

DevJET Software

# Delphi Documentation Guidelines

Baoquan Zuo  
10/12/2010

## Table of Contents

1.	Copyright.....	3
2.	Acknowledgements.....	3
3.	Revision History.....	3
4.	Overview .....	4
5.	XML Documentation .....	5
5.1	Introduction .....	5
5.2	Section Tags.....	6
5.3	Block Tags.....	6
5.4	Inline Tags.....	7
5.5	Tag Usage .....	7
6.	XML Tag Reference.....	7
6.1	<c>.....	7
6.2	<code>.....	8
6.3	<example>.....	8
6.4	<exception>.....	9
6.5	<include>.....	10
6.6	<list>.....	10
6.7	<note> .....	12
6.8	<para> .....	13
6.9	<param> .....	14
6.10	<paramref>.....	14
6.11	<permission>.....	15
6.12	<preliminary> .....	15
6.13	<remarks> .....	15
6.14	<returns>.....	16
6.15	<see> .....	16
6.16	<seealso>.....	16
6.17	<summary> .....	17
6.18	<threadsafety> .....	18
6.19	<typeparam>.....	18
6.20	<typeparamref> .....	19
6.21	<value>.....	19
7.	Write Documentation .....	21
8.	Generate Documentation .....	23
9.	References.....	23

# 1. Copyright

Copyright © 2010, [DevJET Software](#). All rights reserved.

This document may not be included in the distribution of any software application or any other document collection without the permission of the author.

The author has made every effort in the preparation of this content to ensure the accuracy of the information. However, the information is provided without warranty, either express or implied. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by this content.

**Delphi** and **RAD Studio** are registered trademarks of [Embarcadero Technologies, Inc.](#)

[NDoc](#) is a well-known open-source project which provides a documentation generator for .Net.

## 2. Acknowledgements

(Placeholder)

## 3. Revision History

Version	Date	Author	Remarks
V0.9.2	Oct. 12 <sup>th</sup> , 2010	Baoquan Zuo	Initial release

To get the latest version of this document, please visit the website:

**DevJET Software**

<http://www.devjet.net/>

## 4. Overview

When we want to learn a code library, we must read its documentation at first. A professional documentation will help us easily get the key points and understand the design intention of the author. It is no doubtful that one of the most impressive examples is MSDN.

On the other hand, when we play the role of an author, we also need to write documentation to explain our code and design remarks, especially when working together with other developers.

There are two major candidates in software documentation:

- **XML Documentation**
- **Javadoc**

XML documentation was introduced by Microsoft to document .Net code while the format of Javadoc is created by Sun Microsystems to document java source files.

Since the Delphi IDE has officially supported the XML documentation standard since Delphi 2006, so this document is based on XML Documentation.

In the following sections, you will know more about the xml documentation style and the detailed syntax and remarks.

The last sections will tell you how to easily and quickly write xml documentation in the IDE and generate documentation from the source code.

# 5. XML Documentation

## 5.1 Introduction

XML documentation was introduced by Microsoft to document C#/VB.net code. Delphi followed this standard later and the IDE and compiler has supported the style since Delphi 2006.

When the build-in **Help Insight** feature is enabled, a popup window will be shown to display a simple documentation of the current code element under the mouse hover. In addition, the documentation could be surrounded in a region so that you can fold it when needed.

```
{ $REGION 'Documentation' }
/// <summary>Represents a series of bytes in memory.</summary>
/// <remarks>The <c>TBuffer</c> structure is actually a wrapper of a value of
/// <see cref="SysUtils|TBytes" /> while providing some convenient methods and
/// properties.</remarks>
/// <threadsafety static="true" instance="false" />
{ $ENDREGION }
TBuffer = record
  TBuffer Type - Spring.pas (114,3)
  Represents a series of bytes in memory.
  Declared in Spring
  Remarks
  The TBuffer structure is actually a wrapper of a
  value of TBytes while providing some convenient
  methods and properties.
  ;
  t value: Byte);
class var Empty: TBuffer;
```

As you can see, xml documentation is actually a well-formed XML document and starts with `///` as prefix. You can use some meaningful tags and write descriptions within tags. Some tags may contain attributes.

