

# Progress® OpenEdge® MOBILE

Applications Workbook

Applications Workbook for using Progress®  
OpenEdge® Mobile

## CONTENTS

Mobile App .....	4
Starting restmgr1 and restbroker1 .....	6
Creating a new Mobile OpenEdge project .....	7
Creating an include file .....	8
Creating a new Business Entity.....	10
Adding OpenEdge Business Logic to the Business Entity, dsCustomer.cls.....	11
Create the Catalog File .....	12
Testing the OpenEdge Business Entity .....	13
Creating the Mobile App.....	14
Adding the JSDO service to the client .....	17
Adding JavaScript .....	18
Adding datasources (services).....	20
Adding Events .....	20
Add the mapping from the service to the client.....	22
Add Searching and Batching to CustPage .....	24
Add Events to CustPage.....	26
You need to do some mapping for the NameSearch filter .....	<b>Error! Bookmark not defined.</b>
Adding Events to the CustDetail Page .....	29
Mapping SaveCustomer in the CustDetail Page .....	31
Creating Events on the home Page .....	33
Creating Events to populate map on the CustDetail Page .....	33
Events to Save changes to customer contact information .....	34
Modify a property of the Collapsible Control in the CustDetails page.....	34
GeoLocation details .....	36
How to add the GeoLocation Service:.....	36
Events on the CustDetail Page: .....	38
Deploying Mobile Applications .....	40
Deployment overview.....	40
Packaging and Deploying iOS Apps .....	40
Packaging iOS Apps .....	40
Deploying iOS Apps .....	44
Packaging and Deploying Android Apps .....	46
Packaging Android Apps.....	46

---



---

## MOBILE APP

---

This chapter steps you through the creation of a Mobile Application. It describes the creation of a Mobile service with access to an OpenEdge database using Progress Developer Studio for OpenEdge. It also describes how to use the Progress OpenEdge Mobile App Builder to create a simple Mobile App (UI) that accesses this service and displays information from the database.

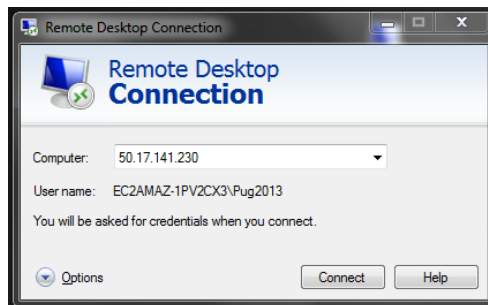
### Important Note:

This workbook requires that you use a Remote Desktop Connection. You can find the Remote Desktop Connection in the Accessories folder.

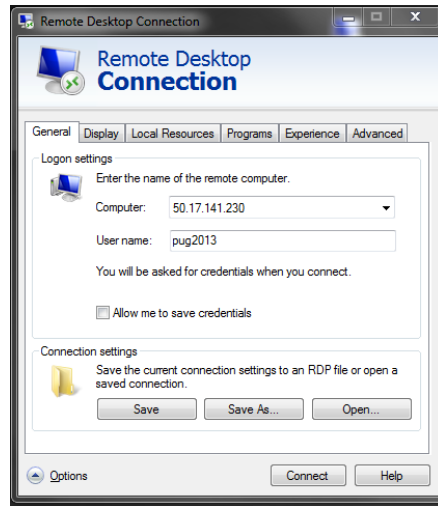
Refer to your table tag for the Computer name and the project id.

The **username** for the machine is **pug2013** and there is no password

When you launch the Remote Desktop Connection, you will see this screen:



1. Enter the **Computer name** that is located the table tag,
2. Click on the **Options dropdown**, and enter **pug2013** for the User name



3. Click on the **Connect** button.
4. On the Windows Security Window, click on the **OK** button
5. Click **Yes** on the Remote Desktop Connection Window

Now your remote session has started.

To start Progress Developers Studio for OpenEdge, click on the **Progress Developers Studio icon on the desktop**

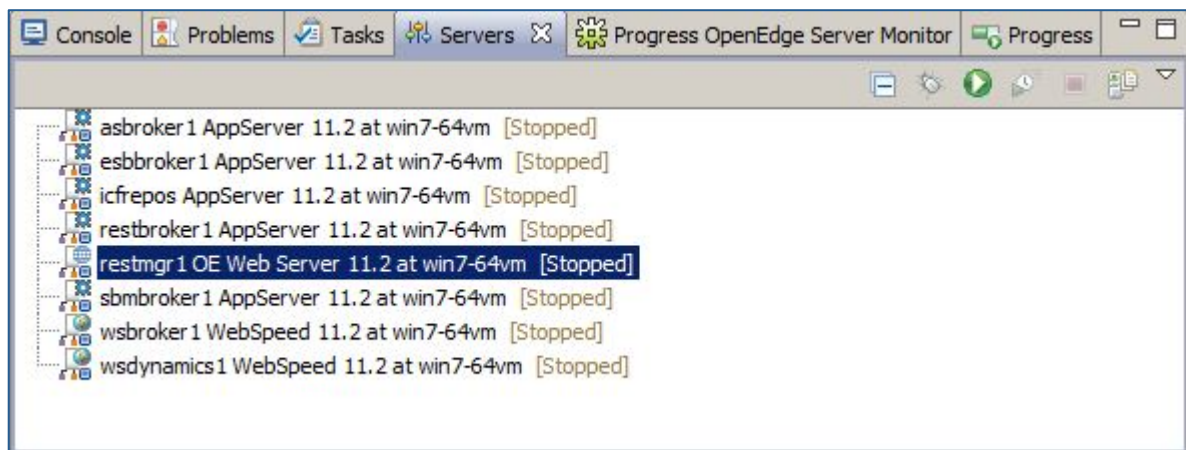
Now you are ready to begin.

## Starting restmgr1 and restbroker1

The restmgr1 and restbroker1 are preconfigured servers. Start the REST Manager, right-click on **restmgr1** and the **restbroker1**

▶▶ To start the restmgr1 and restbroker1:

1. **Right mouse click** on **restmgr1**, choose **Start**. Before moving on to the next step, be sure that the restmgr is in a Started, Synchronized state.
2. **Right mouse click** on **restbroker1**, choose **Start**. Before moving on to the next step, be sure that the restmgr is in a Started, Synchronized state.



## Creating a new Mobile OpenEdge project

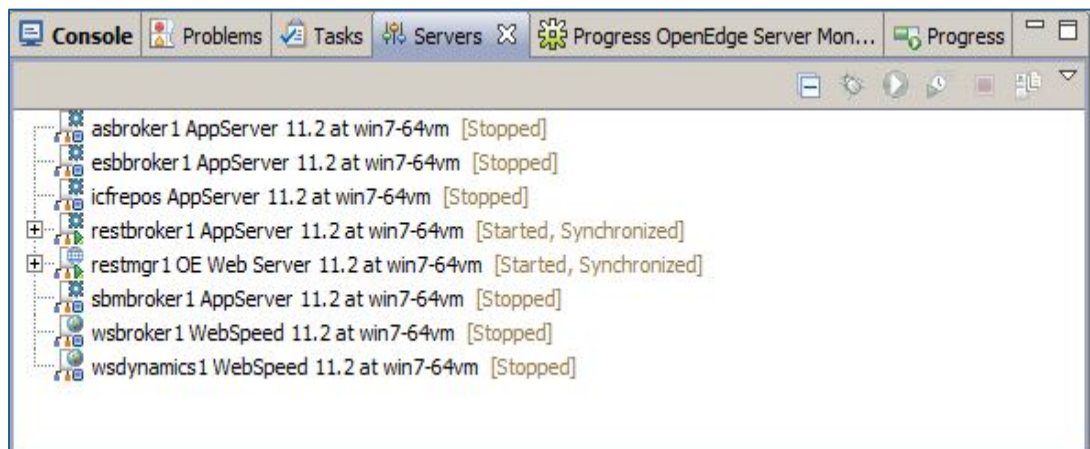
The first thing that you want to do is create a new OpenEdge project.

### To create a new Mobile project:

1. Select **File New, OpenEdge Project**. Name the project **Pug<N>>** and select **Mobile** from the drop-down menu in the Project type configuration. Click **Next**.
2. Click **Next** again to accept the defaults on the AVM and layout options page.
3. On the Define AppServer content module page, select **restbroker1** as your server. Click **Next**.
4. A Mobile service is created automatically. On the Create a Mobile Service page, select **restmgr1** as your server. Click **Next**.
5. On the Create a Mobile App page, select **restmgr1** as your server. Note that Developer Studio automatically appends "App" to your project name to create the name of your Mobile App. Click **Next**.
6. Click **Next** to accept the PROPATH defaults.
7. On this Arcade image, the Sports2000 database has been configured and started for you. Select the sports2000 database by clicking in the box next to sports2000
8. Click **Finish**.
9. Close the browser when prompted for username and password. We will launch the Mobile App Builder later.

### Important Note:

After this step, OpenEdge tries to publish your project. Make sure that your **restmgr1** and **restbroker1** are in a started and synchronized state. This could take up to two minutes. Your screen should look like the following screen shot. If it does not, please contact your instructor.

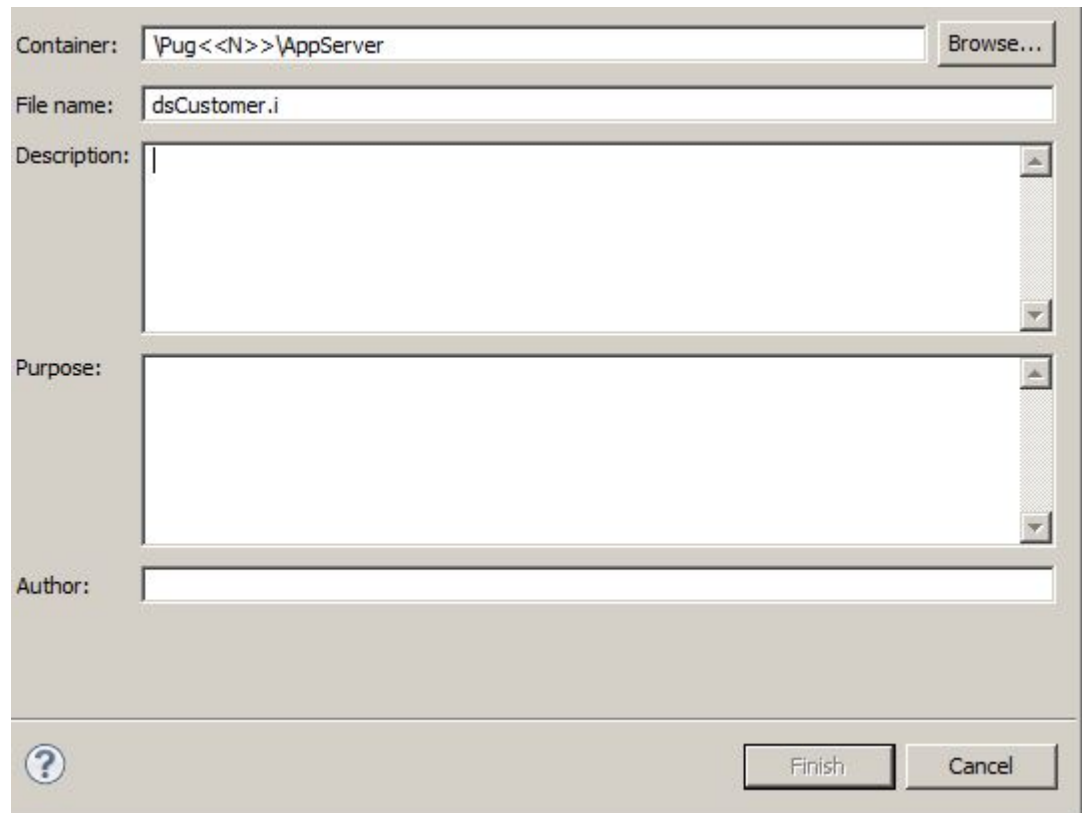


## Creating an include file

Before creating a new Business Entity for the Mobile service, you must create an include file that contains a temp table that describes the schema of the Sports2000 database.

