Chapter 1

Progress Modernization Framework for OpenEdge

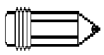
Introduction & Setup

What is PMFO?

Progress Modernization Framework for OpenEdge, or PMFO, is the modernization framework developed by Progress Services. It combines the proven logic of several production-ready, existing ABL frameworks, together with the best practices outlined by the OpenEdge Reference Architecture. While it does include some UI components at present, the majority of the framework is server-oriented and includes functionality to support the following features of a typical web-enabled application:

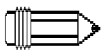
- Session Extensions
- Context Management
- Security Examples
- Sample Databases
- Error Messaging
- Translation Capabilities
- Data Validation Practices
- Extensible Business Entity

Most importantly, due to the number of tools and components in play within the new Progress Application Server (PAS) there is a fully-built sample application provided (DynSports), along with a wizard-based tool for generating custom components. The following training material will cover the use of the existing "Web Wizard" tooling, and illustrate means of manipulating code either after generation or completely by hand. This applies to both the current client-side screens as well as the back-end ABL code.



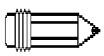
NOTE

The PMFO is not meant to compete with other offerings such as the Kendo UI Builder (KUIB), but to supplement its use where necessary. For organizations competent with JavaScript development, PMFO offers a common set of tools and ready-made widgets using KendoUI. Though first and foremost, PMFO exists to speed development of ABL code and to implement a server back-end as quickly and securely as possible. The choice of UI is considered as transient and is open to any solution that can utilize JSON data over an HTTP/HTTPS connection.



NOTE

Another demo project available for sampling is the simply-named "Sports" which provides a closer "out of the box" experience between use of annotated Business Entities and the Data Object Web Handler solution for exposing your services. Essentially this is the minimal solution for integrating PMFO within your application while still adhering to the bulk of the existing documentation for OpenEdge. In either case, whether you use DynSports or Sports, the exposed API's should be 100% compatible with the JSDO and KUIB when building front-end solutions.



NOTE

The name "Progress Modernization Framework for OpenEdge" is a marketing term to the overall solution of modernizing an ABL application. As we discuss the back-end services and the related repositories for code the term "Spark" may be used. This is the namespace of the internal classes that are part of the OO-ABL solution which bolsters the PMFO approach. For you, the audience, just know that these terms may be used interchangeably to refer to the same family of solutions.

Software Prerequisites

Applications that utilize the OpenEdge REST Adapter and related technologies require a specific set of software to be installed. For purposes of this material we will focus only on the development aspects, and therefore includes some additional tools that aid in this effort. First and foremost, Progress OpenEdge is required with version 11.6 or later recommended. Your training VM should have the necessary products pre-installed but for installation on other development environment the components will be noted in Table 1-1 below. **Note: this material is based on OpenEdge 11.7.2**.

Table 1-1

Component	Notes
Progress Development Studio for OE	REQUIRED - The primary IDE, referred to as PDSOE. Installation of this product alone gives access to a Personal Database, the PAS components, AppServer/WebSpeed, and ABL dev/compile tools.
OE Studio	Also referred to as the AppBuilder, provides GUI development tools. Included in PDSOE.
4GL Development	Provides standard OpenEdge compilation tools. Included in PDSOE.
OE Workgroup or Enterprise RDBMS	Either a workgroup or enterprise database is recommended, though PDSOE provides access to the Personal RDBMS component.
OE Application Server (AppServer/WebSpeed)	The REST adapter requires at least an AppServer for ABL connectivity. Included in PDSOE.
Progress Application Server Dev for OE	Next-generation AppServer product, utilizing a combined Tomcat web server and OE AppServer, included in PDSOE (Win64 only).
Client Networking	Included in PDSOE
Query/RESULTS	Optional, but recommended.
WebSpeed Workshop	Optional, but useful if needed.

In addition to Progress components, some tools may be useful or necessary to development as they can simplify many mundane tasks. Software and minimum versions are listed below in Table 1-2.