You can customize your own tags but there have been so many common tags recommended by Microsoft. Most documentation generator support the recommended usage.

All xml tags could be divided into the following categories:

- **Section Tags**

Defines a top-level section in a topic. e.g. Summary, Remarks, Examples section.

- **Block Tags**

Adds a block to a section. e.g. a paragraph, an image or a code snippet.

- **Inline Tags**

Formats the effect of a text. e.g. bold, italic, underline, hyperlink.

- **Usage Notes:**

- If you want angle brackets to appear in the text of a documentation comment, use **&lt;** and **&gt;**. For example, the string "&lt;text in angle brackets&gt;" will appear as <text in angle brackets>.
- The **cref** attribute in a tag means a reference to a member. You should use this form in Delphi: "**Namespace|Element Path**". e.g. <see cref="Spring.Utils|TDriveInfo.GetDrives" />
- To reference a generic type or member as a target in the cref attribute, you may use curly braces around the "T". e.g. <see cref="Spring.Collections|IList {T}"/>.
- All common HTML tags such as <p>, <b>, <i>, <u>, <a>, <img>, etc. should be also supported.

## 5.2 Section Tags

Section tags are used to define the content of the different sections of the documentation of a type or member. These tags are used as top-level tags.

The following tags are section tags:

Tag	Description
<a href="#">&lt;summary&gt;</a>	A short description of the code element.
<a href="#">&lt;remarks&gt;</a>	Additional information about the code element, supplementing the description in the <summary> tag.
<a href="#">&lt;param&gt;</a>	A parameter of the code element which may be a method, indexed property or a routine.
<a href="#">&lt;typeparam&gt;</a>	A type parameter of a generic type or a generic method
<a href="#">&lt;returns&gt;</a>	A description of the return value of a member.
<a href="#">&lt;value&gt;</a>	A description of the value of a property, field, constant or a variable.
<a href="#">&lt;example&gt;</a>	An example of demonstrating how to use the code.
<a href="#">&lt;exception&gt;</a>	An exception that may be raised the code element.
<a href="#">&lt;permission&gt;</a>	Security permissions required to access a member.
<a href="#">&lt;seealso&gt;</a>	Adds an entry, which may be a link to a member or a URI address, to the <b>See Also</b> section.
<a href="#">&lt;include&gt;</a>	References an xml node in an include file that contains one or more documentation section tags.
<a href="#">&lt;preliminary&gt;</a> <b>[NDoc Compatible]</b>	Represents that the API is preliminary and will be changed in future.
<a href="#">&lt;threadsafety&gt;</a> <b>[NDoc Compatible]</b>	Describes how the code behaviors in multi-threaded context.

## 5.3 Block Tags

Block tags format text within the top-level tags. They are typically used to add structure to the text inside **Remarks** and **Examples** sections.

Tag	Description
<a href="#">&lt;para&gt;</a>	A paragraph.
<a href="#">&lt;code&gt;</a>	A block of code.
<a href="#">&lt;list&gt;</a>	A numbered list, bulleted list or a definition list.
<a href="#">&lt;note&gt;</a> <b>[NDoc Compatible]</b>	A formatted note block.

## 5.4 Inline Tags

Inline tags are typically used inside block tags.

Tag	Description
<a href="#">&lt;c&gt;</a>	A text as an inline code.
<a href="#">&lt;paramref&gt;</a>	A reference to a parameter of a method, a routine or an indexed property.
<a href="#">&lt;typeparamref&gt;</a>	A reference to a type parameter of a generic type or a generic method.
<a href="#">&lt;see&gt;</a>	A link to a member or a hyper link.

