Contact Editor Creation steps

Dynamics CRM Development Task

David Cintado Cámara

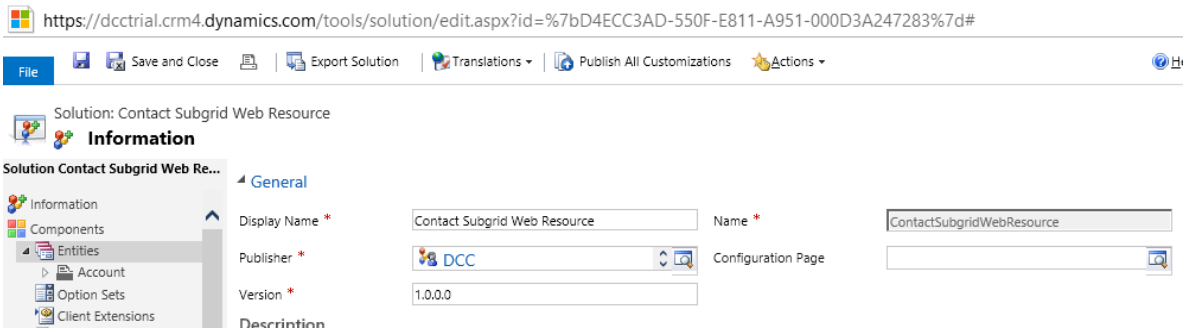
ÍNDICE

1. Introduction

The objective of this document is to summarize the steps I followed to complete the Dynamics CRM Task.

2. CRM

- I created my own Dynamics 365 trial environment.
- I created a solution “” with my own publisher DCC “dcc_”.
- I added the Account form to this solution.

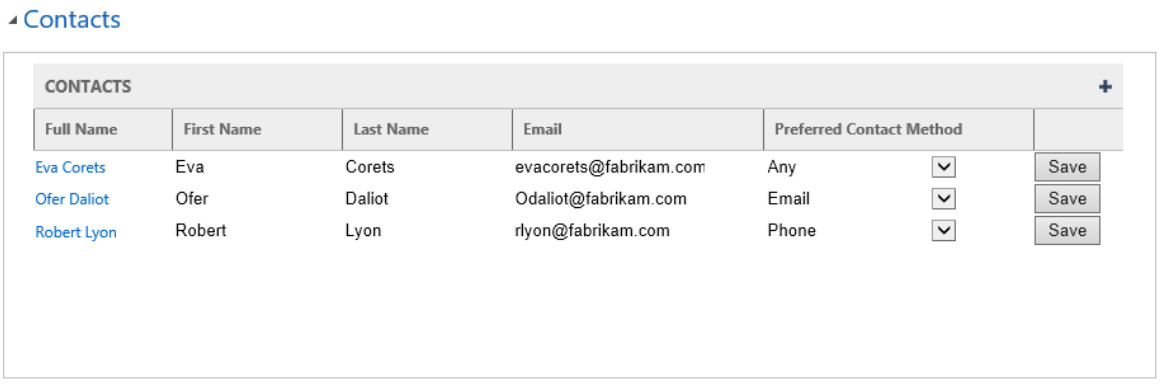


3. Visual Studio 2015

- I created a new solution in VS with a CRM Web Resource Project, where I created the .html, .js, .css.
- I added JQuery using NuGet Package Manager.
- I used Developer Extensions Tool to deploy these component into CRM.

4. The Task

Although the task has three levels, but I did not create three different html web resources. I create just one, and I added the functionality required in the following levels.



- The contacts full name is a hyperlink to open the contact record.

- First Name, Last Name and Preferred Contact Method are editable fields, and with the bottom Save the changes can be saved to update the contact information.

- Last Name field is required; an alert message appears if you try to save a contact without this field filled.

- The + button allows to create a new contact record using the Quick Create form.

- Preferred Contact Method is multilingual, its values will be shown always in the user CRM selected language.

