Always Working For You: ComponentOne Preview for WinForms

Article ID: 1954
Applies To: Studio Enterprise
Author: Steve Basl
Published On: 1/14/2009

Overview

Deep in the ComponentOne Laboratories, our developers have been envisioning new and improved ideas to enhance our current controls. This is no different with ComponentOne Preview for WinForms. A major enhancement has been made to Preview for WinForms: forms can now be exported to PDF Acroforms. In previous versions of Preview for WinForms, RenderInput controls were exported to PDF simply as images, whereas now they are exported to PDF as form fields. With you – the developer – in mind, ComponentOne has included four supported controls: **Text**, **RadioButton**, **ComboBox**, and **CheckBox**.

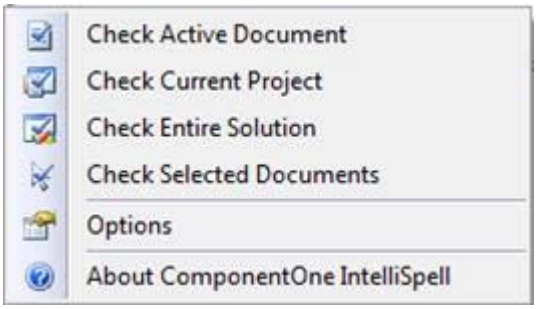
To demonstrate this feature, let's create a sample constructing an employee information form. These forms are extremely useful as they are applicable in every department. Creating the forms with Preview for WinForms is a breeze – just follow the steps below and see how it's done.

Setup the Application

Create a Windows Forms Application

ComponentOne Preview (**C1Preview**) is a Windows Forms based control. In order to use the **C1Preview** control, we must first create a Windows Forms Application.

Open Microsoft Visual Studio and select "File| New Project" from the menu. For this tutorial, we will create a C# Windows Forms Application. Then, choose a unique name for the project. For this sample, name the project "EmployeeForm".