## 5.5 Tag Usage

Element	Sections
<b>Classes/Records</b>	Summary, Remarks , Examples and Thread Safety
<b>Interfaces</b>	Summary, Remarks, Examples
<b>Methods</b>	Summary, Remarks, Parameters, Returns and Exceptions
<b>Properties</b>	Summary, Value, Remarks and Exceptions
<b>Fields</b>	Summary
<b>Constants/Variable</b>	Summary and Value
<b>Delegates</b>	Summary, Remarks, Parameters and Returns

# 6. XML Tag Reference

## 6.1 <c>

This tag is used to mark the text as inline code.

**Syntax:**

```
<c>text</c>
```

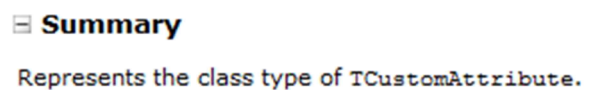
**Remarks:**

The `<c>` tag gives you a way to indicate that text within a description should be marked as code. Use [<code>](#) to indicate multiple lines as code.

**Examples:**

```
/// <summary>Represents the class type of <c>TCustomAttribute</c>.</summary>  
TAttributeClass = class of TCustomAttribute;
```

**Examples Out:**

A screenshot of a summary box. It has a title bar with a minus sign icon and the word "Summary". Below the title bar, the text "Represents the class type of TCustomAttribute." is displayed in a monospaced font.

**Summary**  
Represents the class type of TCustomAttribute.

## 6.2 <code>

This tag is used to add a code snippet.

**Syntax:**

```
<code [lang="language"] >content</code>
```

**Parameters:**

lang:  
The language of the code.

**Remarks:**

**Examples:**

See [examples](#) in the [<example >](#) tag.

## 6.3 <example>

Use this tag to insert an example to demonstrate how to use the code element.

**Syntax:**

```
<example>description</example>
```

**Remarks:**

**Examples:**

```
/// <example>  
/// The following code demonstrates how to create a new guid and use it in  
/// OO-style:  
/// <code lang="Delphi">
```



```

/// procedure TestGuidHelper;
/// var
///   guid: TGuid;
/// begin
///   // generates a new guid.
///   guid := TGuid.NewGuid;
///   // this guid must not be empty.
///   Assert(not guid.IsEmpty);
///   // print the string representation
///   Writeln(guid.ToString);
///   // print the quoted string
///   Writeln(guid.ToQuotedString);
///   // This guid must equal to itself.
///   Assert(guid.Equals(guid));
/// end;
/// </code>
/// </example>

```

### Examples Out:

**Examples**

The following code demonstrates how to create a new guid and use it in OO-style:

**Delphi** [Copy Code](#)

```

procedure TestGuidHelper;
var
  guid: TGuid;
begin
  // generates a new guid.
  guid := TGuid.NewGuid;
  // this guid must not be empty.
  Assert(not guid.IsEmpty);
  // print the string representation
  Writeln(guid.ToString);
  // print the quoted string
  Writeln(guid.ToQuotedString);
  // This guid must equal to itself.
  Assert(guid.Equals(guid));
end;

```

## 6.4 <exception>

This tag is used to describe which exceptions will be raised by the method or routine.

### Syntax:

```
<exception cref="member">description</exception>
```

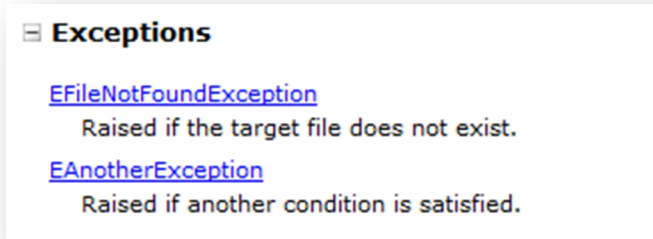
### Remarks:

The cref attribute references to an exception class.

### Examples:

```
/// <exception cref="EFileNotFoundException">Raised if the target file does
/// not exist.</exception>
/// <exception cref="EAnotherException">Raised if another condition is
/// satisfied.</exception>
procedure CheckFileExists(const fileName: string);
```

### Examples Out:



## 6.5 <include>

Allow you to refer to the comments in the specified file.

