amtt Documentation

Release 0.3.0

Ergys Dona

Jun 22, 2017

Contents

amtt stands for "Availability Modelling Translation Toolkit". Its goal is to enable Availability Engineers to easily define their models in a predefined format and be able to translate them to multiple RAMI Software (targets).

CHAPTER 1

Releases

The latest release can be found in the section of the GitHub .

Normally, each release is build for Windows and Linux. Some releases may also be built for MacOS, but if they aren't you can always build by yourself, by downloading the release **source code** and following the *Building from source* guidelines.

Release **packages** (that is, archives with pre-built, ready to run executables) are in *zip* format. Once you have downloaded a release **package**, just extract it to your desired location. Then, in order to run the translator via the user interface:

- If you have downloaded the Windows executable, run `amtt_win32.exe`.
- If you have downloaded the Linux executable, run `amtt_linux`.
- If you have downloaded the MacOS executable, run `amtt_darwin`.

CHAPTER 2

Installation Guide

If you download one of the amtt release bundles, there no need to install anything. The bundle consists of one executable file that can be run either via the user interface (by double clicking on the downloaded file), or from a command line (by specifying the `--cli` option).

However, if you prefer to build and install from source, please refer to *Building from source*.

Runtime dependencies

For amtt to run (either from a bundle file or from a Python installation), you will need to install the following.

1. **Graphviz**

   Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks.

   - To install on Windows, visit .
   - For Linux installation, your distribution package manager may offer a graphviz package. If not, the Downloads section in offers pre-built packages for some Linux distributions (RPM and DEB based).
   - For MacOS, you can install via (the packages in the Graphviz homepage are old).

CHAPTER 3

Building from source

This section outlines the procedure to follow if you need to build a bundle of amtt (which can be useful if you want to distribute the application without having to build it in every machine).

Python version

amtt is written in Python and was developed with **Python 3.5**. It has been tested to work with Python 3.6 as well, although as of the time of writing this text, creating a bundle with Python 3.6+ is not supported. The reason is explained in the following note.

No matter which Python version you choose, you should always download the latest point release for that version. For example, as of the time of writing this text, the latest Python 3.5 version is 3.5.3 and that is what you would choose if you read this text on 13th April 2017.

Note: amtt uses PyInstaller in order to build standalone archives. While amtt should work with any Python 3.5+ version, PyInstaller does not usually support the latest Python version right away. Before proceeding, please check with the home page for the maximum supported Python version. If in doubt, it is recommended that you use Python 3.5 (either 32 or 64 bit).

Requirements

Download and install Python

- For **Windows**, download the (either 32 or 64 bit will work). Upon installing, make sure to check the "Add Python to PATH" option.
- For **Linux**, Python should be available via your package manager. You can also refer to the project, which allows you to install any Python version to any distribution locally (as well as the system meets the pyenv dependencies).
- For **MacOS**, you can install Python via the , via or via the project.

Extra requirements for Windows

In order to build on Windows, you will also need "Microsoft Visual C++ 2010 Redistributable Package". Some links are provided below for your convenience (if they don't work, Google should do the trick).

Choose the appropriate version for your **Python installation**. If you are not sure, just install both:

- x86 (32-bit Python): https://www.microsoft.com/en-us/download/details.aspx?id=5555
- x64 (64-bit Python): https://www.microsoft.com/en-us/download/details.aspx?id=14632

Optional: Use a Python virtual environment

This step is optional, but highly recommended. A Python virtual environment allows you to create a completely isolated environment in which you can install Python packages. This approach avoids cluttering a Python installation with many packages, probably unnecessary for the system (e.g. the packages that this build procedure pulls and installs).

There are many ways to create a Python virtual environment, depending on what you use.

For example, has build-in support for creating and managing virtual environments. However, it is only available for Unix systems.

Another tool that greatly simplifies virtual environment management is virtualenvwrapper. See for installation and usage details (**attention if using Windows:** the installation procedure is different, check the aforementioned link).

Below are some generic instructions, based on the `venv` Python module included in the standard library (no need of any external tools).

1. Choose where you want the virtual environment to be created. Let the path to that directory be `/home/user/path/to/venv/directory`. It is important that this is a local directory, for which you have read and write rights.
2. Create the virtual environment:

   ```
   python -m venv path/to/venv/directory
   ```

   Note for Windows: Under Windows, the Python installer installs a launcher that is used to manage multiple python versions. As the states, explicit is better than implicit. Therefore, make sure to create the virtual environment with the desired python version. For example, to use version 3.5 of Python, execute:

   ```
   py -3.5 -m venv path/to/venv/directory
   ```

   The above command adds all the required executables (`python`, `pip`, etc.) in the `bin` directory within the virtual environment directory. From now on, whenever you need to execute `python`, `pip` or any other executable, you can execute `path/to/venv/directory/bin/python`, `path/to/venv/directory/bin/pip`, etc. respectively.

   You need to "activate" the virtual environment so that the Python-related commands executed from a shell point to the executables within the virtual environment directory. To do this, you have to "source" the activate script residing within the binaries directory of the virtual environment.

   - In Unix systems (Linux, MacOS, etc.), source the activate script:

     ```
     source path/to/venv/directory/bin/activate
     ```

   - In Windows, execute activate.bat:

     ```
     path\to\venv\directory\Scripts\activate.bat
     ```

You will have to do the above every time you open a new terminal or cmd instance.

Clone or download the amtt repository

The amtt repository is located in GitHub: https://github.com/errikos/amtt .

- You can clone the repository by using git:

```
git clone https://github.com/errikos/amtt.git
```

- Or, if you do not want to clone, you can download the master snapshot: https://github.com/errikos/amtt/archive/master.zip

Installing the application

After you have downloaded the source, you can start the installation process.

Change directory, or `cd`, to the directory containing the source code (you will first need to extract archive if you downloaded the master snapshot). Then, you have two options:

- via `pip` (**recommended**):

  `pip` is a package manager for Python packages and is automatically installed if you install Python via the Windows/MacOS installers or the Homebrew/pyenv projects. For package manager based installations in Linux, you may need to install it manually.

  Once inside the source directory, execute the following two commands

```
pip install -U pip setuptools wheel
pip install -e .[docs,build]
```

- without `pip`, via `distutils`:

  `distutils` are bundled in the standard Python distribution.

  1. To fetch the dependencies and build the application

```
python setup.py build
```

  2. To install the application, after building

```
python setup.py install
```

  3. In you encounter an error like the following

```
error: The 'pyexcel' distribution was not found and is required by amtt
```

     then execute the install command again

```
python setup.py install
```

  The above two commands will install the application in your local python distribution (the one that resembles to the executed `python` command) along with all the required dependencies. They will also install the application executables, which can then be called from the command line (see *Running the application* below for more details).

Running the application

Make sure you have installed all *Runtime dependencies*.

There are two ways to run the application. The easier way is via the user interface. However, you can also run the application via the command line. The installation procedure described above installs both.

Open a command prompt (Windows) or a terminal (Linux/MacOS).

- To run the graphical user interface, execute `amtt-gui`.
- To run the command line interface, execute `amtt`.

Building an application bundle

If you want to distribute the application among many users, the easiest way is to create an application bundle, i.e. a standalone executable, that is ready-to-run on any machine.

This bundle will have no dependencies whatsoever and will run on a given machine, even if no Python version is installed on it.

To create an application bundle:

1. Make sure you have followed the installation procedure in *Installing the application*.
2. Install `PyInstaller`:

- via `pip` (easiest, trouble-free method):

```
pip install PyInstaller
```

- without `pip`:
    1. Download the latest PyInstaller package from PyPi: https://pypi.python.org/pypi/PyInstaller/
    2. Extract it, `cd` to the extracted directory and execute:

```
python setup.py install
```

3. While in the project root folder (where `setup.py` resides), execute:

```
python setup.py build_standalone
```

After the process is complete, you will find the bundle in the `dist` directory.

Please note that the build process builds a bundle for the operating system you are currently using. It is not possible to cross-compile and create application bundles for other operating systems.

Therefore, you can only build for Linux from a Linux installation, for MacOS from a MacOS installation and for Windows from a Windows installation.

CHAPTER 4

Importing the generated files

Normally, the generated files are ready to import to the target software. Follow the instructions that are outlined below for each target.

Isograph Availability Workbench

The files generated for Isograph are in XML format. To import them you will need to create a new, empty project.

The steps are outlined below:

1. Choose **File** -> **Import**.
2. In the **Database** tab, select **Type**: XML File.
3. In **File**, click browse and select the XML file generated by the translator.
4. In the **Schema** tab you can see the generated schema (the tables and the columns of each table).
5. Go to the **Table Matches** tab and click "**Auto match**". The external tables should match automatically.
6. Go to the **Column Matches** tab. For each entry in the "**Table match**" drop-down menu, select "**Auto match**". The external columns should match automatically.
7. Save the import template if you wish, by clicking **Save**.
8. Finally, click **Import**. The model should get imported.