# Spark Quick-Start Guide

Building back-end services quickly with the Spark toolkit

March 2018

# Summary

The **Spark** toolkit can assist in modernizing your legacy application when creating a new and modernized solution through use of various products from Progress. Modern application components such as the **Progress Application Server** (PAS) can be used for exposing web-based API’s for accessing data services written in ABL. The power and flexibility of this technology makes it an open-ended frontier for expanding your application’s reach, which can be both a blessing and burden depending on your experience level.

The job of the Spark toolkit is to guide ABL developers (old and new) down a known and viable path. This is a collection of ABL code and practices that work in concert with the PAS product to build a secure, productive, and extensible application. Though most of all, every aspect of the included demos and codebase follows a best-practices approach based on decades of modernization strategies from the Professional Services group as well as continual feedback from Progress Engineering on how best to implement the available technologies in OpenEdge.

The Spark libraries are broken up into 3 distinct repositories to reflect the appropriate audience meant to consume the technology. For those wishing to get started immediately with coding, the “**spark-demo**” repository provides sample projects which can be deployed to a PAS instance with just a few steps. For advanced ABL developers wishing to contribute back to the project, the “**spark-server**” repository will be the avenue for making or requesting changes to the codebase. Similarly, for experienced front-end developers the “**spark-client**” repository will accept contributions for improving KendoUI integrations and customized plugins for the JSDO. **For the purpose of this guide we will be focused on the quick-start approach offered by the “spark-demo” projects.**

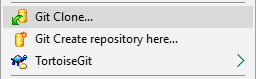
# Prerequisites

To provide the best experience with the **Progress Application Server (PAS)** and **Progress Developer Studio (PDSOE)** it is recommended that you be on the latest service pack of OpenEdge. The demos provided with Spark are compatible with both OpenEdge 11.6 and 11.7, with the latter being preferable due to significant simplifications of security options and improved support for Single Sign-On and OAuth. Use of a **64-bit Windows** installation and **OE 11.7.2 or later** is assumed, and at least the Progress Developer Studio for OE component present.

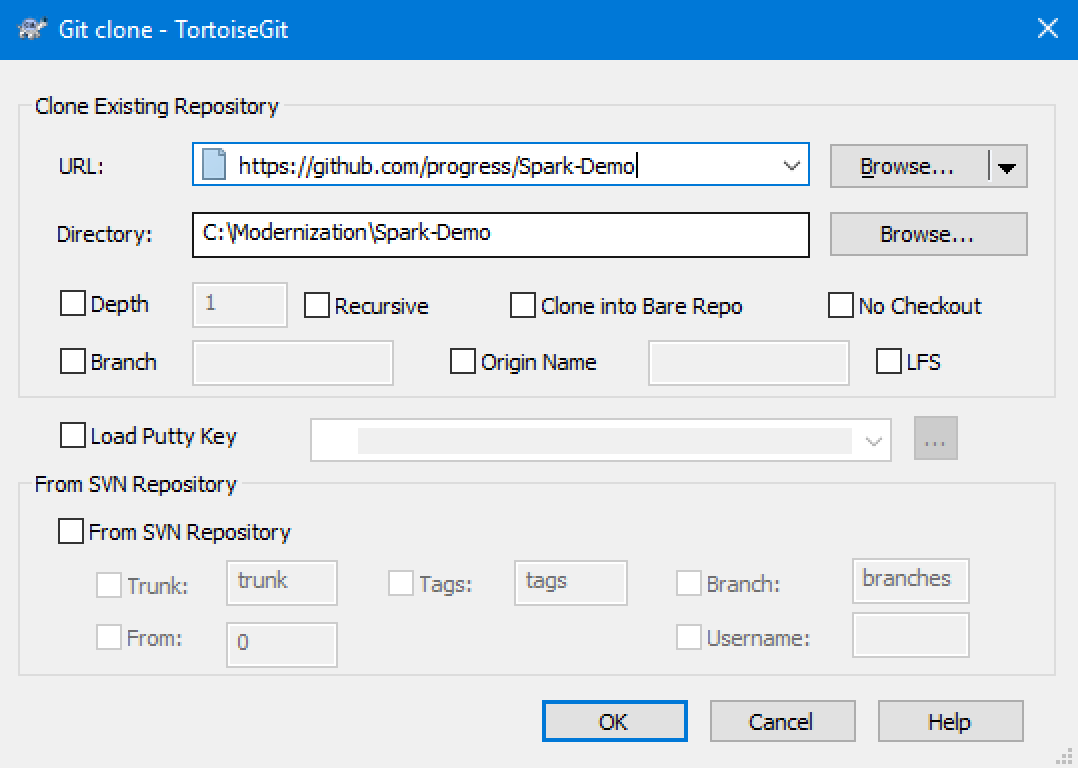
Access to the repositories is provided through **Git** and you may benefit from having a suitable Git client installed on your workstation. As a suggestion, **Git for Windows** and **TortoiseGit** will provide a seamless integration with Window Explorer. Some automated tasks will be performed using **Ant** which will already be present in your DLC directory if using 11.7 or later.

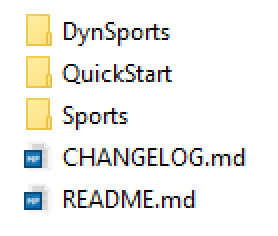
# Obtaining Code

1. Create a directory **C:\Modernization** for all future application code.
2. **Option 1:** Right-click within the new directory to view TortoiseGit options.

****

* 1. Perform a “**Git Clone**” of [**https://github.com/progress/Spark-Demo**](https://github.com/progress/Spark-Demo)



1. **Option 2:** Visit <https://github.com/progress/Spark-Demo/releases> and download the latest available release as either a .ZIP or .TAR.GZ archive (~90MB).
   1. Expand the archive, making sure the top-level directory is named simply “**Spark-Demo**” and contains an immediate “**oe117**” folder within.
2. Confirm the demo code is available by locating the “**oe117**” directory contents:

# Workspace Options

Before proceeding, it may be useful to enable some options within the Progress Developer Studio for OpenEdge (Eclipse) environment. These options will provide a more consistent experience with the actions to be requested in the remainder of this document. Begin by starting the **Developer Studio** and selecting **C:\Modernization** as your workspace location. If PDSOE has already been started under a different workspace, use the option **File > Switch Workspace > Other…** to choose.