### Syntax:

```
<include file='filename' path='tagpath[@name="id"]' />
```

### Remarks:

**WARNING:** Not supported by most tools including Documentation Insight.

## 6.6 <list>

Use <list> tag to insert a numbered list, bulleted list or a definition table.

### Syntax:

```
<list type="bullet" | "number" | "table">
  <listheader>
    <term>term</term>
    <description>description</description>
  </listheader>
  <item>
    <term>term</term>
    <description>description</description>
  </item>
</list>
```

### Parameters:

*term*

A term to define, which will be defined in **description**.

*description*

Either an item in a bullet or numbered list or the definition of a **term**.

### Remarks:

The <listheader> block is used to define the heading row of either a table or definition list. When defining a table, you only need to supply an entry for term in the heading.

Each item in the list is specified with an <item> block. When creating a definition list, you will need to specify both term and description. However, for a table, bulleted list, or numbered list, you only need to supply an entry for description.

A list or table can have as many <item> blocks as needed.

### Examples:

```
/// <remarks>
/// <para>This is a <b>bulleted list</b>:</para>
/// <list type="bullet">
/// <item>Item 1</item>
/// <item>Item 2</item>
/// <item>Item 3</item>
/// </list>
/// <para>This is a <b>numbered list</b>:</para>
/// <list type="number">
/// <item>Item 1</item>
/// <item>Item 2</item>
/// <item>Item 3</item>
/// </list>
/// <para>This is a <b>definition table</b>:</para>
/// <list type="table">
/// <listheader>
/// <term>Fruit</term>
/// <description>Description</description>
/// </listheader>
/// <item>
/// <term>Apple</term>
/// <description>A</description>
/// </item>
/// <item>
/// <term>Banana</term>
/// <description>B</description>
/// </item>
/// <item>
/// <term>Pear</term>
/// <description>C</description>
/// </item>
/// </list>
/// </remarks>
```

### Examples Out:

**Remarks**

This is a **bulleted list**:

- Item 1
- Item 2
- Item 3

This is a **numbered list**:

1. Item 1
2. Item 2
3. Item 3

This is a **definition table**:

Fruit	Description
Apple	A
Banana	B
Pear	C

## 6.7 <note>

Produce a more visible note block within the topic.

### Syntax:

```
<note type="note" | "tip" | "warning" | "caution" | "security" | "security note" | "important"
| "implement" | "caller" | "inherit" | "delphi" | "bcb">content</note>
```

### Remarks:

**Note:** This tag is an extension and compatible with **NDoc**.

### Examples:

```
///  
/// <remarks>  
/// <note type="note">This is a note sample.</note>  
/// <note type="warning">This is a warning sample.</note>  
/// <note type="security">This is a security note.</note>  
/// <note type="delphi">This is a Delphi note.</note>  
/// <note type="bcb">This is a C++ Builder note.</note>  
/// <note type="caller">Notes to callers.</note>  
///  
/// </remarks>
```

### Examples Out:



## 6.8 <para>

This tag is used to add a paragraph.

### Syntax:

```
<para>content</para>
```

### Remarks:

**Tip:** You can also use the <p> tag in HTML.

### Examples:

```
///  
/// <remarks>  
/// <para>This is the first paragraph.</para>  
/// <para>This is the second paragraph.</para>  
/// <para>This is the third paragraph.</para>  
/// </remarks>
```

### Examples Out:

### ☐ **Remarks**

This is the first paragraph.  
This is the second paragraph.  
This is the third paragraph.

## 6.9 <param>

This tag is used to describe a parameter of a method, indexed property or a routine.