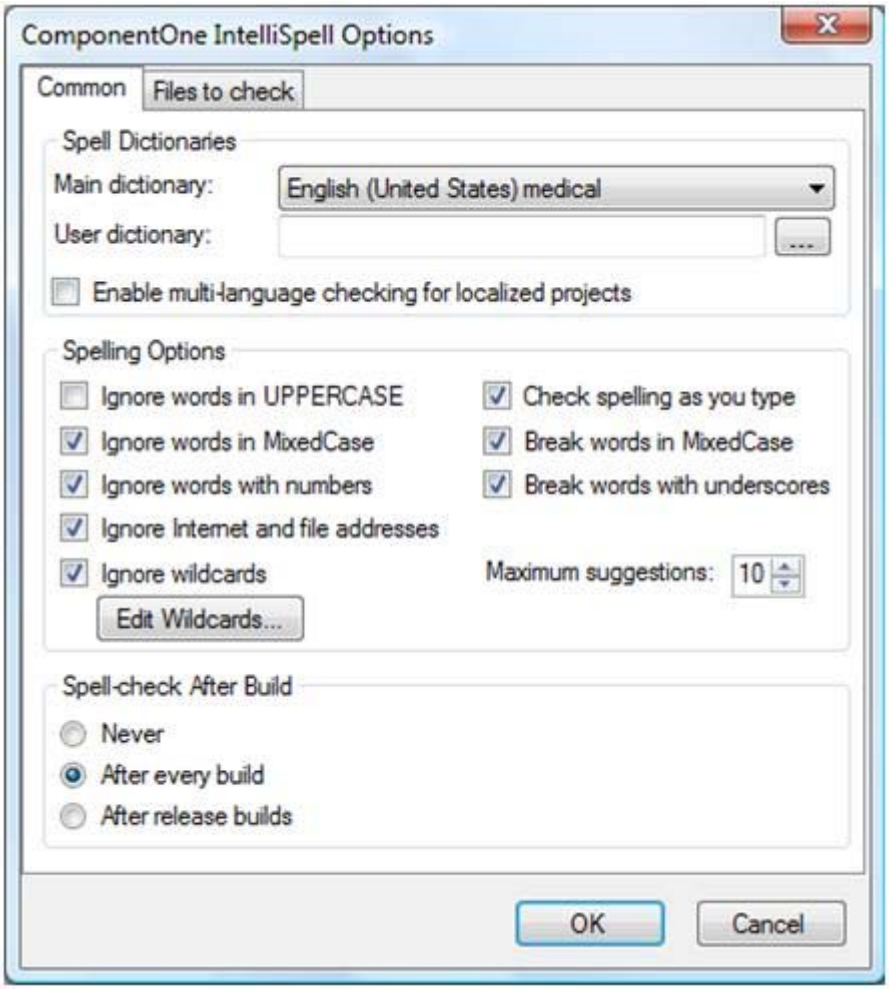
Add Controls to the Form

With the project created, you now have a new form in front of you; let's add the ComponentOne controls needed for this sample.

In Microsoft Visual Studio, go to your Toolbox and locate the **C1PrintDocument** control. Drag and drop the **C1PrintDocument** control onto the form followed by the **C1PrintPreviewControl**. Resize and adjust the form and control so it is where you would like it to be.

Next, we now must set the viewer for **C1PrintDocument**. To do this, click on the SmartTag in the upper-right corner of the control.

In the **C1PrintDocument Tasks** menu, click the checkbox for c1PrintPreviewControl1. Below is an image of what you should see.



We have now given the **C1PrintDocument** control the ability to display its contents on the **C1PrintPreviewControl**. With these steps completed, we are now done with setting up the form on the design surface. The majority of the work is done through code, which includes the formatting of the contents for the PDF form.

Create the Form

With the controls added, we will begin to put the pieces of the puzzle together. Before we design the layout and display it, we first need to create the form and get our declarations complete. Below is the first portion of the code, the declarations.

To use the processes with Preview for WinForms, we first must add it to the declarations. Before the namespace, add the using C1.C1Preview; statement to your code. The code below shows the statement added.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;using C1.C1Preview;
```

The next lines of code are placed in your form automatically, which I will also show as we continue to progress with this example. A few changes will need to be made to the lines, such as giving the name to the document you are creating. For this example, the name of this document is docEmp. Throughout the form you will see this being used so make it something unique.

```
namespace EmployeeForm
{
    public partial class Acroform : Form
```

```
    {
        C1PrintDocument docEmp;
        public Acroform()
        {
            InitializeComponent();
        }
```

Load the Form

You have two options to load the form: the first is going back to the design tab and double-clicking on the form, and the second option is staying within the code tab and using the drop-down lists to place the code for the Form load in place. For this example, we will choose the second option. Under form_load, we will place our document in the form and making some visual changes to it. Also, a line of code will enable a zoom feature within **C1Preview** to enhance our project.

```
private void Acroform_Load(object sender, EventArgs e)
        {
            createEmployeeForm();
            c1PrintPreviewControl1.Document = docEmp;
            c1PrintPreviewControl1.PreviewPane.ZoomMode = C1.Win.C1Preview.ZoomModeEnum.PageWidth;
            c1PrintPreviewControl1.ToolBars.Navigation.Visible = true;
        }
```

Create the Form Fields

Our goal here is to create the form fields needed in our example as well as state the page settings and styles. We will use the four form fields available for Acroforms (**Text**, **RadioButton**, **ComboBox**, and **CheckBox**), and declare unique names and sizes.

First we will code our page settings and styles. For these lines of code, our objective is the page layout, font, form style, and table creation. Add the following code to your Form:

```
private void createEmployeeForm()
        {
            //Creating layout for document, using text, radiobutton, combobox and checkbox.
            docEmp = new C1PrintDocument();
            docEmp.PageLayouts.Default.PageSettings = new C1.C1Preview.C1PageSettings(false, System
                "1in", "1in", "1in", "1in");
            docEmp.AllowNonReflowableDocs = true;
            docEmp.Style.Font = new Font("Times New Roman", 11);
            docEmp.FormsStyle = FormsStyleEnum.WinXp;
            RenderTable rt = new RenderTable(30, 4);
            rt.Cols[0].Width = 1.4;
            rt.Cols[2].Width = 1.4;
            rt.Cols[1].Style.TextAlignHorz = AlignHorzEnum.Right;
```

Setting up the form fields for the doc is very simple. For each field, the same line of code will be used except for the name as it needs to be unique for each. Giving examples of each field's code and their size are documented beneath; the entire code for the fields in this example will be shown after these examples.

The **Text** control in this section will need to be setup as such:

```
RenderInputText rFirstName = new RenderInputText();
rFirstName.Width = 1.8;
```

The **RadioButton** (rb) control in this section will need to be setup as such:

```
RenderInputRadioButton rb1 = new RenderInputRadioButton("< 5 Years", 0);
```

The **ComboBox** control in this section will need to be setup as such:

```
string[] fict_Departments = { "Finance", "Marketing", "Sales", "IT", "Management", "Maintenance" };
RenderInputComboBox rDepartments = new RenderInputComboBox(fict_Departments);
rDepartments.Width = 1.8;
```

The **CheckBox** (cb) control in this section will need to be setup as such:

```
RenderInputCheckBox cb1 = new RenderInputCheckBox(" Dental Care Plan");
```

Now we see the syntax for each field, let's look at how this code is used for our example.

```
            //Text Box Formating.
            RenderInputText rFirstName = new RenderInputText();
            rFirstName.Width = 1.8;
            RenderInputText rLastName = new RenderInputText();
            rLastName.Width = 1.8;
            RenderInputText rStreet = new RenderInputText();
            rStreet.Width = 5.05;
            //Combobox creation.
            string[] fict_Departments = { "Finance", "Marketing", "Sales", "IT", "Management", "Mai
            RenderInputComboBox rDepartments = new RenderInputComboBox(fict_Departments);
            rDepartments.Width = 1.8;
            //Text Box Formating.
            RenderInputText rbirth = new RenderInputText();
            rbirth.Width = 1.8;
            RenderInputText rKname = new RenderInputText();
            rKname.Width = 1.8;
            RenderInputText rKstreet = new RenderInputText();
            rKstreet.Width = 5.05;
            RenderInputText rStart = new RenderInputText();
            rStart.Width = 1.8;
            RenderInputText rSalary = new RenderInputText();
            rSalary.Width = 1.8;
            RenderInputText rKphone = new RenderInputText();
            rKphone.Width = 1.8;
            //Radiobutton layout setup.
            RenderInputRadioButton rb1 = new RenderInputRadioButton("< 5 Years", 0);
            RenderInputRadioButton rb2 = new RenderInputRadioButton("5 - 10 Years", 0);
            RenderInputRadioButton rb3 = new RenderInputRadioButton("11 - 15 Years", 0);
            RenderInputRadioButton rb4 = new RenderInputRadioButton("> 15 Years", 0);
            RenderInputRadioButton rb5 = new RenderInputRadioButton("Single Health Care Plan");
            RenderInputRadioButton rb6 = new RenderInputRadioButton("Family Health Care Plan");
            //Checkbox layout setup.
            RenderInputCheckBox cb1 = new RenderInputCheckBox(" Dental Care Plan");
            RenderInputCheckBox cb2 = new RenderInputCheckBox(" Vision Care Plan");
            RenderInputCheckBox cb3 = new RenderInputCheckBox(" 401k Retirement Plan");
            RenderInputCheckBox cb4 = new RenderInputCheckBox(" Company Stock Option");
            RenderInputCheckBox cb5 = new RenderInputCheckBox(" If address is same as above");
```

Input Fields to the Form

At this point, we have done everything to prepare our Form for the document. The following steps create the look and feel of our document. Recall our setup for the table: we set the table to have a maximum of 30 rows and 4 columns. To declare row 1, column 3, for example, [1, 3] would be used.

As a reference, here are some sample lines of code we will be using.

To add a blank line of code in your table your code would be:

```
rt.Cells[5, 0].Text = " ";
```

Taking this a step further, you can add text to the form. Below is the syntax for text:

```
rt.Cells[0, 0].Text = "First Name:";
```

In addition, the form fields already created in our application now need to be coded to appear on the form. To make this possible, below is a sample line to get you familiar:

```
rt.Cells[10, 0].Area.Children.Add(rb1);
```

Where, rb1 is the name of the field and [10, 0] is the position of the field on the form.

Grasping these examples will allow us to finish the syntax for the example. The following code is the code for the Employee Information Form. Add the following code under the private void createEmployeeForm() section:

```
            //Inputing form fields and text to the form.
            //Start creating doc.
            docEmp.StartDoc();
            docEmp.RenderBlockRichText("Employee Information Form");
            rt.Cells[0, 0].Text = "First Name:";
            rt.Cells[0, 1].Area.Children.Add(rFirstName);
            rt.Cells[0, 2].Text = "  Last Name:";
            rt.Cells[0, 3].Area.Children.Add(rLastName);
            rt.Cells[1, 0].Text = " ";
            rt.Cells[2, 0].Text = "Address:";
            rt.Cells[2, 1].Area.Children.Add(rStreet);
```

```
            rt.Cells[3, 0].Text = " ";
            rt.Cells[4, 0].Text = "Birth Date:";
            rt.Cells[4, 1].Area.Children.Add(rbirth);
            rt.Cells[4, 2].Text = "  Department:";
            rt.Cells[4, 3].Area.Children.Add(rDepartments);
            rt.Cells[5, 0].Text = " ";
            rt.Cells[6, 0].Text = "Start Date:";
            rt.Cells[6, 1].Area.Children.Add(rStart);
            rt.Cells[6, 2].Text = "  Salary:";
            rt.Cells[6, 3].Area.Children.Add(rSalary);
            rt.Cells[7, 0].Text = " ";
            rt.Cells[8, 0].Text = "Years Employed By Company:";
            rt.Cells[8, 0].SpanCols = 4;
            rt.Cells[9, 0].Text = " ";
            rt.Cells[10, 0].Area.Children.Add(rb1);
            rt.Cells[10, 1].Area.Children.Add(rb2);
            rt.Cells[10, 2].Area.Children.Add(rb3);
            rt.Cells[10, 3].Area.Children.Add(rb4);
            rt.Cells[11, 0].Text = " ";
            rt.Cells[12, 0].Text = "Benefits Section - Check all that apply:";
