Power Apps Component Framework Overview

Cathal Noonan - 25th Nov 2021

Senior Technical Consultant at Codec Ireland

Agenda

- What is PCF
- Brief comparison with HTML Web Resources
- Where we can use PCF
- PCF development

What is PCF?

- Power Apps Component Framework
- Also referred to as "Code Components"
- Pro-code
- Replace existing field controls and grids

Comparison with HTML Web Resources

- HTML WebResources:
  - Usually involve separate files for HTML, CSS, JavaScript
  - May include translations using XML files
  - No specific build process needed unless using TypeScript already
- PCF:
  - Build process needed
  - Written in TypeScript
  - RESX files used for translations

Where can we learn about PCF?

- https://docs.microsoft.com/en-us/powerapps/developer/component-framework
  - Official documentation from Microsoft
- https://pcf.gallery
  - Pre-built components
  - Created by the community
  - Typically open-source projects, so you can see how they work

Places we can use PCF controls

- Model Driven Apps (Unified Interface only)
- Canvas Apps
- Power Apps Portals (in preview, since March 2021)
    - Need to assign permissions to Read the Web Resource table (entity) in the Web Roles
    - Need to create Entity Form Metadata or Web Form Metadata
- Custom Pages (preview feature)

PCF Development

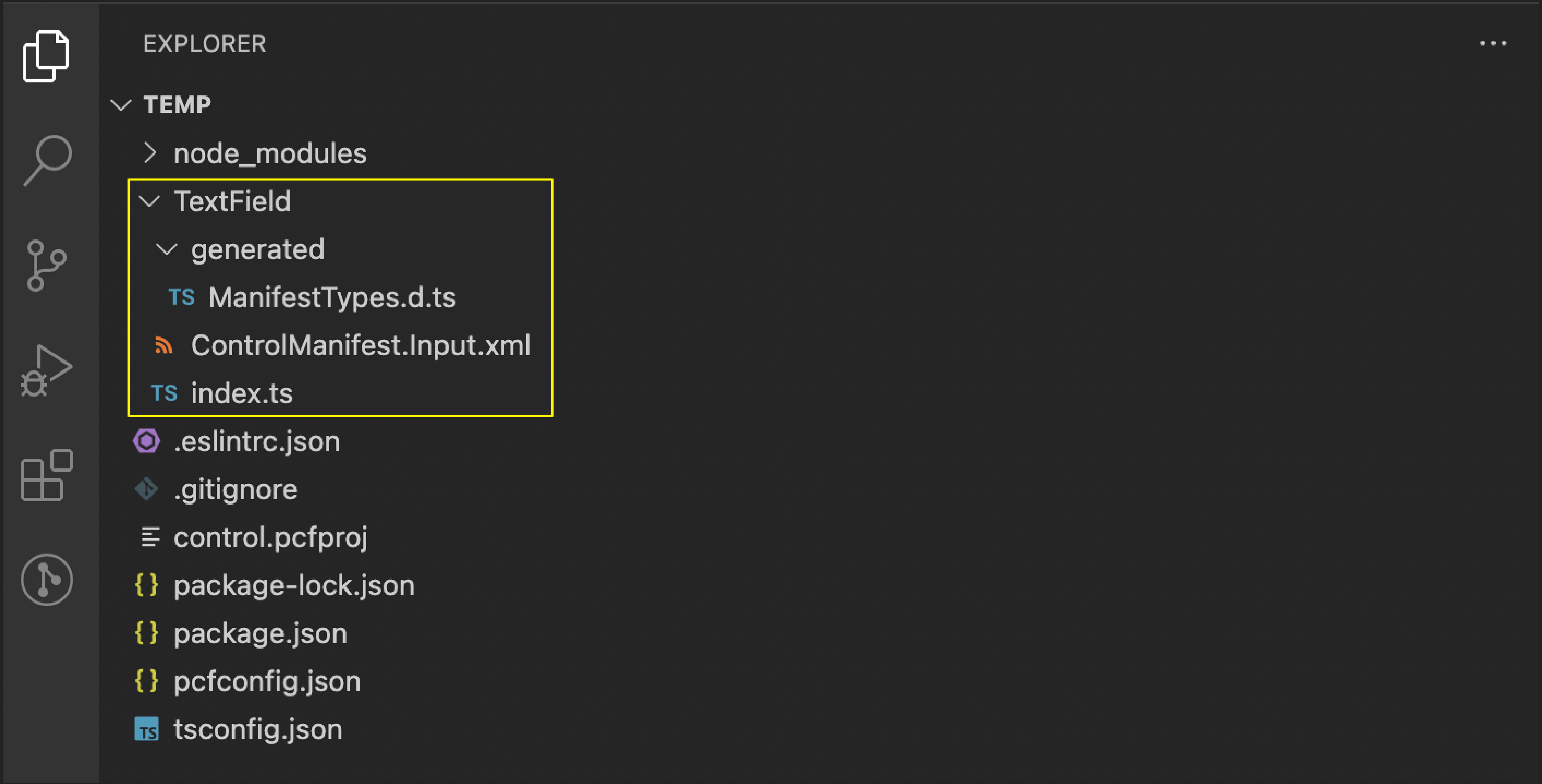
- What software is needed?
  - Node.JS (& npm)
  - dotnet
  - VS Code Extension, or Power Apps Command Line
- Other helpful tools
  - Fiddler AutoResponder, or Charles Proxy

Creating the PCF Project

- pac pcf init
    - --name [-n]
    - --namespace [-ns]
    - --type [-t]
        - Field or DataSet
- > Command:
  >
  > pac pcf init --name TextField --namespace ppug --template field

Project Structure

pac pcf init --name TextField --namespace ppug --template field



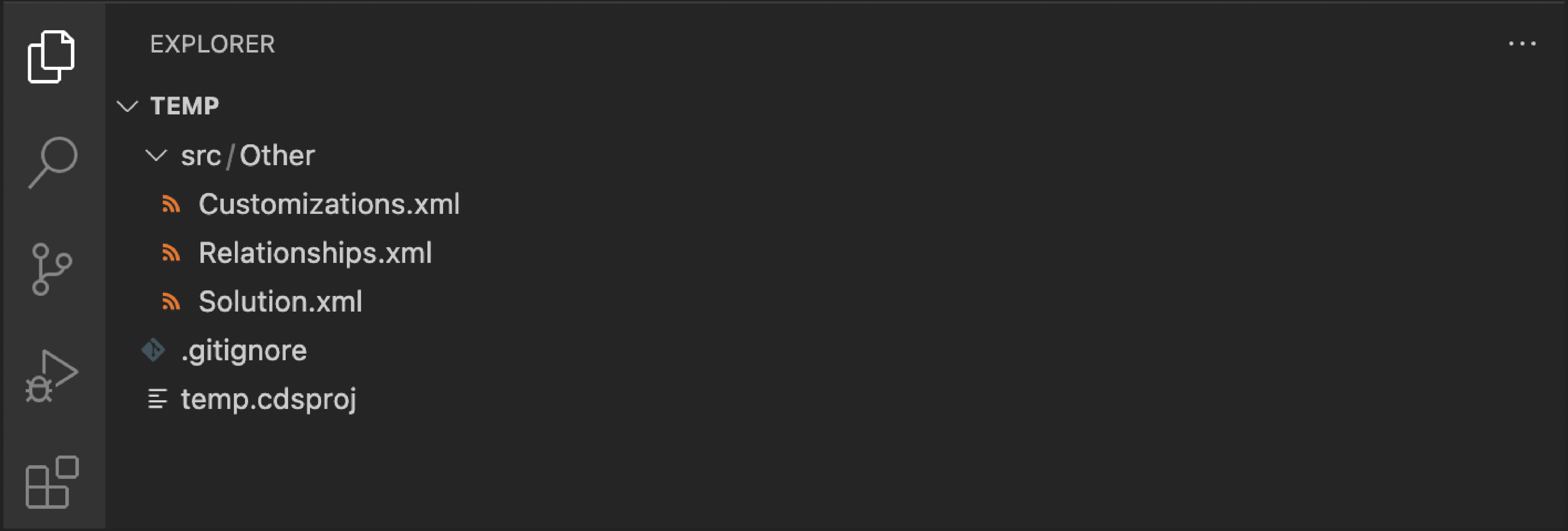
```
EXPLORER
TEMP
  node_modules
  TextField
    generated
      ManifestTypes.d.ts
    ControlManifest.Input.xml
    index.ts
  .eslintrc.json
  .gitignore
  control.pcfproj
  package-lock.json
  package.json
  pcfconfig.json
  tsconfig.json
```

Creating the PCF Solution

- pac solution init
    - --publisher-name [-pn]
    - --publisher-prefix [-pp]
    - --output-directory [-o] (optional)
- > Command:
  >
  > pac solution init --publisher-name PPUG --publisher-prefix ppug

Solution Structure

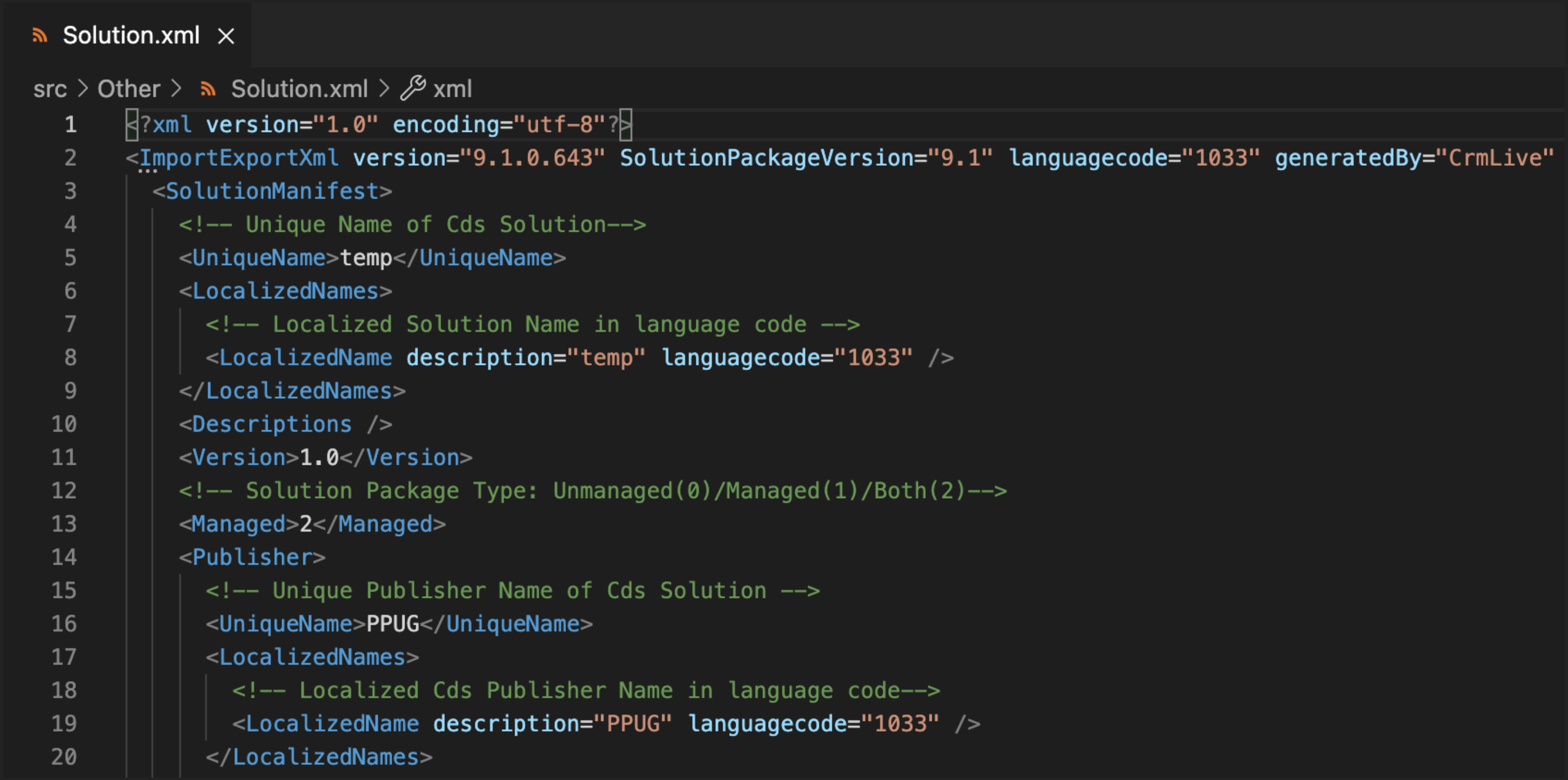
pac solution init --publisher-name PPUG --publisher-prefix ppug



```
EXPLORER
TEMP
  src / Other
    Customizations.xml
    Relationships.xml
    Solution.xml
  .gitignore
  temp.cdsproj
```

Solution XML

pac solution init --publisher-name PPUG --publisher-prefix ppug



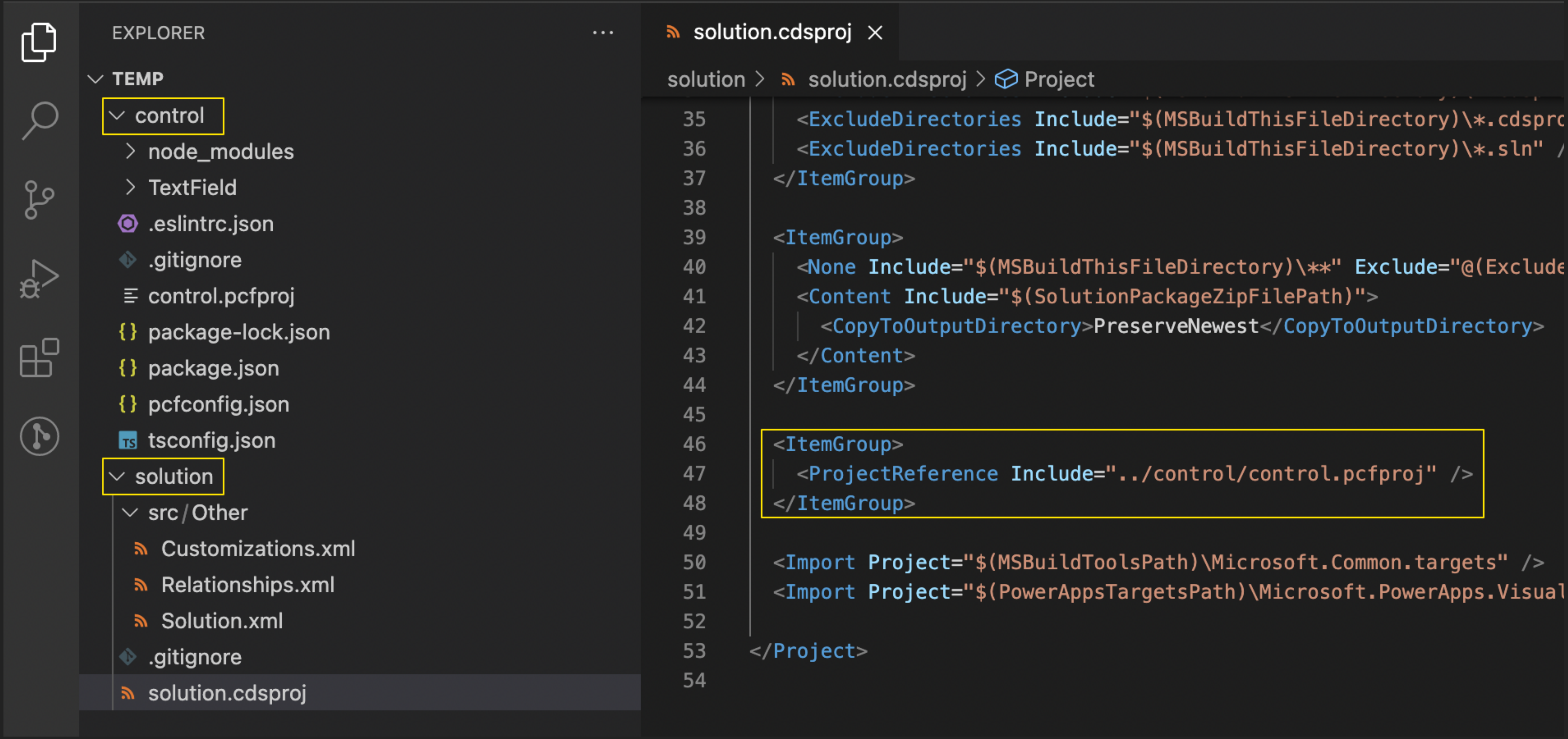
```xml
<?xml version="1.0" encoding="utf-8"?>
<ImportExportXml version="9.1.0.643" SolutionPackageVersion="9.1" languagecode="1033" generatedBy="CrmLive"
  <SolutionManifest>
    <!-- Unique Name of Cds Solution-->
    <UniqueName>temp</UniqueName>
    <LocalizedNames>
      <!-- Localized Solution Name in language code -->
      <LocalizedName description="temp" languagecode="1033" />
    </LocalizedNames>
    <Descriptions />
    <Version>1.0</Version>
    <!-- Solution Package Type: Unmanaged(0)/Managed(1)/Both(2)-->
    <Managed>2</Managed>
    <Publisher>
      <!-- Unique Publisher Name of Cds Solution -->
      <UniqueName>PPUG</UniqueName>
      <LocalizedNames>
        <!-- Localized Cds Publisher Name in language code-->
        <LocalizedName description="PPUG" languagecode="1033" />
      </LocalizedNames>
```

Add the project to the solution

- Change directory into the folder containing the solution
- pac solution add-reference
  - --path [-p]
- > Command:
  >
  > pac solution add-reference --path ../control

Reference added to solution

pac solution add-reference --path ../control

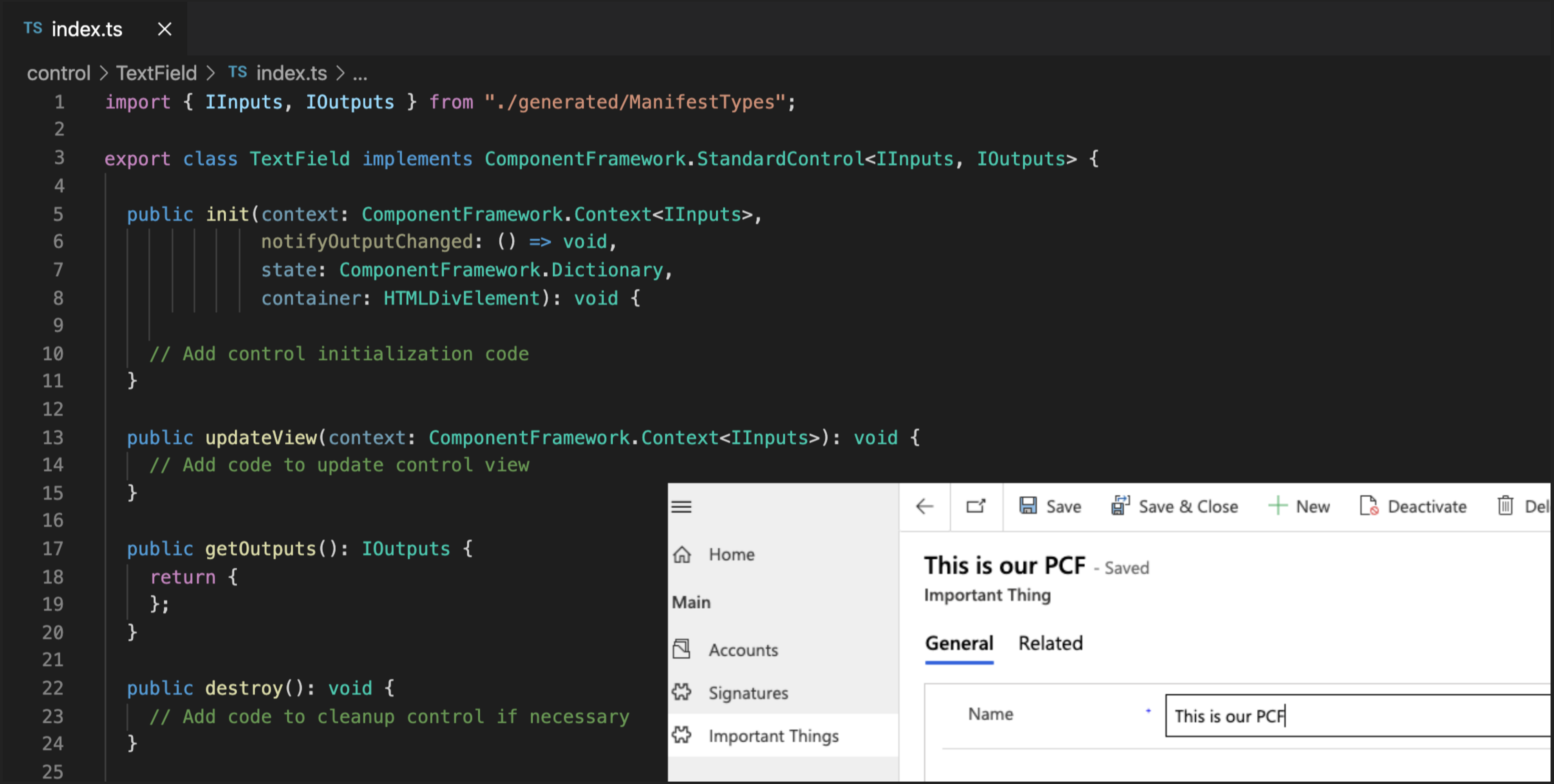


```
      <ExcludeDirectories Include="$(MSBuildThisFileDirectory)\*.cdspro
      <ExcludeDirectories Include="$(MSBuildThisFileDirectory)\*.sln" /
    </ItemGroup>

    <ItemGroup>
      <None Include="$(MSBuildThisFileDirectory)\**" Exclude="@(Exclude
      <Content Include="$(SolutionPackageZipFilePath)">
        <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
      </Content>
    </ItemGroup>

    <ItemGroup>
      <ProjectReference Include="../control/control.pcfproj" />
    </ItemGroup>

    <Import Project="$(MSBuildToolsPath)\Microsoft.Common.targets" />
    <Import Project="$(PowerAppsTargetsPath)\Microsoft.PowerApps.Visual

  </Project>

```

Building and deploying the solution

- Change directory into the solution folder
- For unmanaged solution
  - dotnet build
- For managed solution
  - dotnet build -p:Configuration=Release

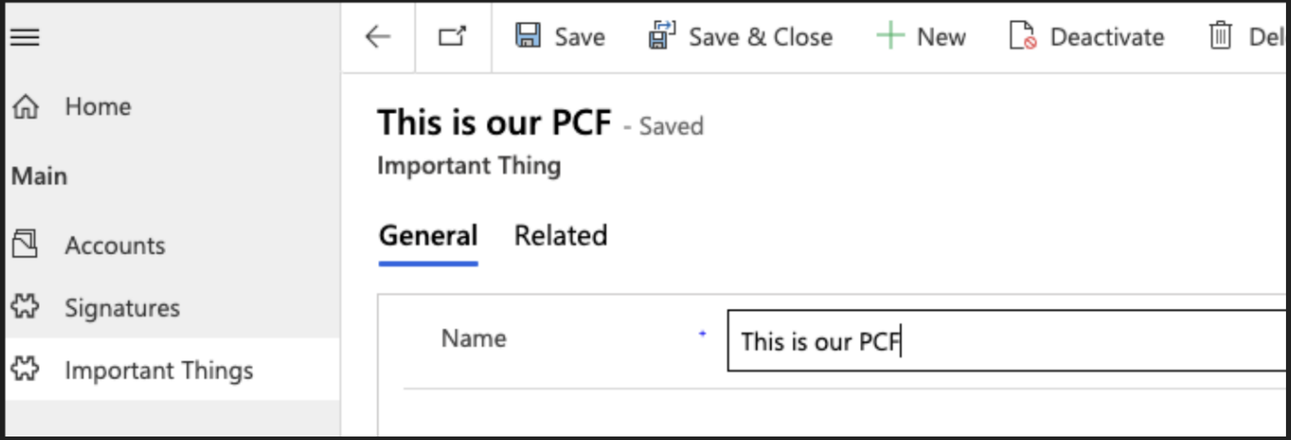
Demo



index.ts

control > TextField > index.ts > ...

```typescript
import { IInputs, IOutputs } from "./generated/ManifestTypes";

export class TextField implements ComponentFramework.StandardControl<IInputs, IOutputs> {

  public init(context: ComponentFramework.Context<IInputs>,
              notifyOutputChanged: () => void,
              state: ComponentFramework.Dictionary,
              container: HTMLDivElement): void {

    // Add control initialization code
  }

  public updateView(context: ComponentFramework.Context<IInputs>): void {
    // Add code to update control view
  }

  public getOutputs(): IOutputs {
    return {
    };
  }

  public destroy(): void {
    // Add code to cleanup control if necessary
  }

