# Introduction

```csharp

// Introduction.cs

using System;

namespace DesignDocuments

{

public class Introduction

{

public string Title { get; } = "Introduction";

public string Content { get; } = "This is a design doc in Pega...";

public void Display()

{

Console.WriteLine($"Title: {Title}");

Console.WriteLine($"Content: {Content}");

}

}

}

```

### Usage Example:

```csharp

// Program.cs

using System;

namespace DesignDocuments

{

class Program

{

static void Main(string[] args)

{

Introduction intro = new Introduction();

intro.Display();

}

}

}

```

### Explanation:

* The Pega design section has been converted into a C# class called `Introduction`.
* The title and content have been defined as properties.
* A `Display` method is included to output the title and content, similar to how it might be presented in a Pega application.
* An example `Program` class demonstrates how to instantiate the `Introduction` class and invoke the `Display` method.

# Workflow

To convert the Pega design section titled "Workflow" into an equivalent .NET format, we will create a simple representation using C#. Below is an example of how you might structure this information in a .NET context, assuming you are creating a class to represent a Workflow.

```csharp

using System;

using System.Collections.Generic;

namespace WorkflowManagement

{

public class Workflow

{

// Properties

public string Title { get; set; }

public List<string> Steps { get; set; }

public string Description { get; set; }

// Constructor

public Workflow(string title, string description)

{

Title = title;

Description = description;

Steps = new List<string>();

}

// Method to add a step to the workflow

public void AddStep(string step)

{

Steps.Add(step);

}

// Method to display workflow details

public void DisplayWorkflow()

{

Console.WriteLine($"Title: {Title}");

Console.WriteLine($"Description: {Description}");

Console.WriteLine("Steps:");

foreach (var step in Steps)

{

Console.WriteLine($"- {step}");

}

}

}

class Program

{

static void Main(string[] args)

{

// Example of creating a workflow

Workflow myWorkflow = new Workflow("Sample Workflow", "Details about workflows in Pega...");

myWorkflow.AddStep("Step 1: Initialize");

myWorkflow.AddStep("Step 2: Process");

myWorkflow.AddStep("Step 3: Complete");

// Display the workflow

myWorkflow.DisplayWorkflow();

}

}

}

```

### Breakdown of the .NET Structure:

* Namespace: We define a namespace `WorkflowManagement` to encapsulate our classes.
* Workflow Class: This class represents the workflow and contains properties like `Title`, `Description`, and a list of `Steps`.
* Constructor: A constructor is defined to initialize the workflow with a title and description.
* Methods:
* `AddStep` allows adding steps to the workflow.
* `DisplayWorkflow` prints out the details of the workflow.
* Main Method: An example workflow is created and displayed in the `Main` method of the application.

This structure allows for easy expansion and modification of workflows in a .NET application while maintaining the essence of the original Pega design.