# Progress Spark Toolkit Evaluation Guide

Evaluating the Progress Spark Toolkit through demo applications

August 2019

# Summary

The **Progress 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 **Progress 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-toolkit-demos**” 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-toolkit**” repository will be the avenue for making or requesting changes to the codebase. **For the purpose of this guide we will be focused on the quick-start approach offered by the “spark-toolkit-demos” 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 at least OpenEdge versions 11.7.4 and 12.0—versions of OE prior to 11.7.4 are no longer recommended. Use of a **64-bit Windows** installation and **OE 11.7.4 or later is required**, and at least the Progress Developer Studio for OE component present for development purposes.

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 11.7.3 or later (executed easily via DLC/bin/proant).

# Demo Project Availability

At present there are two demo projects available, “**Sports**“ and “**DynSports**”. Both projects offer the same named API endpoints via the WEB transport of PASOE, though where these projects differ is in how those endpoints are defined for exposure.

**Sports:** Utilizes the approach of annotated Business Entity classes which are added to a standard ABL Service via the WEB transport. These BE classes follow the prescribed approach for Progress Data Objects (PDO) and utilize an internal class called the DataObjectHandler (DOH) to map HTTP artifacts to ABL constructs and to execute the requested class methods. This approach requires the Progress Developer Studio (PDSOE) to assist with generating those mappings and producing a Data Service Catalog for use with the JSDO on the client side.

**DynSports:** Utilizes the CatalogManager class in the **Progress Spark Toolkit** to dynamically discover any available class or procedure file and exposes internal methods/procedures using default options or specific overrides. This approach performs discovery at runtime (agent startup) and does not rely on the Progress Developer Studio (PDSOE). As such, this requires some additional configuration up front but can become a more automatic approach as the number of exposed entities increases.

In order to maintain consistency with currently-available documentation regarding the Progress Developer Studio, the following content will focus on use of the **Sports** demo for evaluation. This is also the approach utilized in the “**Spark Quick-Start Guide**” document, which may be followed to create a new project from scratch.

# What’s in the Box?

The included demo projects include the following features on the back-end and front-end, in addition to useful tooling and test utilities. This list may be subject to change as new features are added or enhancements are made to improve compatibility with other Progress products such as the KendoUI Builder.

Back-End:

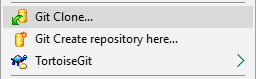
* Authentication examples: form-local, form-oerealm
* Reference implementation of an OERealm class (w/ 2FA)
* Best-practice examples of asserting/clearing user identity
* Use of multiple ABL Services using DataObjectHandler
* Sample use of OE.Web.PingWebHandler for \_oeping
* Sample use of Spark.Core.Handler.FileUpload handler
* Examples of custom Spark manager and service overrides
* Implements the ABL Filter Pattern (AFP) for advanced READ

Testing/Tooling:

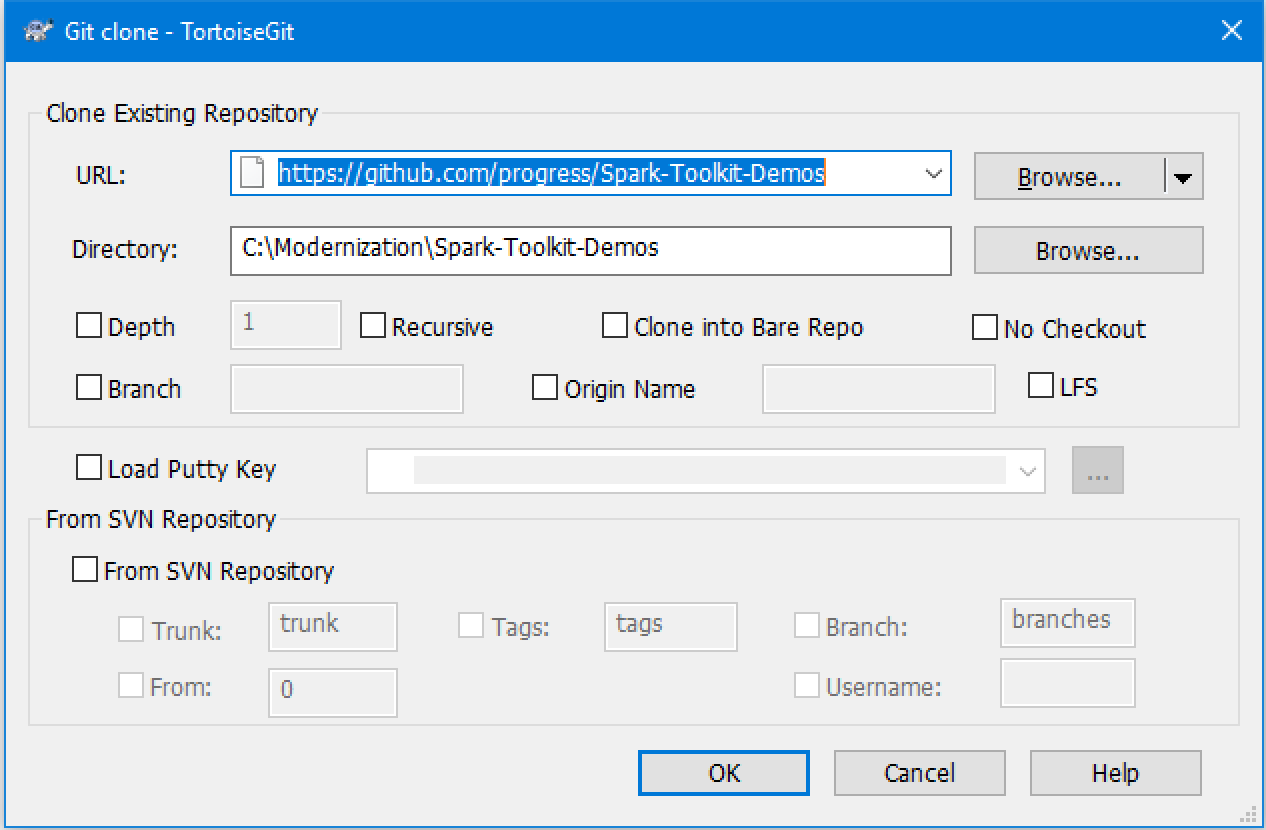
* ANT script for building a suitable PAS instance
* ANT script for creating/loading demo databases
* ANT scripts to start/stop/clean/trim PAS instance
* Test suites for basic ABLUnit sanity checks
* Assorted ABL scripts for debugging and setup

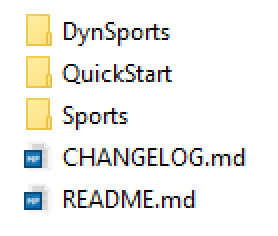
# Obtaining Code

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. Do a “**Git Clone**” of [**https://github.com/progress/Spark-Toolkit-Demos**](https://github.com/progress/Spark-Toolkit-Demos)



1. **Option 2:** Visit <https://github.com/progress/Spark-Toolkit-Demos/releases> and download the latest available release as either a .ZIP or .TAR.GZ archive.
   1. Expand the archive, making sure the top-level directory is named simply “**Spark-Toolkit-Demos**” 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.

# Database Configuration

First, we need databases for our application, and there are 2 databases shipped with the Spark-Toolkit-Demos repository that we can leverage. We can create this easily with some pre-built options.

1. Open a **Proenv** session as Administrator.
2. Navigate to **C:\Modernization\Spark-Toolkit-Demos\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 **Sports2020** 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: Sports2020, C:\Databases\Sports2020\Sports2020.db, Port: 8600
   2. Name: WebState, C:\Databases\WebState\WebState.db, Port: 8500
7. Return to Progress Developer Studio and navigate to **Window > Preferences > 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.

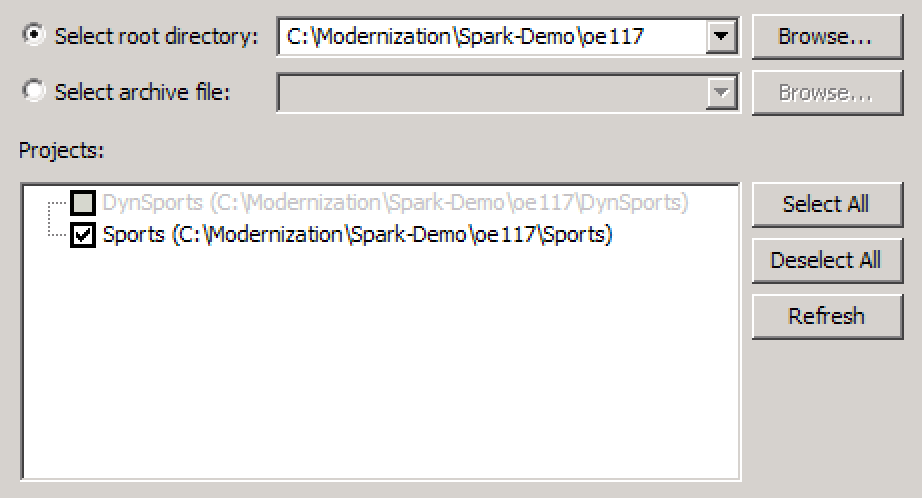
# Demo Project Import

Assuming the preceeding process was followed for obtaining source code, you should have all available demos at the following location:

**C:\Modernization\Spark-Toolkit-Demos\oe117\**

If your filesystem does not reflect the same path, please take a moment to standardize your environment. Use of a consistent path will guarantee a streamlined setup process.

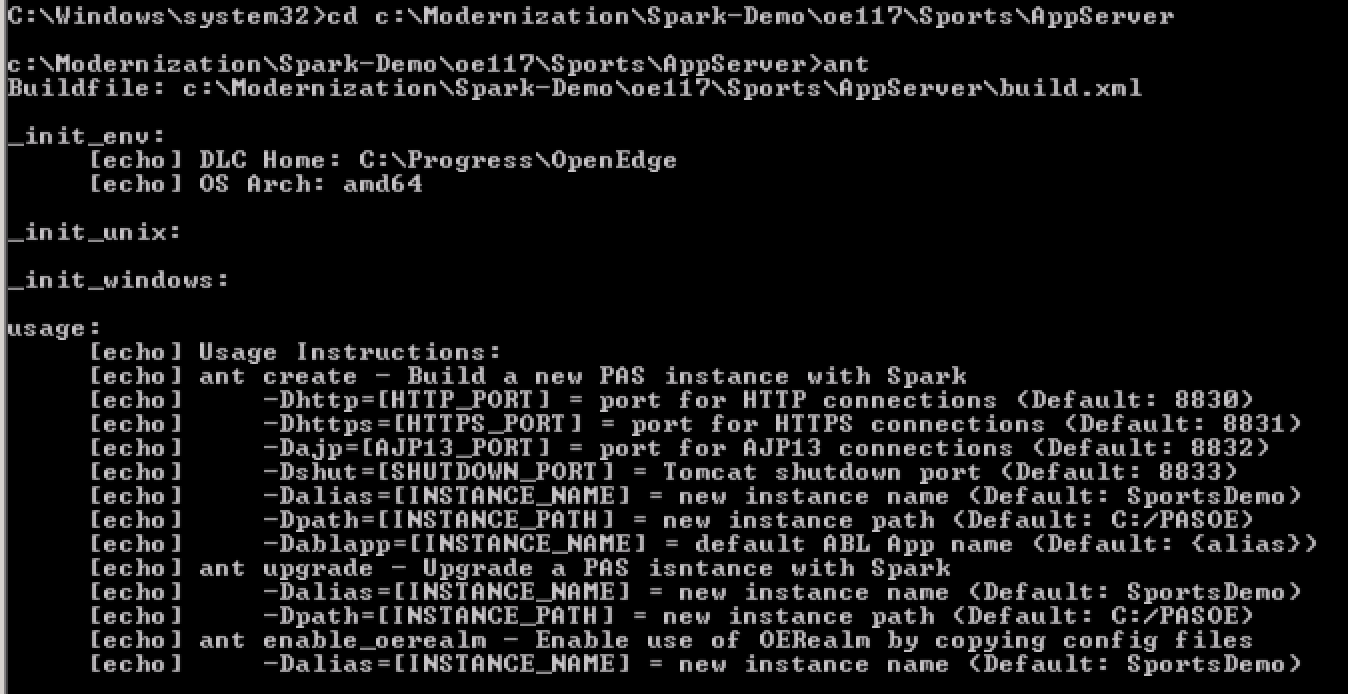
1. Start PDSOE with the **C:\Modernization** workspace selected.
2. Select **File > Import > Existing Projects into Workspace** and click **Next**.
3. Select the root directory **C:\Modernization\Spark-Toolkit-Demos\oe117**
4. In the list of Projects, only check the **Sports** project and click **Finish**.



# PAS Instance Creation

Next we need a dedicated instance for our project. Within the demo project we 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 **Sports** demo project.

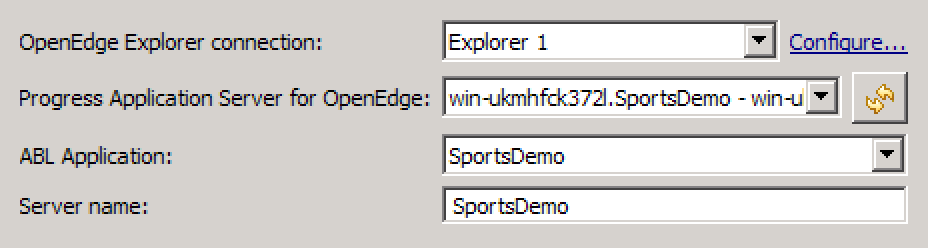
1. Open a **Proenv** session as Administrator.
2. Change to **C:\Modernization\Spark-Toolkit-Demos\oe117\Sports\AppServer**
3. Run the “**ant**” command to view usage instructions.
   1. Assuming the default DLC path, you may need to use the full command path of **C:\Progress\OpenEdge\ant\bin\ant**
   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 **SportsDemo** instance.
   1. Results of the “pasman create” command which runs behind the scenes will be output to a local log file for review.
2. Confirm the new instance was created at **C:\PASOE\SportsDemo**
   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. Configuration files should be present at **/conf/spark** as well.

# PAS Instance Startup

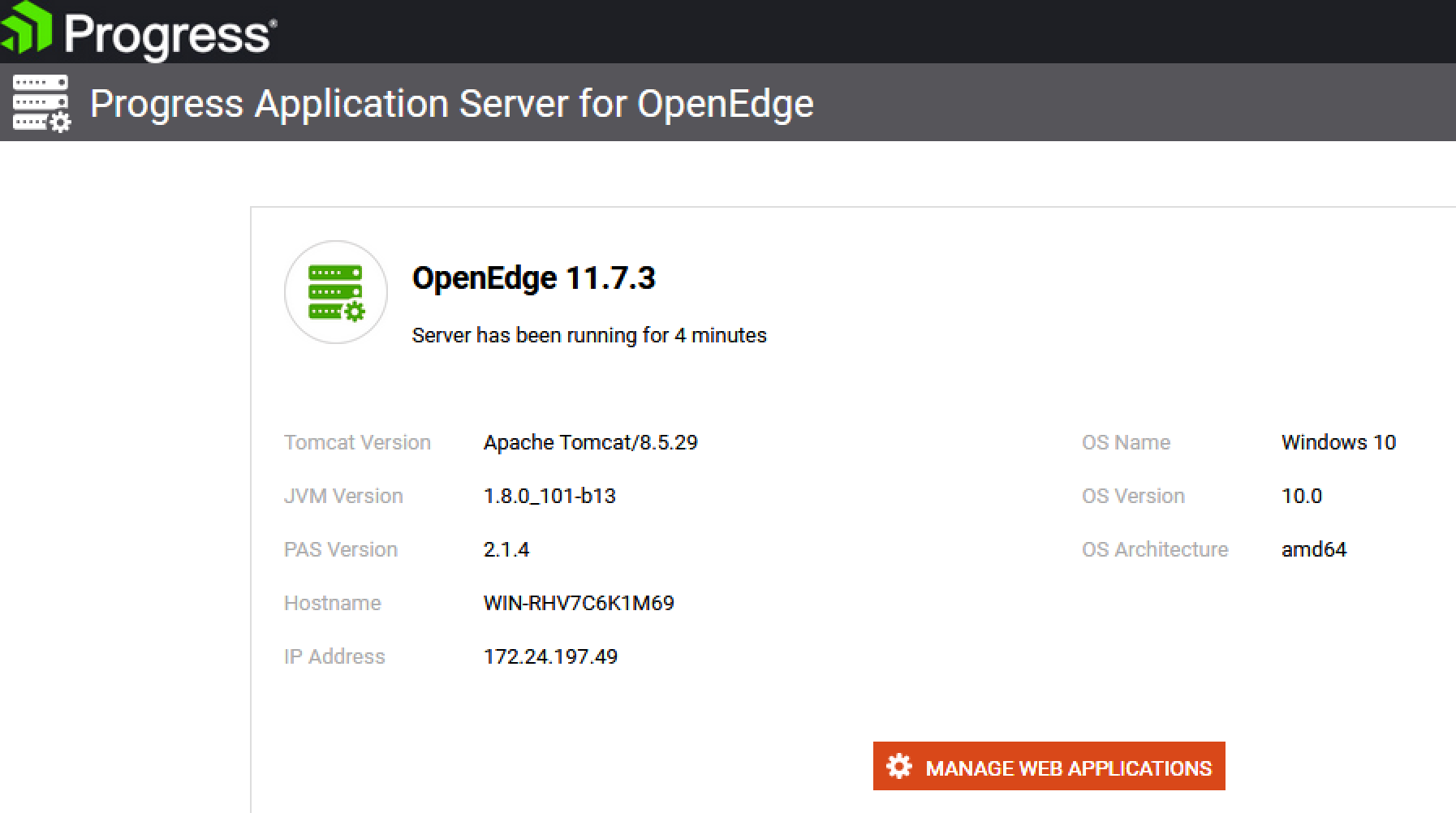
1. Return to the **Developer Studio** application.
2. Select **Window > Show View > Servers**
   1. Use **Window > Show View > Other > Server > Servers** if the option above is not immediately available due to previous use.
3. **Right-Click** in the **Servers** view and select **New > Server**
4. Select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



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



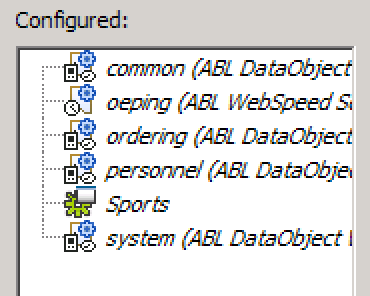
1. Confirm the server’s running status by visiting <http://localhost:8830/>



# Demo Project Publishing

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

1. **Right-Click** on the **SportsDemo** server in the Servers view.
2. Select the “Add and Remove…” option from the list.
3. Add the “**Sports**” WebApp and all services that end with “**🡪 Sports**”.



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:8830/sports/>

At this point you should have a working server instance and a “sports” WebApp for use with any available front-end technologies.