Table 1-2

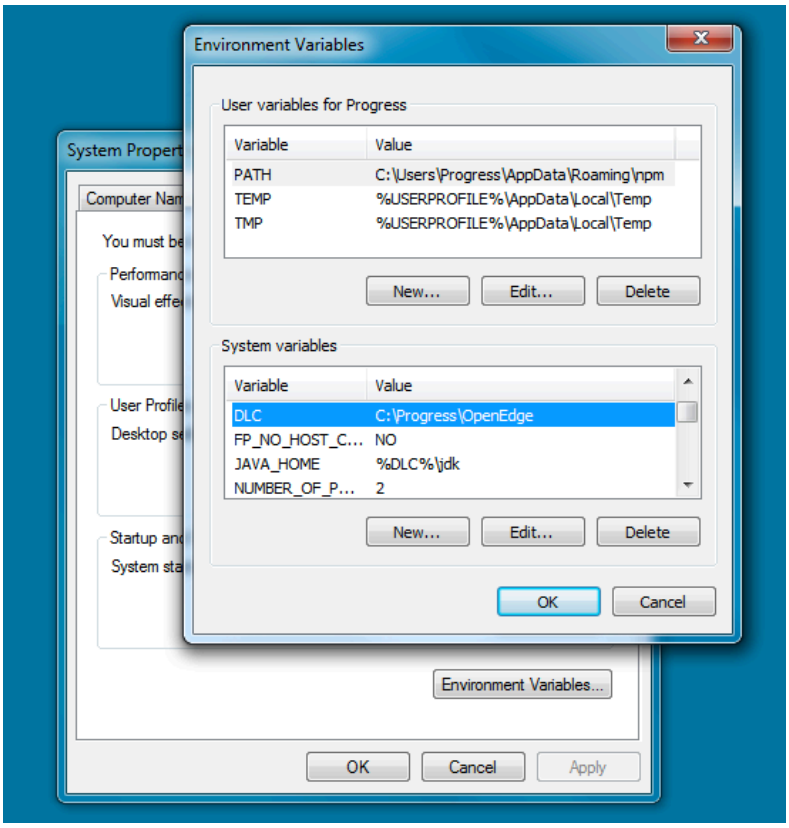
Software	Min. Version	Notes
Apache Ant	In OE 11.7	Used for minification of client libraries and other tasks
Chrome/Firefox	Latest	Multi-platform, OS-agnostic web browsers for testing
Notepad++	Latest	A more powerful text editor than the Windows Notepad
Sublime	Latest	An alternative code editor for static files
Node.js	6.0+	Optional, but recommended for installation of other tools
NPM Tools	Automatic	Additional tools via “npm -g install npm bower grunt”

Environment Setup

As part of the configuration process after installing software, it is recommended to add certain variables to your Windows environment as outlined in Table 1-3. These allow running of command line tools from any directory, and improve productivity by providing easy access to Progress files. If a variable does not exist, simply create a system environment variable by the name listed. **All virtual machines provided for training purposes have been configured with these options already.**

Table 1-3

Variable	Path
DLC	C:\Progress\OpenEdge
ANT_HOME	%DLC%\ant
JAVA_HOME	%DLC%\jdk
PATH	Add the following items **to the end** of the **existing PATH variable**: ;%DLC%;%DLC%\bin;%ANT_HOME%\bin;%JAVA_HOME%\bin



NOTE

After modifying any environment variables, it is necessary to close any open command prompts prior to attempting to use any programs or files now pointed to by the updated paths. Also, it is worth noting that as of Windows 7 you can easily launch a command prompt from any directory: from within the Windows Explorer just **Shift + Right-Click** on a folder and select the option "*Open command window here*".

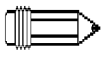
Prerequisite Knowledge

The remaining content for this portion of training depends on knowledge of various basics, many of which should have been covered by earlier training courses. For the sake of completeness, the following topics should have at least been covered prior to continuing:

Table 1-4

Heading	Heading
JavaScript Basics	Absolutely required for all work going forward.
KendoUI	Required, and will be reviewed with necessary depth in coming chapters.
jQuery	Highly recommended, but basics will be covered as needed.
Bootstrap	Optional but useful for understanding how to create responsive layouts.
Font-Awesome	Provides a standard set of icons/glyphs for applications via a special font library, avoiding the need to create these graphic items on your own. Also, they can be scaled, stacked, and colored as necessary.

At first glance, it may seem confusing to know where one tool ends and another begins. In reality, each of these front-end tools layer on top of each other and provide functionality that each excels at. For example, jQuery is a great all-around library of tools for performing common operations in JavaScript. It also provides a simple means of accessing DOM elements by use of "**selectors**", similar to what is used in CSS. These selectors can provide an element object that can then be used by KendoUI to convert to a more complex widget instance or modify a converted widget's properties and behavior. However, none of these libraries themselves truly offer the ability to easily layout a page or handle differences between desktop, tablet, and smartphone devices. To that end, Bootstrap can easily modify a page to suite any device viewport or change of resolution on a desktop application. So, while not absolutely necessary for purposes of covering the remainder of training with PMFO, this is why it is recommended to study up on these toolkits to make development more productive and consistent.