            rt.Cells[12, 0].SpanCols = 4;
            rt.Cells[13, 0].Text = " ";
            rt.Cells[14, 0].Area.Children.Add(rb5);
            rt.Cells[14, 2].Area.Children.Add(rb6);
            rt.Cells[15, 0].Text = " ";
            rt.Cells[16, 0].SpanCols = 4;
            rt.Cells[16, 0].Area.Children.Add(cb1);
            rt.Cells[16, 0].SpanCols = 4;
            rt.Cells[17, 0].Area.Children.Add(cb2);
            rt.Cells[17, 0].SpanCols = 4;
            rt.Cells[18, 0].Area.Children.Add(cb3);
            rt.Cells[18, 0].SpanCols = 4;
            rt.Cells[19, 0].Area.Children.Add(cb4);
            rt.Cells[20, 0].Text = " ";
            rt.Cells[21, 0].Text = "Emergency Contact Information:";
            rt.Cells[21, 0].SpanCols = 4;
            rt.Cells[22, 0].Text = " ";
            rt.Cells[23, 0].Text = "Name:";
            rt.Cells[23, 1].Area.Children.Add(rKname);
            rt.Cells[24, 0].Text = " ";
            rt.Cells[25, 0].Text = "Address:";
            rt.Cells[25, 1].Area.Children.Add(rKstreet);
            rt.Cells[26, 0].Area.Children.Add(cb5);
            rt.Cells[27, 0].Text = " ";
            rt.Cells[28, 0].Text = "Phone #:";
            rt.Cells[28, 1].Area.Children.Add(rKphone);
            docEmp.RenderBlockText(" ");
            docEmp.RenderBlock(rt);
```

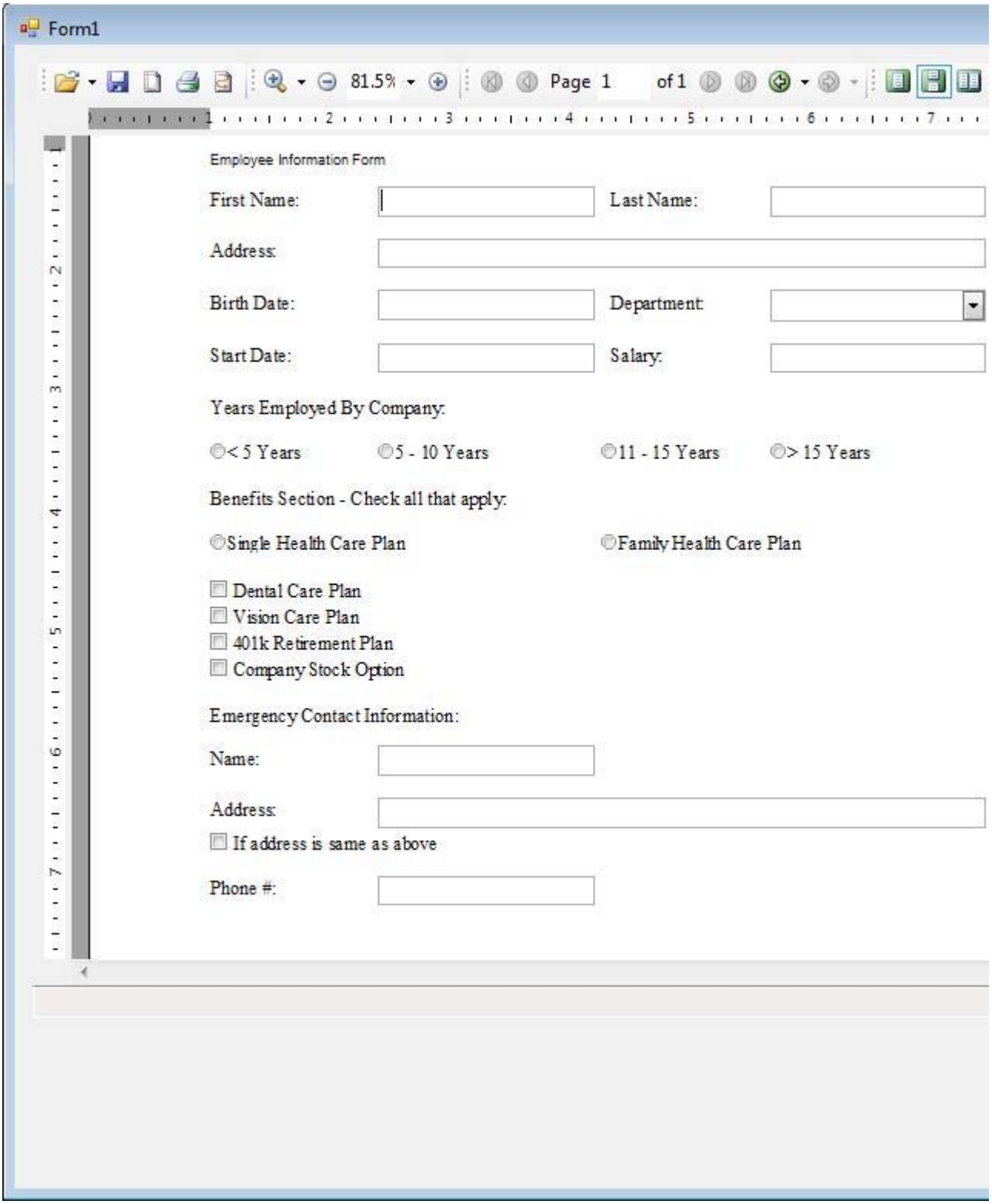
Almost there, now we have the entire doc created, and formatted. Now we need to finalize the document with a really short line of code. Below is the last portion of code we will need for this example.