### To create an Include file:

1. Put the include file in the AppServer folder, to do this, select the **AppServer** folder, then from the menu choose: **File, New, ABL Include**.
2. Select a container and name the file **dsCustomer.i**. Your New ABL Include should look like:



Container: \\Pug<<N>>\\AppServer Browse...

File name: dsCustomer.i

Description:

Purpose:

Author:

? Finish Cancel

3. Click **Finish**

In the Definitions section of the include file, you can enter &F1 to bring in the code or you can copy and paste the following code or copy the code from the pugchallenge folder file1.



```
DEFINE TEMP-TABLE eCustomer NO-UNDO BEFORE-TABLE beCustomer
  FIELD CustNum          AS INTEGER
  FIELD Name             AS CHARACTER
  FIELD Address          AS CHARACTER
  FIELD City             AS CHARACTER
  FIELD State            AS CHARACTER
  FIELD PostalCode       AS CHARACTER
  FIELD Country          AS CHARACTER
  FIELD Phone            AS CHARACTER
  FIELD TextRowID        AS CHARACTER
  INDEX CustNum IS UNIQUE PRIMARY CustNum
  INDEX Name NAME.

DEFINE DATASET dsCustomer FOR eCustomer.
```

4. Save the file: Select **File, Save**. You should see the dsCustomer.i in your AppServer folder.

## Creating a new Business Entity

A Business Entity is an ABL class (CLS) file that includes pre-defined Mobile service annotations.

### ►► To create a new Business Entity:

1. In the Project Explorer, Select the AppServer Folder (this is where the .cls file will be created) Select **File, New, Business Entity**.
2. Enter dsCustomer as the Business entity name
3. Click **Next**

Package root: \Pug<<N>>\AppServer Browse...

Package: Browse...

Business entity name: dsCustomer

Modifiers: ☐ Final ☐ Abstract ☐ Widget pool

Inherits: Browse...

Implements: Add... Remove

Specify the code elements to generate:

☐ Generate default constructor ☐ Generate destructor

☐ Generate super class constructors

☐ Add routine-level error handling

Specify the return value for generated methods:

☒ Throw a Not Implemented exception

☐ Return a default value

Description:

Purpose:

? < Back Next > Finish Cancel

4. In the **Schema file** field, browse to the location of the dsCustomer.ifile you created previously (should be in c:\workspace\Pug<<N>>\AppServer. From the schema displayed, choose **dsCustomer**.
5. Click **Finish**.

## Adding OpenEdge Business Logic to the Business Entity, dsCustomer.cls

When you first create the business entity, it contains stubs for the Create, Read, Update, and Delete (CRUD) operations. You need to add the business logic to do those operations.

### To add the logic to the Business Entity:

1. In the Business Entity source code, look for the **ReaddsCustomer** method. You need to add the business logic, you can do this by entering &F2 which will bring in the code, or you can copy and paste the following code, or browse to folder pug challenge file 2 and copy and paste the code. This reads records from the Sports 2000 database.

```
DEFINE VARIABLE QryPosition AS CHARACTER NO-UNDO.
DEFINE VARIABLE NameFilter AS CHARACTER NO-UNDO.
DEFINE VARIABLE iCount AS INTEGER NO-UNDO.

/* QryPosition|NameSearch */
ASSIGN iCount = NUM-ENTRIES(filter,"|").
IF iCount = 2 THEN
    ASSIGN QryPosition = ENTRY(1,filter,"|")
        NameFilter = ENTRY(2,filter,"|").

MESSAGE "QryPosition" ===== " QryPosition" VIEW-AS ALERT-BOX.
MESSAGE "NameFilter" ===== " NameFilter" VIEW-AS ALERT-BOX.

DEFINE DATA-SOURCE srcCustomer FOR Customer.
EMPTY TEMP-TABLE eCustomer.
BUFFER eCustomer:ATTACH-DATA-SOURCE(DATA-SOURCE srcCustomer:HANDLE).
BUFFER eCustomer:HANDLE:BATCH-SIZE = 10.

IF Namefilter <> "" THEN DO:
    filter = 'WHERE Customer.Name BEGINS "' + Namefilter + '" by Customer.CustNum'.
    DATA-SOURCE srcCustomer:FILL-WHERE-STRING = filter.
END.
ELSE IF QryPosition <> "" THEN DATA-SOURCE srcCustomer:RESTART-ROWID(1) = TO-
ROWID(QryPosition).

BUFFER eCustomer:SET-CALLBACK("AFTER-ROW-FILL", "BuildTxtRowid").
DATASET dsCustomer:FILL() NO-ERROR.
BUFFER eCustomer:DETACH-DATA-SOURCE().
RETURN.
```

2. This code is used to add the rowid in a text format to the temp-table. Go to the end of the code, and just before the "END CLASS" statement enter &F3 to insert the code. Or you can copy the following code or from your desktop, browse to pug challenge and copy and paste file 3 to just. The code segment is:

```
/*-----  
    Purpose:  Add fields to eCustomer Temp-table  
    Notes:  
    -----*/  
METHOD PUBLIC VOID BuildTxtRowid (INPUT DATASET dsCustomer):  
    ASSIGN eCustomer.TextRowID = STRING(ROWID(Customer)).  
END METHOD.
```

3. In the Business Entity source code, look for **updateddsCustomer**, you want to add some code to the stub. If you are doing the keystroke method, type &F4 to insert the code, or copy and paste the following code to the updateddsCustomer method. Or browse to folder pug challenge file 4 and copy and paste the code. This updates records from the Sports 2000 database.

```
DATASET dsCustomer:WRITE-JSON('file', 'c:\dsCustomer.txt', YES).  
    FOR EACH eCustomer:  
        FIND Customer WHERE Customer.CustNum EQ eCustomer.CustNum EXCLUSIVE-LOCK  
        NO-ERROR.  
        IF AVAILABLE Customer THEN BUFFER-COPY eCustomer TO Customer.  
    END.
```

4. Perform a syntax check: right-mouse click on the dsCustomer.cls file and choose **Syntax Check**. If it comes back with OK, then save the file, right-mouse click on dsCustomer.cls and choose **Save**.

## Create the Catalog File

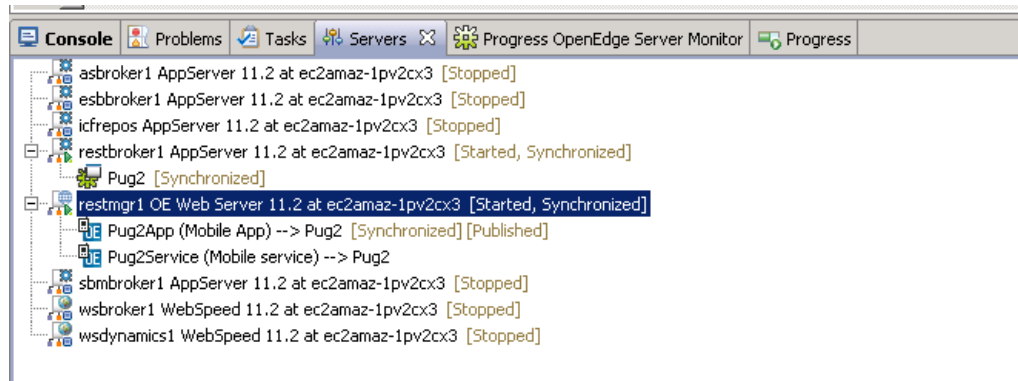
The Catalog file contains all the information about the services and the data for the mobile resource. The catalog file is in .json file format.

### To add the Business Entity to the Mobile Service:

1. In the Project Explorer pane, expand **Defined Services** and right-click **Pug<<N>>Service**. Click **Edit**.
2. Click **Next** on the Edit a Mobile Service page.
3. On the Edit Mobile Service page, select the **check box** next to the dsCustomer.clsfile. Click **Finish**.

#### Important Note:

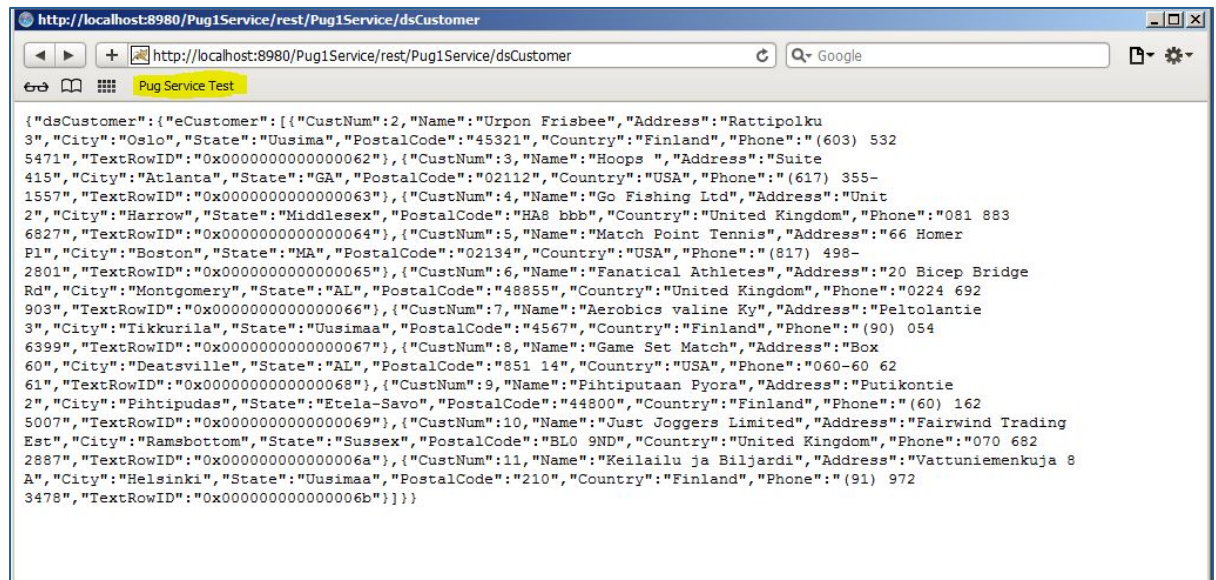
After this step, OpenEdge tries to publish your project. Make sure that your restmgr1 and restbroker1 are in a started and synchronized state. This could take up to two minutes. Your screen should look like the following screen shot. If it does not, please contact your instructor.



## Testing the OpenEdge Business Entity

▶ To test the OpenEdge Service:

1. Launch your browser, there is a bookmark that you will click on. This bookmark called **Pug Service Test**. Click on the bookmark to execute the REST Service. The result of executing this you should see a list of Customer records.



## Creating the Mobile App

You are now ready to use the Mobile App Builder to create the Mobile App. By default, the Mobile App Builder creates a home page that you can customize. You can open the Mobile App Builder at any time by double-clicking your application name in the Project Explorer pane.