### Syntax:

```
<param name="name">description</param>
```

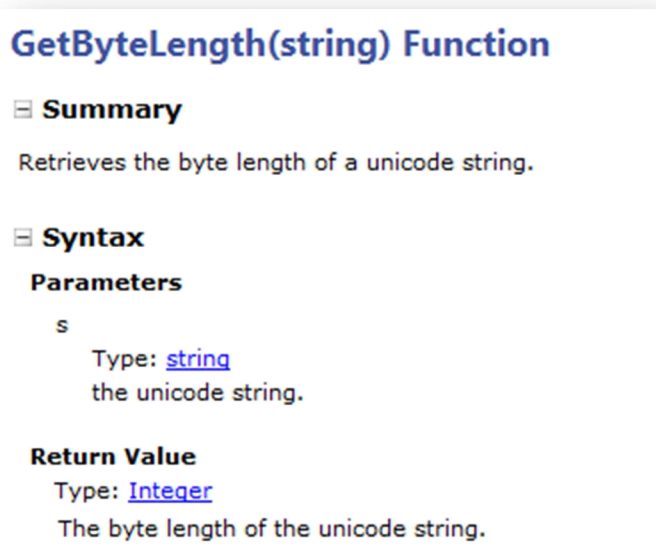
### Remarks:

The count of the <param> tags should be the same as declared.

### Examples:

```
/// <summary>Retrieves the byte length of a unicode string.</summary>  
/// <param name="s">the unicode string.</param>  
/// <returns>The byte length of the unicode string.</returns>  
function GetByteLength(const s: string): Integer;
```

### Examples Out:



**GetByteLength(string) Function**

☐ **Summary**

Retrieves the byte length of a unicode string.

☐ **Syntax**

**Parameters**

s  
Type: [string](#)  
the unicode string.

**Return Value**

Type: [Integer](#)  
The byte length of the unicode string.

## 6.10 <paramref>

This tag is used to mark the text as a reference to a parameter.

**Syntax:**

```
<paramref name="name" />
```

**Remarks:****Examples:**

See examples in [<param>](#).

## 6.11 <permission>

This tag is used to describe the security permission that is required to access the member.

**Syntax:**

```
<permission cref="member">description</permission>
```

**Remarks:**

## 6.12 <preliminary>

This tag is used to mark the documentation of the current code element as preliminary.

**Syntax:**

```
<preliminary>[description]</preliminary>
```

**Parameters:**

*description*

An optional textual message or warning that replaces the default preliminary warning.

**Remarks:**

This tag is an extension tag which is compatible with **NDoc**. You can also use the empty form (`<preliminary />`) and then the default warning message will be shown.

**Examples:**

```
/// <preliminary />
```

**Examples Out:**

**This API is preliminary and subject to change.**

## 6.13 <remarks>

This tag is used to provide additional or detailed information about the code element.

**Syntax:**

```
<remarks>description</remarks>
```

### Examples:

See examples in [<summary>](#).

## 6.14 <returns>

This tag is used to describe the return value of a method or a function.

### Syntax:

```
<returns>description</returns>
```

### Remarks:

### Examples:

See examples in [<param>](#).

## 6.15 <see>

This tag is used to make the text as a link refers to a code element or a URI address.

### Syntax:

```
<see cref="member">[displayName]</see>  
OR  
<see href="URI">[displayName]</see>  
OR  
<see langword="null | sealed | static | abstract | virtual | true | false" />
```

### Remarks:

This tag is an inline link refers to a member, a URI or a language word. Use [<seealso>](#) tag to add a link in the See Also section.

### Examples:

See [examples](#) in [<summary>](#).

## 6.16 <seealso>

This tag is used to add a link to the See Also section.

### Syntax:

```
<seealso cref="member">[displayName]</seealso>  
OR  
<seealso href="URI">[displayName]</seealso>
```

### Parameters:

*member*

The reference to a code element.

*URI*



The reference to a URI address.  
displayName  
The text to display.

**Remarks:**

**Note:** It is not necessary to add entries reference to the parent or other overloads of the current element. Use [<see>](#) to specify a link from within text.

**Examples:**

```
/// <seealso cref="Spring.Collections|IList{T}" />  
/// <seealso href="http://www.devjet.net">DevJET</seealso>
```

**Examples Out:**



## 6.17 <summary>

This tag is used to provide a short description of the code element.

