**DocTemplateBuilder - Detailed Guide**

DocTemplateBuilder is a **fluent builder class** to dynamically populate Word templates with **text fields, images, and tables**. It integrates seamlessly with the DocumentGenerator to produce Word or PDF outputs using **Aspose.Words**.

**1. Creating a Builder Instance**

To start building a template:

// Using default placeholders "<" and ">"

var builder = DocTemplateBuilder.Create();

// Using custom placeholder indicators

var builderCustom = DocTemplateBuilder.Create("<%", "%>");

**Notes:**

* Start/End indicators define how placeholders appear in your Word template.
* Example: <%UserName%> for "UserName" placeholder with <% and %> indicators.

**2. Adding Text Fields**

**2.1 Single Text Field**

builder = builder.WithText("UserName", "John Doe");

**Optional styling:**

builder = builder.WithText("UserName", "John Doe", new DocTextStyle

{

FontName = "Arial",

FontSize = 12,

Bold = true,

ColorHex = "#FF0000"

});

**2.2 Multiple Text Fields**

builder = builder.WithTextFields(new List<DocTemplateTextField>

{

new DocTemplateTextField { Name = "FirstName", Value = "John" },

new DocTemplateTextField { Name = "LastName", Value = "Doe" }

});

**2.3 Extension: Load from Model**

public class UserModel

{

[DocumentPlaceholder(Key = "FirstName")]

public string First { get; set; } = "John";

[DocumentPlaceholder(Key = "LastName")]

public string Last { get; set; } = "Doe";

}

var user = new UserModel();

builder = builder.WithTextFromModel(user);

**Notes:**

* WithTextFromModel<T> automatically scans public properties of a model decorated with [DocumentPlaceholder].

**3. Adding Images**

**3.1 Single Image from Path**

builder = builder.WithImage("ProfilePic", "C:\\Images\\JohnDoe.jpg", new ImageFieldDimension

{

Width = 100,

Height = 100

});

**3.2 Single Image from Stream**

using var fs = new FileStream("C:\\Images\\JohnDoe.jpg", FileMode.Open);

builder = builder.WithImage("ProfilePic", fs, new ImageFieldDimension

{

Width = 100,

Height = 100

});

**3.3 Multiple Images**

builder = builder.WithImages(new List<DocImageInfoRec>

{

new() { Placeholder = "Logo", Path = "C:\\Images\\Logo.png" },

new() { Placeholder = "Signature", Stream = signatureStream }

});

**Notes:**

* Use ImageFieldDimension to control width/height.
* The builder supports **both path-based and stream-based images**.

**4. Adding Tables**

**4.1 Simple Table**

var rows = new List<DocTableRowInputRec>

{

new() { Columns = new List<string> { "Name", "Age" } },

new() { Columns = new List<string> { "John", "30" } },

new() { Columns = new List<string> { "Jane", "25" } }

};

builder = builder.WithTable("UserTable", rows);

**4.2 Advanced Table with Extensions**

var users = new List<UserModel>

{

new() { First = "John", Last = "Doe" },

new() { First = "Jane", Last = "Smith" }

};

builder = builder.WithTable<UserModel>(table =>

{

table.ConfigureTable("UserTable", users);

table.TableDefinitions

.AddColumn("First Name", u => u.First)

.AddColumn("Last Name", u => u.Last);

});

**Notes:**

* WithTable<T> allows **mapping a data model** to a table.
* AddColumn defines which property/value maps to each column.
* Supports **custom cell styling** via PipeStyles.

**5. Building the Template**

After adding fields, images, and tables:

DocTemplateModel templateModel = builder.Build();

* templateModel can now be used to generate Word or PDF documents.

**6. Generating Documents**

Using DocumentGenerator:

var generator = new DocumentGenerator();

// Word

byte[] wordDoc = generator.GenerateWordDocument("Template.docx", templateModel);

// PDF

byte[] pdfDoc = generator.GeneratePdfDocument("Template.docx", templateModel);

**Async versions:**

byte[] wordDocAsync = await generator.GenerateWordDocumentAsync("Template.docx", templateModel);

byte[] pdfDocAsync = await generator.GeneratePdfDocumentAsync("Template.docx", templateModel);

**Notes:**

* Supports SaveFormat.Docx and SaveFormat.Pdf.
* Works with **existing Word templates** containing placeholders.

**7. Common Placeholder Patterns**

| **Placeholder** | **Start/End Indicator** | **Example** |
| --- | --- | --- |
| UserName | < > | <UserName> |
| Logo | <% %> | <%Logo%> |
| Table | { } | {UserTable} |

**8. Summary**

* **Text Fields:** .WithText, .WithTextFields, .WithTextFromModel<T>
* **Images:** .WithImage (path/stream), .WithImages
* **Tables:** .WithTable, .WithTable<T> (advanced mapping)
* **Builder is immutable:** Each .With... returns a new builder instance.
* **Build:** .Build() returns a DocTemplateModel
* **Document Generation:** DocumentGenerator converts template model into Word or PDF.