Window -> Preferences

General

Show heap status: checked

Editors

Text Editors

Insert spaces for tabs: checked

Displayed Tab Width: 4

Show line numbers: checked

Search

Reuse editors: unchecked

Workspace

Refresh using native hooks or polling: checked

Refresh on access: checked

Workspace name: "Your Workspace Name Here"

LocalHistory

Maximum entries per file: 1

Progress OpenEdge

Editor

Case: Lower

Expand keywords: checked

Case keywords: checked

Build -> Automatically syntax check: checked

Server

Remove all files and folders when cleaning server pub dir: checked

Update properties from server before starting/launching: checked



Project Explorer -> View Menu (small icon on panel, shown above)

Customize View

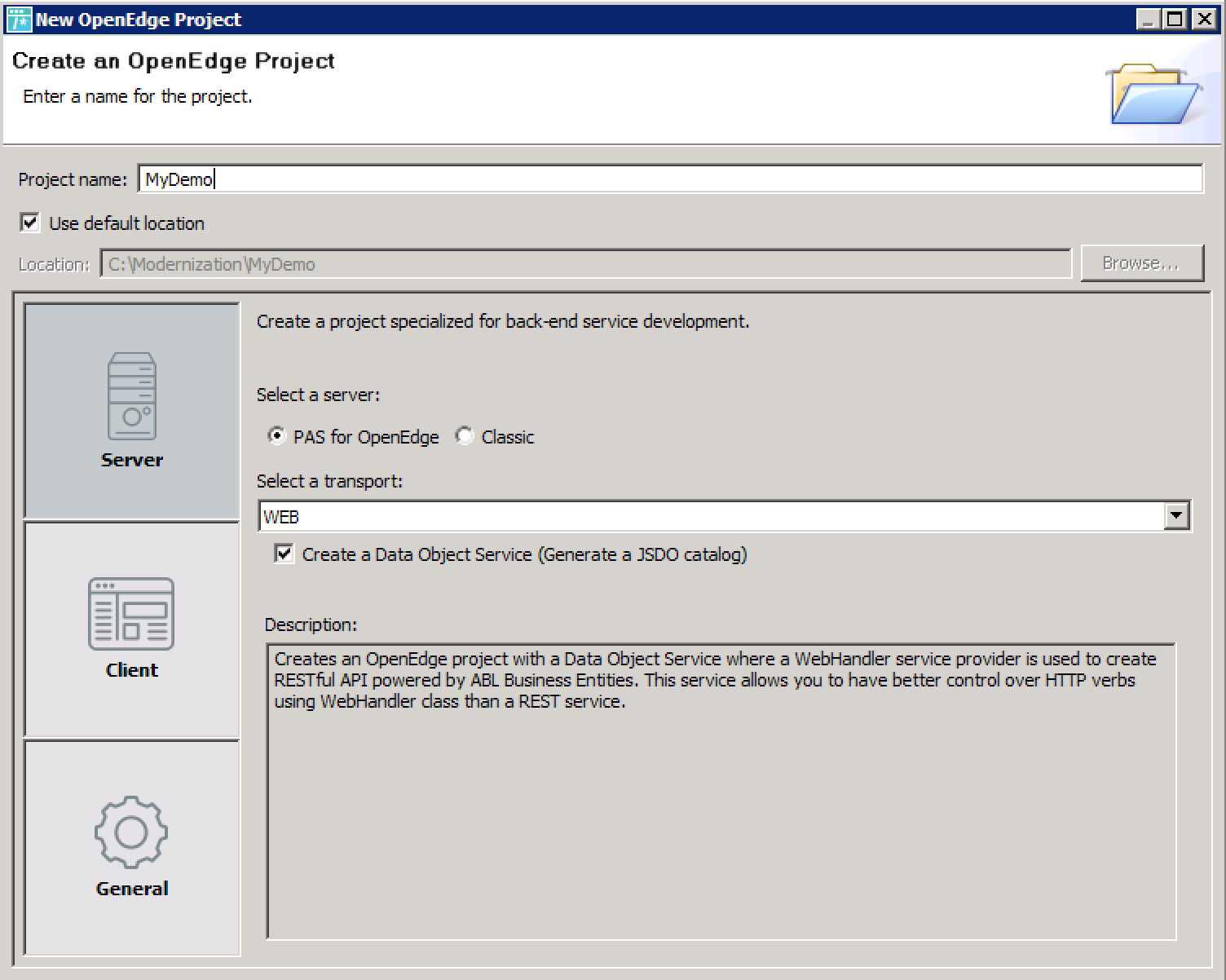
Select the filters to apply (matching items will be hidden)

Unselect \*.pl to view Procedure Library files.

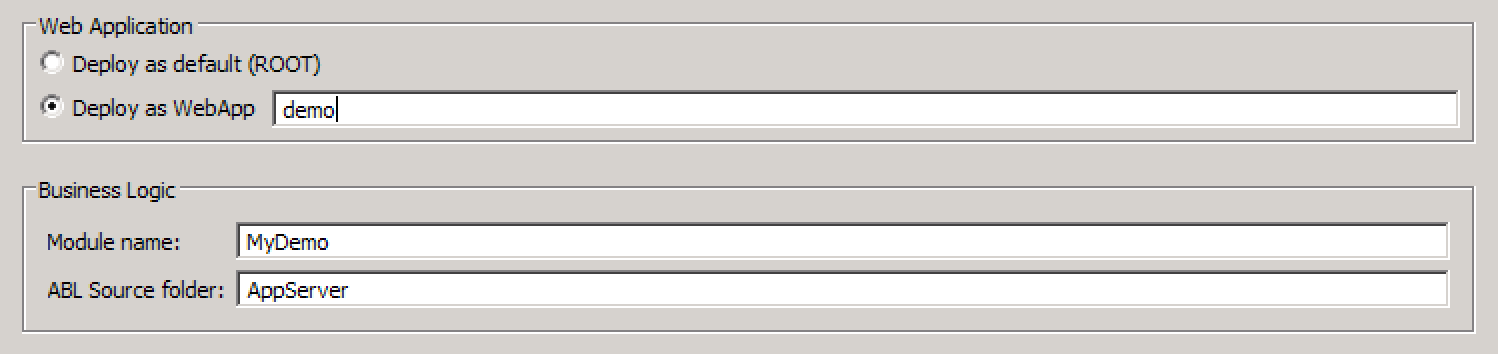
# Demo Project Creation

Creating a ABL Web App project in Progress Developer Studio has few major changes from the standard pattern. In fact, the process is slightly streamlined due to some post-configuration that will be performed later.

