Generated code

# Statement Blocks

- The general syntax is:  
    <# StatementCode #>
- Statement blocks control the flow of processing
  - Enable conditional output
  - Iterate over data
- Code statements are delineated using opening (<#) and closing (#>) text template tags.

# Demo

- Statement Blocks
  - Generate time of generation

# Expression Blocks

- The general syntax is:  
`<#= ExpressionCode #>`
- Add strings to the generated text output
  - Added to the generated transformation class with calls to `ToString()` attached
- Typically, expression blocks are embedded inside text blocks.
  - calculated values from external data
- In text templates, expression blocks are delineated by using opening (`<#=>`) and closing (`#>`) text template tags.

# Demo

- Expression Blocks
  - Generate code to compare run time with the generation time
  - Explore when generation occurs

# Class Feature Blocks

- The general syntax is:  
    <#+ FeatureCode #>
- Add helper functions.
  - Avoid repeating common code.

# Demo

- Class Feature Blocks
  - Refactor comment generation as a method call

# <#@ import #> directive

- Allows references to types in a text template without providing a fully qualified name
  - Think using namespaces

# Demo

- Import Directive
  - Import System.Diagnostics
  - Add breakpoint
  - Step through template

# <#@ output #> directive

- Specifies characteristics of the generated text output
  - File name extension

# Demo

- Output Directive
  - t4.cs

# <#@ include #> directive

- Processes text from the specified file as if that text were included verbatim in the including template

# Demo

- Include Directive
  - Separate out the date time comparison as a separate template

## <#@ assembly #> directive

- Identifies an assembly to be referenced so that you can use types within that assembly from code in the text template
- Using the assembly directive is equivalent to using the Add Reference feature in Visual Studio.

# Demo

- Assembly Directives
  - Create Application Text assembly

# Why should I care?

- Code generation is often better than data driving apps at runtime
  - Generated Code is more straight forward than abstract runtime code
  - Performance is often better
  - Complexity in the template not the app
- Best when you can generate multiple artifacts from a single data source

# Resources

- Editors

- Visual T4

- <http://www.visualt4.com/>

- Tangible Modeling Tools w/T4 Editor

- <http://t4-editor.tangible-engineering.com>

- Template Repositories

- <http://www.codeplex.com/site/search?projectSearchText=t4&sortBy=Relevance&tagName=%2cT4%2c&licenses=|&refinedSearch=true>

# References

- **Generating Artifacts By Using Text Templates**  
<http://msdn.microsoft.com/en-us/library/bb126445.aspx>
- **Visual Studio's T4 Code Generation**  
<http://visualstudiomagazine.com/articles/2009/05/01/visual-studios-t4-code-generation.aspx>
- **T4 Code Generation - Best Kept Visual Studio Secret**  
<http://www.hanselman.com/blog/T4TextTemplateTransformationToolkitCodeGenerationBestKeptVisualStudioSecret.aspx>
- **T4: Text Template Transformation Toolkit**  
<http://www.olegrych.com/2007/12/text-template-transformation-toolkit/>
- **T4 Templates: A Quick-Start Guide for ASP.NET MVC Developers**  
<http://blogs.msdn.com/webdevtools/archive/2009/01/29/t4-templates-a-quick-start-guide-for-asp-net-mvc-developers.aspx>



T4

John M. Miller  
Perpetual Data Systems

[jmiller@pdata.com](mailto:jmiller@pdata.com)

---