Hands-on: Image Analysis

Using Azure Cognitive Services, Computer Vision

In this hands-on lab, you'll explore the Image Analysis functionality of Computer Vision feature of Azure Cognitive Services. There are a number of features that Computer Vision provides, and Dynamics 365 Business Central takes advantage of the Computer Vision services to provide standard, horizontal functionality, that you can use from your own extensions to perform generalized analysis of images. For example, you can analyze image tags to assist you in classifying items in your item catalog, or to recognize age in employee pictures.

Contents

Obtaining Computer Vision API key

In this exercise, you obtain a personal API key that allows you to access the Computer Vision API of Microsoft Azure Cognitive Services. This key is free, is limited to a 7-day trial period, and is required to perform all of the exercises in this hands-on lab.

1. Open your browser and navigate to https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/
2. Click **Try the Computer Vision API**.
3. In the **Try Cognitive Services for free** dialog, under Guest 7-day trial, click **Get started**.
4. In the **Microsoft Cognitive Services Terms** dialog, select the **I agree** checkbox, make sure that **United States** is selected in the **Select your Country / Region** combo box, select the **I accept** checkbox, and then click **Next**.
5. In the **Sign-in to continue** dialog, choose your preferred identity provider. Depending on which provider you choose, the following process may be slightly different.
6. If the next step asks you to authorize azure.microsoft.com to access the account details of your chosen identity provider, select all permissions (if applicable) and then complete the authorization. The **Welcome** dialog will show at some point.
7. In the **Welcome** dialog, click **Got it**. The **Welcome** dialog disappears.
8. The web page now displays your two access points and two API keys. At the same time, you'll receive an e-mail with the link that you can follow to get access to this information. Still, copy the two URLs and two keys, and save them in a file.

Configure Image Analysis Setup in Business Central

In this exercise, you configure your Business Central instance to access the Cognitive Services API on your behalf, using your key, and the endpoint URL you received. This exercise will work in both your local Docker container sandbox, and in the online sandbox.

1. Open your browser and navigate to your Business Central homepage.
2. Click on the **Tell me what you want to do** button () and search for “Image Analysis” and then click the **Image Analysis Setup** page.
3. In the **API URI** field, enter the “v1.0” API endpoint URL that you received, and then click in the **API Key** field.
4. A dialog box shows, asking you whether to append “/analyze” to the URL. Click **Yes** to confirm.
5. In the **API Key** field, enter the value of Key 1 that you received.
6. Close the **Image Analysis Setup** page.

Prepare a demo extension workspace

In this exercise, you create and prepare a new Visual Studio Code workspace that you will use throughout this hands-on lab.

1. In Visual Studio Code, create a new workspace, and name it “Image Analysis”. If you are not sure how to do this, follow the instructions in the **General – Hands-on – Beginner instructions** document.
2. Configure the workspace to use idRange from 50100 to 50109.
3. Clean up your environment by removing the HelloWorld.al file.
4. If you are using a local container sandbox, open the launch.json file, then modify the **server**, **serverInstance**, and **authentication** values to point to your local sandbox environment. If you are not sure exactly which values go here, ask your instructor for help.
5. In the launch.json file, make sure that the following properties are set as shown here:

```
            "startupObjectId": 31,
            "startupObjectType": "Page",
```

6. Make sure to download the symbols from your Business Central instance. Again, if not sure how to do this, follow the instructions in the **General – Hands-on – Beginner instructions** document.

Write a codeunit to test Image Analysis Setup

In this exercise, you write code to use the **Image Analysis Management** codeunit to verify that Image Analysis has been correctly configured, and that Business Central can use your API key. You will also extend the **Item List** page and add an action to invoke your codeunit.

Solution code is shown at the end of the exercise. Try to write as much code without looking or copying/pasting from the solution. The solution is provided as help, not as a cheat-sheet.

1. Create a new file and name it "Codeunit 50101 Image Analysis Client.al".
2. In the editor, enter code to declare codeunit 50101 Image Analysis Client. If you used a snippet to create the codeunit, clean up any automatically created content.
3. Create a function named AnalyzeItemTags that receives the following parameter:

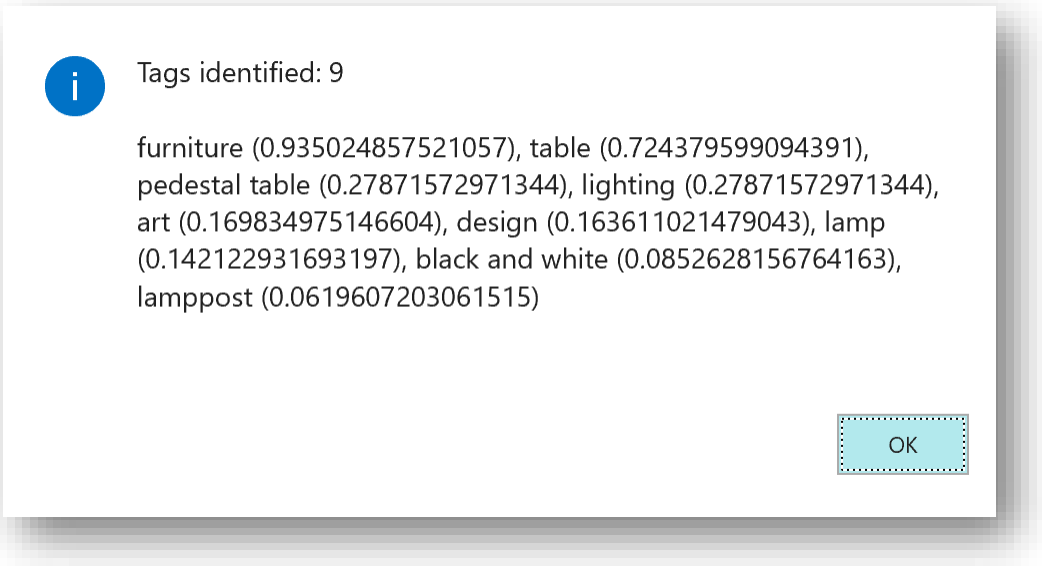
Parameter	Type
Item	Record Item

4. Declare the following local variables for the AnalyzeItemTags function**:**

Parameter	Type
ImageAnalysis	Codeunit "Image Analysis Management"
Result	Codeunit "Image Analysis Result"
ErrorMessage	Text
IsUsageLimitError	Boolean
i	Integer
Tags	Text

5. In the body of the function, write code to check that the item passed into the function contains media in its mediaset. If there are no media, throw an error.
   Hint: Use the Count function of the Picture field.
6. Write code to initialize the **Image Analysis Management** codeunit.
   Hint: use intellisense to inspect the public functions exposed by the codeunit.
7. Write code to pass the first media from the item's mediaset into the SetMedia function of the **Image Analysis Management** codeunit.
8. Write code to invoke the AnalyzeTags function. Pass the appropriate variable into it to receive the results.
   Hint: check which type the AnalyzeTags function expects, then declare a local variable for that type, and pass it into the function.
9. Write code to show an error message if the AnalyzeTags returns false. Read the error message from the **Image Analysis Management** codeunit by invoking its GetLastError function. Declare the necessary variables to be able to successfully invoke this function.
10. Write code to show the number of tags returned from the AnalyzeTags function, and to show the list of all the tags identified. With each tag, show the confidence level for that tag.
    Hint: use the TagCount, TagName, and TagConfidence functions of the **Image Analysis Result** codeunit.
11. Create a new file and name it "PageExtension 50101 Item List Extension.al".

12. In the editor, enter code to declare pageextension 50101 Item List Extension object to extend the **Item List** page. If you used a snippet to create the pageextension, clean up the page by removing layout and global variables sections. Retain the actions section.
13. Write code to add a last action to the **Item** group. Configure the action to invoke the AnalyzeItemTags function of the codeunit you have just created, by passing the current Item record to it.
14. Save all objects, then build and run your extension (press Ctrl+F5).
    Hint: If you used F5 and started a debugging session, and if your debugger stops on errors from standard Business Central code, ignore these errors and continue the execution by pressing F5.
15. In the Item list, select an item with a picture, and then click **… > Actions > Item > Run Image Analysis** to invoke the action you added through the Item List page extension. If everything works correctly, you should se results similar to these:



Solution Codeunit 50101 Image Analysis Client.al

```
codeunit 50101 "Image Analysis Client"
{
    procedure AnalyzeItemTags(Item: Record Item)
    var
        ImageAnalysis: Codeunit "Image Analysis Management";
        Result: Codeunit "Image Analysis Result";
        ErrorMessage: Text;
        IsUsageLimitError: Boolean;
        i: Integer;
        Tags: Text;
    begin
        if Item.Picture.Count = 0 then
            Error('No media available for this item.');
```

```
        ImageAnalysis.Initialize();
        ImageAnalysis.SetMedia(Item.Picture.Item(1));
        if not ImageAnalysis.AnalyzeTags(Result) then begin
            ImageAnalysis.GetLastError(
                ErrorMessage,
                IsUsageLimitError);
            Error('Invocation error: %1', ErrorMessage);
        end;

        for i := 1 to Result.TagCount do begin
            Tags += StrSubstNo('%1 (%2)',
                Result.TagName(i),
                Result.TagConfidence(i));
            if (i < Result.TagCount) then
                Tags += ', ';
        end;
        Message('Tags identified: %1\\%2', Result.TagCount, Tags);
    end;
}
```