NOTE

For the remainder of this material we will be following a step-by-step process, which will be based on the setup process outlined in **Appendix A**. For insight into the original configuration of your working environment, please refer to that content as needed.

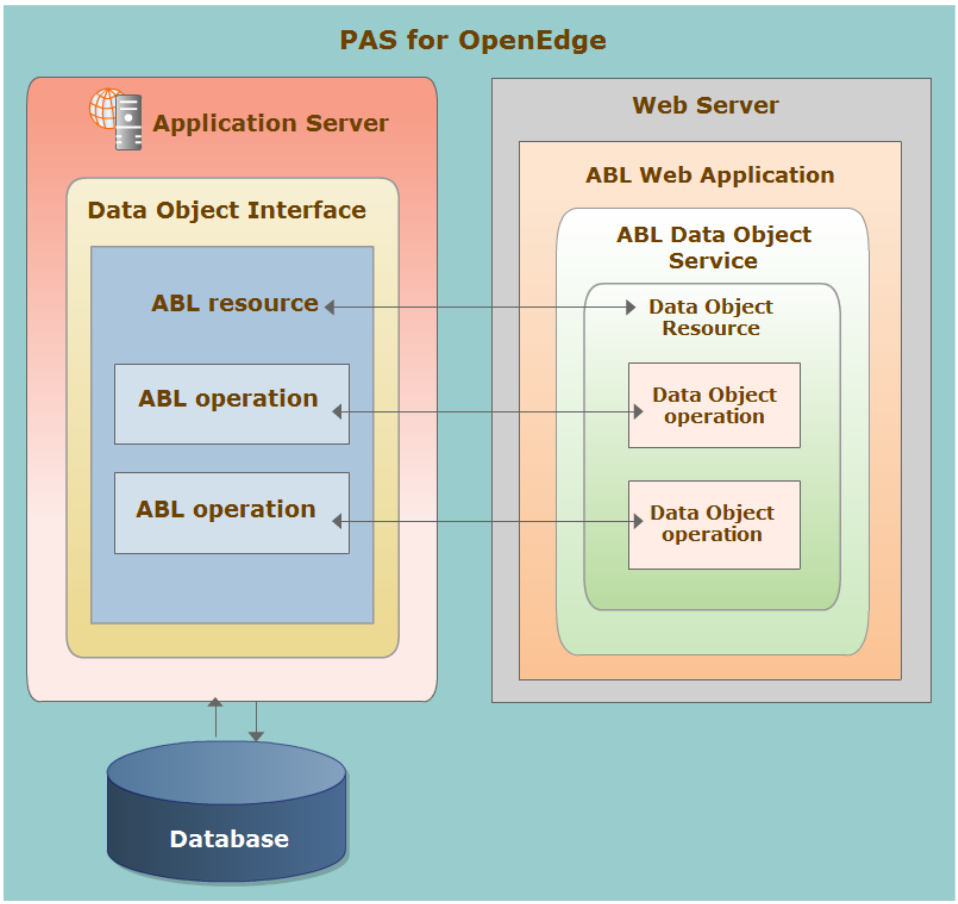
What is PAS?

Before ending this introduction, it should be explained as to what PAS is. This is the next-generation AppServer within OpenEdge, properly referred to as the **Progress Application Server for OpenEdge**, or just "**PASOE**" or "**PAS**". For reference, the previous generation of AppServer is referred to as "Classic" or "**CAS**" for short. Below are a few key differences between these servers.

Table 1-5

Classic AppServer	PAS for OE
AppServer Broker	Multi-Session AppServer (MSAS) Agent
AppServer Agents (separate processes)	Agent Sessions (similar to process threads)
Requires Tomcat for REST	Includes a Tomcat Server
Only AppServer and REST Connections	REST, SOAP, Web, and APSV Connections

In architectural terms, the PAS product is deployed as an "instance" which consists of a Web Server (Tomcat) that contains Web Applications (WebApps). Each of these WebApps contains the ABL code for one or more Services, answering to at least one of four transports: REST, SOAP, APSV, or Web. From there each request is fulfilled by the MSAS Agent which is the AppServer for the PAS instance.



NOTES: