Ozone Build Instructions

DOD GOSS

Version 8.0.0.1-RC1, 2019-12-31

1. Introduction

1.1. Objectives

The purpose of this document is to describe how to build the following projects:

- docs - generating the pdf and html versions from the asciidoc source.
- ozone-framework-client - A bundled and minified version of the frontend.
- ozone-framework-python-server – A local copy of the bundled ZIP file.

1.2. Document Scope

This document is intended for widget developers and those who wish to build the application from source.

It assumes familiarity with the target development environment, such as Microsoft Windows or Linux, as well as basic command line operations.

1.3. Requirements

Before any of the build tasks can run, the development dependencies mustbe installed. The following sections list the tools, the version requirements for recent OWF releases and nominal configuration elements.

Instructions provided below assume a Microsoft Windows development environment for illustrative purposes only.

2. Download and Install Development Dependencies

The build and development process for OWF requires that the following development dependencies be installed:

- NodeJs
- NPM
- Python
- Java Development Kit (JDK) - for frontend automated tests.

Only the dependency versions listed in the tables below are tested and supported. Deviations from the listed versions may cause the build to be unsuccessful.

Table 1. Development Tools and Version Numbers

Dependency	Version	Download Location
Node.js	10.16.0	https://nodejs.org/
NPM [1]	6.9.0	https://www.npmjs.com/
Python	3.7.4	https://www.python.org/
Oracle Java JDK	1.8	http://www.oracle.com/

1. NPM is provided as part of the Node.js distribution and may not need to be installed separately.

Obtain installation media and instructions for the various operating systems from the primary websites for each tool or trusted download source. The default locations are provided below. Once all tools have been installed, the following sections will describe how to configure the environment.

3. Verify Tool Installations

To verify the installation and version of the tools, use the version command for each tool in a Command Prompt Window.

Example commands and output for an OWF 7 environment follow:

Node

```
C:\> node –v
v10.16.0
```

NPM

```
C:\> npm –v
6.9.0
```

Python

```
C:\> python --version
Python {version-python}
```

Java

```
C:\> java –version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
```

4. Build Instructions

4.1. Build & Install Project Modules

4.1.1. OWF Client

To build the frontend client, run the following commands from the `ozone-framework-client` project

```
npm install
npm run bootstrap
npm run clean
npm run build
```

4.1.2. OWF Backend

The backend end runs on Python, a dynamic language, meaning it does not have to build in order to run the application. But a zipped bundle can be produced for the backend by running the following command from the `ozone-framework-python-server` project

```
python setup.py sdist --formats=zip
```

4.2. Run OWF Framework Server (Development)

The `start-dev.sh` script will run migrations against the configured database, load the default data, and start the application in development mode.

Example

```
C:\ozone\ozone-framework-python-server> ./start-dev.sh
```

4.3. Run OWF Framework Server (Production)

The `start.sh` script will run Django's `manage.py collectstatic` command (more information can be found here https://docs.djangoproject.com/en/2.2/ref/contrib/staticfiles/#management-commands), run migrations against the configured database, load the default data, and then host OWF using Waitress, a WSGI server.

Example

```
C:\ozone\ozone-framework-python-server> ./start.sh
```