**Syntax:**

```
<summary>description</summary>
```

**Remarks:**

You can apply this tag to all code elements. Use [<remarks>](#) tag to add additional or detailed description.

**Note:** When applying this tag to a **property**, you should indicate whether the property is readable and writable. (e.g. "Gets or sets the full name of the customer.")

**Examples:**

```
/// <summary>Represents a series of bytes in memory.</summary>  
/// <remarks>The <c>TBuffer</c> structure is actually a wrapper of a value of  
/// <see cref="SysUtils|TBytes" /> while providing some convenient methods and  
/// properties.</remarks>  
/// <threadsafety static="true" instance="false" />  
TBuffer = record  
end;
```

**Examples Out:**

## TBuffer Structure

### Summary

Represents a series of bytes in memory.

### Remarks

The `TBuffer` structure is actually a wrapper of a value of `TBytes` while providing some convenient methods and properties.

### Thread Safety

Public static members of this type are thread safe. Any instance members are not guaranteed to be thread safe.

## 6.18 <threadsafety>

This tag is used to describe how the code element behaves in multi-threaded context.

### Syntax:

```
<threadsafety static="true|false" instance="true|false" />
```

### Parameters:

*static*

Indicates whether static members of this type are safe for multi-threaded operations.

*instance*

Indicates whether instance members of this type are safe for multi-threaded operations.

### Remarks:

Note: This tag can be only applied to classes and structures.

### Examples:

See examples in [<summary>](#).

## 6.19 <typeparam>

This tag is used to describe a type parameter of a generic type or a generic method.

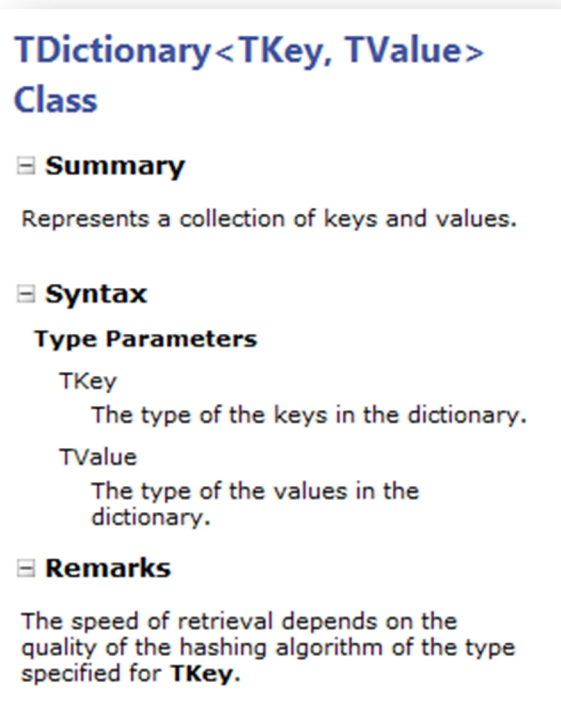
### Syntax:

```
<typeparam name="name">description</typeparam>
```

### Examples:

```
/// <summary>Represents a collection of keys and values.</summary>
/// <typeparam name="TKey">The type of the keys in the
/// dictionary.</typeparam>
/// <typeparam name="TValue">The type of the values in the
/// dictionary.</typeparam>
/// <remarks>The speed of retrieval depends on the quality of the hashing
/// algorithm of the type specified for <typeparamref name="TKey" />.</remarks>
TDictionary<TKey, TValue> = class
end;
```

### Examples Out:



**TDictionnary<TKey, TValue>**  
**Class**

▣ **Summary**

Represents a collection of keys and values.

▣ **Syntax**

**Type Parameters**

TKey  
The type of the keys in the dictionary.

TValue  
The type of the values in the dictionary.

▣ **Remarks**

The speed of retrieval depends on the quality of the hashing algorithm of the type specified for **TKey**.

## 6.20 <typeparamref>

This tag is used to indicate that the text refers to a type parameter.

