**How to retrieve form field bounds from rotated PDF document using C#**

The Syncfusion Essential® PDF is a feature-rich and high performance [**.NET PDF library**](https://www.syncfusion.com/document-processing/pdf-framework/net) used to create, read, and edit PDF documents programmatically without Adobe dependencies. Using this library, you can get the form field bounds on the rotated PDF document page using C#.

**Steps to get a form field bounds on a rotated PDF page programmatically:**

1. Create a new console application project.A screenshot of a computer

   AI-generated content may be incorrect.
2. Install the **[Syncfusion.Pdf.Net.Core](https://www.nuget.org/packages/Syncfusion.Pdf.Net.Core" \t "_blank)** NuGet package as a reference to your console application from [**Nuget.org**](https://www.nuget.org/).

A screenshot of a computer

AI-generated content may be incorrect.

1. Include the following namespaces in the Program.cs file.

**C#**

using **Syncfusion**.Drawing;

using **Syncfusion**.Pdf;

using **Syncfusion**.Pdf.Graphics;

using **Syncfusion**.Pdf.Parsing;

1. Use the following code sample in Program.cs to retrieve the bounds of the form fields on a rotated page.  
   **C#**

// Open the PDF file

**FileStream** inputFileStream = **new** **FileStream**(@"../../../Input.pdf", **FileMode**.Open, **FileAccess**.Read, **FileShare**.ReadWrite);

// Load the PDF document

**PdfLoadedDocument** loadedDocument = **new** **PdfLoadedDocument**(inputFileStream);

// Get the form from the loaded document

**PdfLoadedForm** loadedForm = loadedDocument.Form;

// Get the collection of form fields

**PdfLoadedFormFieldCollection** fieldCollection = loadedForm?.Fields;

// Iterate through each field in the form field collection

foreach (**var** field **in** fieldCollection)

{

// Check if the field is a text box field

**if** (field is **PdfLoadedTextBoxField** loadedField)

{

// Get the bounds of the field considering the page rotation

**var** bounds = **GetRoatedPageFieldBound**(loadedField);

// Draw a red rectangle around the text box field

loadedField.Page.Graphics.**DrawRectangle**(**PdfBrushes**.Red, bounds);

}

}

// Create a memory stream to save the modified PDF document

using **MemoryStream** stream = **new** **MemoryStream**();

// Save the modified document to the memory stream

loadedDocument.**Save**(stream);

// Close the PDF document

loadedDocument.**Close**(true);

// Reset the memory stream position to the beginning

stream.Position = 0;

// Write the modified PDF document to a new file

**File**.**WriteAllBytes**(@"Output.pdf", stream.**ToArray**());

/// <summary>

/// Get the bounds of the field considering the page rotation.

/// </summary>

/// <param name="loadedSignField">The text box field. </param>

/// <returns>The bounds of the field as a RectangleF. </returns>

**RectangleF** **GetRoatedPageFieldBound**(**PdfLoadedTextBoxField** loadedSignField)

{

// Get the page containing the field

**PdfLoadedPage** loadedPage = loadedSignField.Page **as** **PdfLoadedPage**;

float locationX = 0;

float locationY = 0;

**PointF** location = **PointF**.Empty;

**SizeF** size = **SizeF**.Empty;

// Adjust the bounds based on the page rotation

**switch** (loadedPage.Rotation)

{

**case** **PdfPageRotateAngle**.RotateAngle0:

**return** loadedSignField.Bounds;

**case** **PdfPageRotateAngle**.RotateAngle90:

locationX = loadedPage.Size.Height - (loadedSignField.Location.Y + loadedSignField.Bounds.Height);

locationY = loadedSignField.Location.X;

location = **new** **PointF**(locationX, locationY);

size = **new** **SizeF**(loadedSignField.Size.Height, loadedSignField.Size.Width);

**return** **new** **RectangleF**(location, size);

**case** **PdfPageRotateAngle**.RotateAngle180:

locationX = loadedPage.Size.Width - (loadedSignField.Location.X + loadedSignField.Size.Width);

locationY = loadedPage.Size.Height - (loadedSignField.Bounds.Y + loadedSignField.Size.Height);

location = **new** **PointF**(locationX, locationY);

size = **new** **SizeF**(loadedSignField.Size.Width, loadedSignField.Size.Height);

**return** **new** **RectangleF**(location, size);

**case** **PdfPageRotateAngle**.RotateAngle270:

locationX = loadedSignField.Location.Y;

locationY = loadedPage.Size.Width - loadedSignField.Location.X - loadedSignField.Size.Width;

location = **new** **PointF**(locationX, locationY);

size = **new** **SizeF**(loadedSignField.Size.Height, loadedSignField.Size.Width);

**return** **new** **RectangleF**(location, size);

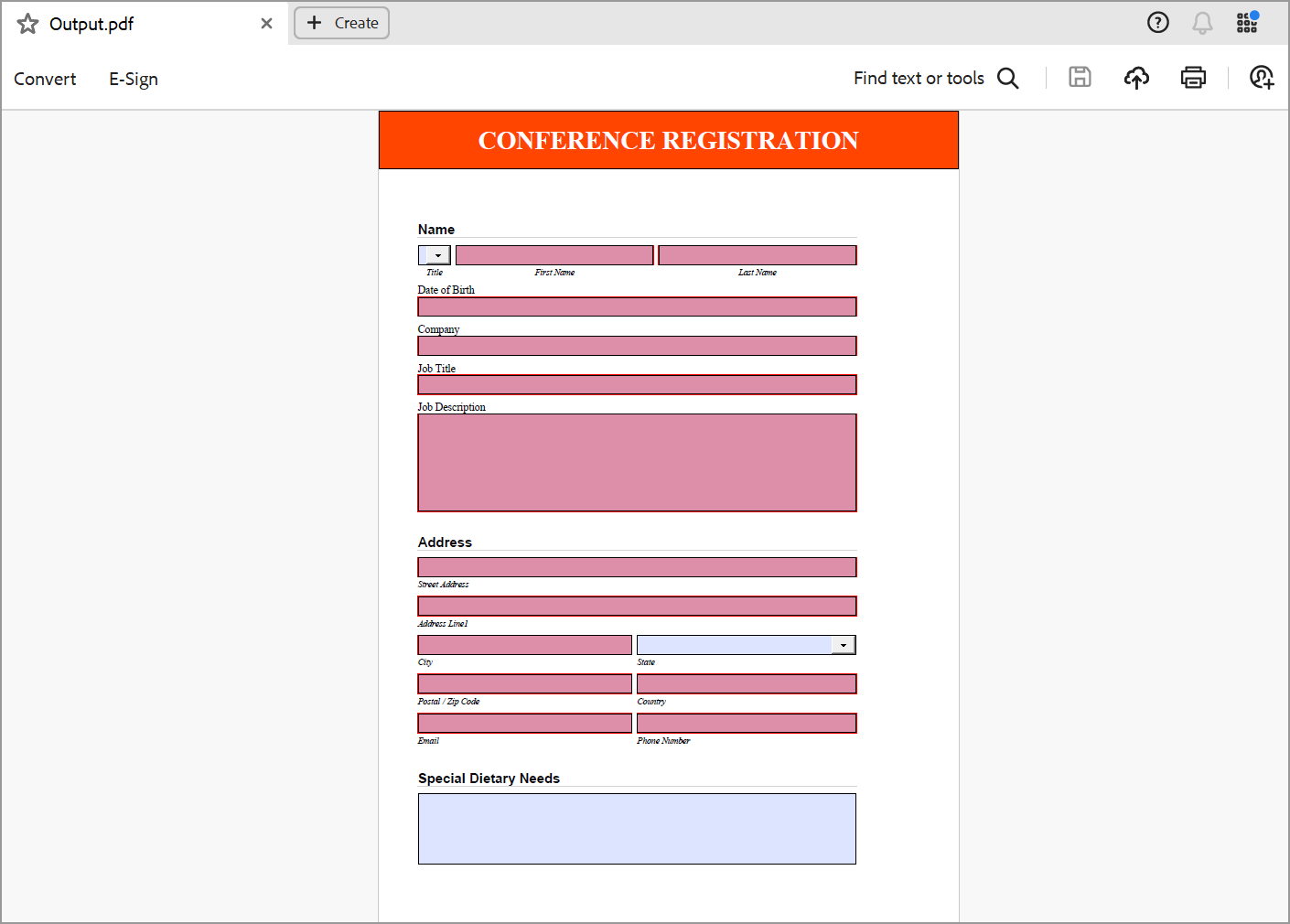
default:

**return** loadedSignField.Bounds;

}

}

A complete working sample can be downloaded from [**Interactive\_Elements\_on\_Rotated\_Page.zip**](https://www.syncfusion.com/downloads/support/directtrac/general/ze/Interactive_Elements_Rotated_Page-1455609935.zip)

By executing the program, you will get the PDF document as follows.

Take a moment to peruse the documentation for [**working with pages**](https://help.syncfusion.com/document-processing/pdf/pdf-library/net/working-with-pages), where you will find other options like inserting, removing, and rearranging pages in PDF document, adding margin, and importing pages from the existing PDF document.

Refer [**here**](https://www.syncfusion.com/document-processing/pdf-framework/net)to explore the rich set of Syncfusion Essential® PDF features.