Microsoft | Power BI

PBICAT

State	Sales Revenue	Units Sold	Customer Count
CA	$5,394,441	681607	10,843
TX	$3,829,475	644747	7,922
FL	$2,558,182	475889	3,577
OR	$1,789,160	215837	3,426
GA	$1,707,677	305316	3,530
WA	$1,681,541	223822	3,284
AZ	$1,586,886	236120	3,321
NC	$1,448,869	261683	3,109
NY	$1,303,529	258122	2,891
CO	$1,037,779	166322	2,155
VA	$993,454	191731	2,190
PA	$887,492	178222	1,917
NM	$833,785	123267	1,604
TN	$728,281	138460	1,579
MA	$569,535	110471	1,266
NJ	$509,734	96096	1,153
CT	$497,090	100693	1,143
SC	$458,267	82863	978
LA	$450,445	91901	963
OH	$416,938	77600	975
UT	$384,891	56343	790
AL	$260,971	47955	559
NH	$256,513	47601	623
RI	$203,680	36552	459
Total	$29,788,617	4849220	62,257

Building Multi-language Reports in Power BI

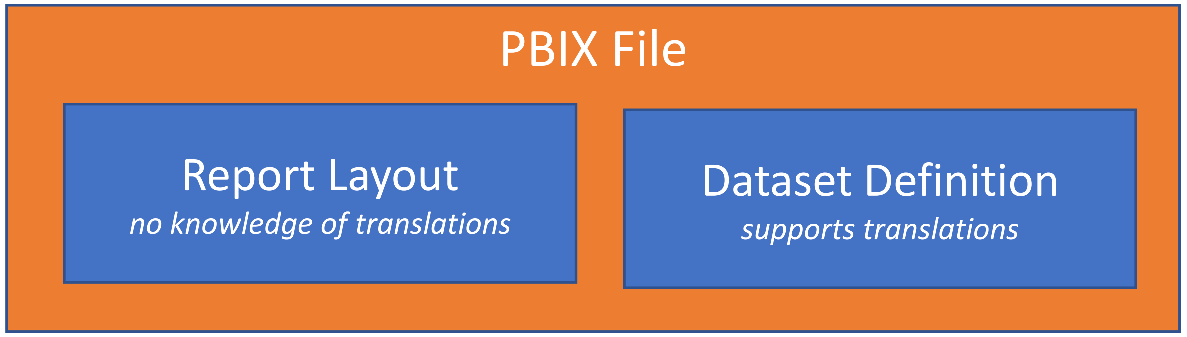
Ted Pattison
Power BI Customer Advisory Team (PBICAT)

Agenda

- ➢ Overview of Multi-language PBIX Development
- Designing Multi-language Reports
- Adding Metadata Translations with TOM
- Embedding Reports with Specific Locales
- Designing Data Models to Support Content Translations
- Setting the Language for Current User using RLS and UserCulture

Overview of Multi-language PBIX Development

- PBIX Localization only supported for data model
  - Report designer has no knowledge of dataset localization support



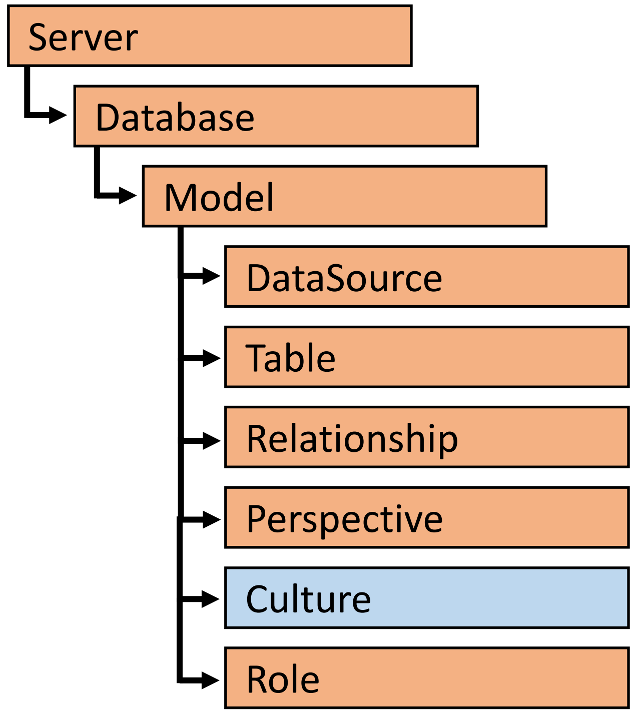
- Metadata translations supported for table names and field name
  - This technique not used to localize metadata labels but not to localize content
  - Requires tricks in the report designer to localize string content on report
- Content translations added using specific data model design
  - Best practices currently based on replicated rows pattern

Workflow for Multi-language PBIX Development

1. Use Power BI Desktop to prepare PBIX to support translations
    - Design data model with tables, columns, measure and hierarchies
    - Create **Localized Labels** table with measures with localizable names
2. Add metadata translations to PBIX
    - Add metadata translations to PBIX project using External Tool support
    - Requires using Tabular Editor or programming Tabular Object Model (TOM)
3. Design and implement content translation strategy
    - Content translation design based replicated rows pattern
    - Can be implemented using Power Query and translation lookup tables

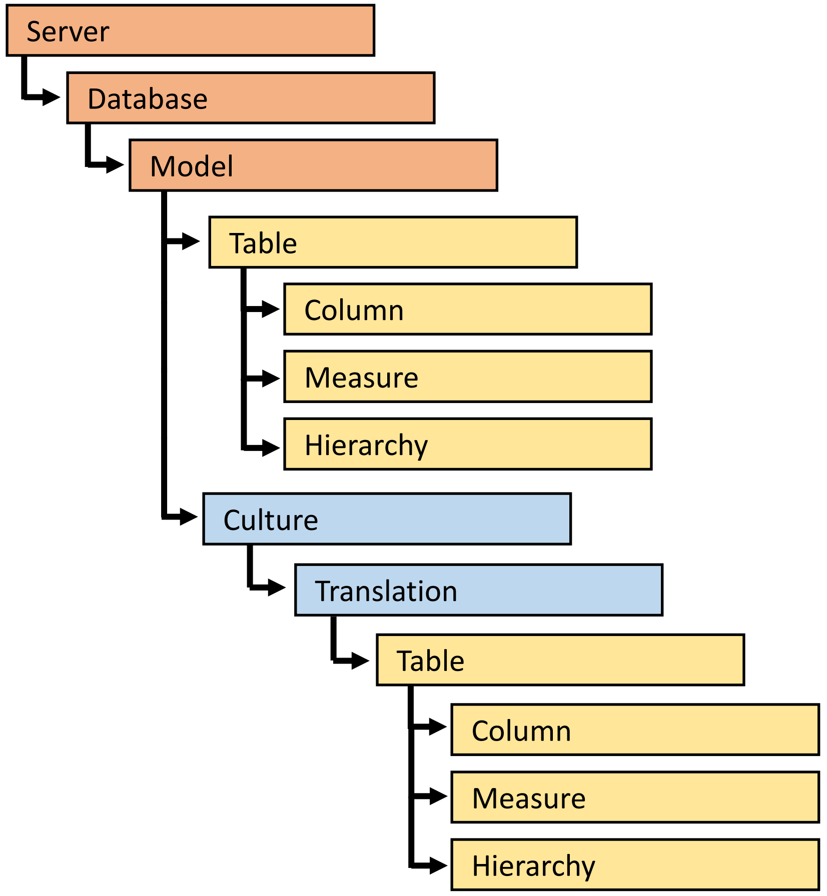
Tabular Object Model (TOM)

- TOM is extension of Analysis Management Object (AMO) client library
  - Created to support programming tabular model databases in SQL Analysis Services
  - Updated to support programming datasets in Power BI Desktop and the Power BI Service
- TOM provides a programmatic way to view/edit data models
  - Creating models
  - importing and refreshing data
  - Assigning roles and permissions
  - Adding support for secondary cultures/languages



Adding Metadata Translations

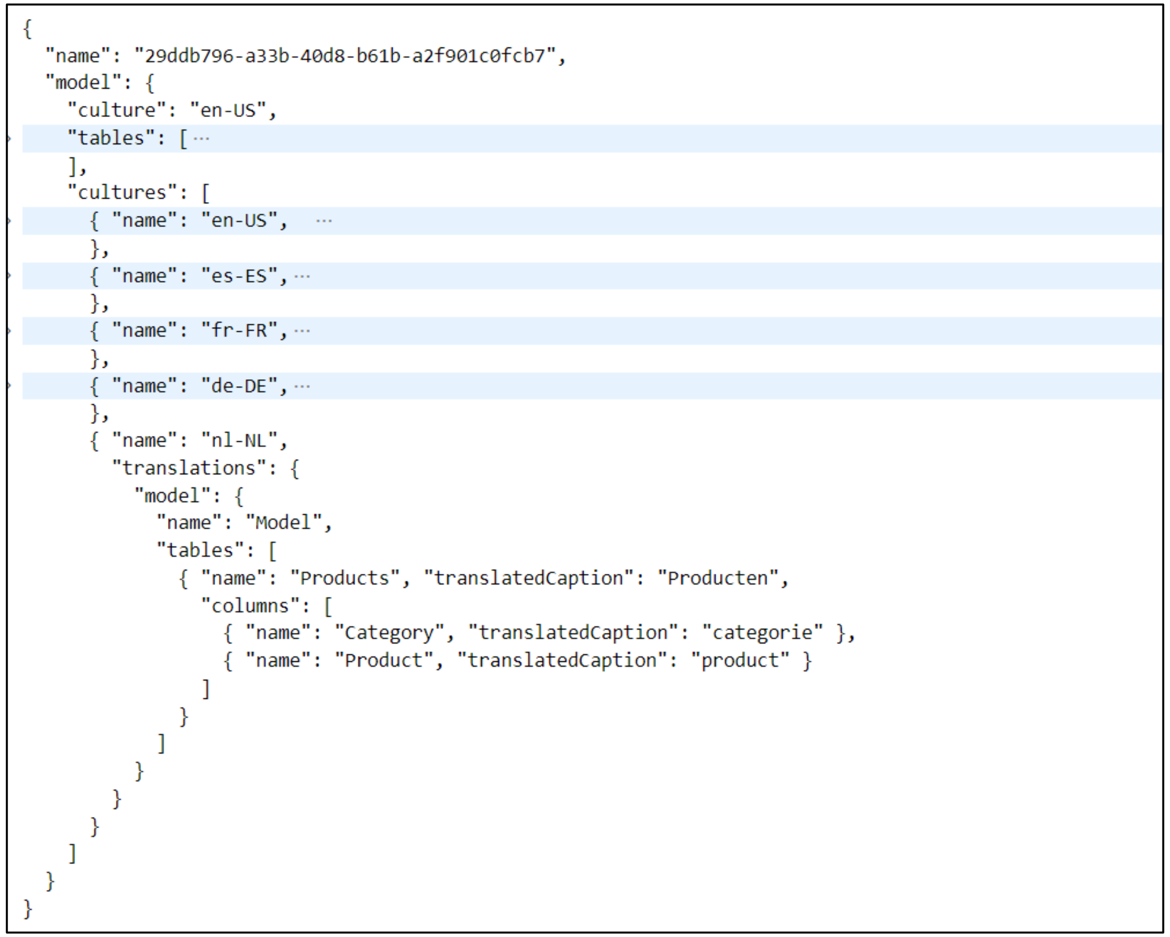
- When you create a new PBIX project…
  - It has a default culture (en-US) but contains no translations
  - You must add translations for each spoken language



```
{
  "name": "29ddb796-a33b-40d8-b61b-a2f901c0fcb7",
  "model": {
    "culture": "en-US",
    "tables": [
      { "name": "Products",
        "columns": [
          { "name": "Category", "dataType": "string" },
          { "name": "Product", "dataType": "string" }
        ] }
    ],
    "cultures": [
      {
        "name": "en-US",
        "translations": {
          "model": {
            "name": "Model",
            "tables": [
              { "name": "Products", "translatedCaption": "Products",
                "columns": [
                  { "name": "Category", "translatedCaption": "Category" },
                  { "name": "Product", "translatedCaption": "Product" }
                ]
              }
            ]
          }
        }
      }
    ]
  }
}
```

Adding Secondary Cultures

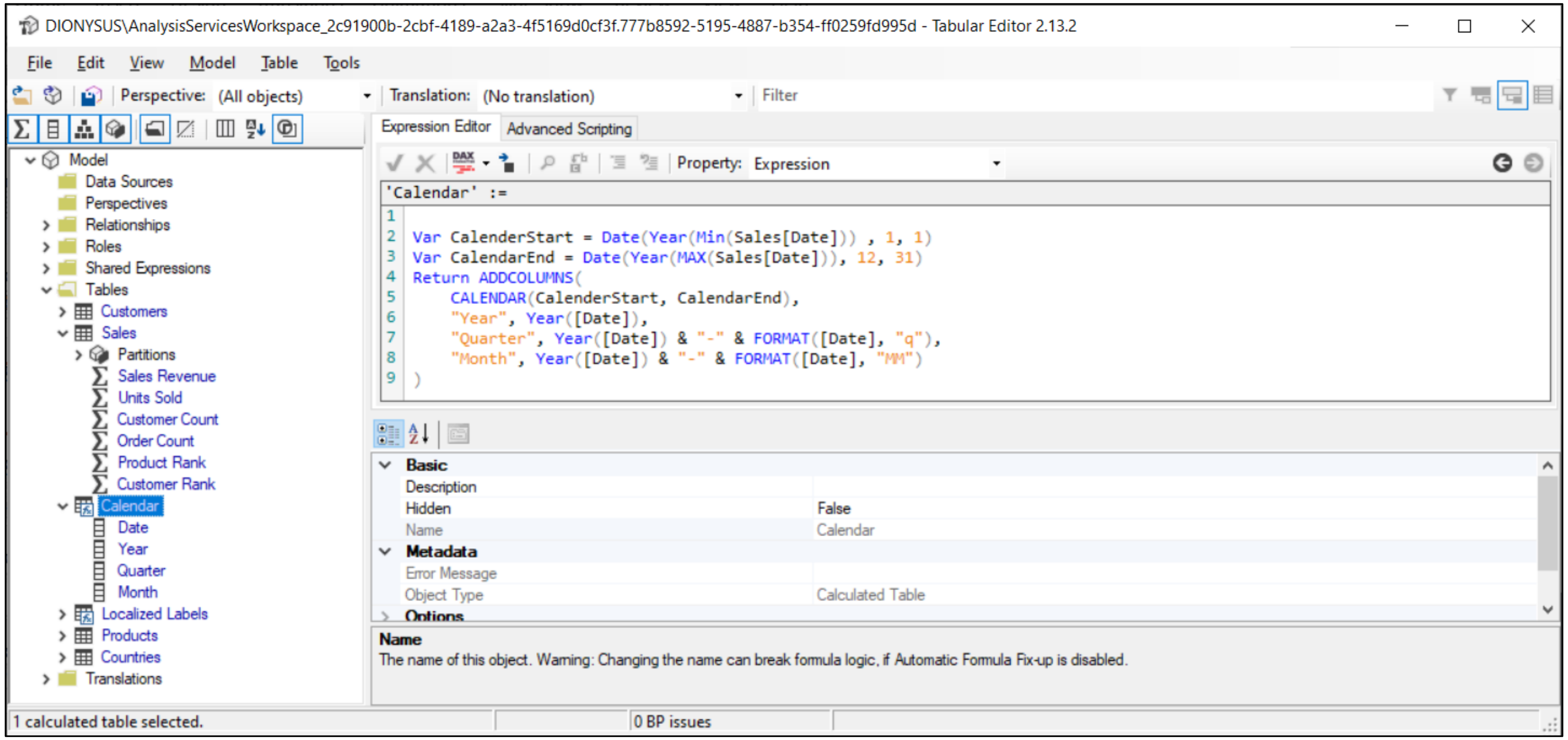
- Translations must be added for each required language
  - Contains **translatedCaption** for each translated string



```
{
  "name": "29ddb796-a33b-40d8-b61b-a2f901c0fcb7",
  "model": {
    "culture": "en-US",
    "tables": [ ···
    ],
    "cultures": [
      { "name": "en-US",  ···
      },
      { "name": "es-ES", ···
      },
      { "name": "fr-FR", ···
      },
      { "name": "de-DE", ···
      },
      { "name": "nl-NL",
        "translations": {
          "model": {
            "name": "Model",
            "tables": [
              { "name": "Products", "translatedCaption": "Producten",
                "columns": [
                  { "name": "Category", "translatedCaption": "categorie" },
                  { "name": "Product", "translatedCaption": "product" }
                ]
              }
            ]
          }
        }
      }
    ]
  }
}
```

Working with Tabular Editor

- Download from https://github.com/otykier/TabularEditor/


DIONYSUS\AnalysisServicesWorkspace_2c91900b-2cbf-4189-a2a3-4f5169d0cf3f.777b8592-5195-4887-b354-ff0259fd995d - Tabular Editor 2.13.2

```
'Calendar' :=

Var CalenderStart = Date(Year(Min(Sales[Date])) , 1, 1)
Var CalendarEnd = Date(Year(MAX(Sales[Date])), 12, 31)
Return ADDCOLUMNS(
    CALENDAR(CalenderStart, CalendarEnd),
    "Year", Year([Date]),
    "Quarter", Year([Date]) & "-" & FORMAT([Date], "q"),
    "Month", Year([Date]) & "-" & FORMAT([Date], "MM")
)
```

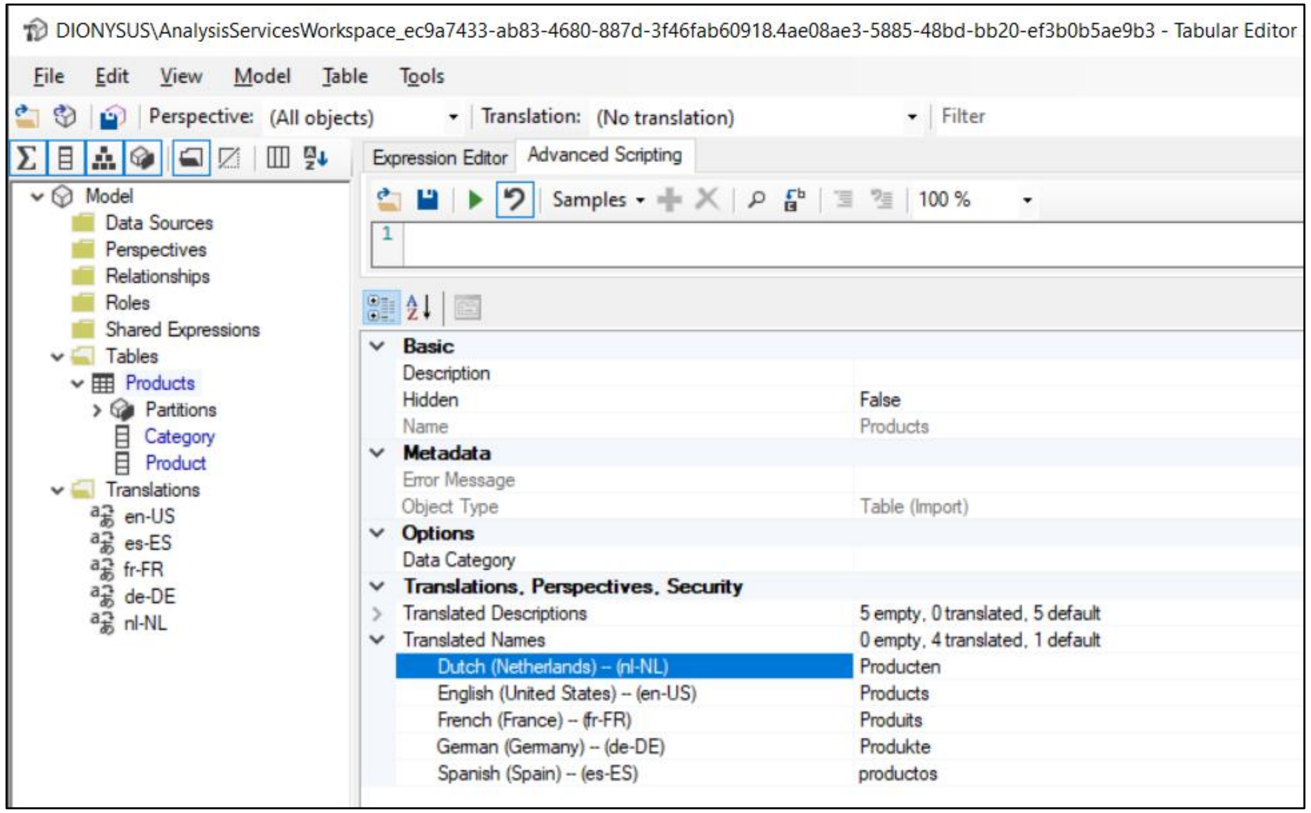
Basic	
Description	
Hidden	False
Name	Calendar
Metadata	
Error Message	
Object Type	Calculated Table
Options	

Name
The name of this object. Warning: Changing the name can break formula logic, if Automatic Formula Fix-up is disabled.

1 calculated table selected. 0 BP issues

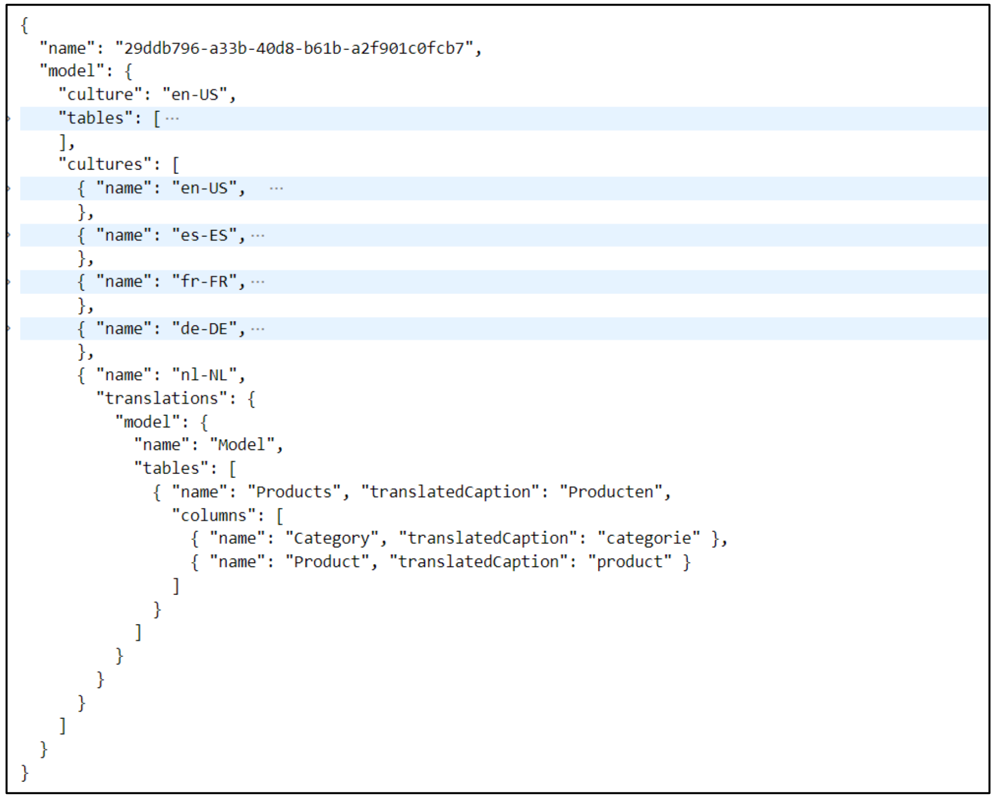
Adding Translations using Tabular Editor

- Tabular Editor can be used to add cultures and populate translations


DIONYSUS\AnalysisServicesWorkspace_ec9a7433-ab83-4680-887d-3f46fab60918.4ae08ae3-5885-48bd-bb20-ef3b0b5ae9b3 - Tabular Editor
File
Edit
View
Model
Table
Tools
Perspective: (All objects)
Translation: (No translation)
Filter
Expression Editor
Advanced Scripting
Samples
100 %

- Model
  - Data Sources
  - Perspectives
  - Relationships
  - Roles
  - Shared Expressions
  - Tables
    - Products
      - Partitions
      - Category
      - Product
  - Translations
    - en-US
    - es-ES
    - fr-FR
    - de-DE
    - nl-NL

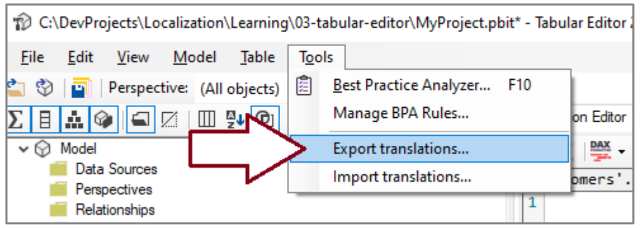
Basic	
Description	
Hidden	False
Name	Products
Metadata	
Error Message	
Object Type	Table (Import)
Options	
Data Category	
Translations, Perspectives, Security	
Translated Descriptions	5 empty, 0 translated, 5 default
Translated Names	0 empty, 4 translated, 1 default
Dutch (Netherlands) – (nl-NL)	Producten
English (United States) – (en-US)	Products
French (France) – (fr-FR)	Produits
German (Germany) – (de-DE)	Produkte
Spanish (Spain) – (es-ES)	productos

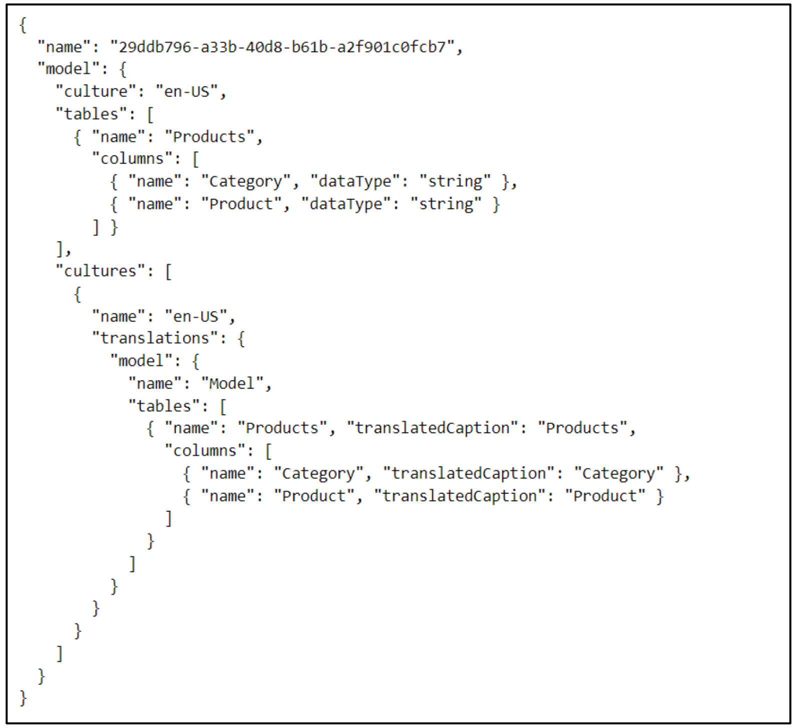


```
{
  "name": "29ddb796-a33b-40d8-b61b-a2f901c0fcb7",
  "model": {
    "culture": "en-US",
    "tables": [ ···
    ],
    "cultures": [
      { "name": "en-US",  ···
      },
      { "name": "es-ES", ···
      },
      { "name": "fr-FR", ···
      },
      { "name": "de-DE", ···
      },
      { "name": "nl-NL",
        "translations": {
          "model": {
            "name": "Model",
            "tables": [
              { "name": "Products", "translatedCaption": "Producten",
                "columns": [
                  { "name": "Category", "translatedCaption": "categorie" },
                  { "name": "Product", "translatedCaption": "product" }
                ]
              }
            ]
          }
        }
      }
    ]
  }
}
```

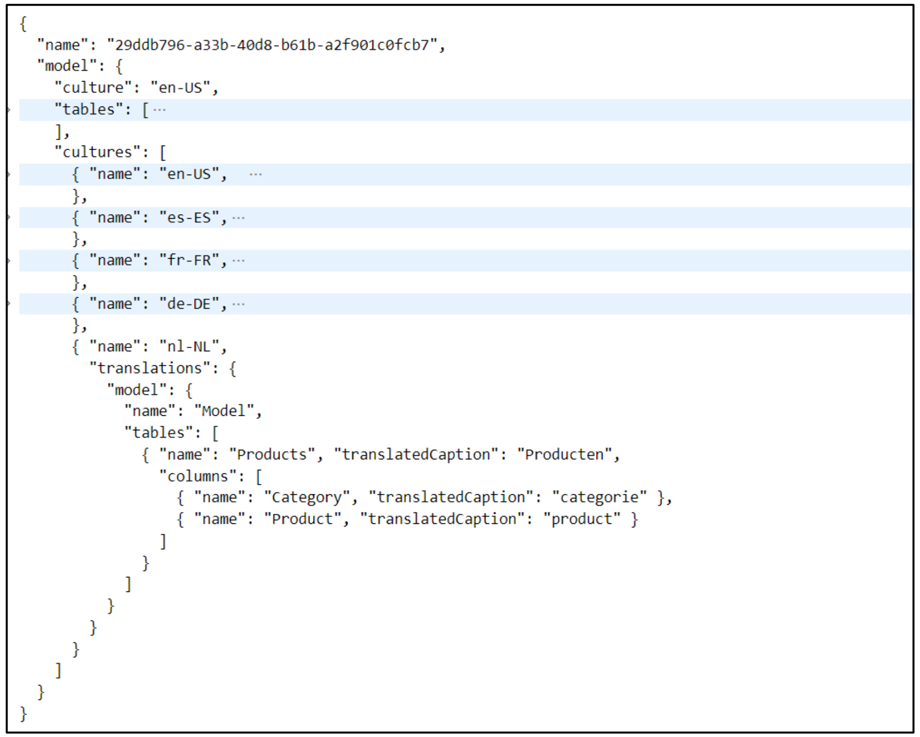
Exporting/Importing Metadata Translations

- Tabular Editor supports exporting/importing translations
  - Export metadata after populating translations for default language
  - Design workflow with human translators to add translations
  - Import updated translation back into PBIX project using Tabular Editor





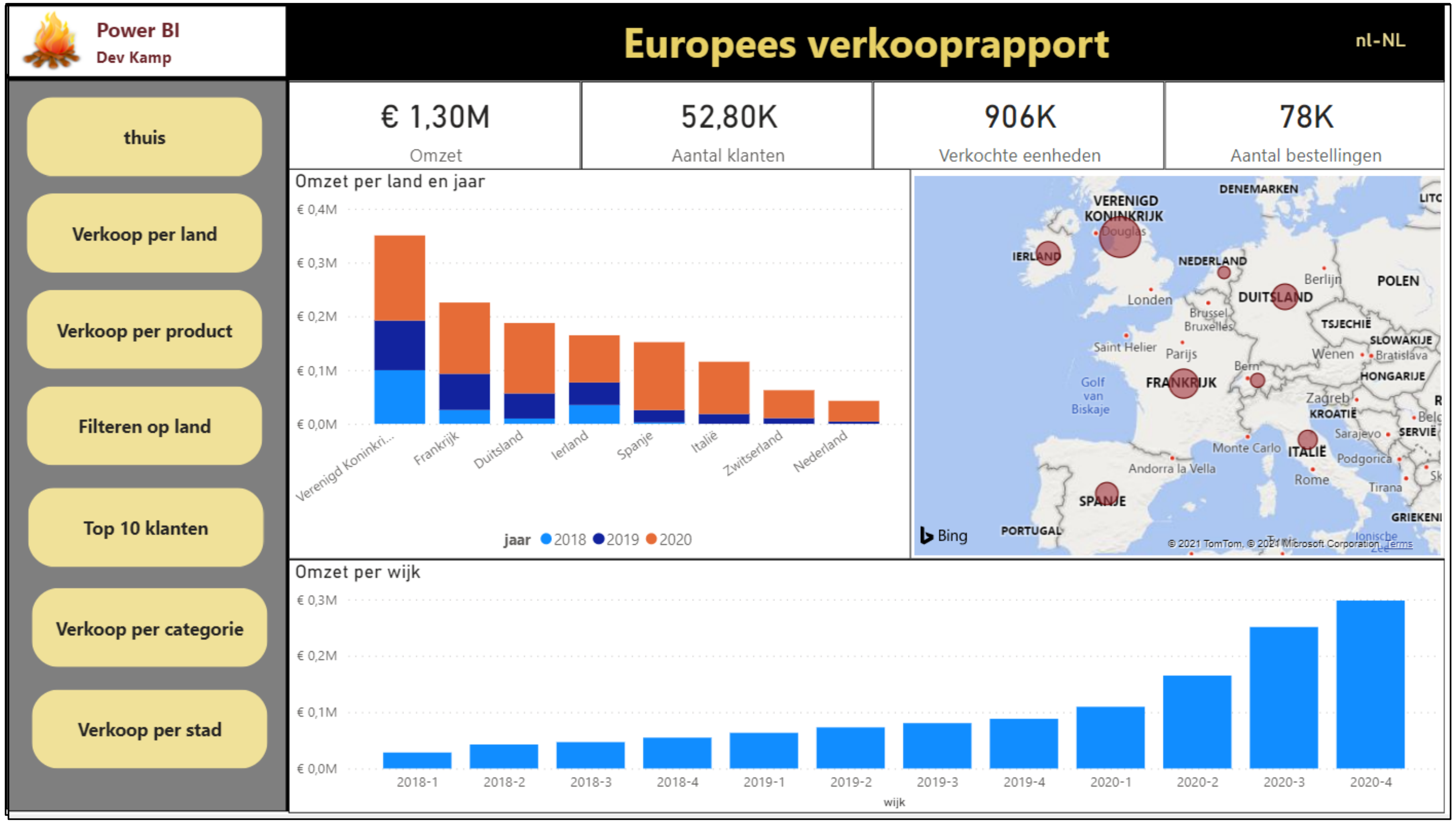
```
{
  "name": "29ddb796-a33b-40d8-b61b-a2f901c0fcb7",
  "model": {
    "culture": "en-US",
    "tables": [
      { "name": "Products",
        "columns": [
          { "name": "Category", "dataType": "string" },
          { "name": "Product", "dataType": "string" }
        ] }
    ],
    "cultures": [
      {
        "name": "en-US",
        "translations": {
          "model": {
            "name": "Model",
            "tables": [
              { "name": "Products", "translatedCaption": "Products",
                "columns": [
                  { "name": "Category", "translatedCaption": "Category" },
                  { "name": "Product", "translatedCaption": "Product" }
                ]
              }
            ]
          }
        }
      }
    ]
  }
}
```



```
{
  "name": "29ddb796-a33b-40d8-b61b-a2f901c0fcb7",
  "model": {
    "culture": "en-US",
    "tables": [ ···
    ],
    "cultures": [
      { "name": "en-US",  ···
      },
      { "name": "es-ES", ···
      },
      { "name": "fr-FR", ···
      },
      { "name": "de-DE", ···
      },
      { "name": "nl-NL",
        "translations": {
          "model": {
            "name": "Model",
            "tables": [
              { "name": "Products", "translatedCaption": "Producten",
                "columns": [
                  { "name": "Category", "translatedCaption": "categorie" },
                  { "name": "Product", "translatedCaption": "product" }
                ]
              }
            ]
          }
        }
      }
    ]
  }
}
```

European Product Sales Demo

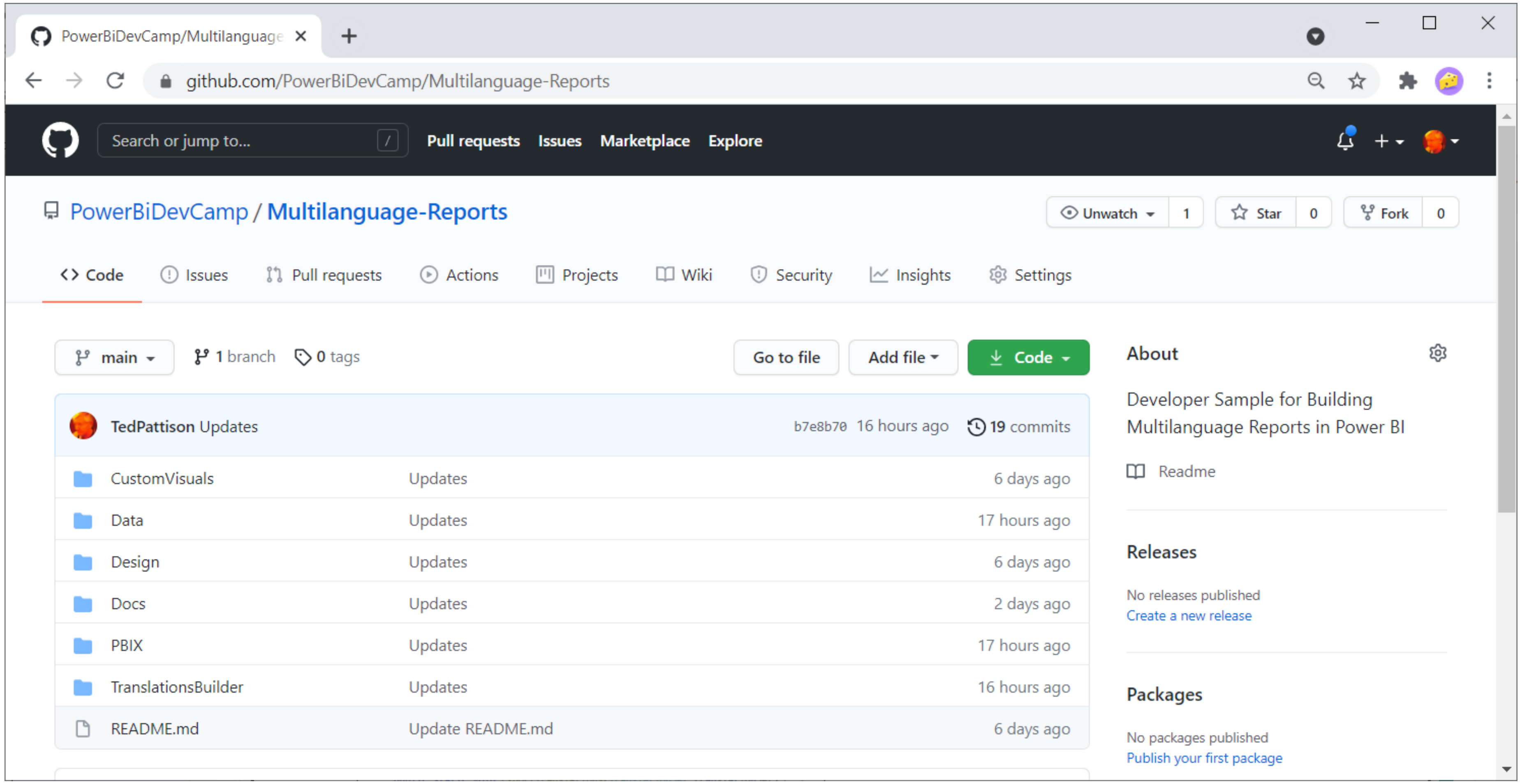
- Live demo at https://multilanguagereportdemo.azurewebsites.net/
  - English
  - Spanish
  - French
  - German
  - Dutch



GitHub Repository for Multilanguage-Reports Sample

All sample code for this developer samples available for download

https://github.com/PowerBiDevCamp/Multilanguage-Reports



PowerBiDevCamp / Multilanguage-Reports

1 branch 0 tags

TedPattison Updates b7e8b70 16 hours ago 19 commits

CustomVisuals	Updates	6 days ago
Data	Updates	17 hours ago
Design	Updates	6 days ago
Docs	Updates	2 days ago
PBIX	Updates	17 hours ago
TranslationsBuilder	Updates	16 hours ago
README.md	Update README.md	6 days ago

About

Developer Sample for Building Multilanguage Reports in Power BI

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Agenda

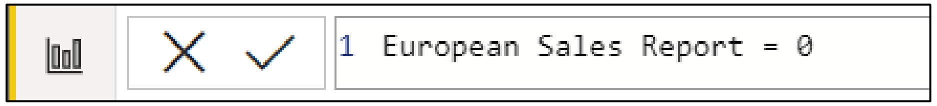
- ✓ Overview of Multi-language PBIX Development
- ➢ Designing Multi-language Reports
- Adding Metadata Translations with TOM
- Embedding Reports with Specific Locales
- Designing Data Models to Support Content Translations
- Setting the Language for Current User using RLS and UserCulture

Preparing a PBIX Project for Translations

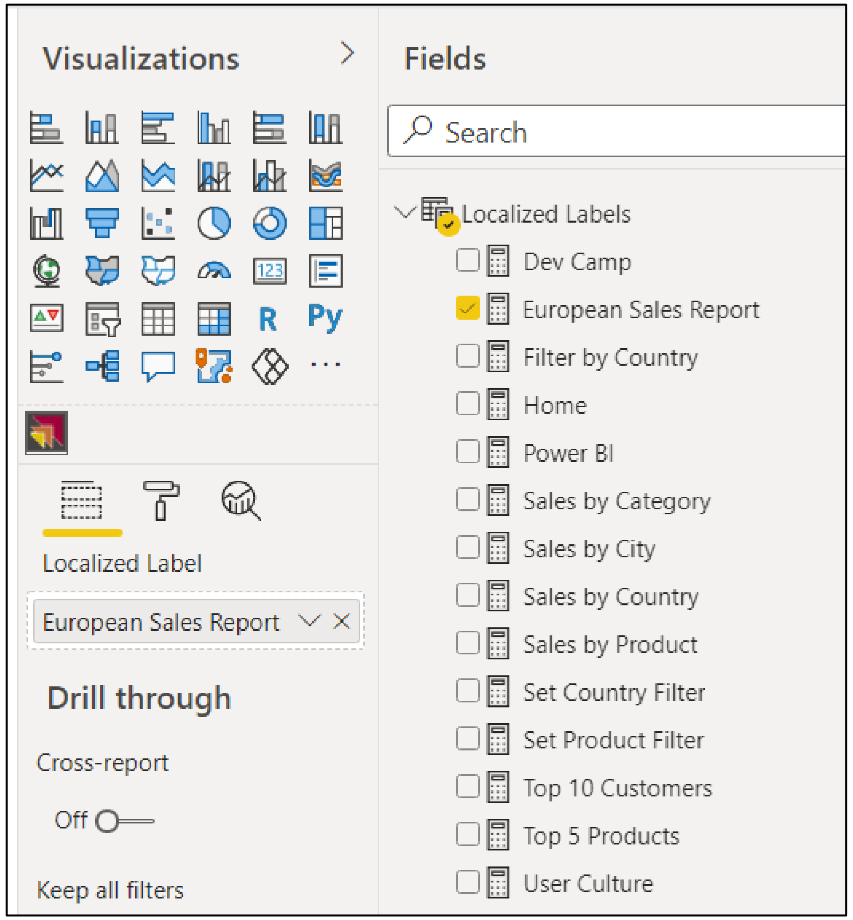
- Plan localization from the start
  - Much harder to work with pre-existing PBIX created without localization in mind
- Plan for content growth
  - Some languages have content wider than English
  - Include padding for translated content
- Avoid report design techniques that do not support localization
  - Don’t add literal text in visuals such a textboxes or button
  - Don’t display page tabs - their content cannot be localized

Localized Labels Table

- Create a new table in data model named **Localized Labels**
- Add a measure for any string content that needs to be localized
- Set measure expressions to 0



```
1  European Sales Report = 0
```

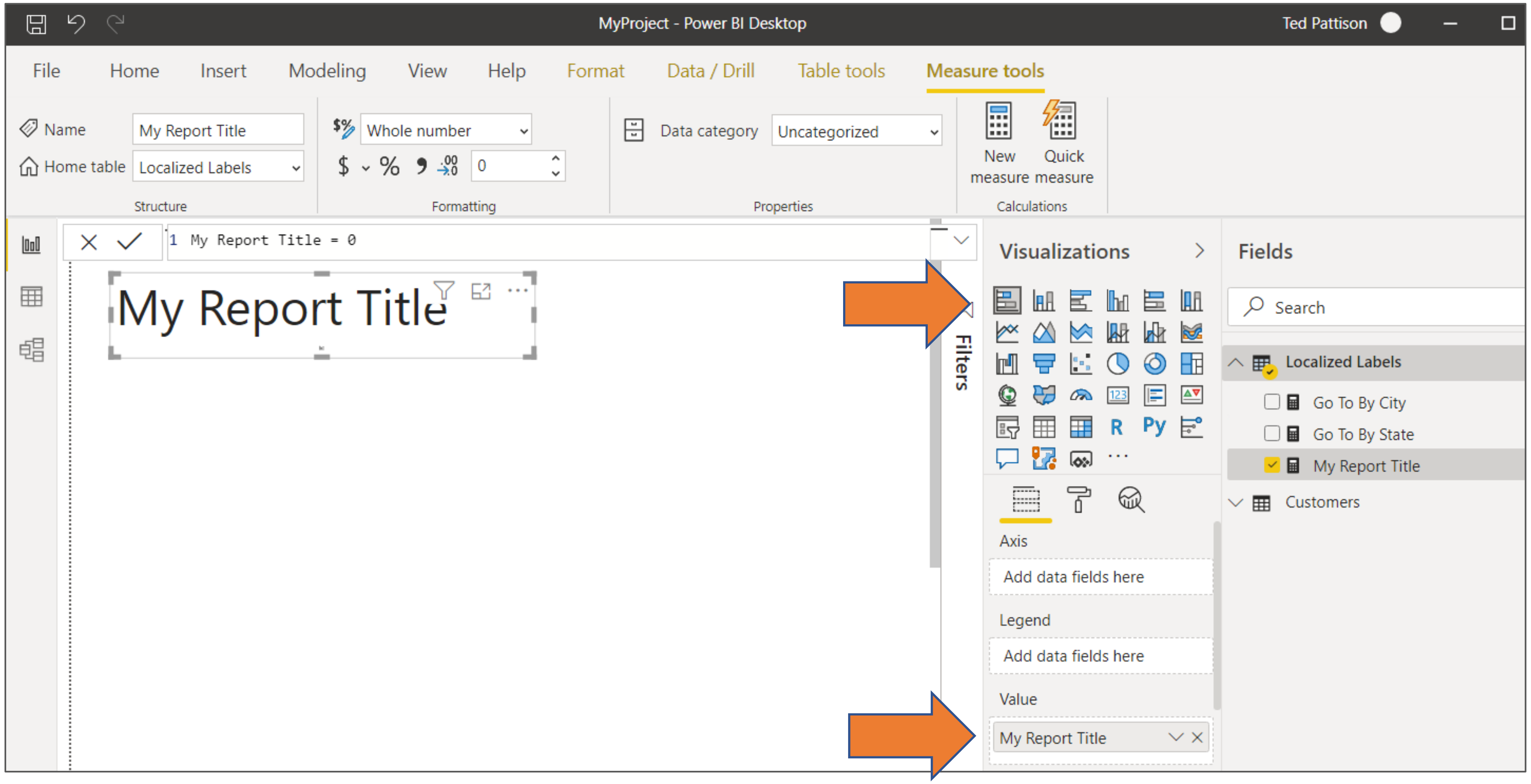


Localized Labels
- Dev Camp
- [✓ European Sales Report]
- Filter by Country
- Home
- Power BI
- Sales by Category
- Sales by City
- Sales by Country
- Sales by Product
- Set Country Filter
- Set Product Filter
- Top 10 Customers
- Top 5 Products
- User Culture

Localized Label: European Sales Report

Surfacing a Localized Label

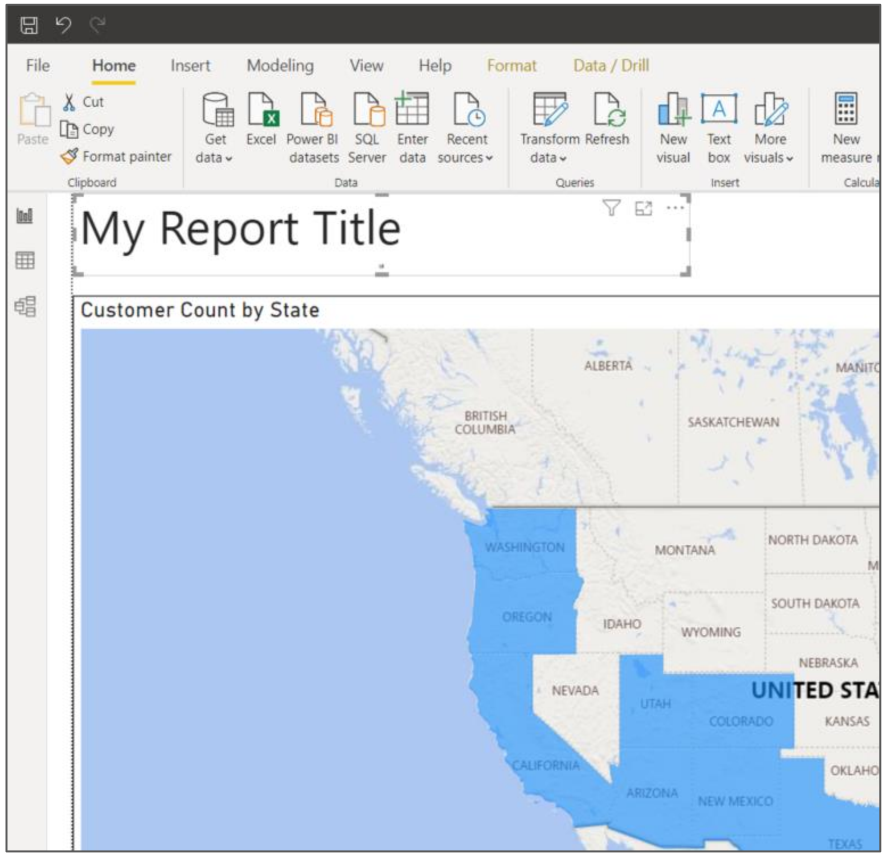
- You can surface localized label using one of core Power BI visuals
  - Add Stacked Barchart visual and add localized label in **Values** data roles

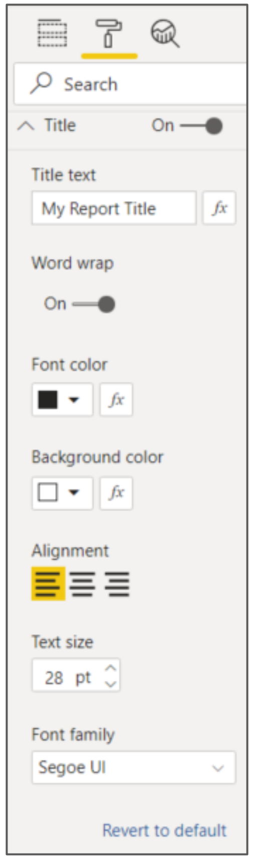


```
1 My Report Title = 0
```

Configuring Localized Label Display Properties

- Using Format pane to configure label display properties
  - Design experience for this technique is limited





Search

Title On

Title text: My Report Title

Word wrap: On

Font color

Background color

Alignment

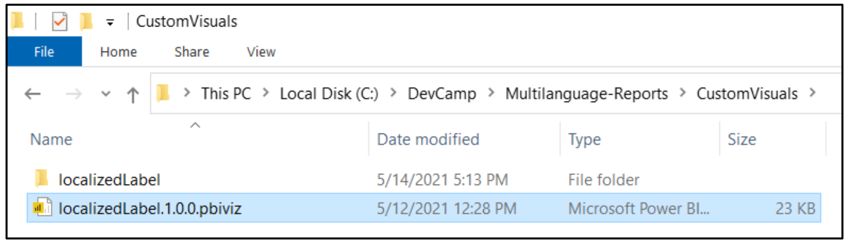
Text size: 28 pt

Font family: Segoe UI

Revert to default

Developing a Custom Visual: LocalizedLabel

- Using a custom visual to surface localized labels
  - Sample provides custom visual project

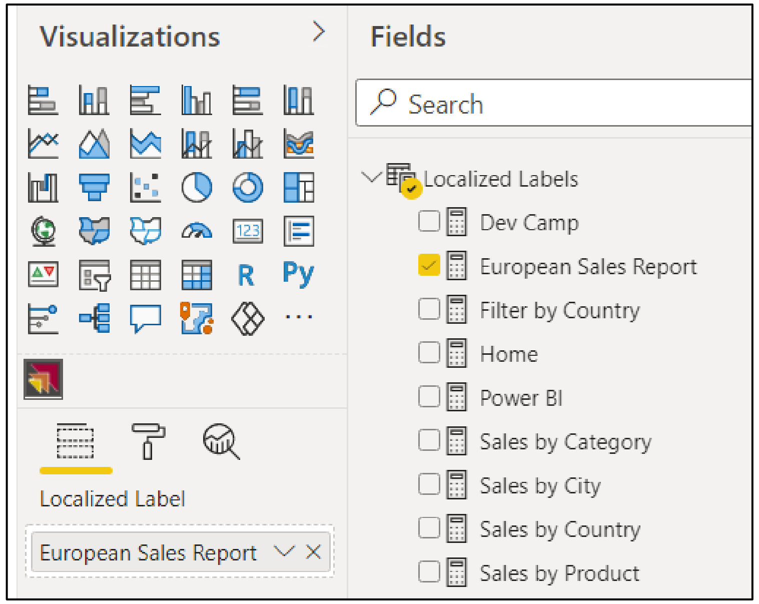


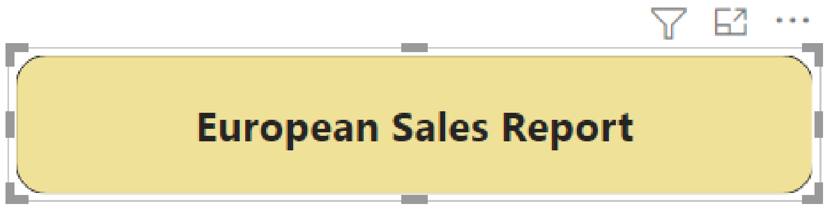
CustomVisuals

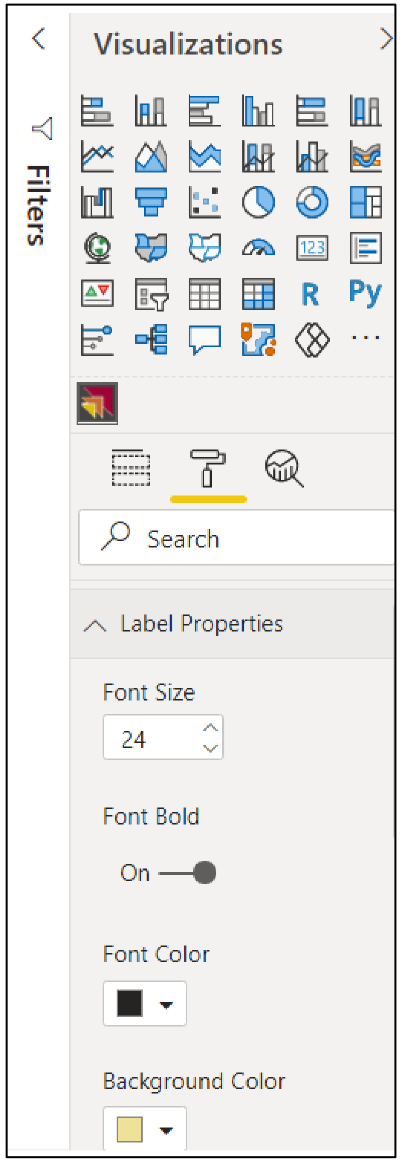
This PC > Local Disk (C:) > DevCamp > Multilanguage-Reports > CustomVisuals

Name	Date modified	Type	Size
localizedLabel	5/14/2021 5:13 PM	File folder	
localizedLabel.1.0.0.pbiviz	5/12/2021 12:28 PM	Microsoft Power BI...	23 KB

  - Provides more flexibility in configuring display properties





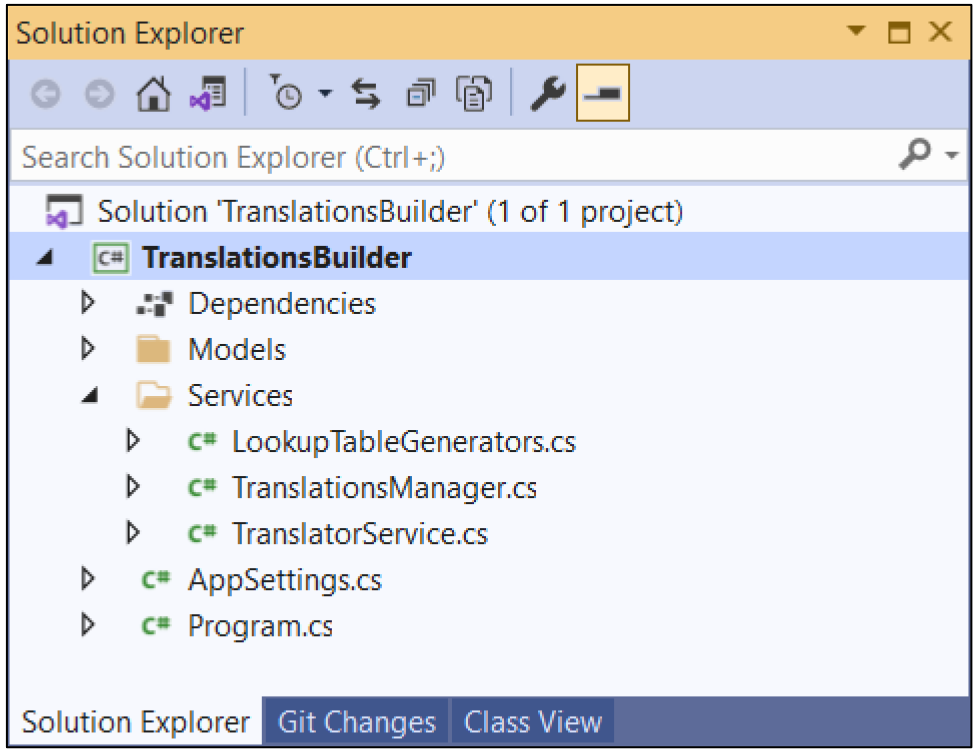


Agenda

- ✓ Overview of Multi-language PBIX Development
- ✓ Designing Multi-language Reports
- ➢ Adding Metadata Translations with TOM
- Embedding Reports with Specific Locales
- Designing Data Models to Support Content Translations
- Setting the Language for Current User using RLS and UserCulture

Translations Builder sample application

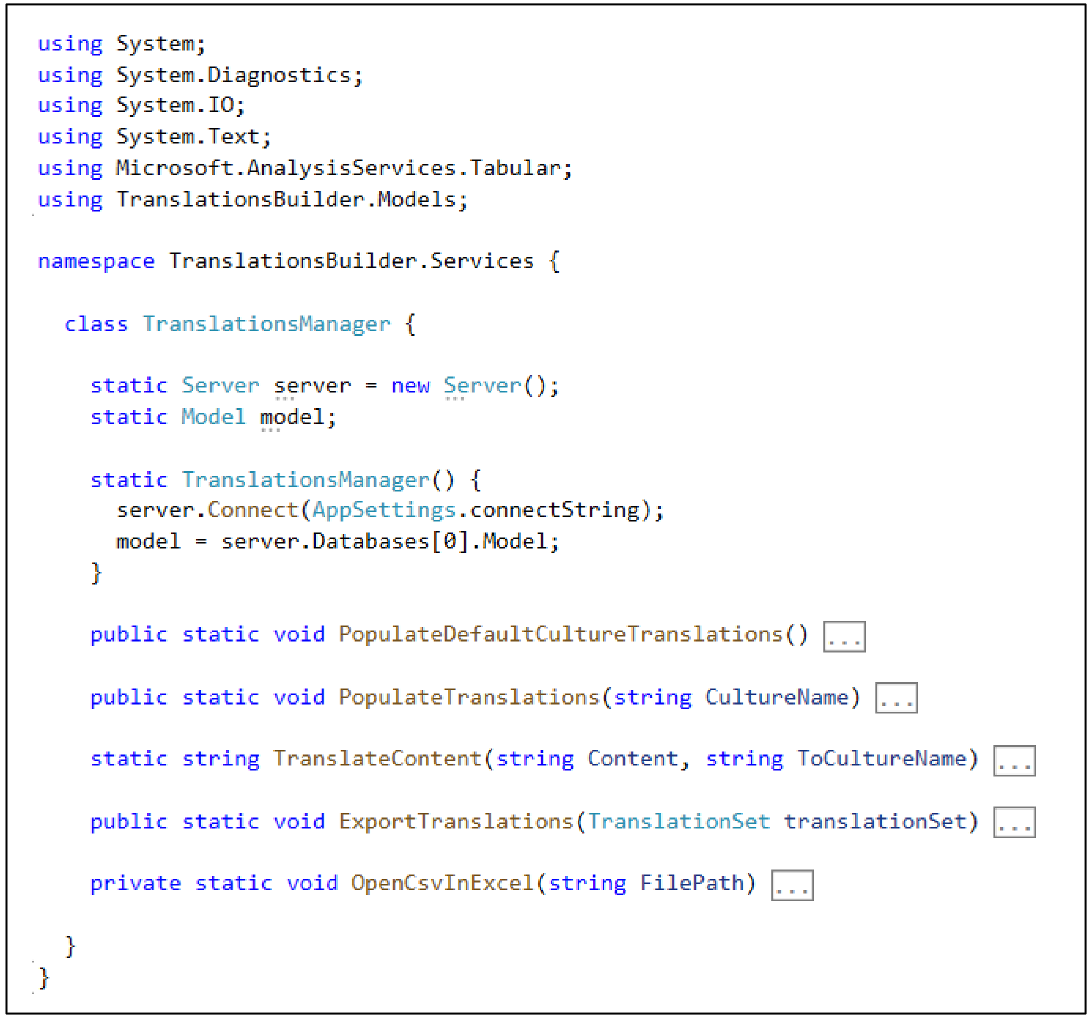
- C# console application which programs TOM to add translations
  - Provides code to populate translations of default culture
  - Provides code to add secondary cultures and populate their translations
  - Calls to Microsoft Translator Service to create starting set of translations for each language



```
Solution 'TranslationsBuilder' (1 of 1 project)
  TranslationsBuilder
    Dependencies
    Models
    Services
      LookupTableGenerators.cs
      TranslationsManager.cs
      TranslatorService.cs
    AppSettings.cs
    Program.cs
```

TranslationsManager class

- TranslationsManager class contains logic to add cultures and add translations



```csharp
using System;
using System.Diagnostics;
using System.IO;
using System.Text;
using Microsoft.AnalysisServices.Tabular;
using TranslationsBuilder.Models;

namespace TranslationsBuilder.Services {

  class TranslationsManager {

    static Server server = new Server();
    static Model model;

    static TranslationsManager() {
      server.Connect(AppSettings.connectString);
      model = server.Databases[0].Model;
    }

    public static void PopulateDefaultCultureTranslations() ...

    public static void PopulateTranslations(string CultureName) ...

    static string TranslateContent(string Content, string ToCultureName) ...

    public static void ExportTranslations(TranslationSet translationSet) ...

    private static void OpenCsvInExcel(string FilePath) ...

  }
}
```