Soluition: PageExtension 50101 Item List Extension.al

```
pageextension 50101 "Item List Extension" extends "Item List"
{
    actions
    {
        addlast(Item)
        {
            action(TestImageAnalysis)
            {
                Caption = 'Run Image Analysis';
                Image = Picture;
                ApplicationArea = All;

                trigger OnAction();
                var
                    Test: codeunit "Image Analysis Client";
                begin
                    Test.AnalyzeItemTags(Rec);
                end;
            }
        }
    }
}
```

Develop functionality to analyze faces

In this exercise you'll write the necessary code to analyze the faces of employees in the **Employee** table by invoking Computer Vision API of Azure Cognitive Services. You'll write code similar to what you did with the Item List, except this time you'll do this for the Employee List. Instead of showing image tags, this time you'll show the number of people in the photo, together with their gender and age.

1. In the Image Analysis Client codeunit, create a new function similar to the AnalyzeItemTags function and name it AnalyzeEmployeePhoto. This new function will invoke the Computer Vision API in a similar way as AnalyzeItemTags, except that this time you will invoke the AnalyzeFaces function, and you will use FaceCount, FaceGender, and FaceAge functions to analyze the result content. The function should fail if the employee record does not have a picture.
   Hint: **Table Employee** does not use mediaset, but media to store pictures. You will have to use a different approach to both check and pass the value from the table to the SetMedia function. To check whether the value in Media field exists, you have to check whether the MediaId is equal to the empty (uninitialized) guid. If you are still unsure exactly how to perform the check, then check the exercise solution.
2. Create a new object pageextension 50102 "Employee List Extension" that extends the **Employee List** page. Inside, define a new action that passes the current employee record to the AnalyzeEmployeePhoto function you created in the previous step.
3. Build, deploy, and run the extension.
4. To test the function, click **Finance > Employees**, then **Navigate > Employee > Run Image Analysis**. If you did everything correctly, you should see a result such as this:

Solution: The AnalyzeEmployeePhoto function

```
procedure AnalyzeEmployeePhoto(Rec: Record Employee)
var
    ImageAnalysis: Codeunit "Image Analysis Management";
    Result: Codeunit "Image Analysis Result";
    ErrorMessage: Text;
    IsUsageLimitError: Boolean;
    i: Integer;
    Faces: Text;
```

```
    begin
        if IsNullGuid(Rec.Image.MediaId) then
            Error('No picture available for this employee.');

        ImageAnalysis.Initialize();
        ImageAnalysis.SetMedia(Rec.Image.MediaId());
        if not ImageAnalysis.AnalyzeFaces(Result) then begin
            ImageAnalysis.GetLastError(
                ErrorMessage,
                IsUsageLimitError);
            Error('Invocation error: %1', ErrorMessage);
        end;

        for i := 1 to Result.FaceCount do begin
            Faces += StrSubstno('%1 (%2)',
                Result.FaceGender(i),
                Result.FaceAge(i));
            if (i < Result.FaceCount) then
                Faces += ', ';
        end;
        Message('Faces identified: %1\\%2', Result.FaceCount, Faces);
    end;
```

Solution: PageExtension 50102 Employee List Extension.al

```
pageextension 50102 "Employee List Extension" extends "Employee List"
{
    actions
    {
        addlast("E&mployee")
        {
            action(ImageAnalysis)
            {
                Caption = 'Run Image Analysis';
                Image = Picture;
                ApplicationArea = All;

                trigger OnAction();
                var
                    Test: codeunit "Image Analysis Client";
                begin
                    Test.AnalyzeEmployeePhoto(Rec);
                end;
            }
        }
```

```
    }
}
```

Develop functionality to analyze colors

Finally, you will learn how to perform color analysis. You will use the same codeunits to invoke Computer Vision API and process the invocation results, except this time you will use the functions specific for the color analysis.

In the Image Analysis Client codeunit, create a new function similar to other functions you created, and name it AnalyzeCompanyLogo. This new function will invoke the Computer Vision API in a similar way as AnalyzeItemTags, except that it will invoke the AnalyzeColors function and then you'll use DominantColorForeground and DominantColorBackground to show dominant foreground and background colors. The function should fail if the company information record does not have a picture.