4.1 Task Level 1

These are the steps I followed to create the Contact editor web resource.

Create an HTML Web Resource that:

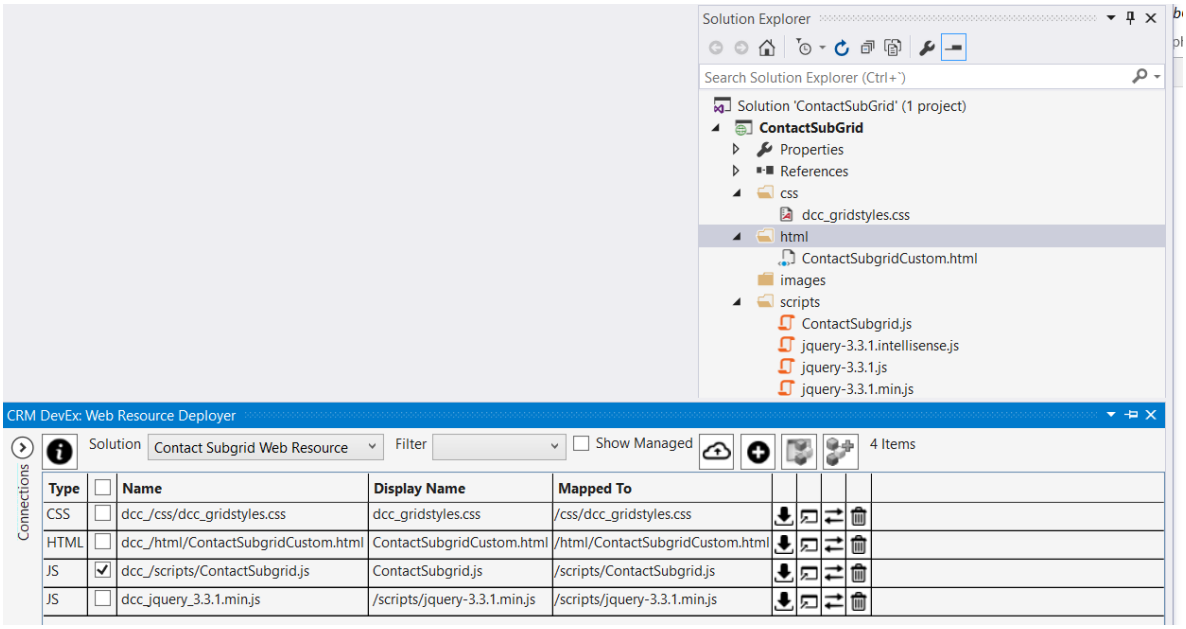
- **Can be placed on an Account Form** :
    - I inserted a new Tab, new section, and a web resource into the Account main form.

- **Shows a grid of Contacts that are associated with the parent Account (a similar UI Style to the standard sub-grid)**:
    - I created a table in html.
    - Using javascript and web API, I created a function to retrieve the contacts related with the current account. (`function GetContact(successCallback)`)
    - I created a .css file, trying to copy the CRM styles.

- **Allows selecting and opening of a contact form from the grid** :
    - Using javascript, each cell in the column Full Name is a hyperlink to open the contact.

- **Allows adding new contacts using a '+' button.**
    - I added an image as a button to open Contact Quick Create Form. When the quick create form is opened, the parent account field is informed.
    - I used an image that already exists in CRM.

- **Create a Visual Studio solution to deploy the HTML Web Resource**
    - I created a solution in VS, and I used Developer Extensions Tools to deploy each web resource into my CRM solution.

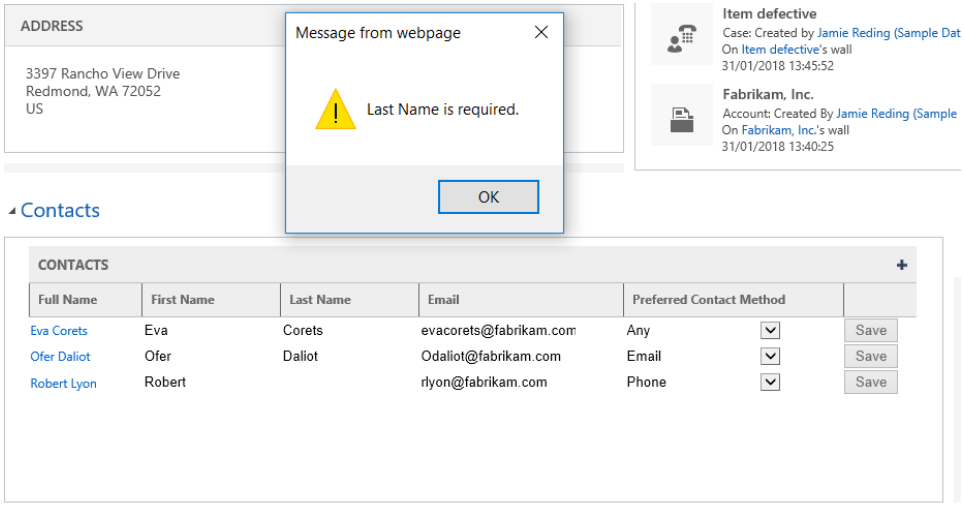


4.2 Task Level 2

I modified the web resource created for the level 1.

Create an HTML Web Resource that additionally:

- **Allows inline grid editing of the fields:**
    - **First Name**
    - **Last Name**
    - **Preferred Contact Method (As a drop down)**
    - I update my javascript function to retrieve these fields.
    - I added new columns in the html table and edit their properties.
    - I created a function to update entities with javascript and web API.(`function UpdateEntity(clientURL, entityId, entityType, entityTobeUpdated, apiVersion) {}`
    - To show Preferred Contact Method I created a <select> component. To set the selected value I use the retrieved data.
- **The Last Name must be a mandatory field and cannot be blank.**
    - I created a javascript validation to control the Last Name is filled before the update can be saved.



4.3 Task Level 3

I modified the web resource created for the level 2.

Create an HTML Web Resource that additionally:

• Load option set values and labels from the CRM Metadata rather than hard code

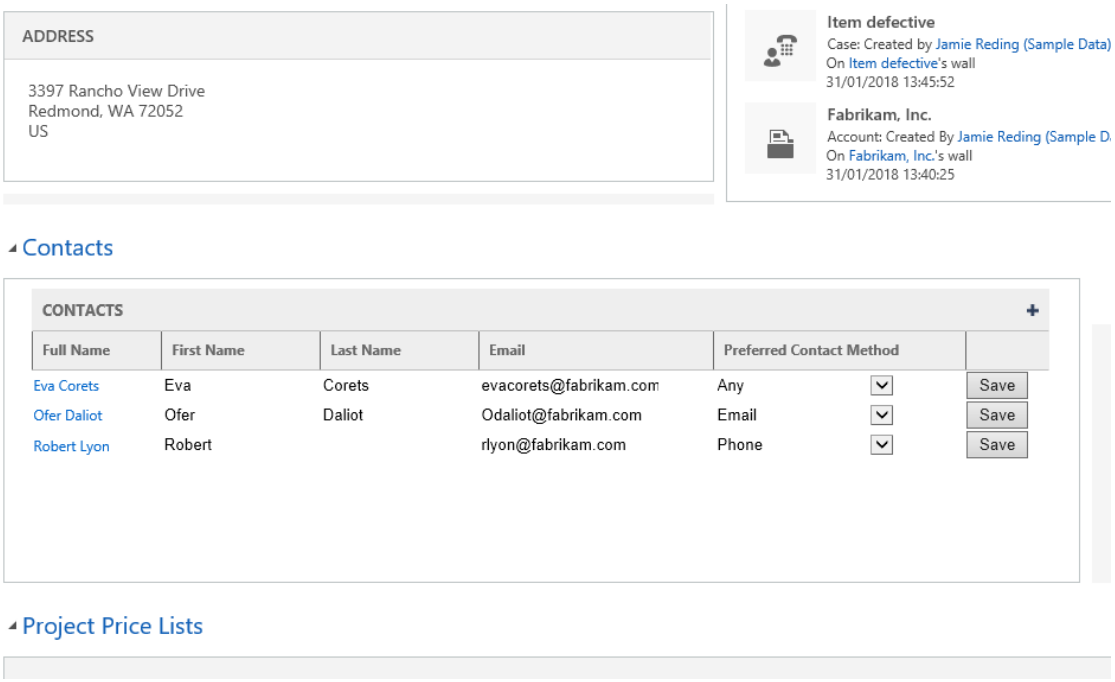
- I created a javascript function to retrieve the metadata of the option set. Another function get the information retrieved and create a html <select> component dynamically, using the current crm user language. (`function GetMetadataByName(entityLogName, attLogName, successCallback)`)

• Provide Multi-Language support for English and Spanish (Grid Labels and Drop Down Values etc.) based on the user's language preference

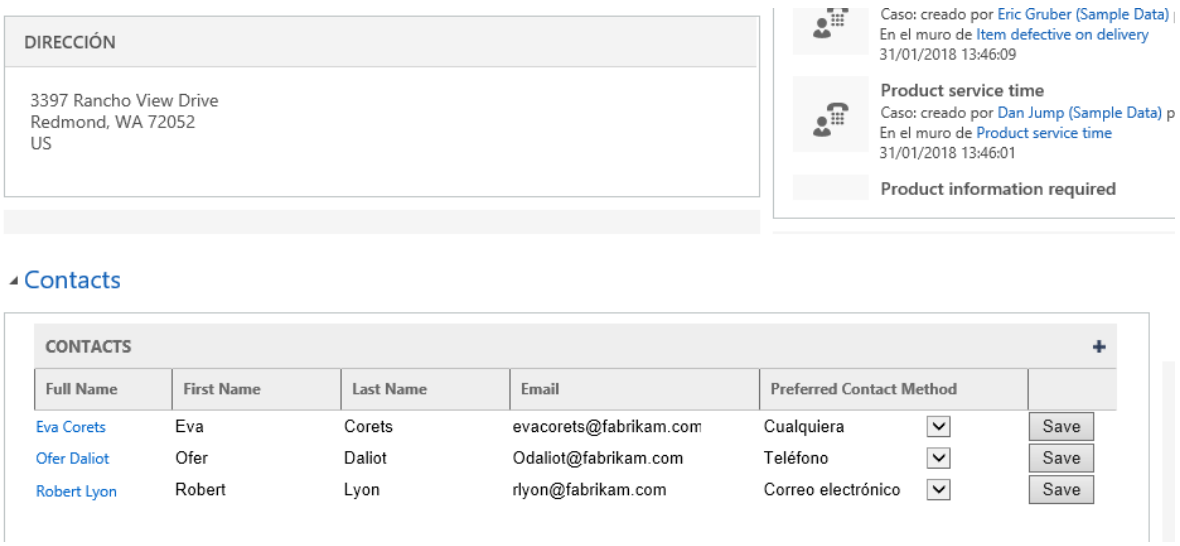
- The function that creates the select component uses the UserLocalizedLabel, to show the labels in the user language.

```
prefConMethOptions = prefConMethOptions + "<option value='" +
data.OptionSet.Options[i].Value +
                "'>" + data.OptionSet.Options[i].Label.UserLocalizedLabel.Label +
"</option>";
```

English:



Spanish:



Note: I did not translate the title of the columns or the value of the button Save, although the information is available in the javascript code.