### To launch the Mobile App Builder:

Bring up your Chrome browser and enter in the URL: **mobile.progress.com**

**Progress ID:** pug2013

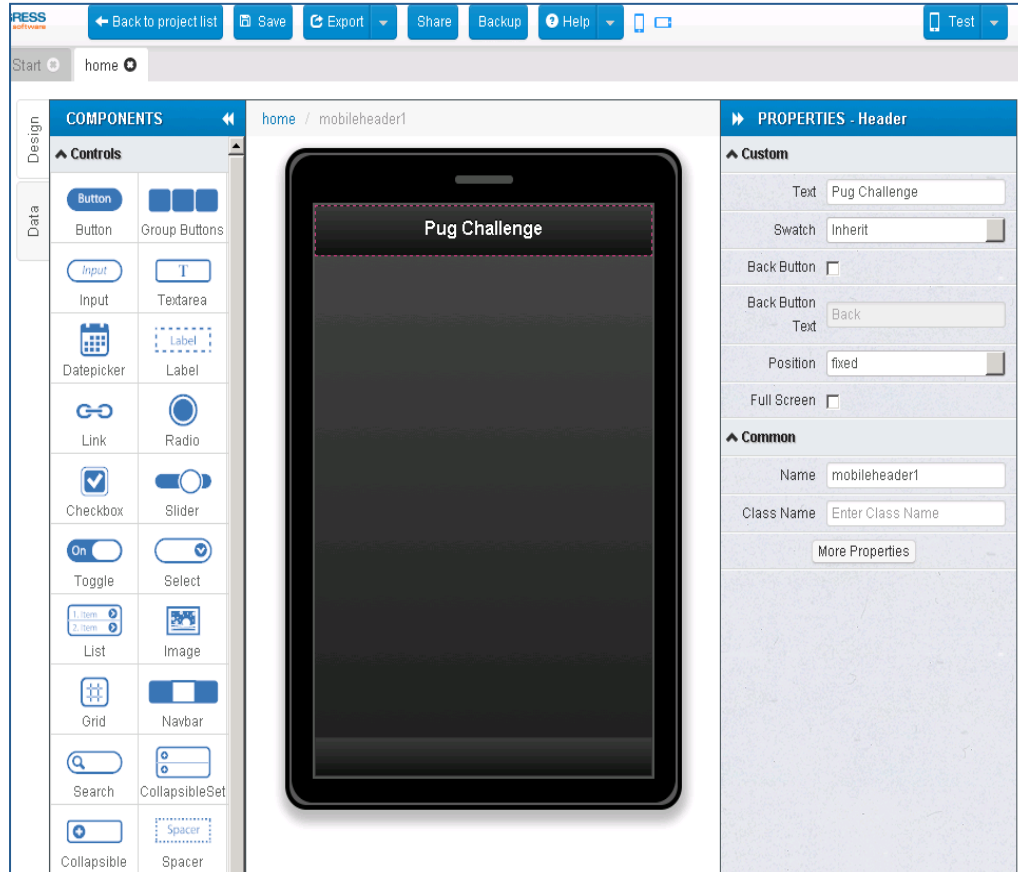
**Password:** pug2013

From the Dashboard, select **Apps**, and choose **Pug<<N>>App** and click the **Open** button

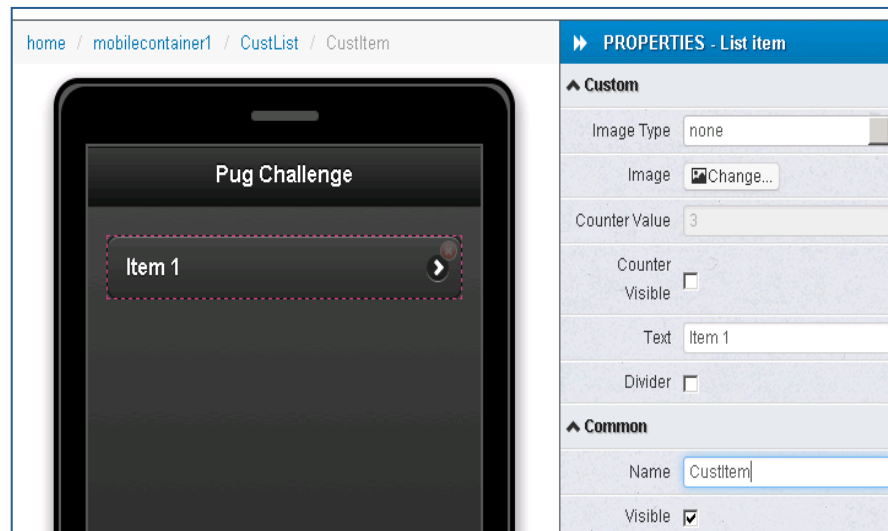


### To customize the client UI:

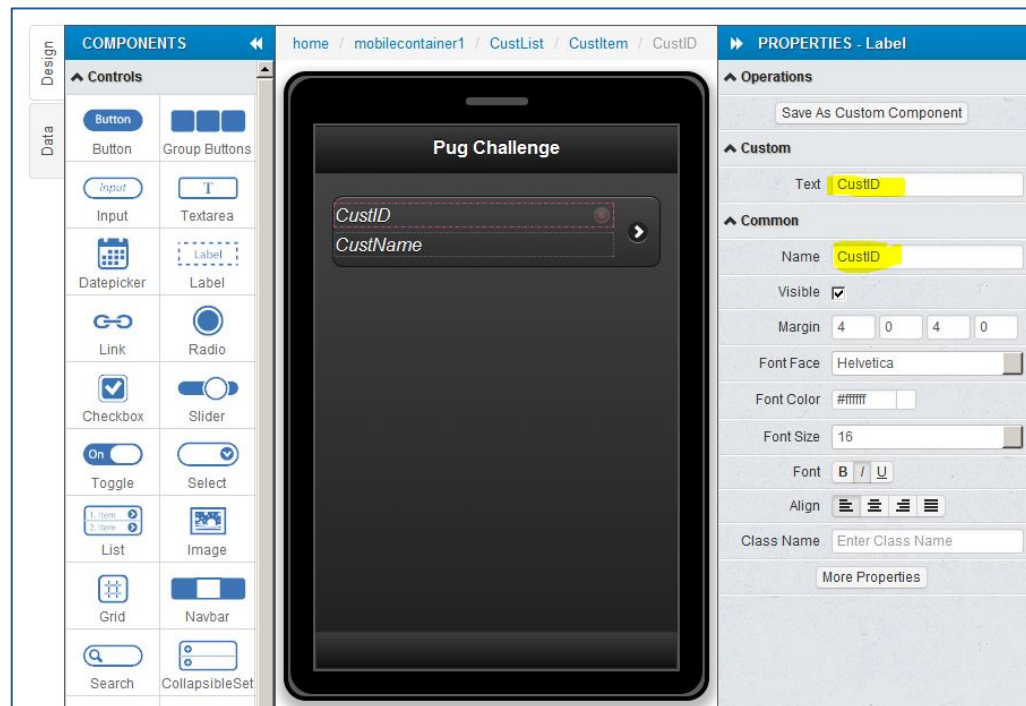
1. In the Project pane, expand the Pages folder and click the home page.
2. In the properties for **home page**, uncheck the property **Show Footer**
3. Click on the Caption and change the text properties to **Pug Challenge**.



4. Add a list to the page by dragging and dropping the **List Component** from the Components pane. In the Properties pane, change the **Name** field to **CustList** and change the **Items** field to 1.
5. Click on **Item 1** on the screen, this brings you the List Items. You will see in the properties List Item, change the name to be **CustItem**



6. Drag and drop 2 labels on top of the list item. In the properties for the label, Change the properties of the first labels as follows:  
 Label1: **Name** and **Text** of the label to **CustID**  
 Label2: **Name** and **Text** of the label to **CustName**



5. Click on the **Save** button





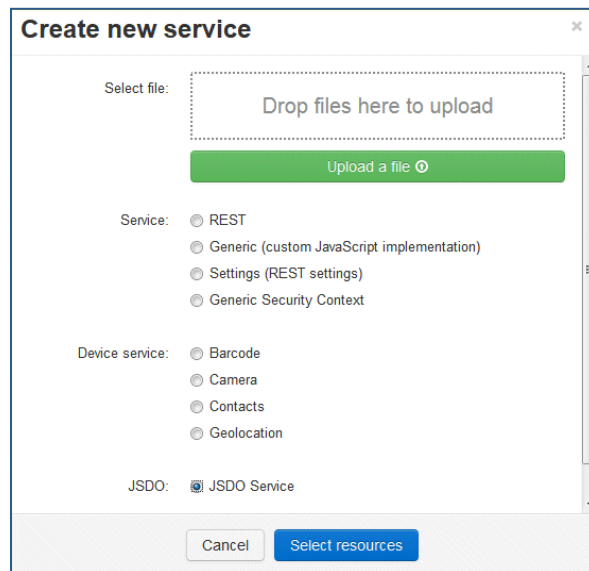
## Adding the JSDO service to the client

An OpenEdge JavaScript data object (JSDO) provides client access to ABL data. A JSDO service is a coding service in the Mobile App Builder that simplifies the mapping between JSDO data and the HTML elements of a Mobile App.



To add the JSDO service to the client:

1. In the Project pane, select **Create New Service**.
2. Select the **JSDO Service** radio button.



3. Click **Upload a file** and browse to the Pug<<N>>Service.json file. The file is located in **c:\workspace\Pug<<N>>\WebContent**.
4. Click **Select resources**.
5. Select **Pug<<N>>Service.dsCustomer** and click **Create services**. You should see that it created the services that were defined in the .json file
6. Click **Close**.
7. In the Project pane, under Services, expand Pug<<N>>Service.ds.Customer and click on **Pug<<N>>Service\_ds\_Customer\_Settings**.
8. You want to change the default values for the catalogURI and the serviceURI. If you are doing the key sequence, in the default value for catalogURI, enter &F5A. For serviceURI enter &F5B in the default value. Or you can look in the pug challenge folder on your desktop for a file 5 and open the file and copy the value of the catalogURI and the serviceURI from this file into the default values
9. Click on the **Save** button

## Adding JavaScript

In this example, you need to add JavaScript for the Initialization and for Batching.



### To add JavaScript in the Mobile AppBuilder:

1. Create the JavaScript Initialization code. Select **Create New, JavaScript**. Enter name: **JSDOInitialize**, click **Create JavaScript**. You need to add the JavaScript code, if you are doing the key sequence, enter &F6, or from your desktop from your pug challenge directory copy file 6 and paste the code, or you can copy and paste the code:.

**Important...if you copy this code, be sure to change N to your respective number.**

```
function JSDOInitialize() {
var settings;
var cMsg = "ok";
var pdsession;
try {
    /* CHANGE THIS TO POINT TO YOUR SETTINGS SERVICE */
    settings = Pug<<N>>>Service_dsCustomer_Settings;
    pdsession = new progress.data.Session();
    var loginResult =
        pdsession.login(settings.serviceURI, "", "");
    if (loginResult != progress.data.Session.LOGIN_SUCCESS) {
        console.log('ERROR: Login failed with code: ' + loginResult); switch
        (loginResult) {
            case progress.data.Session.LOGIN_AUTHENTICATION_FAILURE:
                cMsg = 'Invalid user-id or password';
                break;
            case progress.data.Session.LOGIN_GENERAL_FAILURE:
                default:
                cMsg = 'Service is unavailable';
                break;
        }
    }
} catch (e) {
    cMsg = "Failed to log in";
    console.log(e.stack);
}
if (cMsg != "ok") {
    alert(cMsg); return;
}
pdsession.addCatalog(settings.catalogURI);
}
```

2. Click **Save**

3. Create another JavaScript file, Select **Create New JavaScript**, Enter name: **CustomerBatching**, click **Create JavaScript**. You need to add the JavaScript code, if you are doing the key sequence, enter &F7, or from your desktop from your pug challenge directory copy file 7 and paste the code, or you can copy and paste the code:.

```
function Batching() {
    localStorage.setItem('CustRowid', "");
    var CurrentPage = 0;
    localStorage.setItem('CustomerCurrentPageTag', CurrentPage);
    var PageRowidArray = new Array();
    PageRowidArray[CurrentPage] = 'Empty';
    var PageRowidStr = JSON.stringify(PageRowidArray);
    localStorage.setItem('CustomerPageRowidTag', PageRowidStr);
}

function BtnPrev() {
    var PageRowid = localStorage.getItem('CustomerPageRowidTag');
    var CurrentPage = parseInt(localStorage.getItem('CustomerCurrentPageTag')) - 1;
    localStorage.setItem('CustomerCurrentPageTag', CurrentPage);
    var PageRowidArray = jQuery.parseJSON(PageRowid);
    localStorage.setItem('CustRowid', PageRowidArray[CurrentPage]);
    if (CurrentPage == 0) {
        Tiggzi("ButtonPrev").attr("disabled", true);
    };
    $.mobile.silentScroll(0);
}

function BtnNext() {
    var PageRowid = localStorage.getItem('CustomerPageRowidTag');
    var CurrentPage = parseInt(localStorage.getItem('CustomerCurrentPageTag')) + 1;
    localStorage.setItem('CustomerCurrentPageTag', CurrentPage);
    var PageRowidArray = jQuery.parseJSON(PageRowid);
    PageRowidArray[CurrentPage] = localStorage.getItem('CustRowid');
    var PageRowidStr = JSON.stringify(PageRowidArray);
    localStorage.setItem('CustomerPageRowidTag', PageRowidStr);
    if (PageRowidArray[CurrentPage] != "Empty") {
        Tiggzi("ButtonPrev").removeAttr("disabled");
    };
    $.mobile.silentScroll(0);
}
```