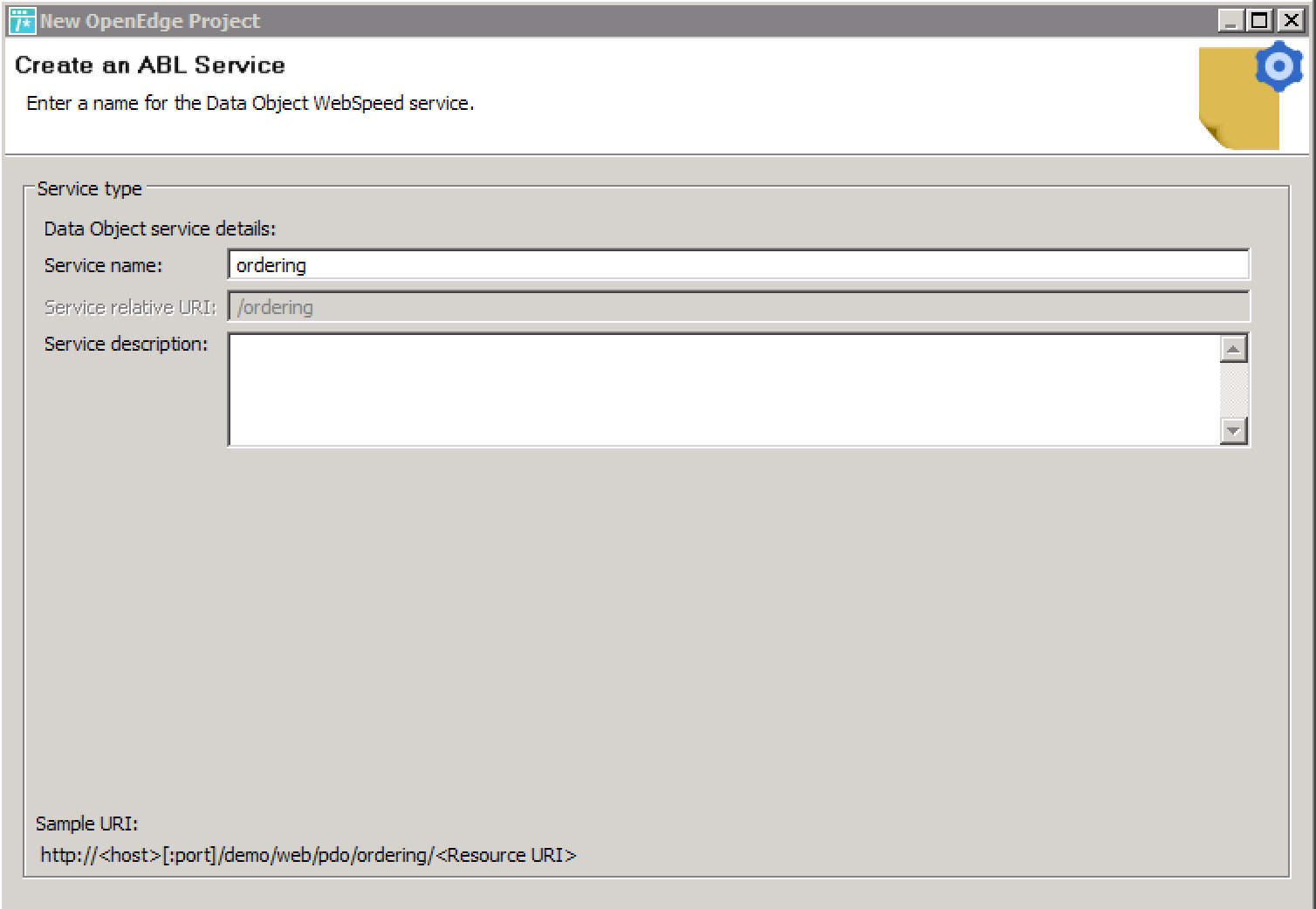
1. Start the Developer Studio, selecting **C:\Modernization** as your workspace.
   1. This is required on the first start-up of PDSOE, otherwise you must select the option **File > Switch Workspace > Other…** to choose.
2. Select **File > New > OpenEdge Project** from the menu at the top.
3. For the Project Name, let’s use “**MyDemo**” as a simple example.
4. Select the “**Server**” option with type “**PAS for OpenEdge**”.
5. Select the “**WEB**” transport and check the box to create a **Data Object Service**.



1. Press the **Next** button, and with **Deploy as WebApp** selected, rename the deployed application to “**demo**” (this is how we can control the WebApp’s URL).
2. If any PAS servers are pre-selected, unselect them and press **Next**.



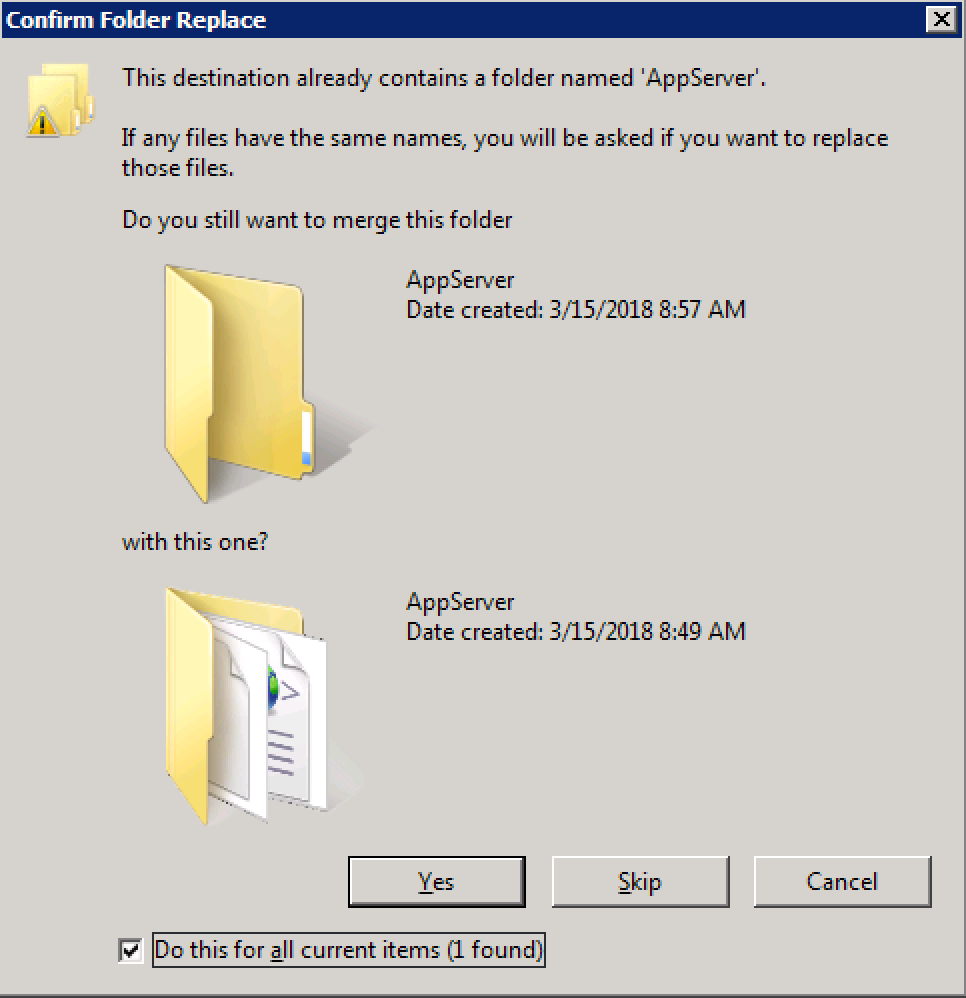
1. For our initial Data Object service let’s call it “**ordering**”, noting that the change in service name is reflected in the relative URI in the disabled field below.
2. Now it should be possible to press **Finish** as we will keep the defaults for the AVM, project layout, PROPATH, and databases. We will manage these next.



# Demo Project Configuration

Adding support for the Spark toolkit has been made easy thanks to a set of files that will augment our new project. This will provide the additional Procedure Library files and adjustments to your PROPATH. **It is important that these steps be performed prior to creating your own project code or altering the PROPATH for your specific application needs.** Repeat these steps for **any** project in the future where the Spark toolkit can assist you.

1. **Close** the Progress Developer Studio.
2. For the following steps use the **Windows Explorer** to access files:
   1. Navigate to **C:\Modernization\Spark-Demo\oe117\QuickStart**
   2. Select ALL of the contents of this folder and press **Ctrl-C** to copy.
   3. Navigate to your new project at **C:\Modernization\MyDemo**
   4. Press **Ctrl+V** to paste, overwriting/replacing any files if prompted.



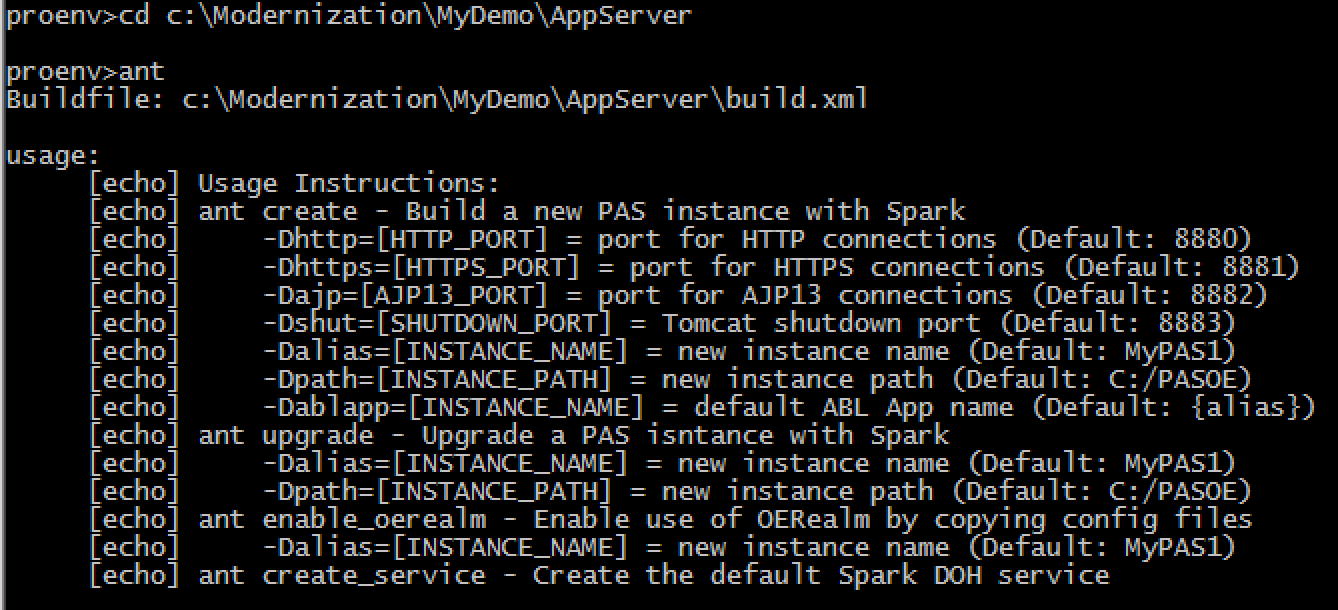
1. **Open** the Progress Developer Studio, which will utilize the new PROPATH items.

**Note:** One of the files added to the project is a **ROOT.map** located in the **PASOEContent\WEB-INF\OpenEdge** folder. This mapping file will provide a full listing of services when you visit the **<webapp>/web/pdo/** URI for your new PAS instance. This may not be secure or ideal for Production purposes but can be tremendously helpful for Development. You may wish to avoid including this file within your final published application at runtime.

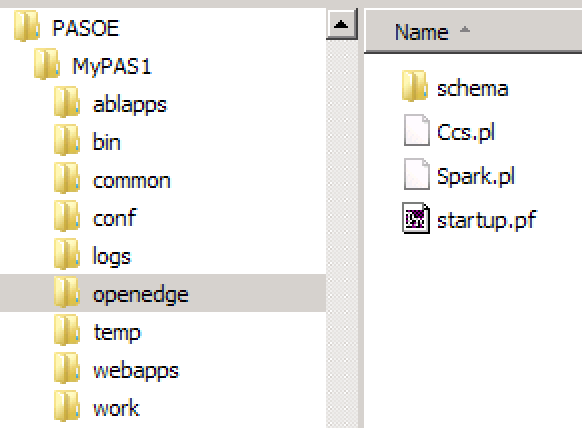
# PAS Instance Creation

Next we need a dedicated instance for our project. With the Spark files in place we now have an Ant task to make this much easier to perform via the command line. Once our new instance is created we will be able to publish our demo project.

1. Open a **Proenv** session as Administrator.
2. Change directory to **C:\Modernization\MyDemo\AppServer**
3. Run the “**ant**” command to view usage instructions.
   1. Assuming the default DLC path, you may need to use the full command path of **C:\Progress\OpenEdge\ant\bin\ant**
   2. Note the “**ant create**” usage with various options available.
   3. For our purpose here we will take all of the given defaults.

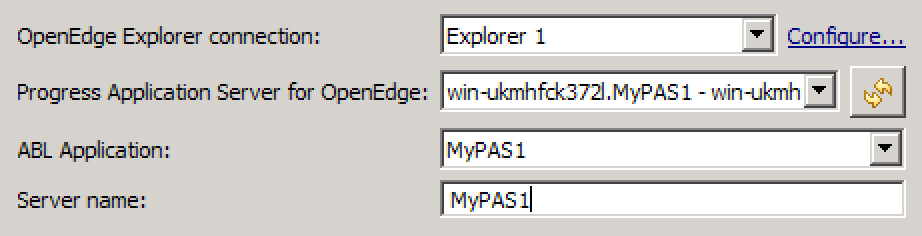


1. Run the “**ant create**” command to create the default **MyPAS1** instance.
   1. Results of the “pasman create” command, which runs behind the scenes, will be output to a log file (in same directory) for review.
2. Confirm the new instance was created at **C:\PASOE\MyPAS1**
   1. There should be a **Ccs.pl** and **Spark.pl** in the **/openedge** folder.
   2. The **PROPATH** for this instance should include these libraries.
   3. Default configuration files (\*.json) should also be present at **C:\PASOE\MyPAS1\conf\spark**



# PAS Instance Startup

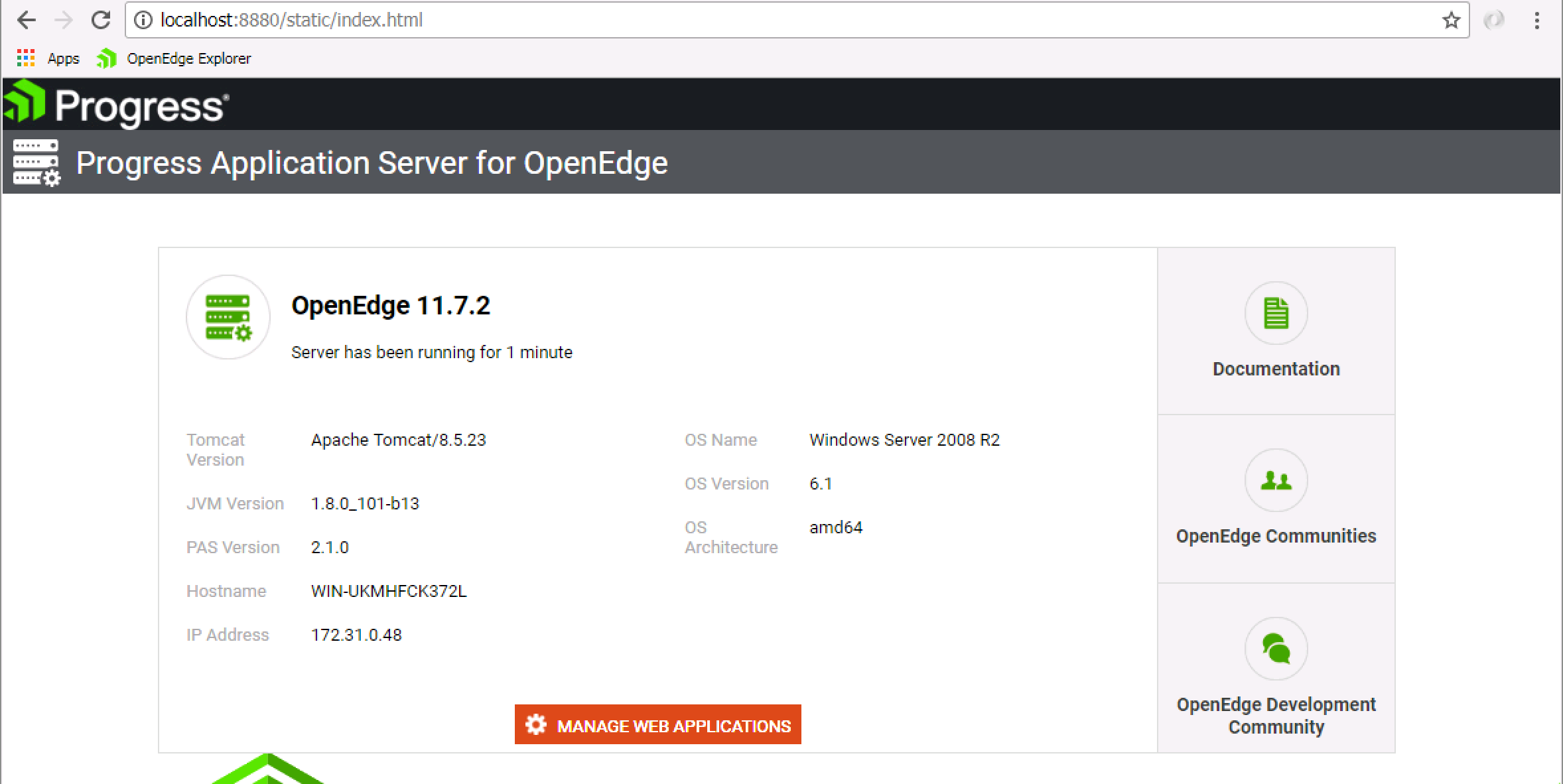
1. Return to the **Developer Studio** application.
2. Select **Window > Show View > Servers**
   1. Use **Window > Show View > Other > Server > Servers** if the option above is not immediately available due to previous use.
3. **Right-Click** in the **Servers** view and select **New > Server**
4. Select the option **Progress Application Server for OpenEdge**.
5. Press the **Next** button and locate your **MyPAS1** server.
6. Set the server name as just “**MyPAS1**”, and press **Finish**.



1. **Right-Click** on the new server and select **Start**, wait for “Started” status.