### Syntax:

```
<typeparamref name="name" />
```

### Examples:

See [<typeparam>](#) tag.

## 6.21 <value>

This tag is used to describe the value of a member (property, field, constant, etc.).

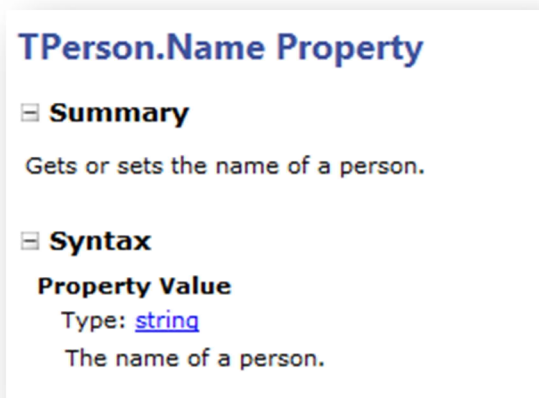
### Syntax:

`<value>description</value>`

### Examples:

```
TPerson = class
private
  fName: string;
public
  /// <summary>Gets or sets the name of a person.</summary>
  /// <value>The name of the person.</value>
  property Name: string read fName write fName;
end;
```

### Examples Out:

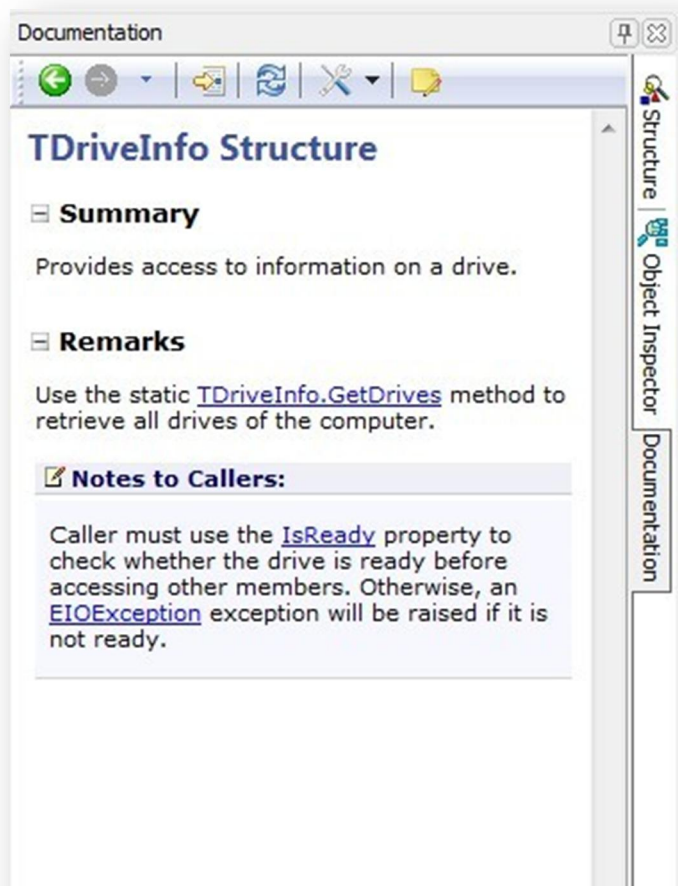


The screenshot shows a documentation window titled "TPerson.Name Property". It contains two expandable sections: "Summary" and "Syntax". The "Summary" section is expanded and shows the text "Gets or sets the name of a person." The "Syntax" section is also expanded and shows the text "Property Value" followed by "Type: [string](#)" and "The name of a person.".