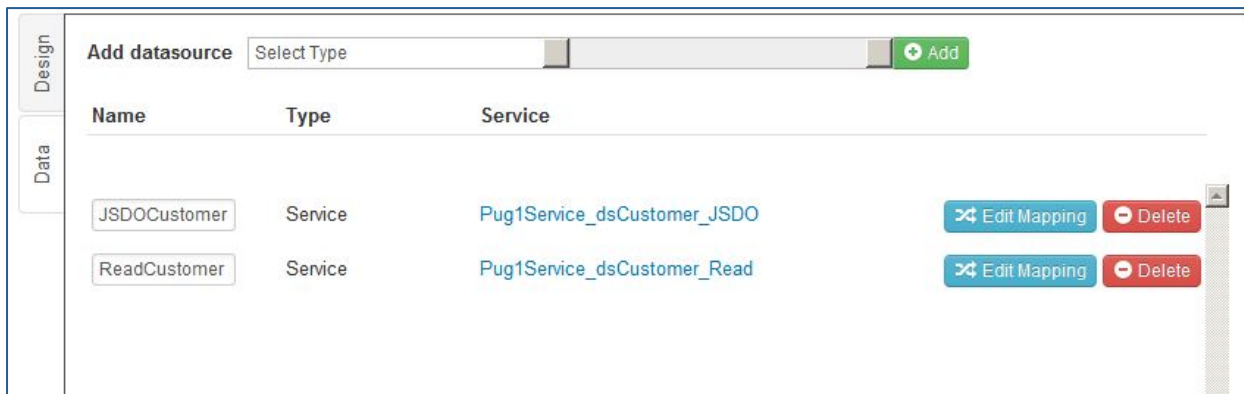
4. Click **Save**

## Adding datasources (services)

In this section, you are adding the catalog file and the operations that you want to access on this page.

▶▶ To add a datasource in the Mobile AppBuilder:

1. Click **home** in the component view to bring up the home page. Click on the **Data tab**, in the drop-down menu labeled **Add data source**, choose **Service**, and then choose **Pug<<N>>Service\_dsCustomer\_JSDO**. Click **Add** and name the service **JSDOCustomer**
2. Add another data source by choosing Service again. Choose **Pug<<N>>Service\_dsCustomer\_Read**. Click **Add** and name the service **ReadCustomer**
3. Click on the **Save** button  
(Note: In this screen shot, it states pug1, your project may be pug2, or pug3 and so on)



## Adding Events

Events happen in response to an action. In this section, you will add events to initialize your session, for batching and to access the operations in the JSDO.

▶▶ To add an Event in the Design tab of the Mobile AppBuilder:

- 1 Expand **Events** and add the events.

Want to execute the initialize and batching JavaScript: (In the JavaScript editor, you can enter &F8 to bring the code into the editor, or copy and paste from here)

Component: **home**

Event: **Load**





Action: **Run JavaScript**

In the editor box enter **JSDOInitialize(); Batching();**

Click **Add Event**

Component: **home**



Event: **Load**  
 Action: **Invoke Service**  
 Datasource: **JSDOCustomer**  
 Click **Add Event**

Component	Event	Order	Action	Details	Show All
Select Compor	Select Event		Select Action		+ Add event
home	Load	1. ↓	Run JavaScript	Handler: JSDOInitialize(); Batching();	 
home	Load	2. ↑	Invoke service	Datasource: JSDOCustomer	 

- Click on the **Data** tab, need to add an event

This is telling it that when the JSDOInstance has loaded successfully, then invoke the Read Service.

Component: **JSDOCustomer**  
 Event: **Success**  
 Action: **Invoke Service**  
 Datasource: **ReadCustomer**  
 Click **Add Event**

Component	Event	Order	Action	Details	Show All
Select Compor	Select Event		Select Action		+ Add event
JSDOCustomer	Success	1.	Invoke service	Datasource: ReadCustomer	 

- Click **Save**

## Add the mapping from the service to the client

Mapping enables you to map the fields from the datasource to the client UI components

▶▶ To do the mapping in the interface to the datasource:

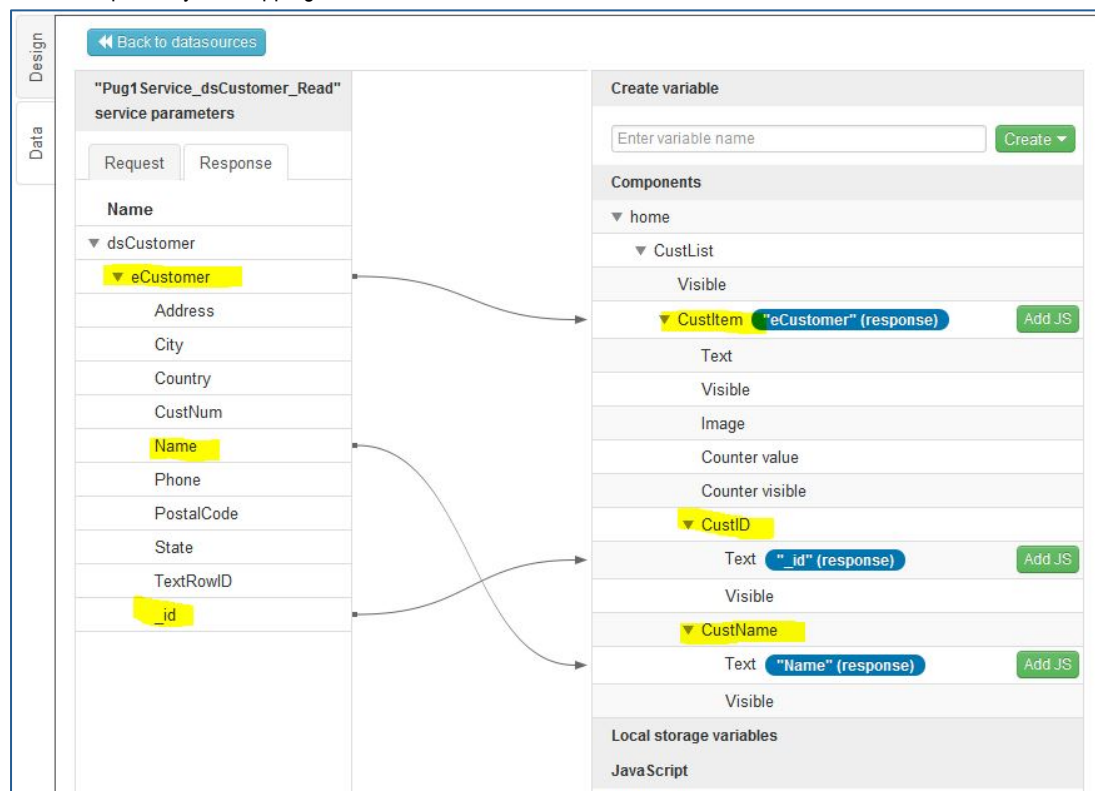
1. In the **Data tab**
2. Click **Edit mapping** next to the ReadCustomer service.
3. Select the **Response** tab.
4. Expand **dsCustomer** and all of its components on the left, and do the same for **home** and all of its components on the right.
5. Map as follows

**eCustomer -> CustItem**

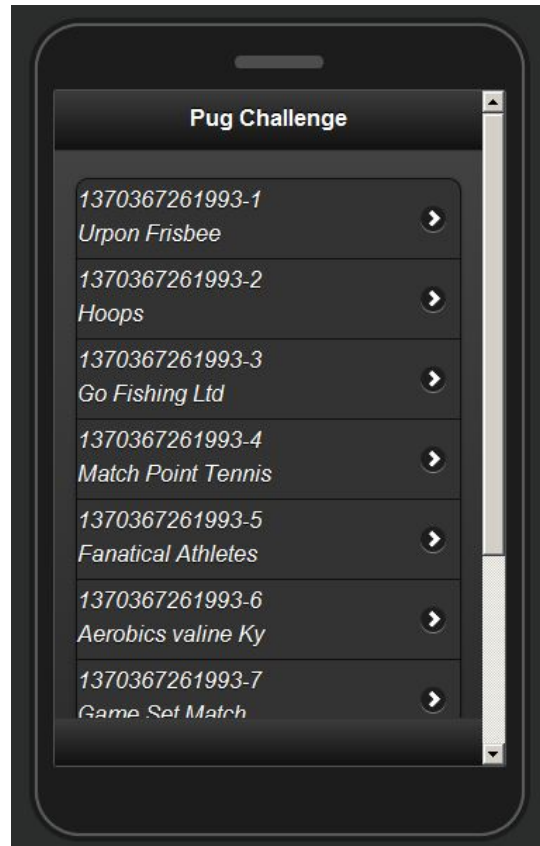
**Name -> CustName**

**\_id -> CustID**

When completed, your mapping should look like:

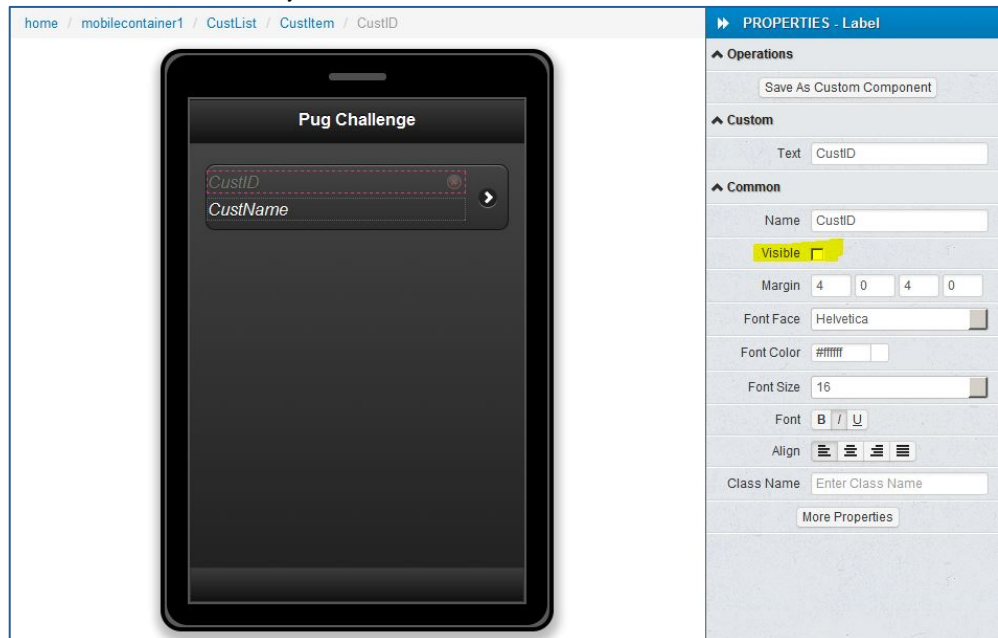


6. Click **Save** to save your project
7. Click **Test** to run your project. When you run the project, you should see a screen as follows:



8. You don't need to have the CustID, so go back to the **Design tab in the home page**, click on CustID and adjust the properties:

**Visible** take the check out so you won't see the field.



9. Click **Save**

## Add Batching and Searching to CustPage

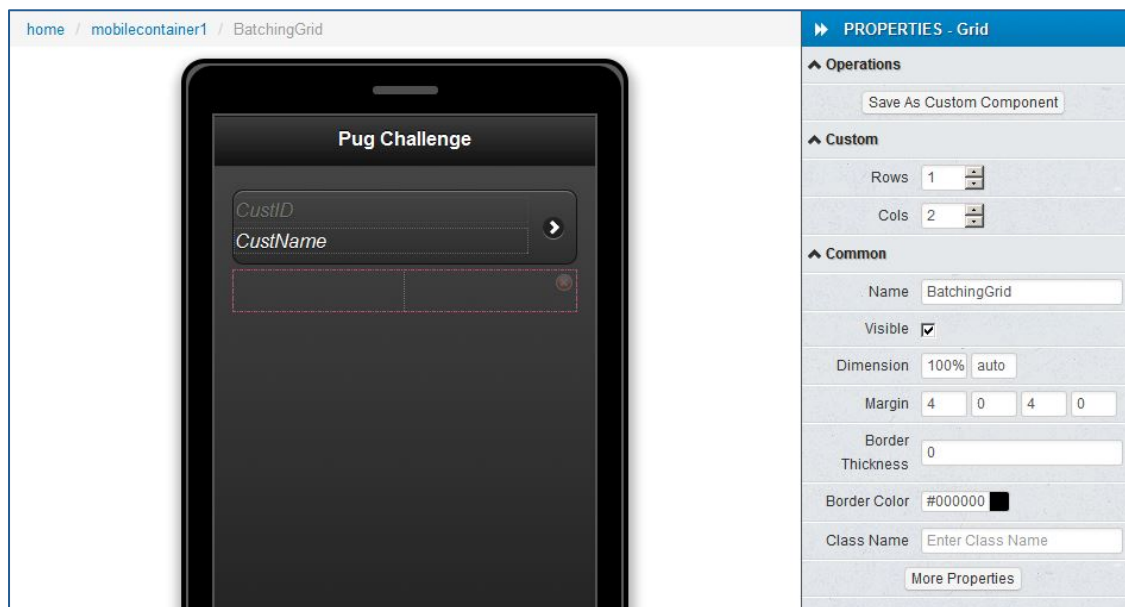
This will require that you add additional components to the CustPage. You will add a grid, with two buttons, Prev and Next and a second grid with the two columns, input and a button. Input so you can enter data and search on that data and a button so you can clear the text.

▶▶ To add input and button components to do the batching and searching:

1. From Components, choose **grid**, add it to the design canvas, put the grid under the **CustList** Component that you created earlier.
2. Change the **properties** of the grid:

**Name : BatchingGrid**

**Rows : 1**



3. Add a **button** in each cell in the grid, Adjust the properties as follows:

Button on the Left:

**Text: Prev**

**Icon: Left Arrow**

**Name: ButtonPrev**

Button on the Right:

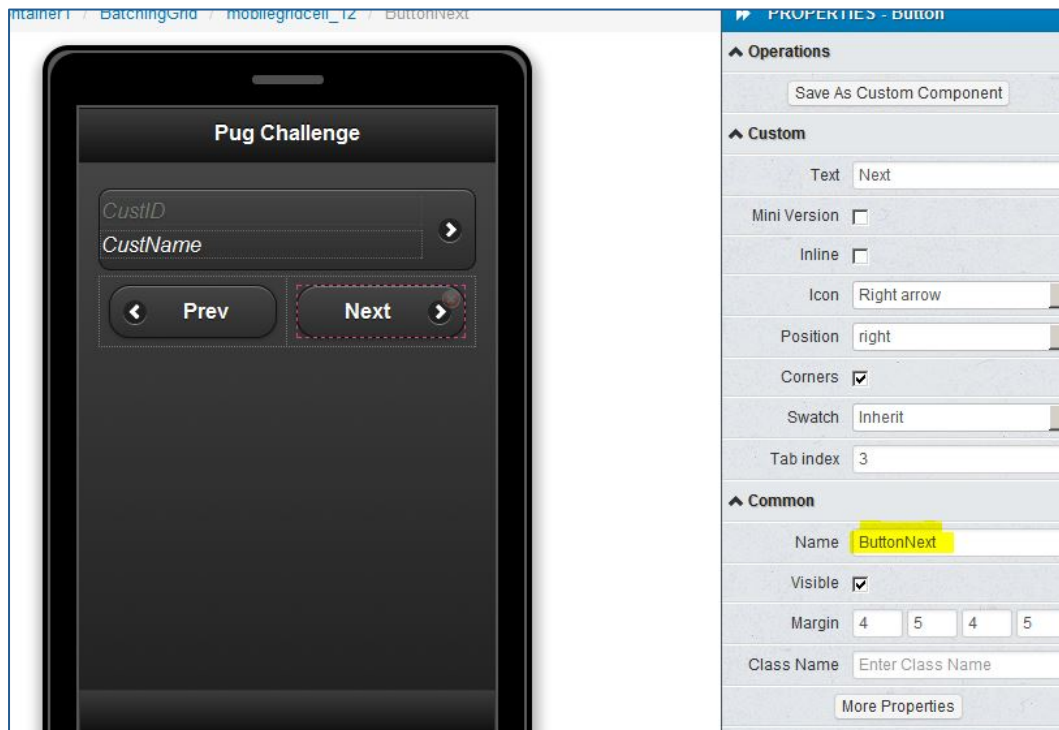
**Text: Next**

**Icon: Right Arrow**

**Position: right**

**Name: ButtonNext**



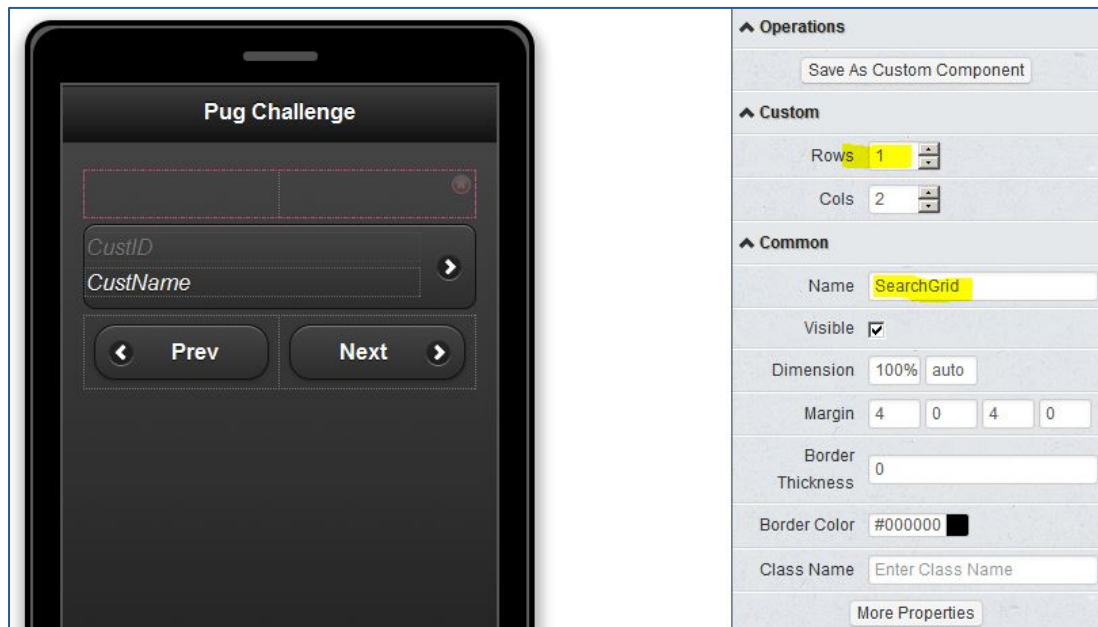


4. Add a **grid** component to the design canvas above the CustList component.

5. Change the **properties** of the grid:

**Name : SearchGrid**

**Rows : 1**



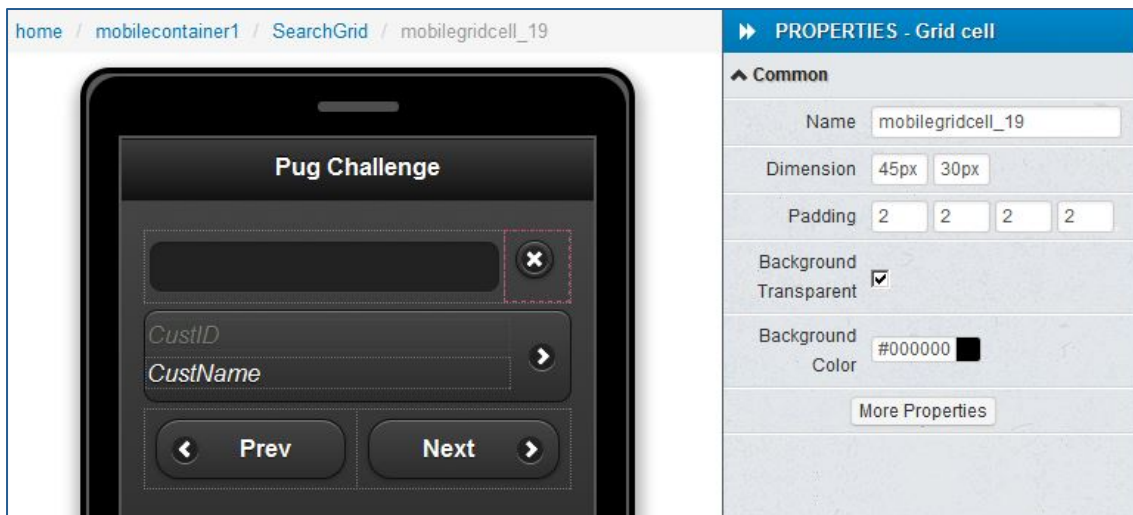
6. Select the **Input** component and put it in the cell on the left, and put a **button** in the cell on the right.

7. Change the properties of the input to be:

**Remove the word Input from the Text**

**Name: NameSearch**

8. Change the properties of the button to be:  
**Position: notext**  
**Icon: Delete**  
**Name: ButtonClear**
9. Select the first grid cell in the **SearchGrid**
10. Select the second grid cell in the **SearchGrid**
11. Change the properties of the grid cell to be:  
**Dimension: 45px 30px**
12. Click on the **Save** button



## Add Events to CustPage

You want events to fire in response to entering in values in the NameSearch and when you click on the delete button. This is done from the **Design tab**, **expand the Events**:

 To add the events to the CustPage:

1. Create an Event on **NameSearch**:  
 Component: **NameSearch**  
 Event: **Input**  
 Action: **Invoke service**  
 Details: **ReadCustomer**  
 Click **Add Event**
2. Create an Event for click of the **ButtonClear** button (if you are doing the key sequence, in the JavaScript editor, type &F9, or you can copy the code from the pug challenge folder, file 9 and paste in editor)  
 Component: **ButtonClear**  
 Event: **Click**  
 Action: **Run JavaScript**  
 Details: **Tiggzi("NameSearch").val(""); Batching();**  
 Click **Add Event**

**Important:** in the val("") those are two single quotes.

3. Create a second event for click of the **ButtonClear** button:  
Component: **ButtonClear**  
Event: **Click**  
Action: **Invoke Service**  
Details: **ReadCustomer**  
Click **Add Event**

Create Events for the Prev and Next Buttons:

1. Component: **ButtonPrev**  
Event: **Click**  
Action: **Run JavaScript**  
Details: **BtnPrev();**  
Click **Add Event**
2. Component: **ButtonPrev**  
Event: **Click**  
Action: **Invoke service**  
Details: **ReadCustomer**  
Click **Add Event**
3. Component: **ButtonNext**  
Event: **Click**  
Action: **Run JavaScript**  
Details: **BtnNext();**  
Click **Add Event**
4. Component: **ButtonNext**  
Event: **Click**  
Action: **Invoke service**  
Details: **ReadCustomer**  
Click **Add Event**
5. Click on the **Save** button

## You need to do some mapping for the NameSearch filter

This will enable you to filter based on the data that is being entered in NameSearch and get the Query Position for the batching so you know where to begin the batch of records.

 **To map the fields for the NameSearch:**

1. Click on the **Data** Tab
2. Select **Edit Mapping** next to the ReadCustomer Service
3. In the **Request** tab
4. You need to create a local variable: **CustRowid** On the right hand side, under Create variable, enter **CustRowid** and hit the **Create** button, **select local storage variable**.

5. Map **filter** to **NameSearch** by dragging and dropping NameSearch to filter
6. Click the **Add JS** button **next** to filter
7. This code will create QryPosition|NameSearch. If you are doing the key sequence, enter &F0 in the JavaScript editor. You can copy the code from the Pug Challenge folder, file 10. The function you want to add is:  
**return localStorage.getItem('CustRowid') + '|' + value;**
8. Click **Save and Return**
9. Click on the **Response** Tab
10. To get the query position for each record, you need to Map **TextRowID** to **CustRowid**
11. Click on **Save** to Save the Project
12. Click on the **Test** button to test the project
13. Enter in "**Lif**" in the NameSearch, and you should see the customers narrow down to that search. Hit the **Clear** button to get back the list, click on the **Next** button to get the next set of records.
14. Click on the **Save** button

### Create the CustDetail page from a Plug-in

Now that you know how to build a page, the next section enables you to focus on linking to the details page and using a GeoLocation Service. You will create the CustDetail from a Plug-in.

#### To create the CustDetail Plugin:

1. In the Mobile AppBuilder, Select **Create New** button
2. Select From **Plug-in**
3. Select **CustDetail Plug-in** by putting a clicking in the checkbox
4. Select the **Import selected plug-ins** button
5. Select Keep Current at the Appery plug-ins were installed successfully section, click on **Apply settings**
6. Click on the **Save** button

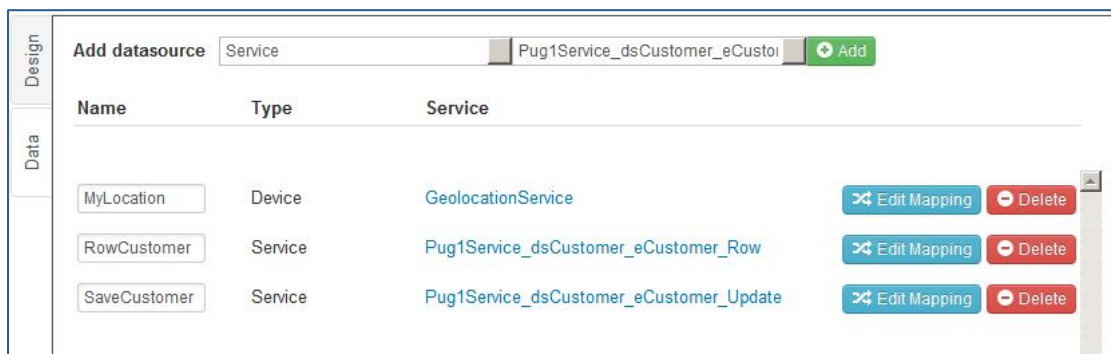
### Adding JSDO Services to the CustDetail Page

For the CustDetail page, you will be adding RowService. This will read a row from the database.

#### To add JSDO Services to the CustDetail:

1. Open the **CustDetail** Page
2. Click on the **Data** tab

3. You are going to add the Row Service. This service is reading data a single record from the customer dsCustomer that is stored in local cache.
4. Click on the drop down next to Add datasource
5. Choose **Service**
6. Click on the drop down to choose the service and choose **Pug<<N>>Service\_dsCustomer\_eCustomer\_Row**
7. Click the **Add** button
8. Change the name from restservice1 to **RowCustomer**.
9. Add a second datasource for updating, from Add datasource select **Service**
10. Select **Pug<<N>>\_dsCustomer\_eCustomer\_Update**
11. Click the **Add** Button
12. Change the name the Service from restservice1 **SaveCustomer**
13. Your screen should look like:



- 14 Click on the **Save** button

## Adding Events to the CustDetail Page

The event is so when the page appears, it will call the RowCustomer Service and display the current record in the cache in the CustDetail page.

To add events the CustDetail Plugin:

1. Open the CustDetail Page in the Design view by clicking on the **Design tab**, you need to add a couple of events. Expand the **Events tab**

Component: **CustDetail**





Event: **Page show**

Action: **Invoke Service**

Datasource: **RowCustomer**

Click the **Add Event** button

**Important Note:** Make sure that the order of events is that the RowCustomer is before the InitializeMap. If they are not in that sequence, click on the arrows to make them in that sequence.

Component	Event	Order	Action	Details	Show All
CustDetail	Page show		Select Action		+ Add event
CustDetail	Page show	1. ↓	Invoke service	Datasource: RowCustomer	 
CustDetail	Page show	2. ↑	Run JavaScript	Handler: InitializeMap();	 

2 Click on the **Save** button

## Mapping RowCustomer in the CustDetail Page

You need to map the records from the service to the page so that the records will be displayed on the page.

 To do the RowCustomer mapping on the CustDetail page

- 1 Go to the **CustDetails** page, click on the **Data** tab, you want to do some mapping from the **RowCustomer**
2. Select **Edit mapping** next to **RowCustomer**
3. You need to create a local storage variable, call it **LocalCustID**, click **Create Local storage variable**
4. In the **Request** tab, map **id** with **LocalCustID**
5. In the **Response** Tab, expand so you can see all fields  
map the following:

**Expand HeaderGrid**  
**CustNum ->CustomerNum**  
**Name ->CustomerName**

**Expand ContactCollapse, expand ContactCollapsible\_Content, ContactGrid**  
**Address -> CustomerAddress**  
**City -> CustomerCity**  
**Country -> CustomerCountry**  
**Phone -> CustomerPhone**  
**PostalCode -> CustomerPostalCode**  
**State -> CustomerState**



## 7. Click on **Save** button

## Mapping SaveCustomer in the CustDetail Page

You are mapping the records from the interface that you want to be saved to the database.

### ▶▶ To do the SaveCustomer mapping on the CustDetail page

- 1 Click on the **Back to DataSources** button
- 2 Select **Edit mapping** next to **SaveCustomer**
- 3 In the **Request** Tab, expand so you can see all fields  
map the following:

**Expand HeaderGrid**  
**CustomerNum ->CustNum**  
**CustomerName ->Name**

Expand ContactCollapse, expand ContactCollapsible\_Content, ContactGrid  
 CustomerAddress -> Address  
 CustomerCity -> City  
 CustomerCountry -> Country  
 CustomerPhone -> Phone  
 CustomerPostalCode ->PostalCode  
 CustomerState -> State  
 LocalCustID -> \_id



4 Click on the **Back to DataSources** button







## Creating Events on the home Page

You want to create two events to fire to enable you to navigate from home to the CustDetail page

 **To create the events on the home page**

1. Select the **home page**.
2. In the **Design** Tab
3. **Expand Events**
4. Create first event as follows:  
Component: **CustItem**  
Event: **Click**  
Action: **Set local storage variable**  
Details: Variable name: **LocalCustID**  
Bind to component **checked**  
Target Component: **CustID**  
Property name: **Text**  
Click **Add** event
5. Create second event as follows:  
Component: **CustItem**  
Event: **Click**  
Action: **Navigate to page**  
Details: Page: **CustDetail**  
Transition effect: **default**  
Click **Add** event

Your screen should look like this:

Component	Event	Order	Action	Details	Show All
CustItem	Click		Select Action		Add event
CustItem	Click	1. ↓	Set local storage variable	Variable name: LocalCustID, Target Component: Cust...	 
CustItem	Click	2. ↑	Navigate to page	Page: CustDetail, Transition effect: default, Reverse: f...	 

6. Click on the **Save** button

## Creating Events to populate map on the CustDetail Page

You want to create events on the CustDetail page that will enable you to populate the map and enable you to save changes when you update the customer contact information.

 **To create the events on the CustDetail page**

1. Select the **CustDetail** page.


2. Select the **Data** tab
3. Expand **Events**
4. You will add an event for the RowCustomer Service, in the **Data** tab. This will initialize the map  
Component: **RowCustomer**  
Event: **Success**  
Action: **Run JavaScript**  
Details: **PopulateMap();**  
Click **Add** event
5. Add another event to navigate to the home page when you save changes to contact information:  
Component: **SaveCustomer**  
Event: **Success**  
Action: **Navigate to page**  
Page **home**  
Transition effect: **default**  
Click **Add** event
6. Press the Show All button to see the events. Your screen should look like this:

Component	Event	Order	Action	Details	Show All
SaveCustomer	Success		Select Action		Add event
MyLocation	Success	1.	Run JavaScript	Handler: GetMyLocation()	
RowCustomer	Success	1.	Run JavaScript	Handler: PopulateMap();	
SaveCustomer	Success	1.	Navigate to page	Page: home, Transition effect: default, Reverse: false, ...	

7. Click **Save**

## Events to Save changes to customer contact information

This will enable you to modify the customer contact information and save the changes back to the OpenEdge Database.

 To create the events to save the changes to contact data on the CustDetail page

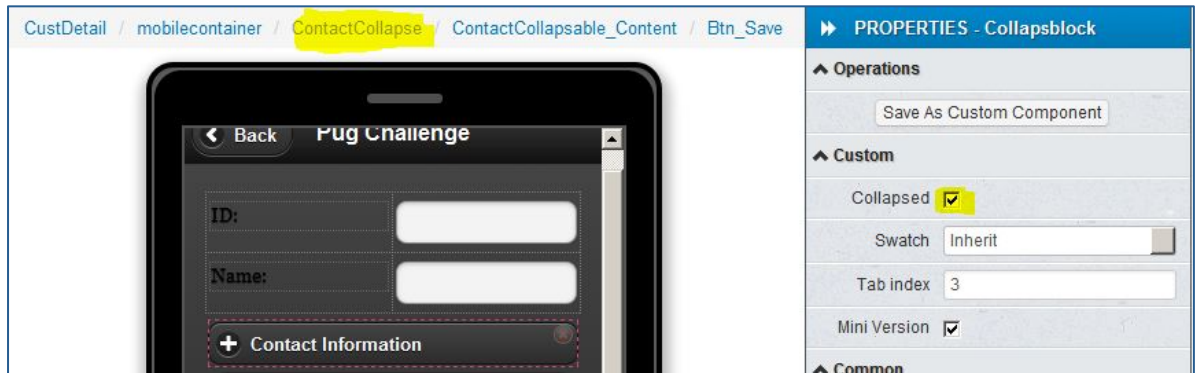
1. Click on the Design Tab
2. Click on the Save button
3. Review the existing Event for Btn\_Save. This event is JavaScript code asking for confirmation of saving the data.
4. Add a new Event invoke the update JSDO operation  
Component: **Btn\_Save**  
Event: **Click**  
Action: **Invoke service**  
Datasource: **SaveCustomer**  
Click **Add** event

## Modify a property of the Collapsible Control in the CustDetails page

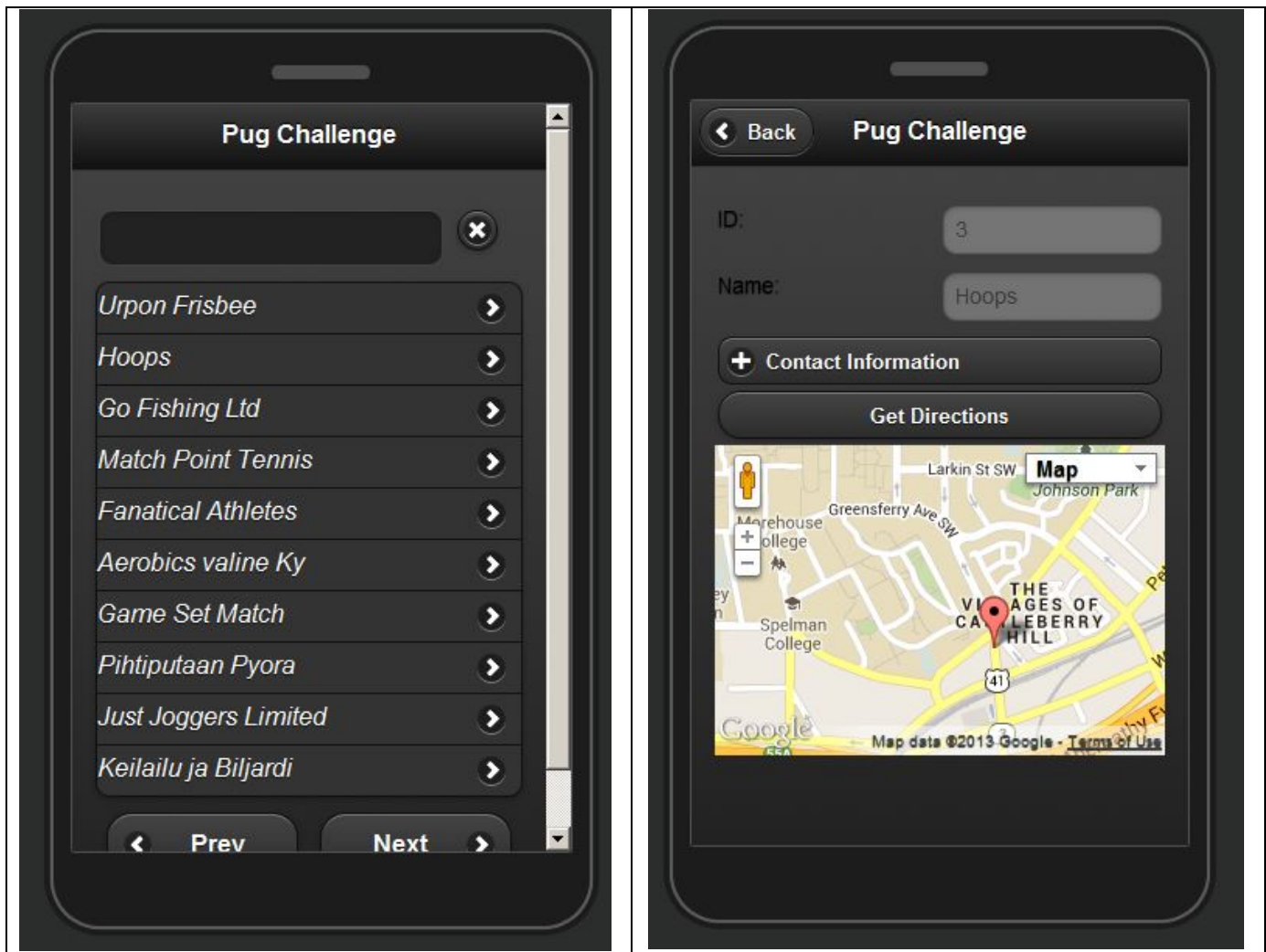
To let the user expand the Collapsible Control, you need to set a property. Right now the control is rendered expanded.

▶▶ To set the property of the Collapsible Control

1. In the Component bar across the top of the screen, select **ContactCollapsible**
2. In the properties, check Collapsed



3. Click the **Save** button
4. Click the **Test** button
5. Your application should look like the following: On the left, it shows a list of customers, on the right is when you have clicked on a customer on the left and you want to see the details. The screenshot on the right contains the details.



## GEOLOCATION DETAILS

The GeoLocation is already in the template, these notes will help you see how it was done.

### How to add the GeoLocation Service:

1. Click on the **Create New** button in the Mobile AppBuilder
2. Select **Service**
3. Select **Geolocation**
4. Click on the **Create new service** button

In the Project, there is some JavaScript called MapScripts, here is the JavaScript:

```
var DirectionsDisplay;
var DirectionsService = new google.maps.DirectionsService();

function InitializeMap() {
    DirectionsDisplay = new google.maps.DirectionsRenderer();
};

function displayDirections(sourceAddress, destinationAddress, map, panel) {
    var request = {
        origin: sourceAddress,
        destination: destinationAddress,
        travelMode: google.maps.DirectionsTravelMode.DRIVING
    };
    DirectionsService.route(request, function(response, status) {
        if (panel != null) {
            DirectionsDisplay.setPanel(panel);
        }
        DirectionsDisplay.setMap(map);

        if (status == google.maps.DirectionsStatus.OK) {
            DirectionsDisplay.setDirections(response);
        } else {
            alert("Directions query unsuccessful. Response status: " + status);
        }
    });
}

function PopulateMap()
{
    var destinationAddress;
    var destinationAddress = Tiggzi('CustomerAddress').val() + ', '
        + Tiggzi('CustomerCity').val() + ', '
        + Tiggzi('CustomerState').val() + ', '
        + Tiggzi('CustomerPostalCode').val() + ', '
        + Tiggzi('CustomerCountry').val();

    var map = Tiggzi("directionsMap");

    if (map.isInitialized)
    {
        if (destinationAddress != undefined)
        {
            map.options['address'] = destinationAddress;
            map.refresh();
        }
    }
    else
    {
        setTimeout(function(){PopulateMap()},3000);
    }
}
```

```

function GetMyLocation() {
    var mylat = localStorage.getItem("MyLatitudeTag");
    var mylog = localStorage.getItem("MyLongitudeTag");
    var mylocation = new google.maps.LatLng(mylat, mylog);

    var destinationAddress = Tiggzi('CustomerAddress').val() + ', '
        + Tiggzi('CustomerCity').val() + ', '
        + Tiggzi('CustomerState').val() + ', '
        + Tiggzi('CustomerPostalCode').val() + ', '
        + Tiggzi('CustomerCountry').val();

    var map = Tiggzi("directionsMap").gmap;
    var panel = Tiggzi("directionsPanel")[0];
    displayDirections(mylocation, destinationAddress, map, panel);
}

```

## Events on the CustDetail Page:

In the Design Tab, when you look at the Events, you see this event:

Component: CustDetail  
 Event: Page show  
 Action: Run JavaScript InitializeMap();

The JavaScript does an initialization with google maps

There is a button, Get Directions, the Event for the button is as follows:

Component: BtnDirections  
 Event: Click  
 Action: Invoke Service Datasource: MyLocation

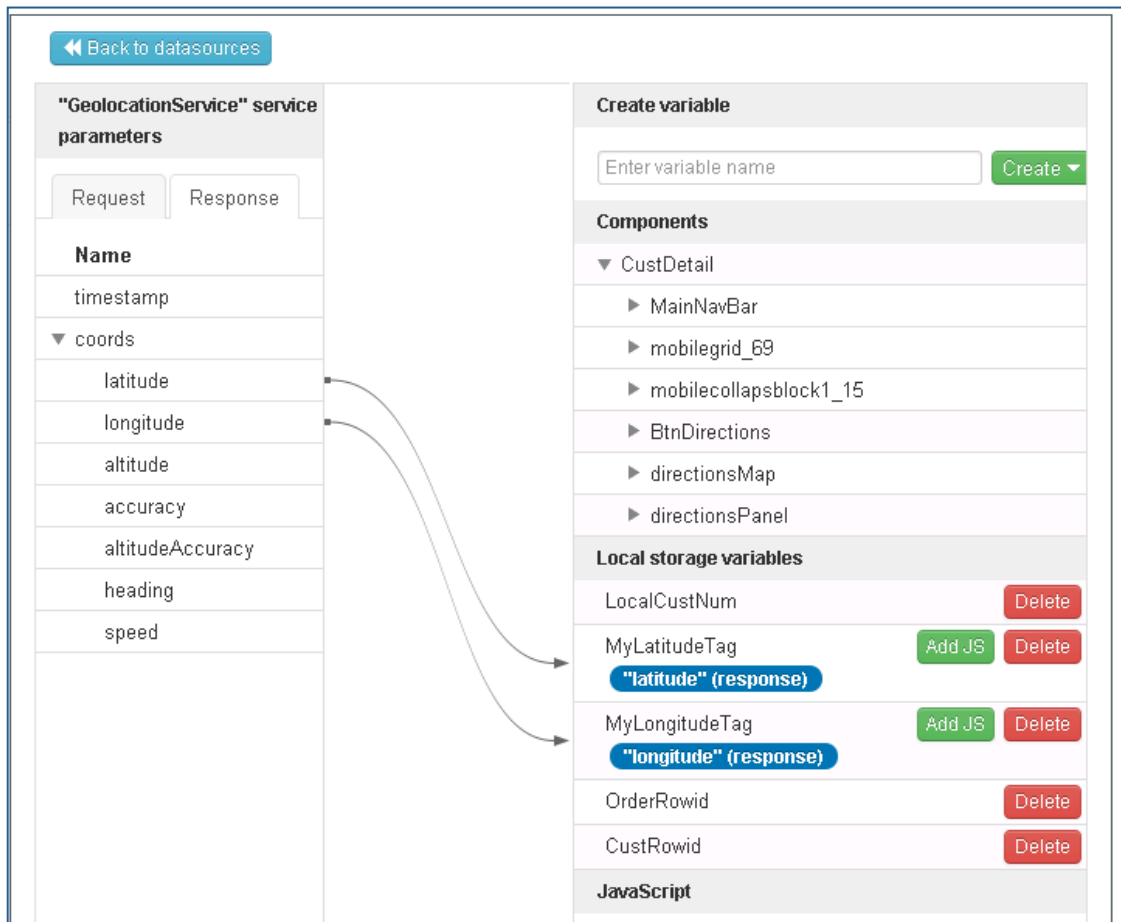
Datasource MyLocation is the GeoLocation Service. If you click on the Data tab, you can see the definition for MyLocation

MyLocation was defined by doing the following:

Click on the Create New button  
 Choose Service  
 Under Device service, choose Geolocation

Click on the Edit Mappings button in the Data tab to see the mappings for the service

Click on the Response Tab:



When the GeoLocation Service, MyLocation, is called, it will map the latitude and longitude with the local storage variables MyLatitudeTag and MyLongitudeTag. These values will be used in the JavaScript function GetMyLocation().

Click on the Data tab, you see these events:

Component: ReadService

Event: Success

Action: Run JavaScript PopulateMap()

The JavaScript gets the component values from the User Interface, and it maps the address based on those values in destinationAddress.

Component: MyLocation

Event: Success

Action: Run JavaScript GetMyLocation()

The JavaScript function GetMyLocation uses the local storage variables to determine current location in the map. The map will also use destinationAddress and draw the map between those points.

---

# DEPLOYING MOBILE APPLICATIONS

---

This chapter provides information on how to package up your app to see it on the Android or Apple device. This will use the same test/development Web Server that we have built the application on.

## Deployment overview

There are three options for packaging and deployment of a Mobile App:

iOS App

Android App

Table 3 lists the terminology used in this chapter to describe the considerations for deployment of different types of mobile applications.

## Packaging and Deploying iOS Apps

Packaging a Mobile App for iOS (IPA file) requires a certificate and a provisioning profile. These credentials are required during development as well as deployment.

---

**Note:** When deploying iOS Apps to different environments (development and testing vs. production deployment), different source code must be deployed. The URIs for logging into Mobile Web applications and accessing their services in each environment **must** be different absolute URIs. The JavaScript source for production deployment should have the required absolute URIs hard coded to log into all Mobile Web applications whose Mobile services are accessed by the iOS App.

---

## Packaging iOS Apps



**To build and package a Mobile App for iOS:**

1. In Progress Developer Studio for OpenEdge (Developer Studio), right-click on the project and choose **Properties, Progress OpenEdge, Mobile App Builder**.
2. On the Build options page, select **iOS**. Click **OK**.
3. Expand the **Mobile Apps** folder in the **Project** pane and double-click on the name of the Mobile App. The Mobile App Builder will open in a browser.
4. In the Mobile App Builder, expand **Project** and click **App settings**. Click the **iOS binary IPA properties** tab.