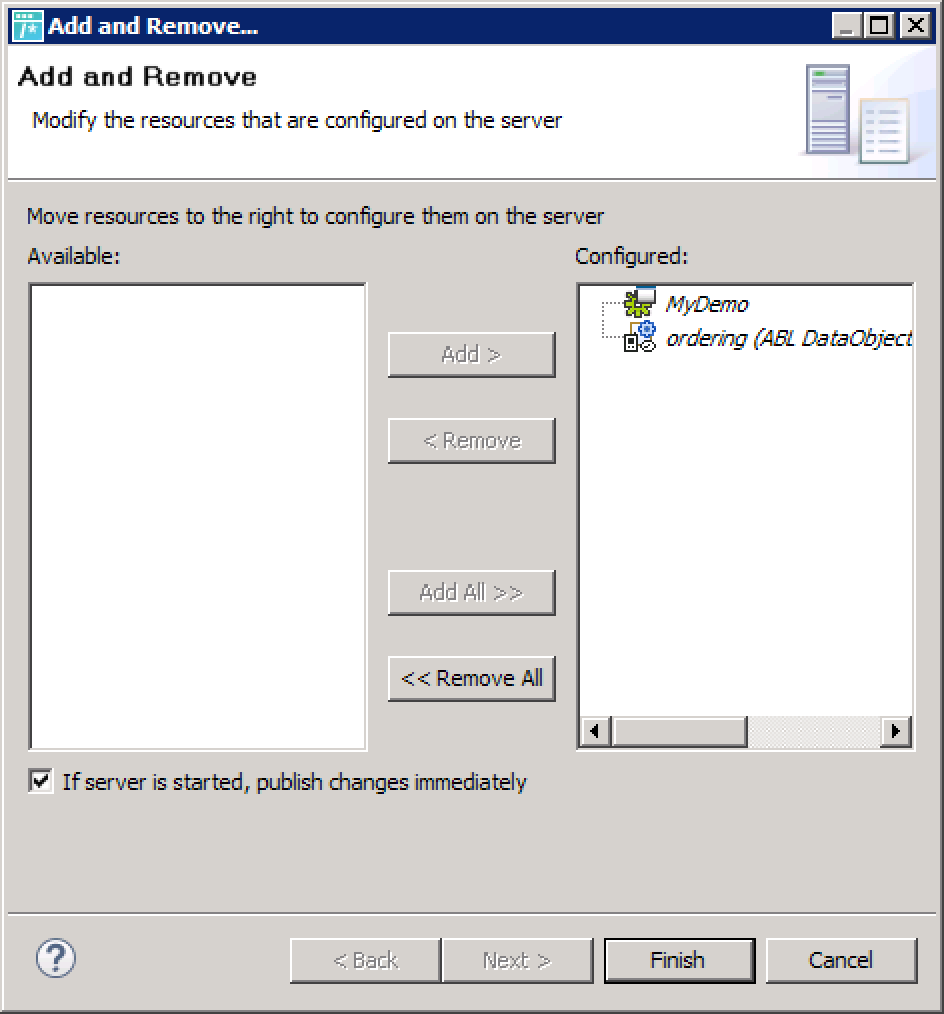
1. Confirm the server’s running status by visiting <http://localhost:8880>



# Demo Project Publishing

With the server running, we can publish our new project from the Developer Studio.

1. **Right-Click** on the **MyPAS1** server in the Servers view.
2. Select the “Add and Remove…” option from the list.
3. Add the “**MyDemo**” WebApp and “**ordering 🡪 MyDemo**” service.



1. Click on **Finish** to begin publishing to the server instance.
2. Wait up to a few minutes for the **[Started, Synchronized]** status.
3. Confirm the new WebApp is available at <http://localhost:8880/demo/>

At this point we should have a working server instance and an available WebApp. Now we need to add some resources to our “ordering” service and connect a database.

# Database Configuration

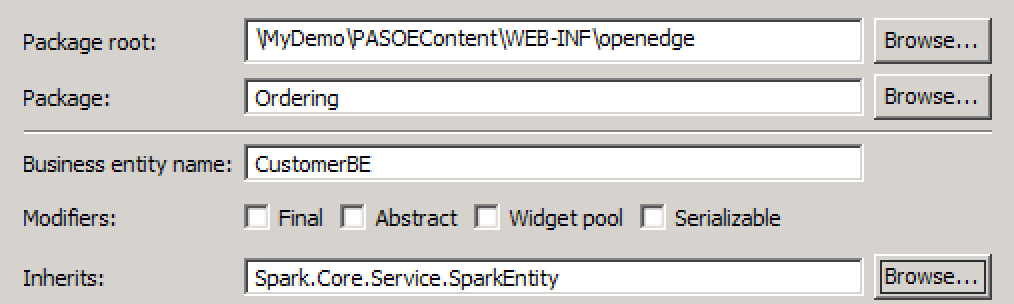
We need a database for our application, and there are 2 databases shipped with the Spark-Demo repository that we can leverage. If you already have a database that you wish to use, that would work as well. We can create this easily with some pre-built options.

1. Open a **Proenv** session as Administrator.
2. Navigate to **C:\Modernization\Spark-Demo\support\schema**
3. Run the “ant” command to view script usage instructions.
   1. By default databases will be created in **C:\Databases**
4. Run the command “**ant create**” to create the directory and 2 new databases.
   1. Databases should be created from available schema and data.
   2. A default “**spark**” domain should be added to each database.
5. Confirm that the **Sports2000** and **WebState** databases exist in **C:\Databases**
6. Create and start two new database servers via **OpenEdge Management** (accessed at <http://localhost:9090>):
   1. Name: Sports2000, C:\Databases\Sports2000\Sports2000.db, Port: 8600
   2. Name: WebState, C:\Databases\WebState\WebState.db, Port: 8500
7. Return to Progress Developer Studio, **Right-Click** on your project and select **Properties**, then **Progress OpenEdge > Database Connections**
   1. Add the two databases using **localhost** and the ports as stated above.
   2. **Select both databases** for the project and click on **OK**.
   3. **Right-Click** on the project, select **Progress OpenEdge > Restart OpenEdge AVM** to ensure the new DB connections are established.
8. Open the project file **AppServer\startup.pf** via a right-click, selecting Text Editor.
   1. Add the following lines to connect to these new databases:
      1. -db Sports2000 -H localhost -S 8600
      2. -db WebState -H localhost -S 8500
   2. Save the file and manually copy to the CATALINA\_BASE/openedge folder of our new PAS instance (**C:\PASOE\MyPAS1\openedge**).
   3. **Note:** When building your own application, replace these connections with the appropriate connection(s) to your application database(s).
9. **Right-click** on the **MyPas1** server and **Restart** your PAS instance to pick up the new database connections when starting the ABL Application.