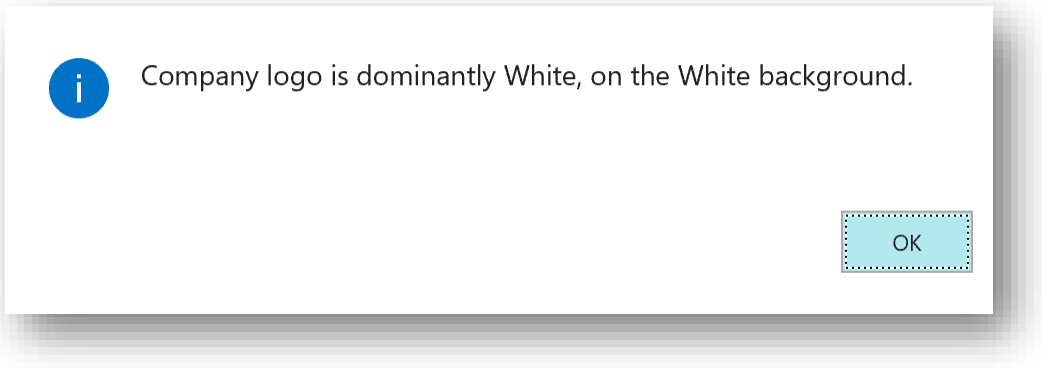
Hint: Table **Company Information** does not use mediaset or media, but BLOB to store pictures. You will have to use a different approach to both check and pass the value from the table to the **Image Analysis Management** codeunit function. You will need a variable of type record, subtype TempBlob, and it should be a temporary table.

To check whether the value in Picture field exists, you have to use the HasValue function.

To assign the picture to TempBlob, you can use the following code:

```
        Rec.CalcFields(Picture);
        TempBlob.Blob := Rec.Picture;
```

1. Create a new object pageextension 50103 "Company Information Extension" that extends the **Company Information** page. Inside, define a new action that passes the current company information record to the AnalyzeCompanyLogo function you created in the previous step.
2. Build, deploy, and run the extension.
3. To test the function, search for the Company Information page, then click **Navigate > Run Image Analysis**. If you did everything correctly, you should see a result such as this:



Solution: AnalyzeCompanyLogo function

```
    procedure AnalyzeCompanyLogo(Rec: Record "Company Information")
    var
        TempBlob: Record TempBlob temporary;
        ImageAnalysis: Codeunit "Image Analysis Management";
        Result: Codeunit "Image Analysis Result";
        ErrorMessage: Text;
```

```
        IsUsageLimitError: Boolean;
        i: Integer;
        Tags: Text;
    begin
        if not Rec.Picture.HasValue() then
            Error('No media available for this item.');

        Rec.CalcFields(Picture);
        TempBlob.Blob := Rec.Picture;

        ImageAnalysis.Initialize();
        ImageAnalysis.SetBlob(TempBlob);
        if not ImageAnalysis.AnalyzeColors(Result) then begin
            ImageAnalysis.GetLastError(
                ErrorMessage,
                IsUsageLimitError);
            Error('Invocation error: %1', ErrorMessage);
        end;

        Message('Company logo is dominantly %1, on the %2 background.',
            Result.DominantColorForeground(),
            Result.DominantColorBackground());
    end;
```

Solution: PageExtension 50103 Company Information Extension.al

```
pageextension 50103 "Company Information Extension" extends "Company Information"
{
    actions
    {
        addlast(Navigation)
        {
            action(ImageAnalysis)
            {
                Caption = 'Run Image Analysis';
                Image = Picture;
                ApplicationArea = All;

                trigger OnAction();
                var
                    Test: codeunit "Image Analysis Client";
                begin
                    Test.AnalyzeCompanyLogo(Rec);
                end;
            }
```

```
        }
    }
}
```

Challenge yourself

This exercise provides no step-by-step instructions and requires you to write the code entirely from scratch. However, the logic and functionality you invoke will be very similar to the previous exercises.

Item Tag Analysis: find all chairs

Create functionality that analyzes all items in the database, and then shows the list of those items that contain a picture of a chair. The confidence level of the tag must be above 67% for you to accept the identification.

Employee Faces: find all persons younger than declared

Create functionality that analyzes all employees in the database. For each employee, check whether their registered age in the employee record matches the age recognized from the picture, and whether their registered gender matches the gender recognized from the picture. Finally, show the list of all employees whose picture is obviously outdated (shows a younger person) or whose gender does not match.