TranslationsManager.PopulateDefaultCultureTranslations

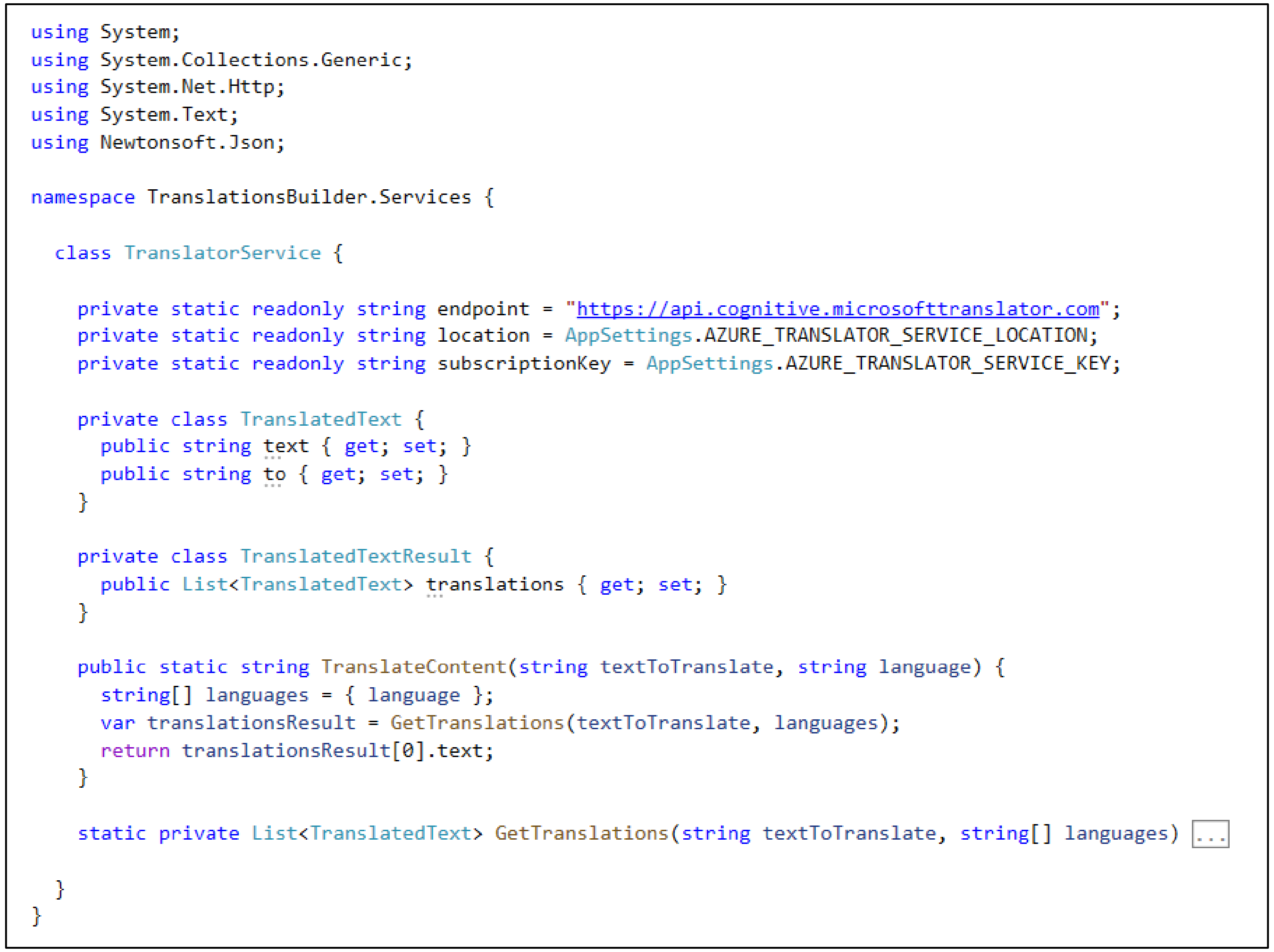
- This code adds translation to the default culture

```
public static void PopulateDefaultCultureTranslations() {

  Culture defaultCulture = model.Cultures[model.Culture];
  Console.Write("Settings data model translations for default culture of " + defaultCulture.Name);

  //*** enumerate through tables
  foreach (Table table in model.Tables) {
    Console.Write(".");
    defaultCulture.ObjectTranslations.SetTranslation(table, TranslatedProperty.Caption, table.Name);
    //*** enumerate through columns
    foreach (Column column in table.Columns) {
      if (column.Type != ColumnType.RowNumber) {
        Console.Write(".");
        defaultCulture.ObjectTranslations.SetTranslation(column, TranslatedProperty.Caption, column.Name);
      }
    };
    //*** enumerate through measures
    foreach (Measure measure in table.Measures) {
      Console.Write(".");
      defaultCulture.ObjectTranslations.SetTranslation(measure, TranslatedProperty.Caption, measure.Name);
    };
    //*** enumerate through hierarchies
    foreach (Hierarchy hierarchy in table.Hierarchies) {
      Console.Write(".");
      defaultCulture.ObjectTranslations.SetTranslation(hierarchy, TranslatedProperty.Caption, hierarchy.Name);
    };
  }
  // save changes back to data model
  model.SaveChanges();
  Console.WriteLine();
}
```

```
{
  "name": "29ddb796-a33b-40d8-b61b-a2f901c0fcb7",
  "model": {
    "culture": "en-US",
    "tables": [
      { "name": "Products",
        "columns": [
          { "name": "Category", "dataType": "string" },
          { "name": "Product", "dataType": "string" }
        ] }
    ],
    "cultures": [
      {
        "name": "en-US",
        "translations": {
          "model": {
            "name": "Model",
            "tables": [
              { "name": "Products", "translatedCaption": "Products",
                "columns": [
                  { "name": "Category", "translatedCaption": "Category" },
                  { "name": "Product", "translatedCaption": "Product" }
                ]
              }
            ]
          }
        }
      }
    ]
  }
}
```

TranslatorService class

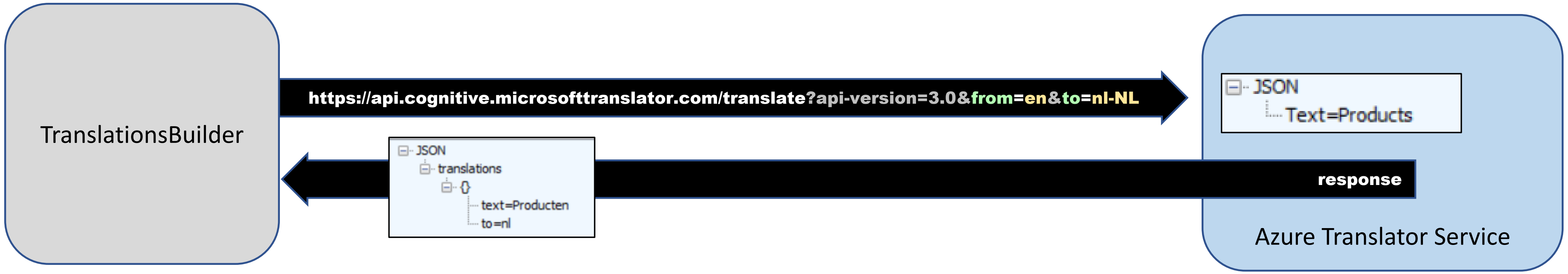
- Abstracts away call to Microsoft Translator service to translate content



```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Text;
using Newtonsoft.Json;

namespace TranslationsBuilder.Services {

  class TranslatorService {

    private static readonly string endpoint = "https://api.cognitive.microsofttranslator.com";
    private static readonly string location = AppSettings.AZURE_TRANSLATOR_SERVICE_LOCATION;
    private static readonly string subscriptionKey = AppSettings.AZURE_TRANSLATOR_SERVICE_KEY;

    private class TranslatedText {
      public string text { get; set; }
      public string to { get; set; }
    }

    private class TranslatedTextResult {
      public List<TranslatedText> translations { get; set; }
    }

    public static string TranslateContent(string textToTranslate, string language) {
      string[] languages = { language };
      var translationsResult = GetTranslations(textToTranslate, languages);
      return translationsResult[0].text;
    }

    static private List<TranslatedText> GetTranslations(string textToTranslate, string[] languages) ...

  }
}
```

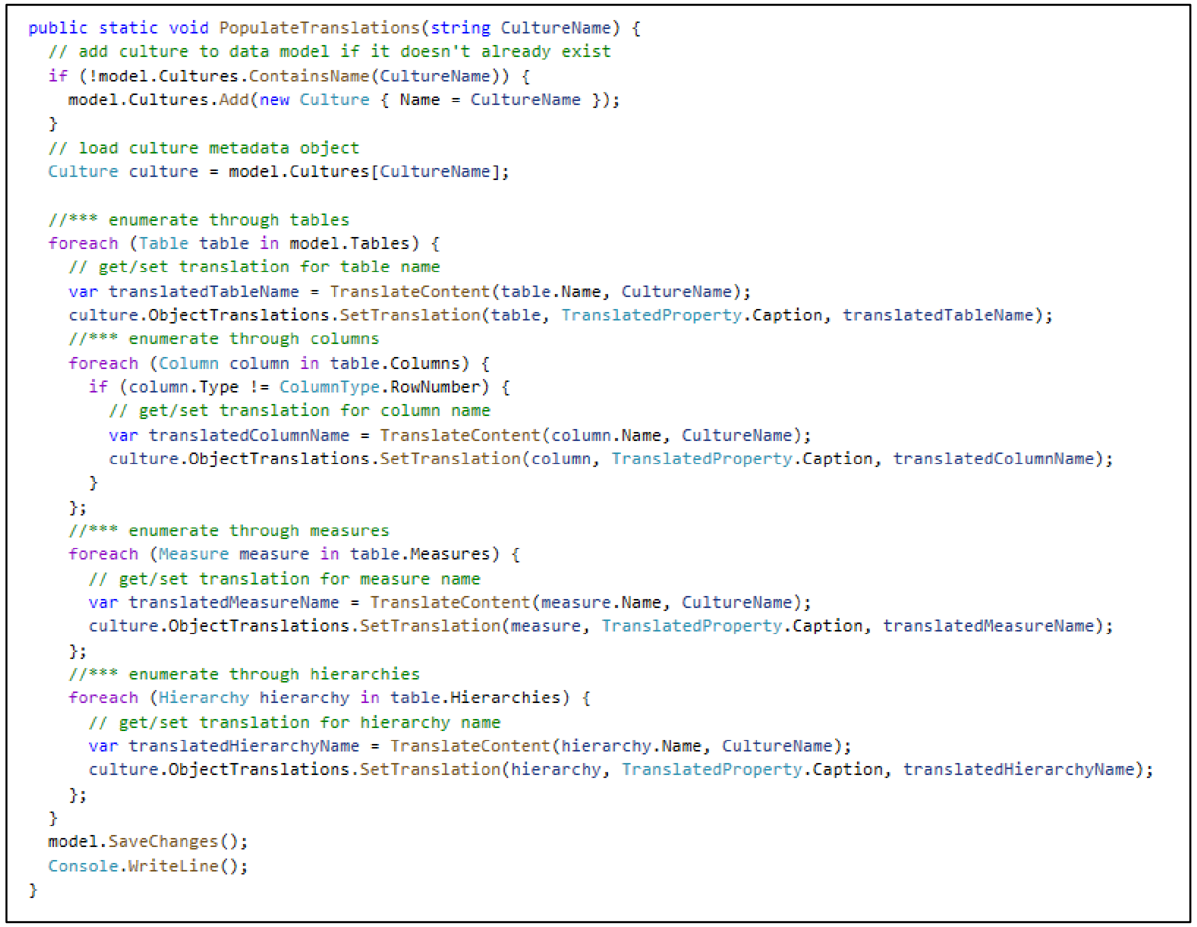
TranslatorService.GetTranslations

```
static private List<TranslatedText> GetTranslations(string textToTranslate, string[] languages) {

  string targetLanguages = "";
  foreach (string language in languages) {
    targetLanguages += "&to=" + language;
  }

  string route = "/translate?api-version=3.0&from=en" + targetLanguages;
  object[] body = new object[] { new { Text = textToTranslate } };
  var requestBody = JsonConvert.SerializeObject(body);

  using (var client = new HttpClient())
  using (var request = new HttpRequestMessage()) {

    // prepare HTTP request
    request.Method = HttpMethod.Post;
    request.RequestUri = new Uri(endpoint + route);
    request.Content = new StringContent(requestBody, Encoding.UTF8, "application/json");
    request.Headers.Add("Ocp-Apim-Subscription-Key", subscriptionKey);
    request.Headers.Add("Ocp-Apim-Subscription-Region", location);

    // transmit HTTP request
    HttpResponseMessage response = client.Send(request);

    // extract translated content from HTTP response body
    string result = response.Content.ReadAsStringAsync().Result;
    List<TranslatedTextResult> convertedResult = JsonConvert.DeserializeObject<List<TranslatedTextResult>>(result);
    return convertedResult[0].translations;
  }

}
```

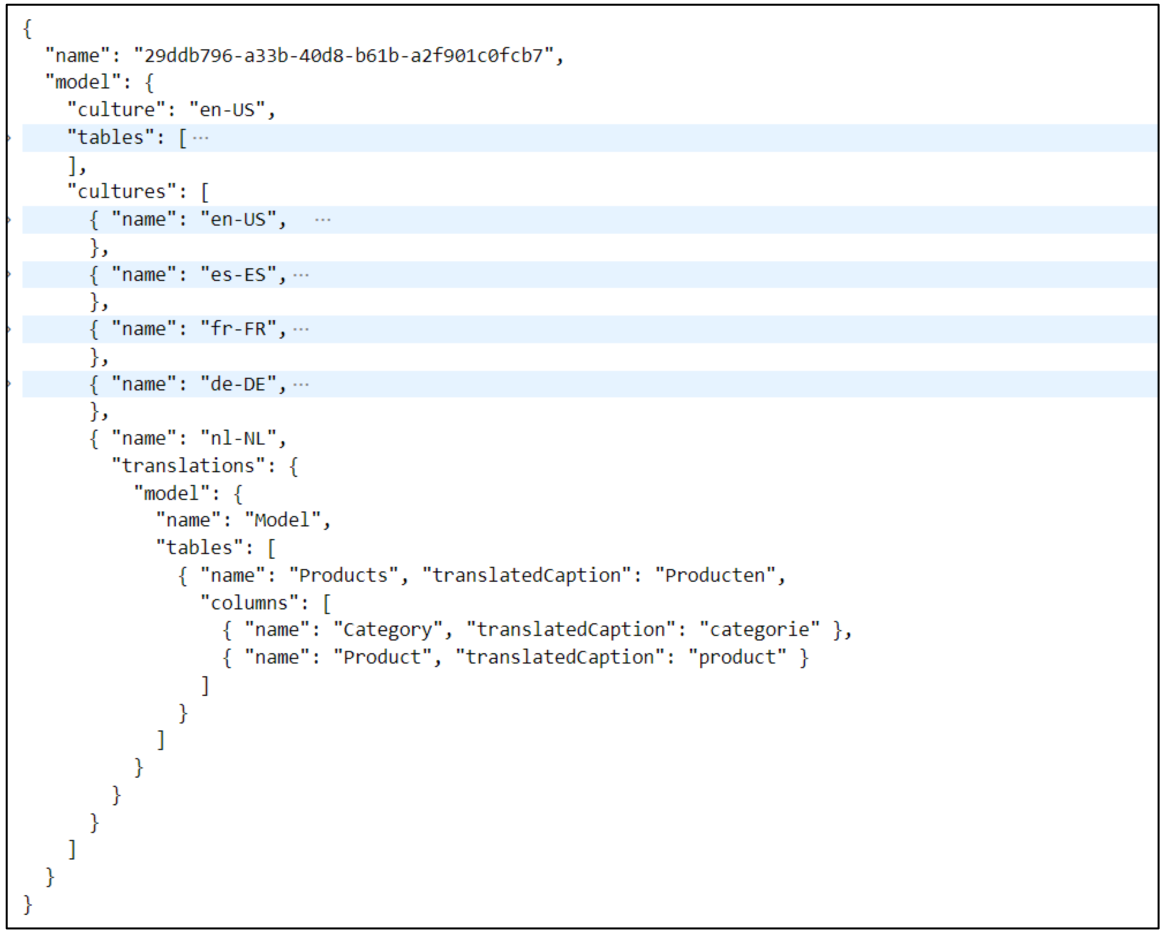


TranslationsManager.PopulateTranslations

- Add translations for each of the secondary languages



```csharp
public static void PopulateTranslations(string CultureName) {
  // add culture to data model if it doesn't already exist
  if (!model.Cultures.ContainsName(CultureName)) {
    model.Cultures.Add(new Culture { Name = CultureName });
  }
  // load culture metadata object
  Culture culture = model.Cultures[CultureName];

  //*** enumerate through tables
  foreach (Table table in model.Tables) {
    // get/set translation for table name
    var translatedTableName = TranslateContent(table.Name, CultureName);
    culture.ObjectTranslations.SetTranslation(table, TranslatedProperty.Caption, translatedTableName);
    //*** enumerate through columns
    foreach (Column column in table.Columns) {
      if (column.Type != ColumnType.RowNumber) {
        // get/set translation for column name
        var translatedColumnName = TranslateContent(column.Name, CultureName);
        culture.ObjectTranslations.SetTranslation(column, TranslatedProperty.Caption, translatedColumnName);
      }
    };
    //*** enumerate through measures
    foreach (Measure measure in table.Measures) {
      // get/set translation for measure name
      var translatedMeasureName = TranslateContent(measure.Name, CultureName);
      culture.ObjectTranslations.SetTranslation(measure, TranslatedProperty.Caption, translatedMeasureName);
    };
    //*** enumerate through hierarchies
    foreach (Hierarchy hierarchy in table.Hierarchies) {
      // get/set translation for hierarchy name
      var translatedHierarchyName = TranslateContent(hierarchy.Name, CultureName);
      culture.ObjectTranslations.SetTranslation(hierarchy, TranslatedProperty.Caption, translatedHierarchyName);
    };
  }
  model.SaveChanges();
  Console.WriteLine();
}
```



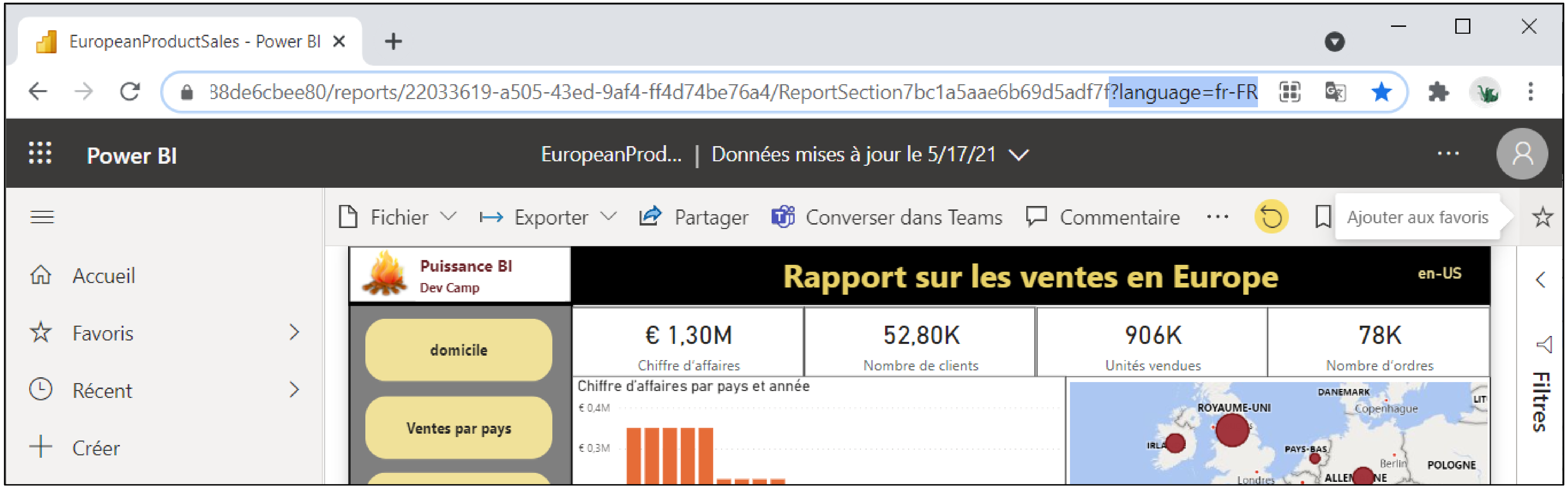
```json
{
  "name": "29ddb796-a33b-40d8-b61b-a2f901c0fcb7",
  "model": {
    "culture": "en-US",
    "tables": [⋯
    ],
    "cultures": [
      { "name": "en-US",   ⋯
      },
      { "name": "es-ES", ⋯
      },
      { "name": "fr-FR", ⋯
      },
      { "name": "de-DE", ⋯
      },
      { "name": "nl-NL",
        "translations": {
          "model": {
            "name": "Model",
            "tables": [
              { "name": "Products", "translatedCaption": "Producten",
                "columns": [
                  { "name": "Category", "translatedCaption": "categorie" },
                  { "name": "Product", "translatedCaption": "product" }
                ]
              }
            ]
          }
        }
      }
    ]
  }
}
```

Agenda

- ✓ Overview of Multi-language PBIX Development
- ✓ Designing Multi-language Reports
- ✓ Adding Metadata Translations with TOM
- ➢ Embedding Reports with Specific Locales
- • Designing Data Models to Support Content Translations
- • Setting the Language for Current User using RLS and UserCulture

Loading Reports with Locale using a Query String Parameter

- You can load report using specific locale
  - Add language query string parameter to report URL
  - Requires workspace in dedicated capacity
  - This technique **will not** change result for UserCulture function in DAX



EuropeanProductSales - Power BI

38de6cbee80/reports/22033619-a505-43ed-9af4-ff4d74be76a4/ReportSection7bc1a5aae6b69d5adf7f?language=fr-FR

Power BI   EuropeanProd... | Données mises à jour le 5/17/21

Fichier   Exporter   Partager   Converser dans Teams   Commentaire   Ajouter aux favoris

Accueil
Favoris
Récent
Créer

Puissance BI Dev Camp

Rapport sur les ventes en Europe   en-US

domicile
Ventes par pays

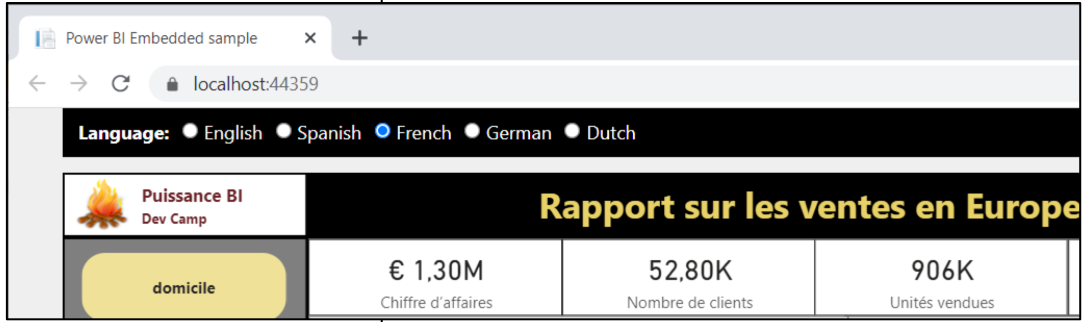
€ 1,30M	52,80K	906K	78K
Chiffre d'affaires	Nombre de clients	Unités vendues	Nombre d'ordres

Chiffre d'affaires par pays et année

Using localeSettings in Embed Config

- You can load report using specific locale by adding **localeSettings** in embed config
  - This technique **will** change result for **UserCulture** function in DAX

```
config = {
  type: "report",
  id: reportId,
  embedUrl: embedUrl,
  accessToken: embedToken,
  tokenType: models.TokenType.Embed,
  settings: {
    panes: {
      pageNavigation: { visible: false },
      filters: { visible: false }
    }
  },
  localeSettings: { language: "fr-FR", formatLocale: "fr-FR" }
};

// embed report with specific locale
let report = powerbi.embed(reportContainer, config);
```


Power BI Embedded sample

localhost:44359

Language: English Spanish [✓ French] German Dutch

Puissance BI
Dev Camp

Rapport sur les ventes en Europe

domicile

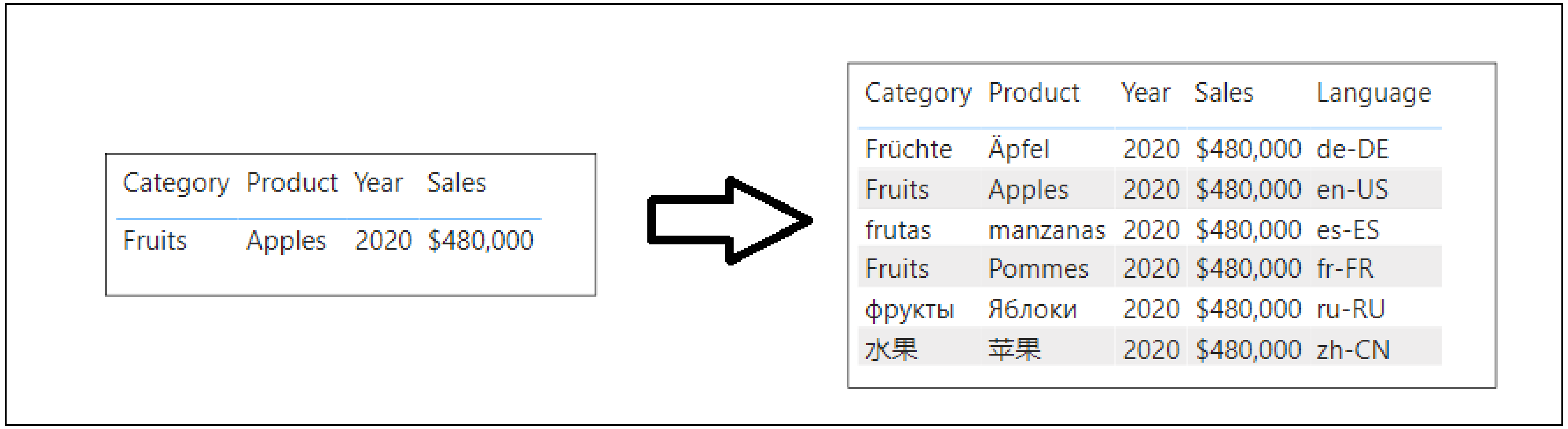
€ 1,30M	52,80K	906K
Chiffre d'affaires	Nombre de clients	Unités vendues

Agenda

- ✓ Overview of Multi-language PBIX Development
- ✓ Designing Multi-language Reports
- ✓ Adding Metadata Translations with TOM
- ✓ Embedding Reports with Specific Locales
- ➢ Designing Data Models to Support Content Translations
- • Setting the Language for Current User using RLS and UserCulture

Content Localization Through Row Replication

- Power BI does not currently support dynamic column evaluation
  - When this support is added, it will drastically change the strategy
  - Until then, row replication is best way to localize content in addition to metadata
- Row replication strategy involves adding separate rows for each language
  - Strategy implemented on dimension tables with text values requiring localization
  - Row filtering used to exclusively filter rows to one specific language

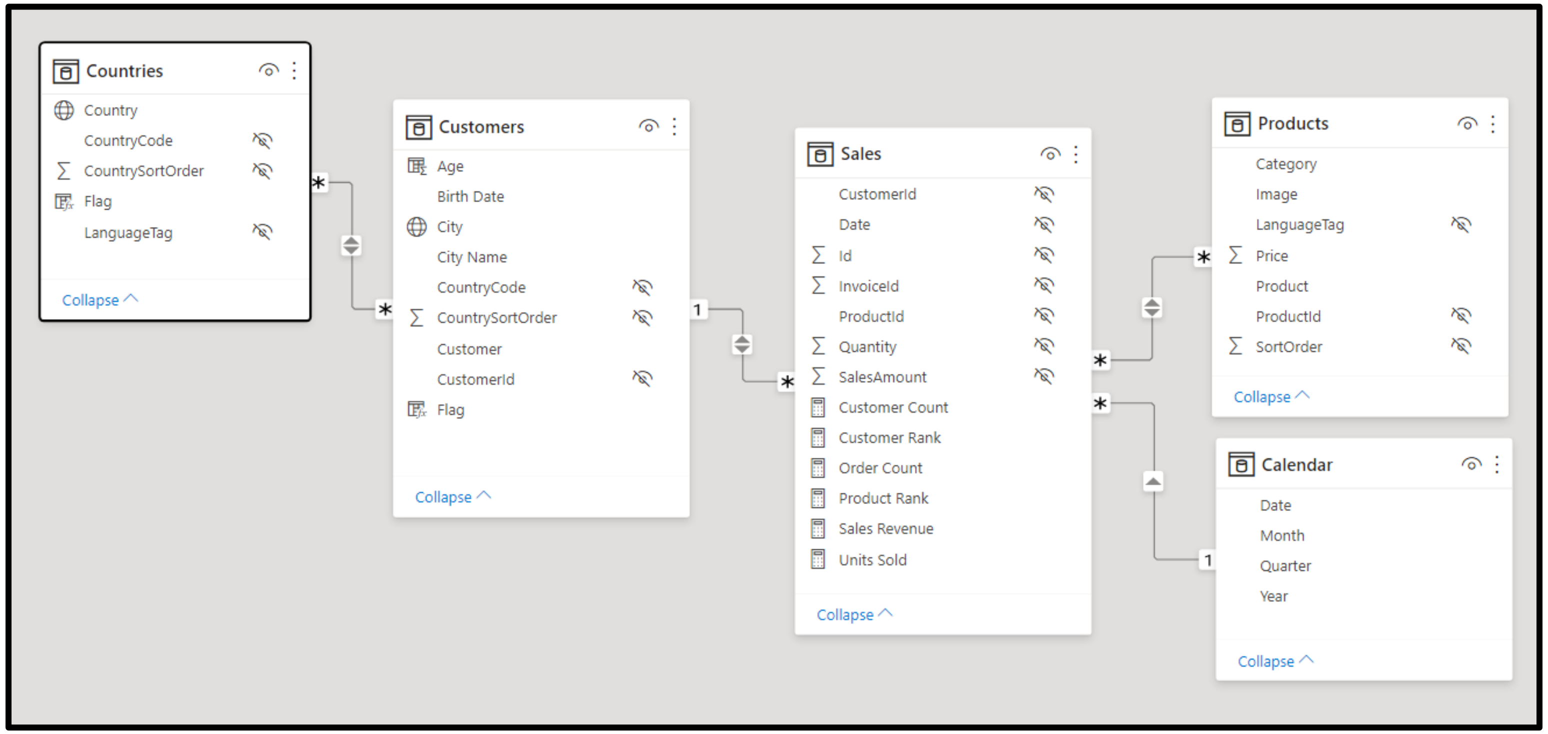


Category	Product	Year	Sales
Fruits	Apples	2020	$480,000

Category	Product	Year	Sales	Language
Früchte	Äpfel	2020	$480,000	de-DE
Fruits	Apples	2020	$480,000	en-US
frutas	manzanas	2020	$480,000	es-ES
Fruits	Pommes	2020	$480,000	fr-FR
фрукты	Яблоки	2020	$480,000	ru-RU
水果	苹果	2020	$480,000	zh-CN

Data Model Implementation of Row Replication Strategy

- Row replication requires configuring many-to-many relationships
  - Many-to-many relationships can have significant impact on performance and scale
  - Row replication used for **Products** table
  - **Customers** table too big for row replication
  - Country names factored into **Countries** table to localize country named

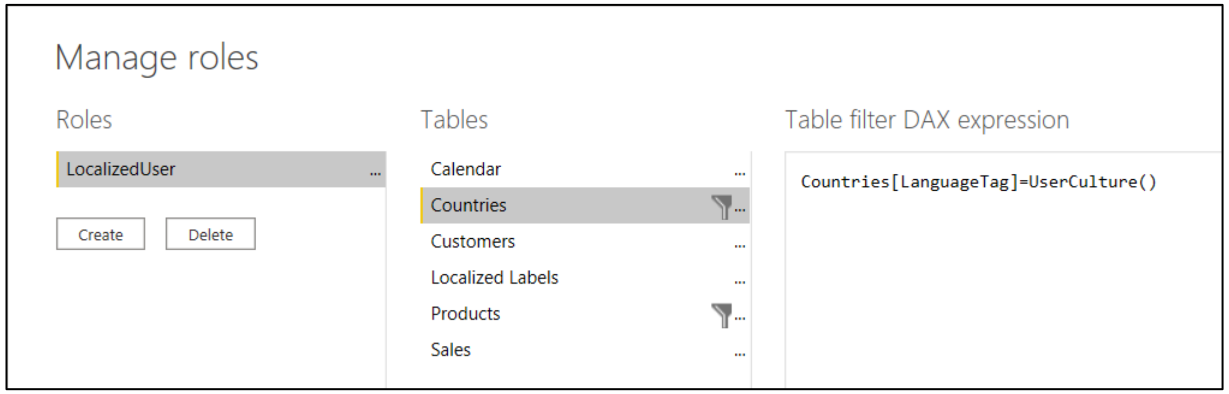


Agenda

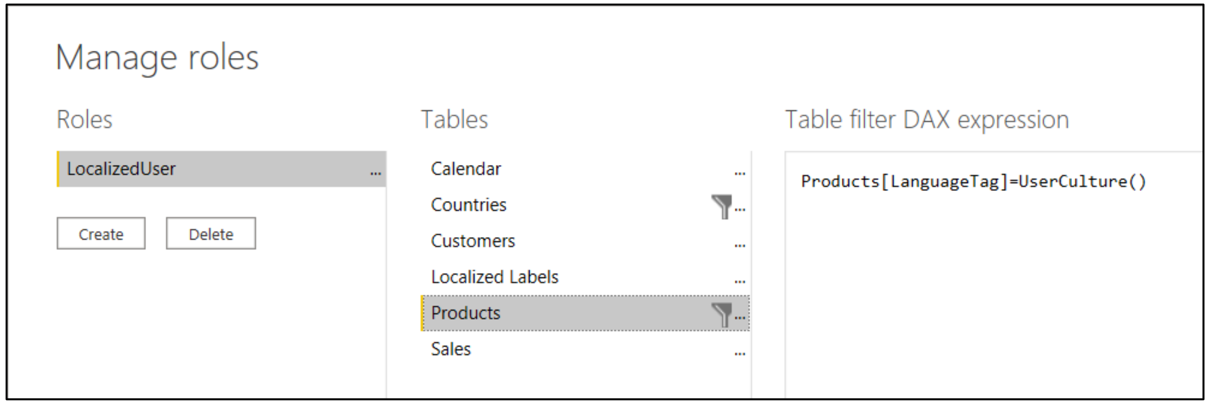
- ✓ Overview of Multi-language PBIX Development
- ✓ Designing Multi-language Reports
- ✓ Adding Metadata Translations with TOM
- ✓ Embedding Reports with Specific Locales
- ✓ Designing Data Models to Support Content Translations
- ➢ Setting the Language for Current User using RLS and UserCulture

Loading Report with Replicated Row Strategy

- App-Owns-Data developer can load report using Row-level Security roles
  - Create RLS role named **LocaizedUser**
  - Create RLS filter expression to filter translated rows using **UserCulture**

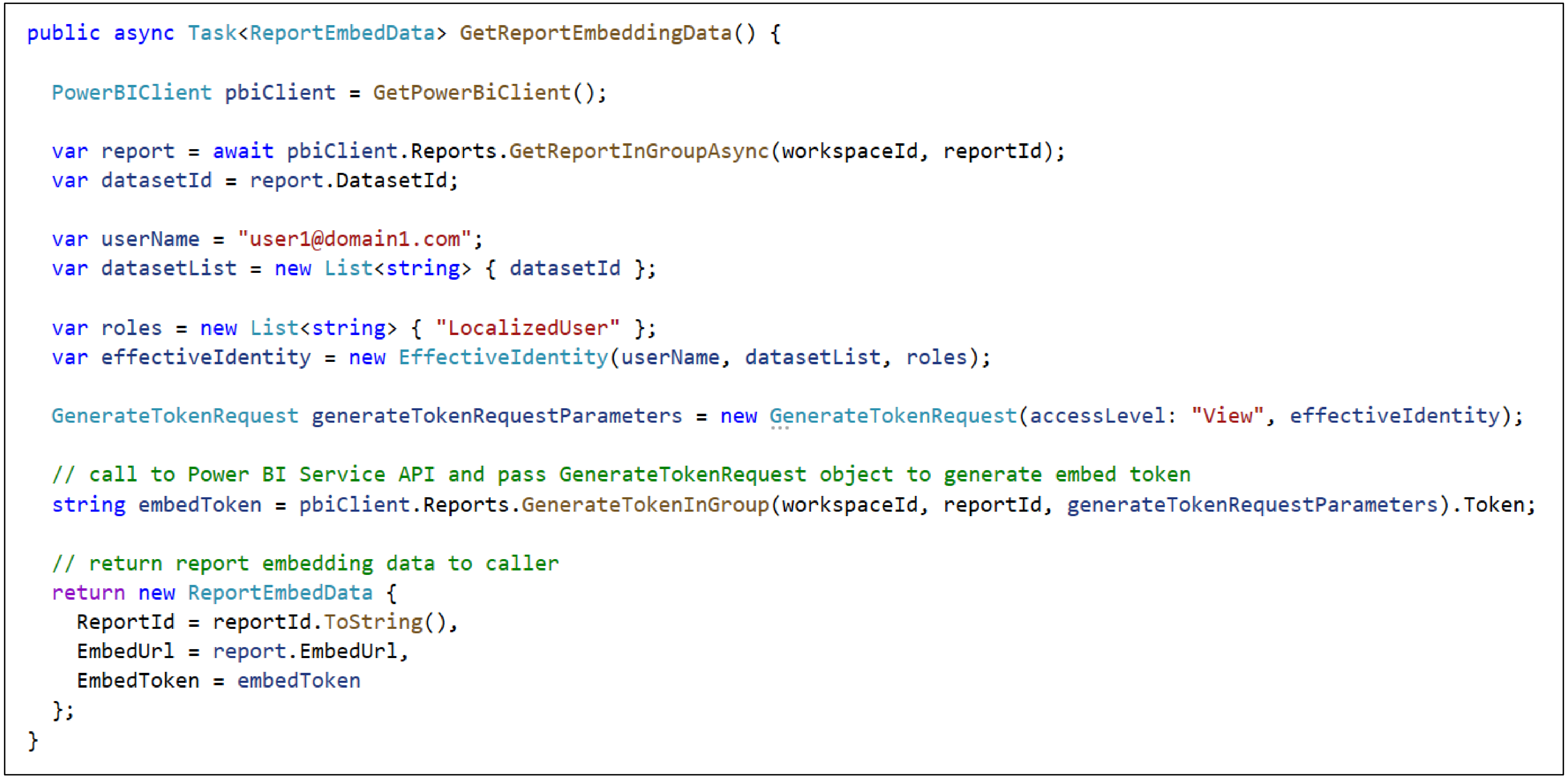


```
Countries[LanguageTag]=UserCulture()
```



```
Products[LanguageTag]=UserCulture()
```

Generating the Embed Token with LocalizedUser Role



```
public async Task<ReportEmbedData> GetReportEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var report = await pbiClient.Reports.GetReportInGroupAsync(workspaceId, reportId);
  var datasetId = report.DatasetId;

  var userName = "user1@domain1.com";
  var datasetList = new List<string> { datasetId };

  var roles = new List<string> { "LocalizedUser" };
  var effectiveIdentity = new EffectiveIdentity(userName, datasetList, roles);

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "View", effectiveIdentity);

  // call to Power BI Service API and pass GenerateTokenRequest object to generate embed token
  string embedToken = pbiClient.Reports.GenerateTokenInGroup(workspaceId, reportId, generateTokenRequestParameters).Token;

  // return report embedding data to caller
  return new ReportEmbedData {
    ReportId = reportId.ToString(),
    EmbedUrl = report.EmbedUrl,
    EmbedToken = embedToken
  };
}
```

Summary

- ✓ Overview of Multi-language PBIX Development
- ✓ Designing Multi-language Reports
- ✓ Adding Metadata Translations with TOM
- ✓ Embedding Reports with Specific Locales
- ✓ Designing Data Models to Support Content Translations
- ✓ Setting the Language for Current User using RLS and UserCulture

Summary

- ✓ Overview of Multi-language PBIX Development
- ✓ Programming Translations with TOM
- ✓ Designing Workflows to Manage Translations
- ✓ Multi-language Report Design with Translations
- ✓ Embedding Reports with Specific Locales