5. Enter the Mobile App name in the **Label** field.
6. Enter the Bundle ID in the **Bundle ID** field. The Bundle ID has to match what is on your Apple Development Account and needs to be a unique identifier at Apple. this ID will be linked to multiple aspects of your application.
7. The icon is the graphic that will launch the Mobile App on an iOS device. Upload an icon file using the **Browse** button in the **Icons** area. The graphs have to be in PNG format and match the resolution listed on the top of each image (ie: 57 pixels by 57 pixels etc). You should upload all of them, otherwise your application might have an icon on the iPhone, but not have one of the iPad.
8. Upload a launch image file for the application using the **Browse** button in the **Launch Images** area. On the absence of a graph for a specific resolution, a Tiggzi graph will be shown instead.
9. You are suppose to get the development certificate from the Apple IOS Development Website (.cer file), import that certificate into Mac-OS Key Chain (so only the place where you need a Mac machine), and then export that certificate from the Mac-OS Key Chain (.p12 file and associated password). To upload the certificate:
  - a. Click **Change** next to Certificate file.
  - b. Click **Upload file**.
  - c. Click **Upload a file** and locate the certificate. Once the upload is complete, click **Back to files list**.
  - d. Click on the certificate file and click **Select**.
10. Enter the certificate password. This is the password that you entered when you exported the certificate (.cer file) in the Mac-OS Key Chain Utility (.p12 file).
11. To upload the provisioning profile:
  - a. Click **Change** next to Provisioning profile file.
  - b. Click **Upload file**.
  - c. Click **Upload a file** and locate the provisioning profile. Once the upload is complete, click **Back to files list**.
  - d. Click on the provisioning profile file and click **Select**.
12. Click **Save**.

General	MKDirectionsApplicationSupportedModes	<div> <div>^</div> <div> MKDirectionsModeCar  MKDirectionsModeBus  MKDirectionsModeTrain  MKDirectionsModeSubway  MKDirectionsModeStreetCar  MKDirectionsModePlane  MKDirectionsModeBike  MKDirectionsModeFerry  MKDirectionsModeTaxi  MKDirectionsModePedestrian  MKDirectionsModeOther </div> <div>v</div> </div>
External resources	UIAppFonts	<div> <div>Enter comma-separated values</div> <div>v</div> </div>
Android binary	UIApplicationExitsOnSuspend	<div> <div>NO</div> <div>v</div> </div>
Android permissions	UIBackgroundModes	<div> <div> audio  location  voip  newsstand-content  external-accessory  bluetooth-central  bluetooth-peripheral </div> <div>v</div> </div>
iOS binary	UIDeviceFamily	<div> <div> 1  2 </div> <div>v</div> </div>
iOS keys	UIFileSharingEnabled	<div> <div>NO</div> <div>v</div> </div>
Push notifications	UIInterfaceOrientation	<div> <div>UIInterfaceOrientationPortrait</div> <div>v</div> </div>
	UILaunchImageFile	<div> <div>Default.png</div> <div>v</div> </div>
	UIMainStoryboardFile	<div> <div>Enter file name</div> <div>v</div> </div>
	UINewsstandApp	<div> <div>NO</div> <div>v</div> </div>
	UIPrerenderedIcon	<div> <div>NO</div> <div>v</div> </div>
	UIRequiredDeviceCapabilities	<div> <div> telephony  wifi  sms  still-camera  auto-focus-camera  front-facing-camera  camera-flash  video-camera  accelerometer  gyroscope  location-services  gps  magnetometer  gamekit  microphone  opengles-1  opengles-2  armv6  armv7  peer-peer  bluetooth-le </div> <div>v</div> </div>
	UIRequiresPersistentWiFi	<div> <div>NO</div> <div>v</div> </div>
	UIStatusBarHidden	<div> <div>YES</div> <div>v</div> </div>
	UIStatusBarStyle	<div> <div>UIStatusBarStyleDefault</div> <div>v</div> </div>
	UISupportedExternalAccessoryProtocols	<div> <div>Enter comma-separated values</div> <div>v</div> </div>
	UISupportedInterfaceOrientations	<div> <div>UIInterfaceOrientationPortrait</div> <div>v</div> </div>

In the Mobile AppBuilder, the above page is called iOS Keys and you need to specify what hardware capabilities your application will need permission for. The most basic ones are the `UISupportedInterfaceOrientations` which will allow your application to work in Portrait or Landscape mode, but your application might also use GPS, Camera, etc.

13. In Developer Studio, right-click on the name of the Mobile App in the Mobile Apps folder and choose **Copy source local** to get the updated Mobile App in the Developer Studio project.
14. To package the IPA, right-click on the name of the Mobile App in the Mobile Apps folder and choose **Export local**. The IPA file will in the bin directory.

You can test your app during development by copying the IPA directly to an Apple device. Your REST Catalog and REST Service have to use a URI that is publicly available on the Internet (ie. localhost or computer name would not work)



**To copy the IPA to an iOS device:**

1. Open iTunes on your computer.
2. Drag the IPA into the iTunes Library.
3. Plug the iOS device into the computer.
4. Drag the IPA from the iTunes Library onto the icon showing the device.

If you get an installation error, verify that the UUID of the device is registered correctly and that the provisioning profile includes that device.

## Deploying iOS Apps

Building a Mobile App for iOS (IPA file) requires registration as an Apple Developer, a developer certificate, and a provisioning profile. The sections below point you to information to help you create your own credentials. Note that a Mac is required for this. For more detailed information visit:

<https://developer.apple.com/devcenter/ios/index.action>

### Apple Developer

To register as an Apple Developer, visit this website:

<https://developer.apple.com/programs/register/> **iOS**

### Provisioning Portal

To create the certificate and provisioning profile, use the iOS Provisioning Portal. This portal is part of the iOS Dev Center. To access the portal, log in at the following link with the Apple ID you used to register as a developer:

<https://developer.apple.com/devcenter/ios/index.action>

Click on the iOS Provisioning Portal link.

### Certificates

The following video has more information on certificates:

<https://developer.apple.com/ios/manage/overview/index.action>

Note that a certificate must be in P12 format in order to be used in the Mobile Application Builder.

### App IDs

An App ID has three parts: Description, Bundle Seed ID (App ID Prefix), and Bundle Identifier (App ID Suffix). The Description of the App ID can be anything, e.g., OEMobile. The Bundle Seed ID (App ID Prefix) is assigned for you and is assigned to be your Team ID. The Bundle Identifier can be unique for each application or use a wildcard for many applications. Although it is possible to use "\*" by itself, the recommended practice is to use the following notation: `com.progresssoftware.*` or `com.progresssoftware.applicationname`.

It is important to remember what you entered for the Bundle Identifier. This will be used when creating the IPA in the Mobile App Builder. The following link describes Bundle Identifiers:

[http://developer.apple.com/library/ios/#DOCUMENTATION/FileManagement/Conceptual/understanding\\_utis/understand\\_utis\\_conc/understand\\_utis\\_conc.html](http://developer.apple.com/library/ios/#DOCUMENTATION/FileManagement/Conceptual/understanding_utis/understand_utis_conc/understand_utis_conc.html)

The following video has information about app IDs:

<https://developer.apple.com/ios/manage/overview/index.action>

## Devices

Each device must be registered. This is accomplished by using the device UUID. The UUID can be found by running a free UUID app on the device. The following video has more

information on devices:

<https://developer.apple.com/ios/manage/overview/index.action#>

## Provisioning Profiles

A Provisioning Profile has four parts: Profile Name, Certificate(s), App ID, and Device(s). The Profile Name can be anything, e.g., OEMobile. The following video has more information on provisioning:

<https://developer.apple.com/ios/manage/overview/index.action#>

The following link also has more information:

[http://developer.apple.com/library/ios/#technotes/tn2250/\\_index.html](http://developer.apple.com/library/ios/#technotes/tn2250/_index.html)

# Packaging and Deploying Android Apps

Packaging a Mobile App (APK file) for Android requires a signing key/certificate, but during development and testing, the signing key/certificate inputs are not required.

**Note:** When deploying Android Apps to different environments (development and testing vs. production deployment), different source code must be deployed. The URIs for logging into Mobile Web applications and accessing their services in each environment **must** be different absolute URIs. The JavaScript source for production deployment should have the required absolute URIs hard coded to log into all Mobile Web applications whose Mobile services are accessed by the Android App.

## Packaging Android Apps



To build and package a Mobile App for Android:

1. In Progress Developer Studio for OpenEdge (Developer Studio), right-click on the project and choose **Properties, Progress OpenEdge, Mobile App Builder**.
2. On the Build options page, select **Android**. Click **OK**.
3. To open the Mobile App within the Mobile App Builder, double-click on the name of the Mobile App. The Mobile App Builder will open in a browser.
4. From within the Mobile App Builder, choose **Project, App settings** and click **Android binary**

General	Label	MyMobileApp
External resources	Icon	<b>Name: none</b>
Android binary	Version code	1
Android permissions	Version name	1.1
iOS binary	Package name	com.tiggzi.project3952
iOS keys	Min. SDK version	Android 2.2.x
Push notifications	Target SDK version	Android 2.3.3 - 2.3.4
	Max. SDK version	Android 2.3.3 - 2.3.4
<b>Signing key/certificate</b>		
<input checked="" type="checkbox"/> Autogenerate		
	Keystore file	<b>Name: none</b>
	Key alias	gotiggr.project
	Key password	8Tl9hv
	Keystore password	e2lDrC

General	Label	MobileApp
External resources	Icon	Change... Name: none
Android binary APK properties	Version code	1
iOS binary IPA properties	Version name	1.1
Push notifications	Package name	com.tiggzi.project2167
	Min. SDK version	Android 2.1.x
	Target SDK version	Android 2.1.x
	Max. SDK version	Android 2.3.3 - 2.3.4
Signing key/certificate		
	<input checked="" type="checkbox"/> Autogenerate	
	Keystore file	Change... Name: none
	Key alias	gotiggr.project
	Key password	fAUlOW
	Keystore password	IBWw3j

5. Enter the Pug1 App name in the **Label** field.
6. The launcher icon is the visual representation of your app on the mobile device.  
To upload an icon:
  - a. Click **Change** next to **Icon**.
  - b. Click **Upload file**.
  - c. Click **Upload a file** and locate the icon file. Once the upload is complete, click **Back to images list**.
  - d. Click on the icon file and click **Select**.
7. Click **Save**.
8. In Developer Studio, right-click on the Mobile App in the Mobile Apps folder and choose **Copy source local** to get the updated Mobile App to the Developer Studio project.
15. To package the APK, right-click on the Mobile App in the Mobile Apps folder and choose **Export local**. The APK file will be in the bin directory
16. During development, there are several ways to get the newly created APK on an Android device for testing.
  1. Create an account on <http://dropbox.com>.
  2. Click and drag the APK into dropbox.
  3. Open <http://dropbox.com> from a browser on the mobile device and log in.
  4. Click on the APK from within the dropbox application. This will begin downloading the application onto the device.
  5. When the download is complete, click on the downloaded file on the mobile device. Follow the device directions for installing the app.



---

**Third party acknowledgements** — See the "Third party acknowledgements" section on page 17.

February 2013  
Last updated with new content: Release 11.2.0  
Product Code: 4496; R11.2.0

For the latest documentation updates see OpenEdge Product Documentation on PSDN (<http://communities.progress.com/pcom/docs/DOC-16074>).

**Progress Software**

Progress Software Corporation (NASDAQ: PRGS) is a global software company that simplifies the development, deployment and management of business applications on premise or on any Cloud, on any platform and on any device with minimal IT complexity and low total cost of ownership.

**Worldwide Headquarters**

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA  
Tel: +1 781 280-4000 Fax: +1 781 280-4095 On the Web at: [www.progress.com](http://www.progress.com)

For regional international office locations and contact information, please refer to the Web page below: [www.progress.com/worldwide](http://www.progress.com/worldwide)

Progress and OpenEdge are trademarks or registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and other countries. Any other marks contained herein may be trademarks of their respective owners. Specifications subject to change without notice.

© 2013 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.