```



This is our PCF - Saved

Important Thing

Name: This is our PCF

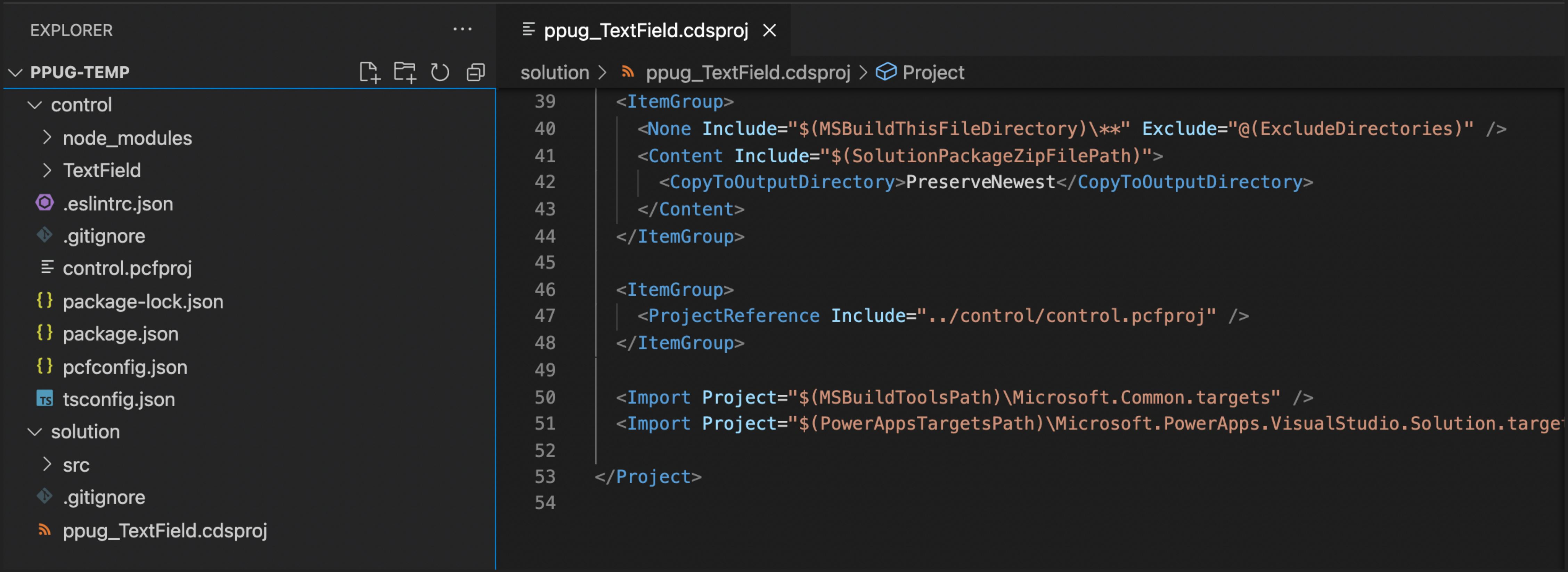
Demo

- Creating the project & solution
- Building out the source code
- Deploying the solution
- Tools for Local Development

Demo (1 of 5)

- Run the command above to create the project
- Run the command above to create the solution
- Run the command to add the project to the solution

Demo (1 of 5)

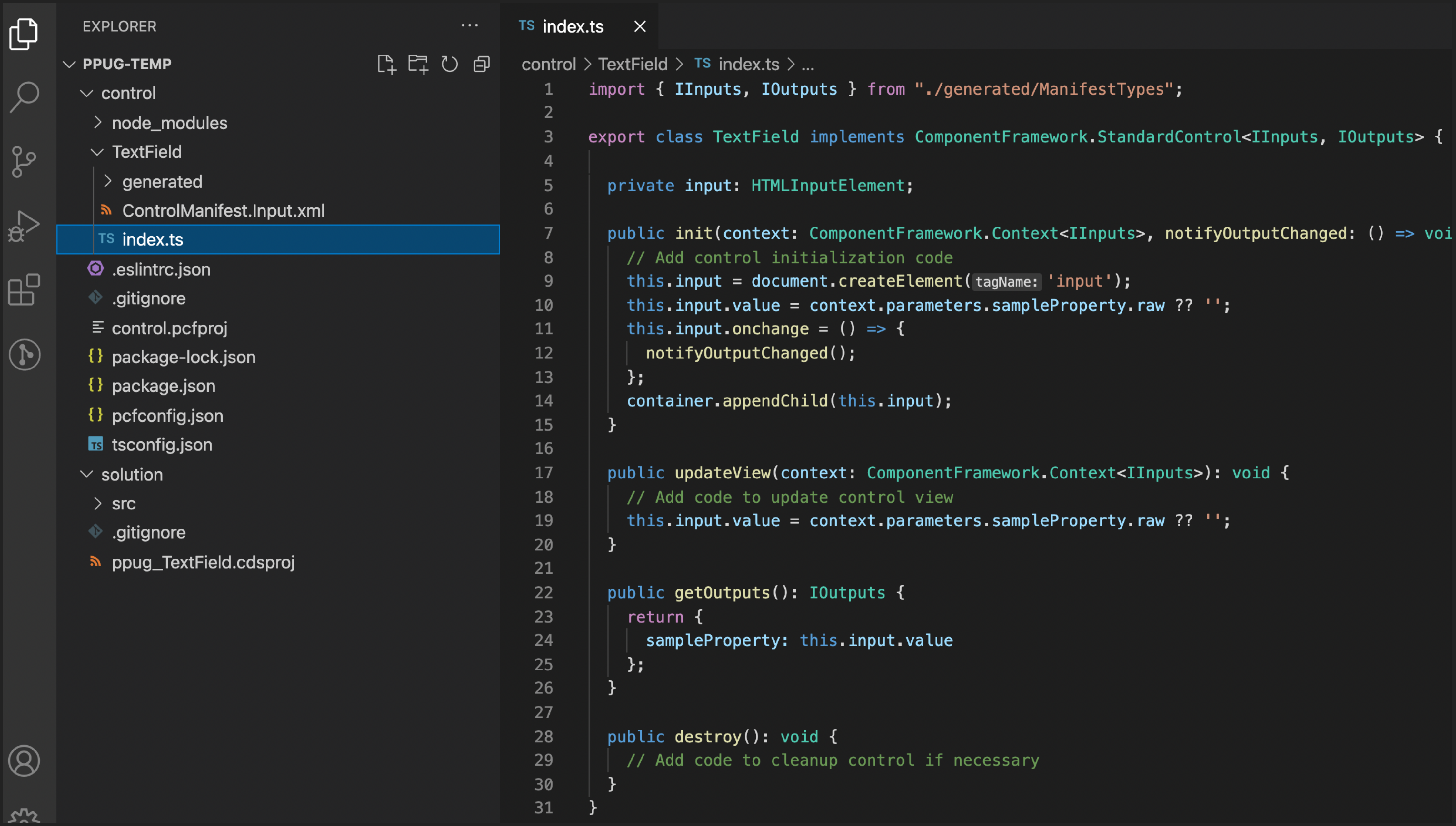


```xml
  <ItemGroup>
    <None Include="$(MSBuildThisFileDirectory)\**" Exclude="@(ExcludeDirectories)" />
    <Content Include="$(SolutionPackageZipFilePath)">
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </Content>
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="../control/control.pcfproj" />
  </ItemGroup>

  <Import Project="$(MSBuildToolsPath)\Microsoft.Common.targets" />
  <Import Project="$(PowerAppsTargetsPath)\Microsoft.PowerApps.VisualStudio.Solution.targe

</Project>

```

Demo (2 of 5)

- Create a basic control using `document.createElement()`
    - Receive the field value on load using `init`
- For the demo, will not implement the following straight away:
    - Receive new values using `updateView`
    - Returns the updated value using `notifyOutputChanged`
- Start the project locally using `npm run start watch`

Demo (2 of 5)



```
EXPLORER
PPUG-TEMP
  control
    node_modules
    TextField
      generated
      ControlManifest.Input.xml
      index.ts
    .eslintrc.json
    .gitignore
    control.pcfproj
    package-lock.json
    package.json
    pcfconfig.json
    tsconfig.json
  solution
    src
    .gitignore
    ppug_TextField.cdsproj
```

index.ts

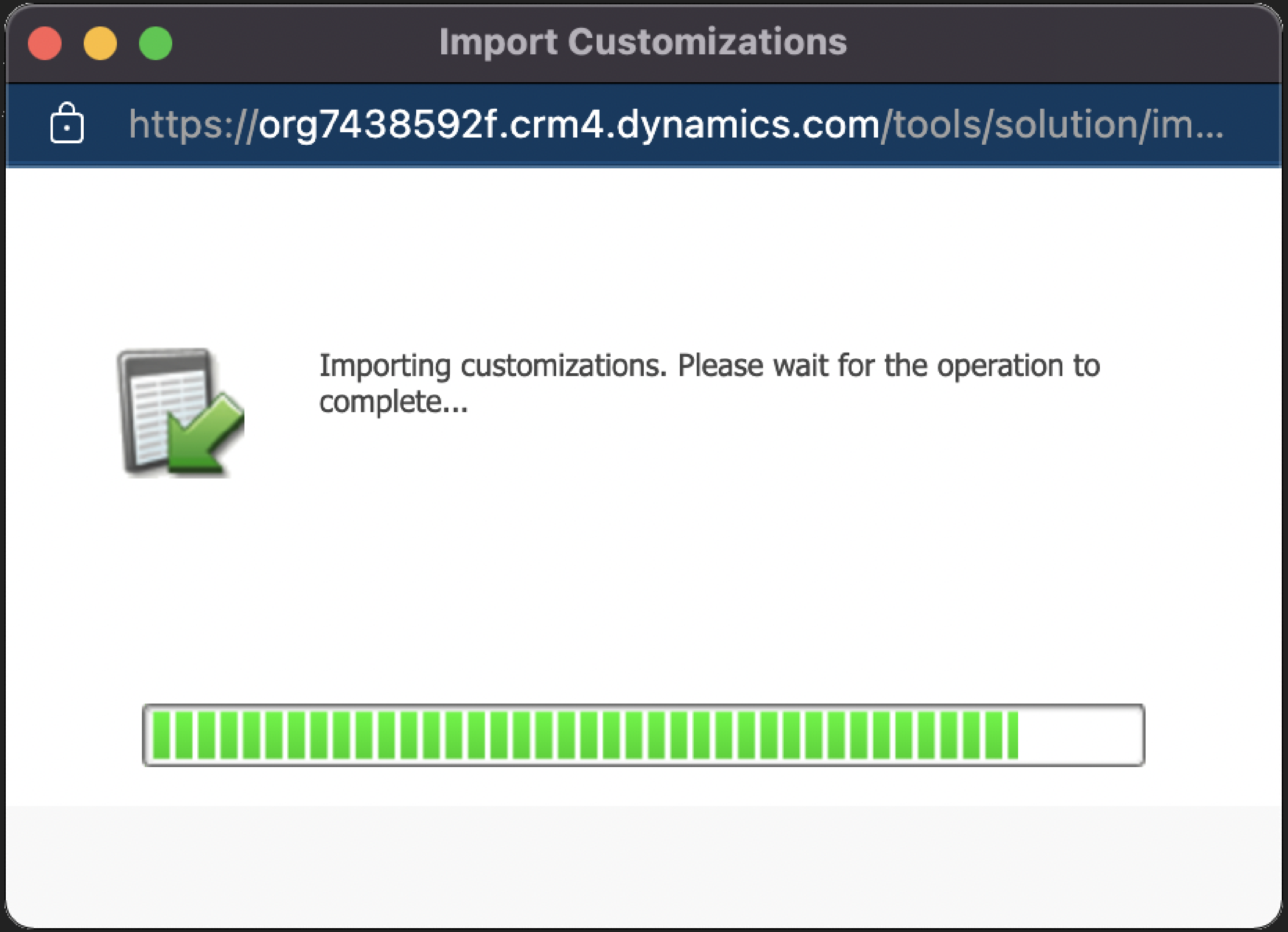
control > TextField > index.ts > ...

```typescript
import { IInputs, IOutputs } from "./generated/ManifestTypes";

export class TextField implements ComponentFramework.StandardControl<IInputs, IOutputs> {

  private input: HTMLInputElement;

  public init(context: ComponentFramework.Context<IInputs>, notifyOutputChanged: () => voi
    // Add control initialization code
    this.input = document.createElement(tagName: 'input');
    this.input.value = context.parameters.sampleProperty.raw ?? '';
    this.input.onchange = () => {
      notifyOutputChanged();
    };
    container.appendChild(this.input);
  }

  public updateView(context: ComponentFramework.Context<IInputs>): void {
    // Add code to update control view
    this.input.value = context.parameters.sampleProperty.raw ?? '';
  }

  public getOutputs(): IOutputs {
    return {
      sampleProperty: this.input.value
    };
  }

  public destroy(): void {
    // Add code to cleanup control if necessary
  }
}
```

Demo (3 of 5)

- Build and deploy the solution
- Configure the control using the classic form editor

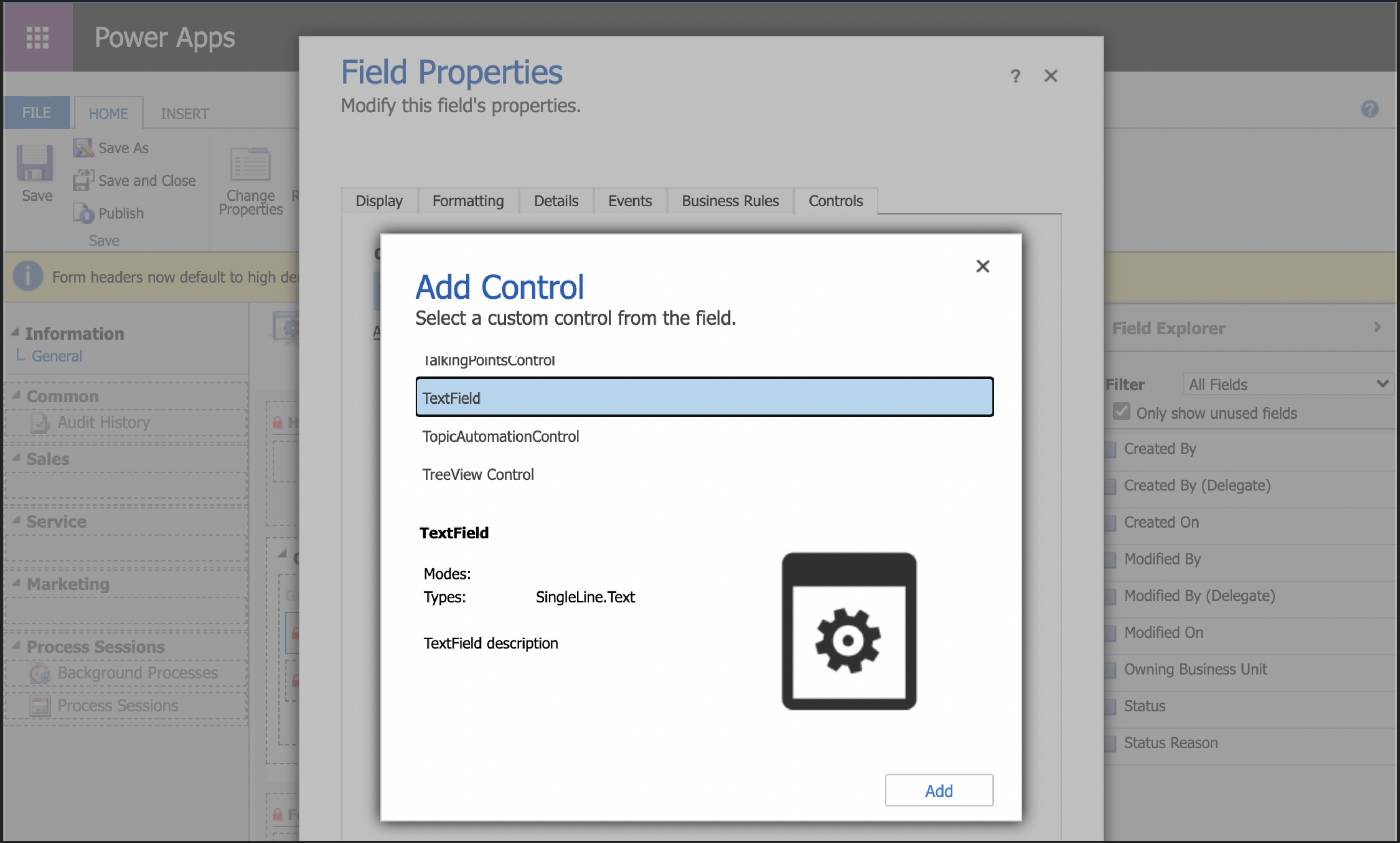
Demo (3 of 5)


Import Customizations

https://org7438592f.crm4.dynamics.com/tools/solution/im...

Importing customizations. Please wait for the operation to complete...

Demo (3 of 5)



Power Apps

FILE HOME INSERT

Save, Save As, Save and Close, Publish, Change Properties

Form headers now default to high de

Information: General; Common: Audit History; Sales; Service; Marketing; Process Sessions: Background Processes, Process Sessions

Field Properties

Modify this field's properties.

Display | Formatting | Details | Events | Business Rules | Controls

Add Control

Select a custom control from the field.

TalkingPointsControl

TextField

TopicAutomationControl

TreeView Control

TextField

Modes:	
Types:	SingleLine.Text

TextField description

Add

Field Explorer

Filter: All Fields

☑ Only show unused fields

Created By, Created By (Delegate), Created On, Modified By, Modified By (Delegate), Modified On, Owning Business Unit, Status, Status Reason

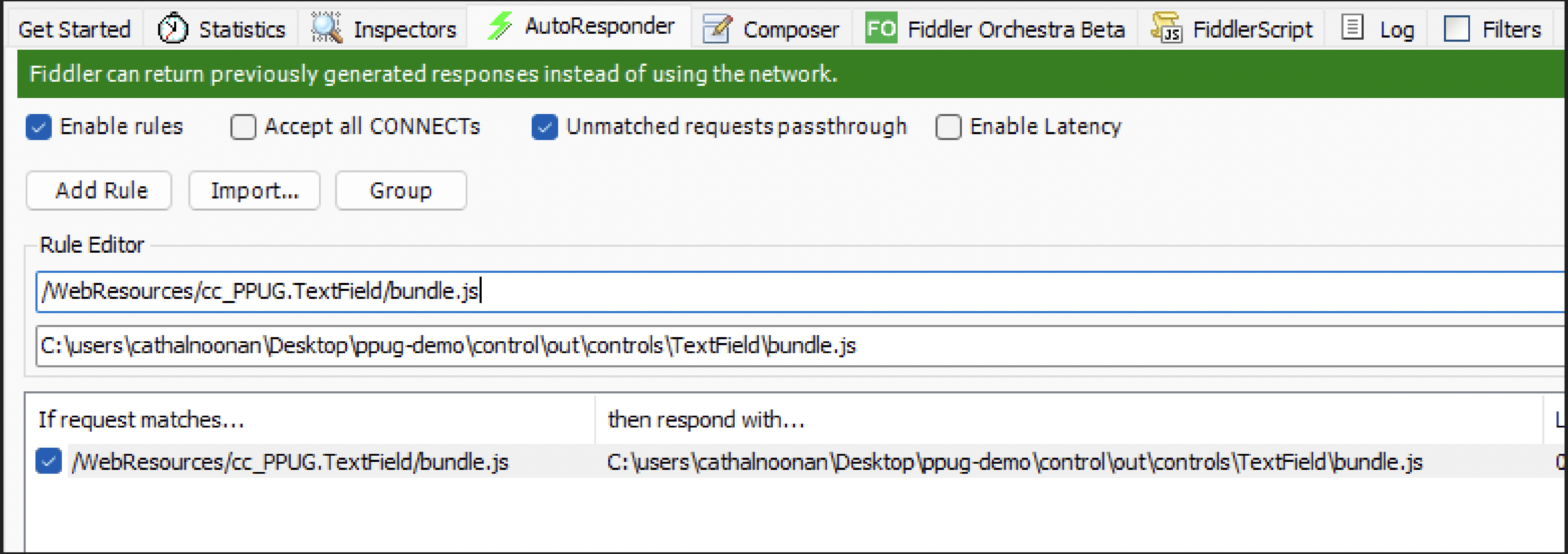
Demo (4 of 5)

- Start fiddler
- Configure the AutoResponder rule
  - Reloading the page in D365 will show the new code changes without deploying the solution
- Implement the remaining functionality to handle field value changes:
  - `updateView`
  - `notifyOutputChanged`

Demo (4 of 5)

Fiddler, AutoResponder configuration

- Publisher Prefix: **PPUG**
- Control Name: **TextField**



Fiddler can return previously generated responses instead of using the network.

✓ Enable rules ☐ Accept all CONNECTs ✓ Unmatched requests passthrough ☐ Enable Latency

Rule Editor

/WebResources/cc_PPUG.TextField/bundle.js

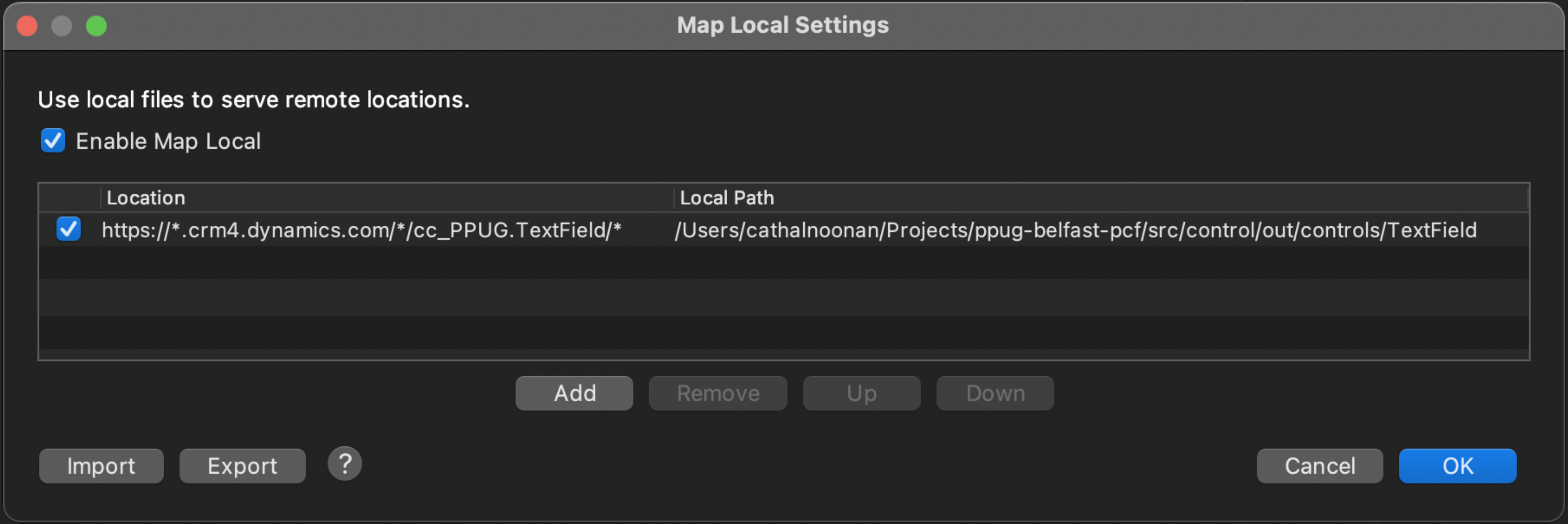
C:\users\cathalnoonan\Desktop\ppug-demo\control\out\controls\TextField\bundle.js

If request matches...	then respond with...
✓ /WebResources/cc_PPUG.TextField/bundle.js	C:\users\cathalnoonan\Desktop\ppug-demo\control\out\controls\TextField\bundle.js

Demo (4 of 5)

Charles, Map Local configuration

- Publisher Prefix: **PPUG**
- Control Name: **TextField**



Map Local Settings

Use local files to serve remote locations.

✓ Enable Map Local

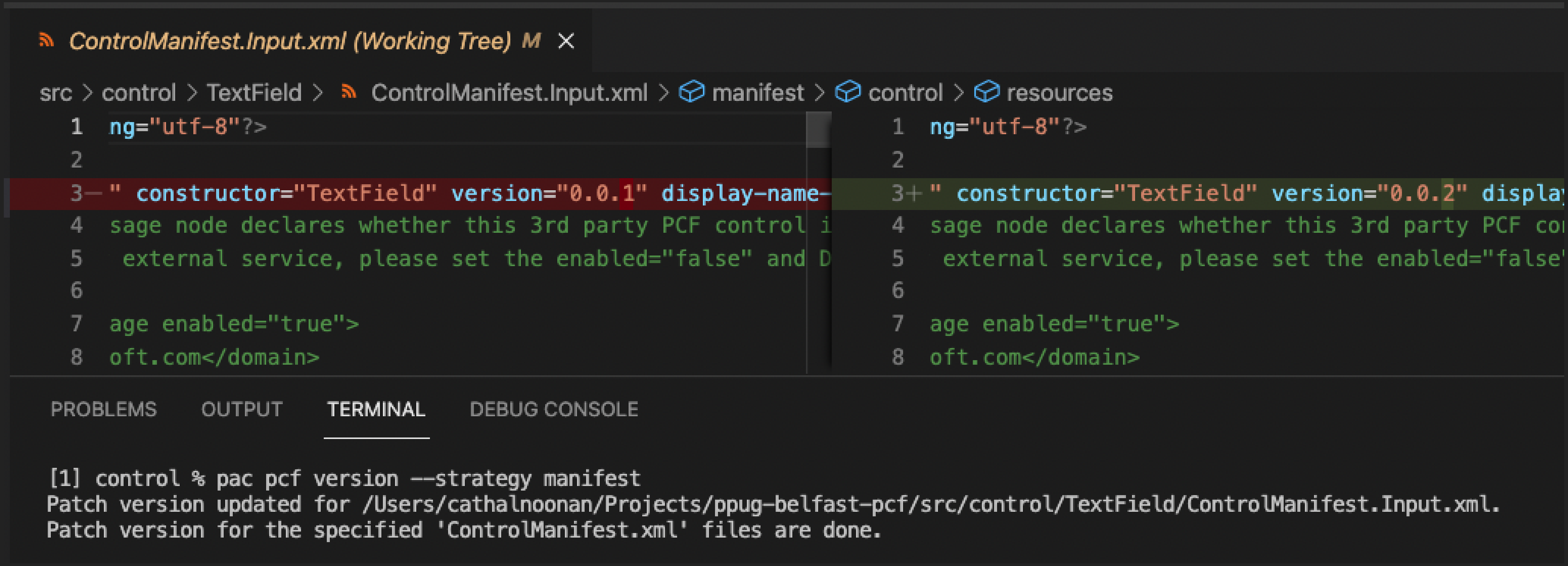
	Location	Local Path
✓	https://*.crm4.dynamics.com/*/cc_PPUG.TextField/*	/Users/cathalnoonan/Projects/ppug-belfast-pcf/src/control/out/controls/TextField

Demo (5 of 5)

- Update the PCF version number
- Deploy the solution
- Stop fiddler & reload the page

Demo (5 of 5)

pac pcf version --strategy manifest



ControlManifest.Input.xml (Working Tree) M ×

src > control > TextField > ControlManifest.Input.xml > manifest > control > resources

```
ng="utf-8"?>

" constructor="TextField" version="0.0.1" display-name-
sage node declares whether this 3rd party PCF control i
 external service, please set the enabled="false" and D

age enabled="true">
oft.com</domain>
```

```
ng="utf-8"?>

" constructor="TextField" version="0.0.2" displa
sage node declares whether this 3rd party PCF co
 external service, please set the enabled="false'

age enabled="true">
oft.com</domain>
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[1] control % pac pcf version --strategy manifest
Patch version updated for /Users/cathalnoonan/Projects/ppug-belfast-pcf/src/control/TextField/ControlManifest.Input.xml.
Patch version for the specified 'ControlManifest.xml' files are done.
```

Extra demo (time permitting)

- Replace `document.createElement` with React & Fluent UI
  - Copy and paste a working example into the project
  - Suggest having a look at the FluentUI docs rather than getting too heavy on the details

Thank you for listening!

Slides & source code from the demo

https://github.com/cathalnoonan/ppug-belfast-202110