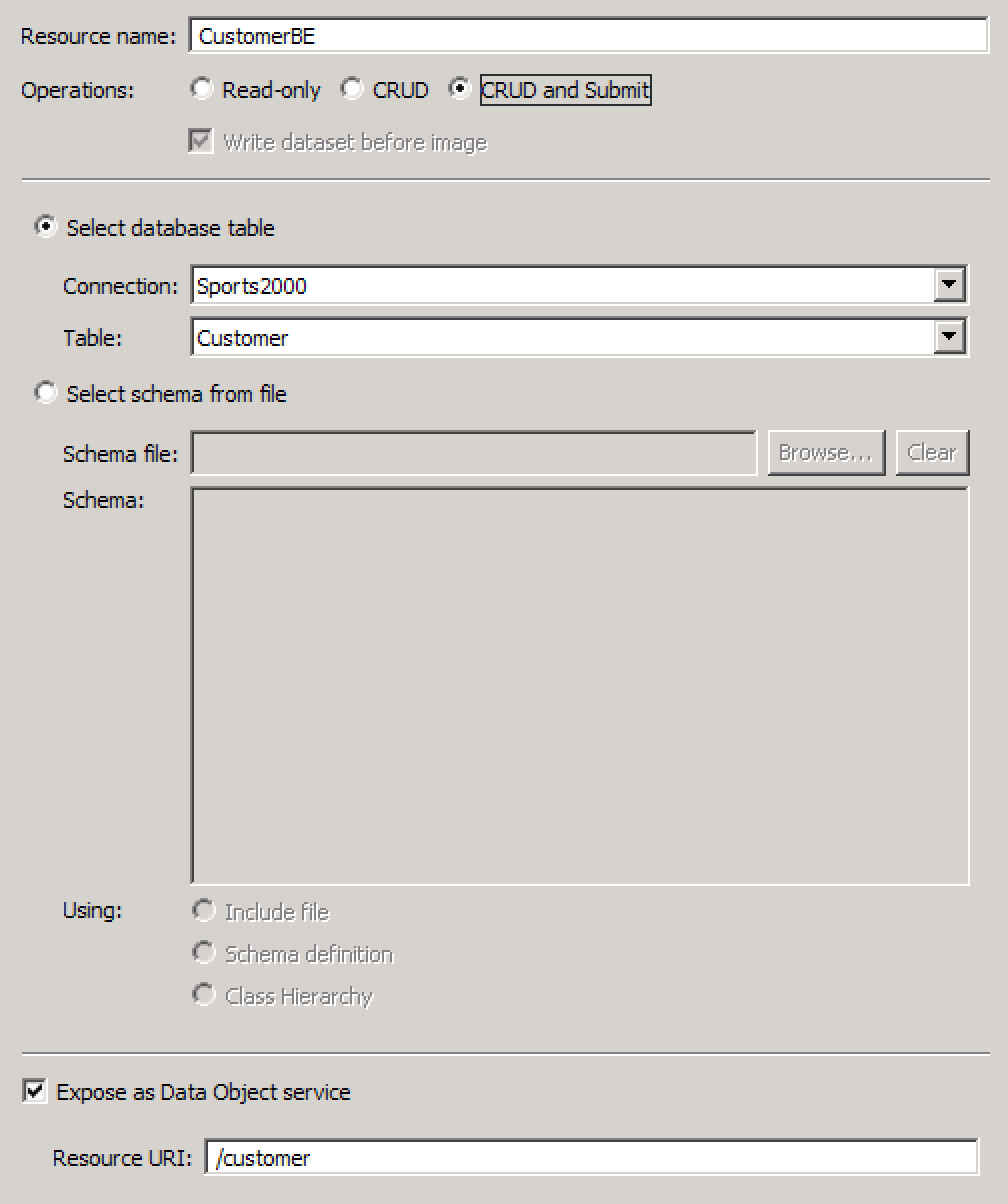
# Resource Creation

We already have an available service “**ordering**”, but we lack any resources that will reside under that service. For that we need a suitable Business Entity class. The process for this will utilize the standard wizard in PDSOE with a few post-generation modifications.

1. Before proceeding, select **OpenEdge > Refresh Cache** from the top menu.
2. Right-Click on the project and select **New > Business Entity**
3. Change the **Package Root** to **\MyDemo\PASOEContent\WEB-INF\openedge**
   1. This location keeps code specific to your WebApp when deployed.
4. Set the **Package** as “**Ordering**” to help organize the class file created.
5. Set the **Business entity name** as “**CustomerBE**”.



1. Change the **Inherits** class to **Spark.Core.Service.SparkEntity** and click **Next**.
2. Select the “**CRUD and Submit**” option, and select **Database Table**.
   1. Connection: Sports2000
   2. Table: Customer
3. Leave the **Expose as a Data Object** service option checked
4. Change the **Resource URI** to simply “/customer” and click **Finish**.
   1. If your screen resolution is low, you may not see the entire modal window—note the following screenshot for comparison if needed.
5. The new file will be created and opened for editing automatically.



# Code Refactoring

Since we inherited from the **Spark.Core.Service.SparkEntity** class, we need to make some minor adjustments to accommodate this particular class. The SparkEntity provides some advanced features over the stock **OpenEdge.BusinessLogic.BusinessEntity** class, even though it still inherits that original class.

1. Remove the “define data-source” line at the top of the class file.
2. Adjust the **constructor** method as follows:

ProDataSet = dataset dsCustomer:handle.

extent(DataSourceArray) = 1.

create data-source DataSourceArray[1].

DataSourceArray[1]:add-source-buffer(buffer Customer:handle, ?).

ProDataSource = DataSourceArray.

extent(SkipListArray) = 1.

SkipListArray[1] = "".

SkipList = SkipListArray.

1. Just above “**method public void ReadCustomerBE**” add the following:

@openapi.openedge.method.property(name="mappingType", value="AFP").