```
            //Finished creating doc.
            docEmp.EndDoc();
        }

    }
}
```

Run the Application

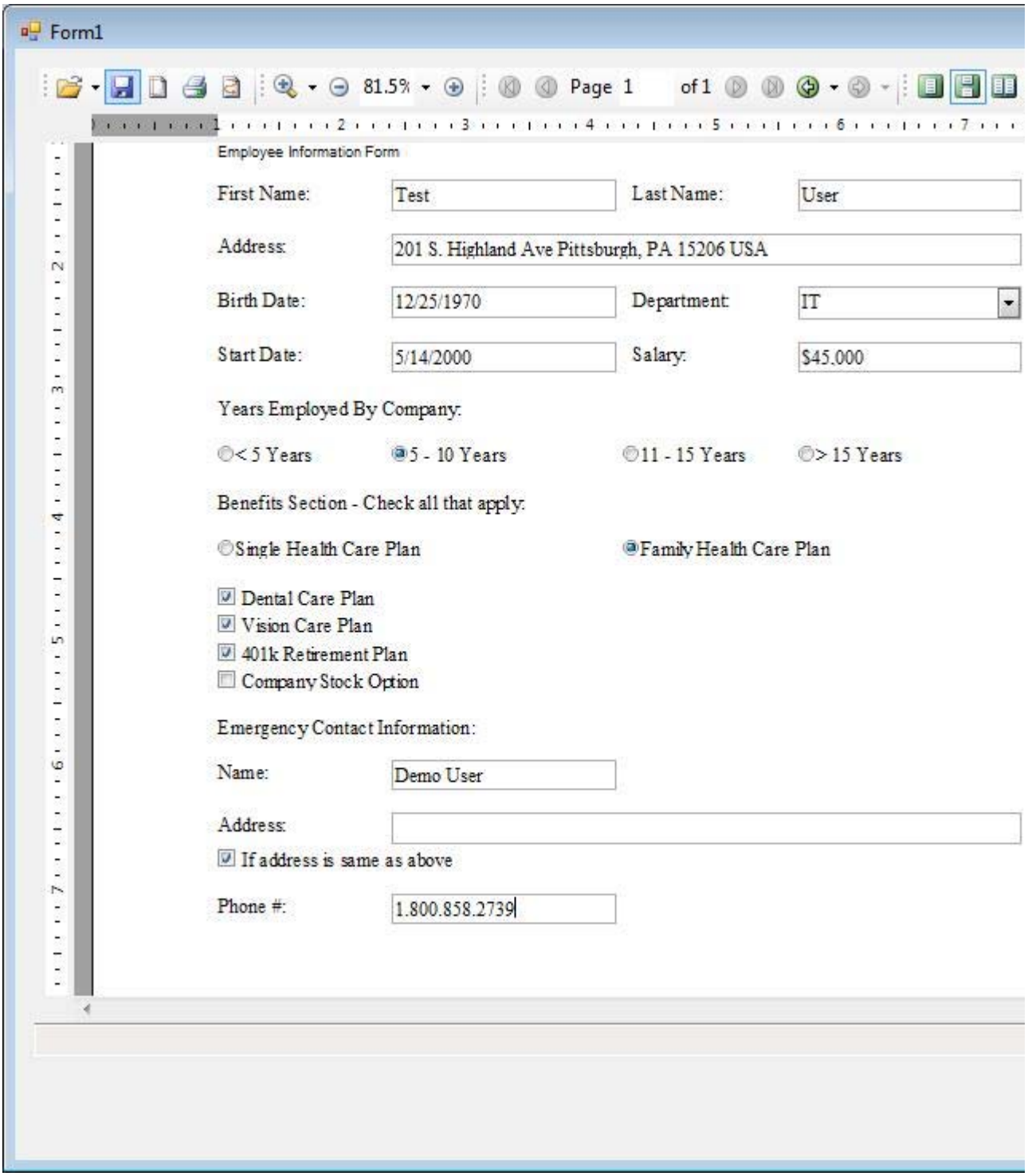
Press **F5** to run the application. You will see the Employee Information Form we created.



Edit and Save the Form

With the Employee Information Form running, lets demo the form. Fill out the form, and interact with the **RadioButton**, **ComboBox**, and **Checkbox** controls. Also the form has the zoom feature, zooming in and out can be done with ease, as well as opening, saving, and printing the forms.

As you can see below, the form is fully interactive which will save time and make PDF file creation very simple. Every field is complete and the **Save** button is highlighted to save the file. The save feature will make forms paperless, saving money for the company, helping the company become greener, and providing an easier way of access.



Conclusion

Finally, with a little bit of work we turned a hard copy paper system into digital system with endless results. Knowing what we can do with ComponentOne Preview for WinForms, it's time to leave the unledgible, wrinkled, coffee-stained form days behind. With ease, we created a professional looking form.

Get started today. Download the trial version of ComponentOne Studio for WinForms, which includes ComponentOne Preview for WinForms.

Free Download