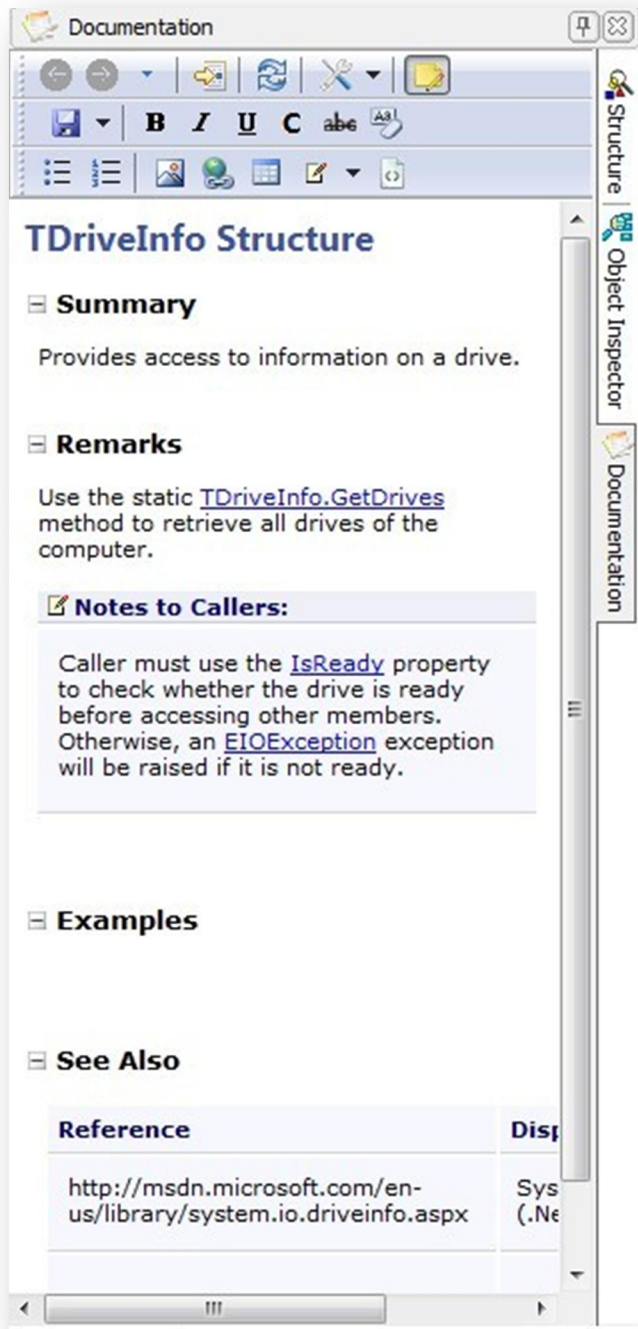
## 7. Write Documentation

It is obvious that we can write xml documentation by typing, or use some customized live templates to simplify input. But they are still not so easy to write and maintain and it will make you boring some day.

We are very happy to write documentation because we have developed a smart tool, which is named as **Documentation Insight**, while developing **Spring Framework for Delphi**. Documentation Insight is an add-in for RAD Studio IDE and it provides a **WYSIWYG** editor to help you visually view and edit xml documentation. The first beta has been released. Here are some snapshots of the product:



The above snapshot shows the reading mode of Documentation Insight.



With Documentation Insight, you can easily bold/italic/underline/strikethrough text, start a bulleted list, a numbered list or a definition table, or insert an image, a hyper link, a code snippet or a note block, etc.

It's so wonderful since you don't have to remember the complicated syntax and deal with messy xml tags.

For more information about **Documentation Insight**, please visit our homepage:

<http://www.devjet.net>

## 8. Generate Documentation

**RAD Studio** supports the documentation generation by using the built-in **Together** module. At first, you need to make sure the option “**Generate XML Documentation**” in project options is checked and **Modeling Support** for your projects is also enabled. Then switch to **Model View** tab, and choose “**Generate Documentation**” in the popup menu.

## 9. References

- Michael Taylor. [.NET Documentation Guidelines](#)
- Wm. Eric Brunsen, Dyncity, LLC. [XML Documentation Comments Guide](#)
- Mike Elliott. [C# and XML Source Code Documentation](#)
- Anson Horton. **XML Documentation in C#**
- NDoc. [Tags Supported by NDoc](#)
- MSDN. [Recommended Tags for Documentation Comments](#)