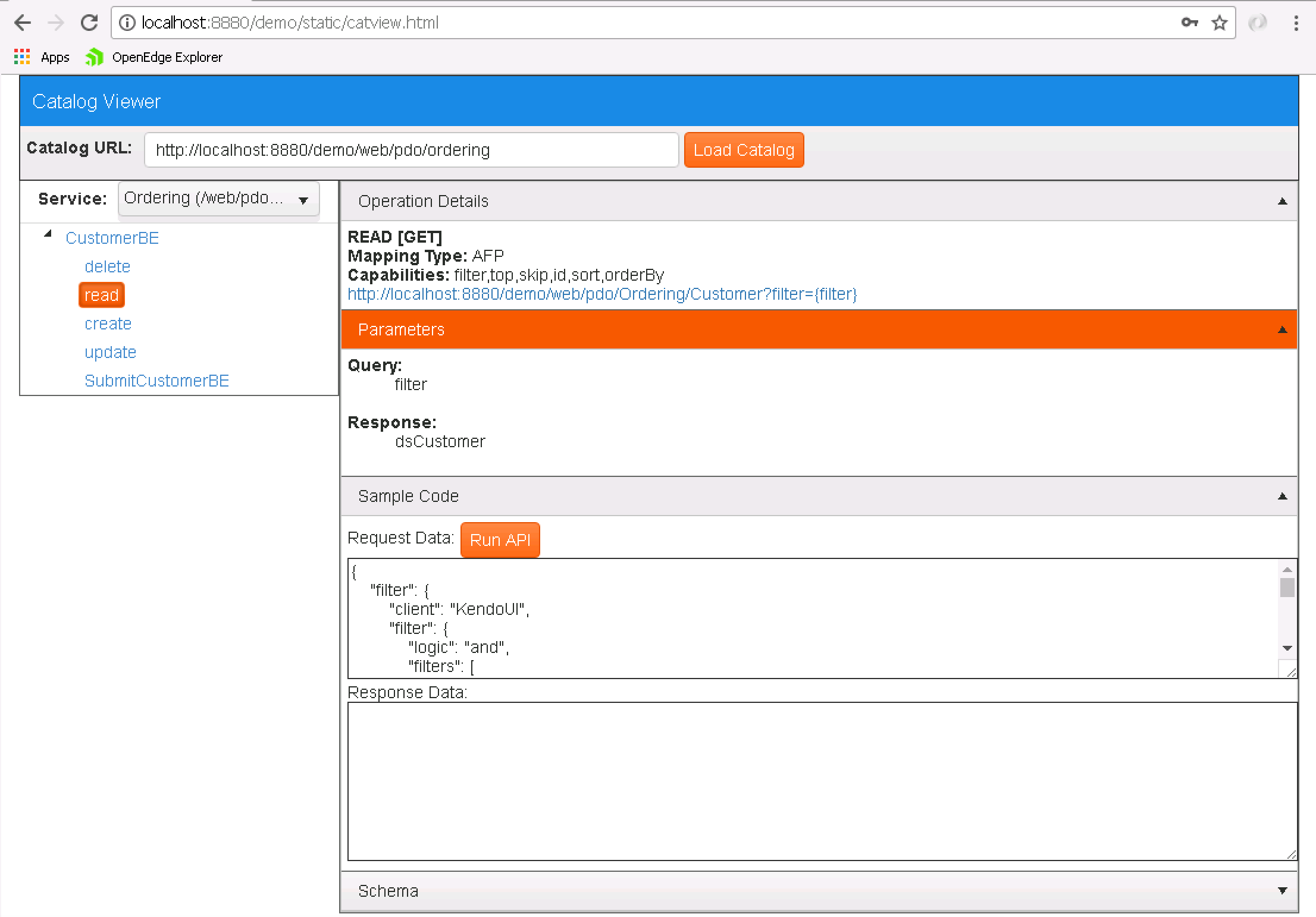
@openapi.openedge.method.property(name="capabilities", value="filter,top,skip,id,sort,orderBy").

1. Replace the statement “**super:ReadData(filter)**” with the following:

define variable iNumRecs as int64 no-undo.

super:ReadData(filter, iNumRecs, output dataset dsCustomer by-reference).

1. Open the **Ordering/customerbe.i** include file and adjust the schema as follows:
   1. **Add** a field “**id”** as type **character**.
   2. **Add** a field “**seq**” as type **integer**, with **initial** value **?**.
   3. **Remove** the “**primary**” keyword on any existing indexes.
   4. **Add** an index “**pkSeq**” as **primary/unique** on “**seq**”.
   5. **Save** your changes to the file.
2. **Save** and **compile** the **CustomerBE** file to pick up changes to the include file.
3. Expand the list of **Defined Services** in the project, double-click on “**ordering**”.
4. Click on **Next**, select the new **CustomerBE.cls** class, and click **Finish**.
5. **Right-Click** on the **MyPAS1** server and select **Publish**.
   1. If the server is already configured to perform an automatic publish, simply wait for PDSOE to acknowledge the recent changes and to publish the updates to the PAS instance.
6. Confirm the new CustomerBE resource is available by accessing the catalog at <http://localhost:8880/demo/web/pdo/ordering>
7. Test the new resource by visiting <http://localhost:8880/demo/static/catview.html> and loading from the catalog URL referenced in the previous step.



1. Expand the **CustomerBE** resource and click on the **read** operation.
2. Expand the **Sample Code** area and click on **Run API**.
   1. Confirm a response is returned for the dsCustomer dataset.

At this point you should have an annotated Business Entity (BE) which is capable of leveraging any other classes or services within the Spark toolkit directly. In addition, the modifications made to the default BE class pattern and annotations will provide for filtering, sorting, and paging of results through the AFP mapping type. This pattern utilizes the natural format of KendoUI’s KendoGrid widget and supports returning the record count within the READ operation. For more information about AFP, please consult the supplemental documentation included with the Spark toolkit.

# Enhancing Security

So far we’ve been working with the Anonymous security model, as we have not been required to log in. This allowed us to confirm that our server was operational and capable of returning data. But now we need to begin adjusting the security model to lock out unauthorized users.

1. Open the **oeablSecurity.properties** file in **/MyDemo/PASOEContent/WEB-INF**
2. Change the **client.login.model** value from “anonymous” to “form”.
   1. Leave the **http.all.authmanager** set to “local” to utilize users.properties
3. **Save** the file, **Publish**, and **Restart** the **MyPAS1** instance.
4. Return to the **Catalog Viewer** at <http://localhost:8880/demo/static/catview.html>
5. Attempt to run the CustomerBE’s **read** operation again.
   1. A dialog should appear requiring you to log in.
   2. Use “dev” and “progress” as the username/password, and click **Login**.
   3. The dialog should disappear if the login was a success.
6. Re-run the API request to confirm that authentication worked and data returned.