# PyPI Repositories

## Overview

Artifactory fully supports PyPI repositories providing:

1. The ability to provision PyPI packages from Artifactory to the `pip` command line tool from all repository types.
2. Calculation of Metadata for PyPI packages hosted in Artifactory's local repositories.
3. Access to remote PyPI repositories (such as `https://pypi.org/`) through Remote Repositories which provide proxy and caching functionality.
4. The ability to access multiple PyPI repositories from a single URL by aggregating them under a Virtual Repository.
5. Compatibility with the setuptools and its predecessor distutils libraries for uploading PyPI packages.

**Running on Windows**

To use Artifactory PyPI repositories on Windows, make sure to set the required environment variables for Python and Pip.

Note that on Windows platforms, *`%HOME%\pip\pip.ini`* replaces the *`pip.conf`* file described in the sections below and should be reachable through your HOME path.

**Page Contents**

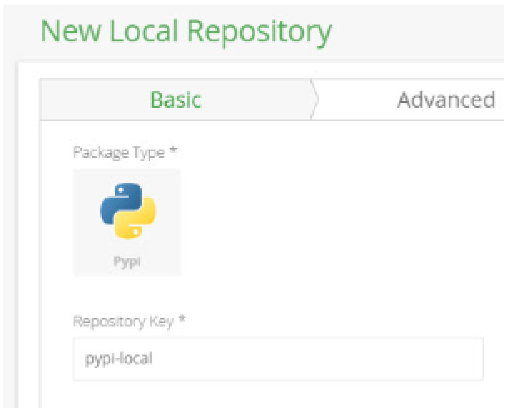
**Integration Benefits**

JFrog Artifactory and PyPI Repositories

## Configuration

### Local Repositories

To create a new PyPI local repository, in the New Local Repository screen, set the **Package Type** to **PyPI.**



Repository Key: pypi-local

### Remote Repositories

**Requires Artifactory 5.8.10, 5.9.7, 5.10.4 and above or 6.x**

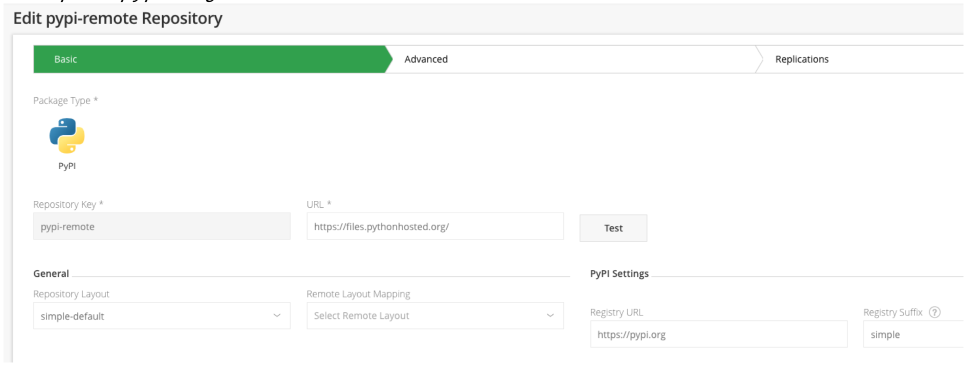
Due to several changes introduced to PyPI in April 2018, to proxy *`https://pypi.org/`* using a remote repository, you need to work with Artifactory 5.8.10, 5.9.7, 5.10.4 or above (including 6.x).

A Remote Repository defined in Artifactory serves as a caching proxy for a registry managed at a remote URL such as https://pypi.python.org/.

Artifacts (such as .whl files) requested from a remote repository are cached on demand. You can remove downloaded artifacts from the remote repository cache, however you can not manually deploy artifacts to a remote PyPI repository.

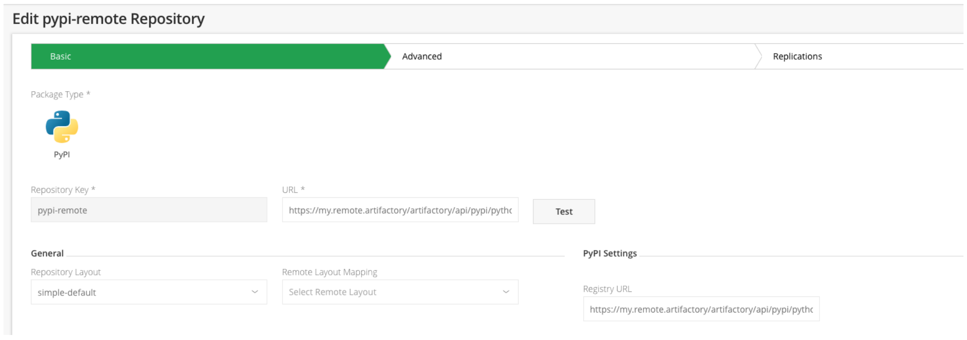
To create a repository to proxy a remote PyPI repository follow the steps below:

1. In the **Admin** module under **Repositories | Remote,** select "New"
2. Set the **Package Type** to **PyPI** and enter the **Repository Key** value.

3. The URL and Registry URL settings depend on whether you are proxying the public external PyPI repository, or a PyPI repository hosted on another Artifactory server.
   **For a public, external PyPI repository:** Change the **URL** field to `https://files.pythonhosted.org/`, and set the **Registry URL** field to `https://pypi.org/` as shown below:
   

   Repository Key *: pypi-remote
   URL *: https://files.pythonhosted.org/
   Repository Layout: simple-default
   Remote Layout Mapping: Select Remote Layout
   Registry URL: https://pypi.org
   Registry Suffix: simple

   **For a PyPI repository hosted on another Artifactory instance:** Set the remote repository's PyPI API URL in both the **URL** field and the **Registry URL** field. For example, to proxy a PyPI repository called "*python-project*" hosted by an Artifactory instance at `https://my.remote.artifactory/artifactory/`, you would set both the **URL** field and the **Registry URL** to https://my.remote.artifactory/artifactory/api/pypi/python-project as shown below:

   Repository Key *: pypi-remote
   URL *: https://my.remote.artifactory/artifactory/api/pypi/pytho
   Repository Layout: simple-default
   Remote Layout Mapping: Select Remote Layout
   Registry URL: https://my.remote.artifactory/artifactory/api/pypi/pytho

   > **PyPI remote repository URL**
   >
   > You should not include "/pypi" or "/simple" in the the PyPI remote repository URL. These suffixes are added by Artifactory when accessing the remote repository.
   >
   > If you use a custom PyPI remote repository, you need to make sure it has a simple index (directory listing style) accessible by `<URL>/simple`.

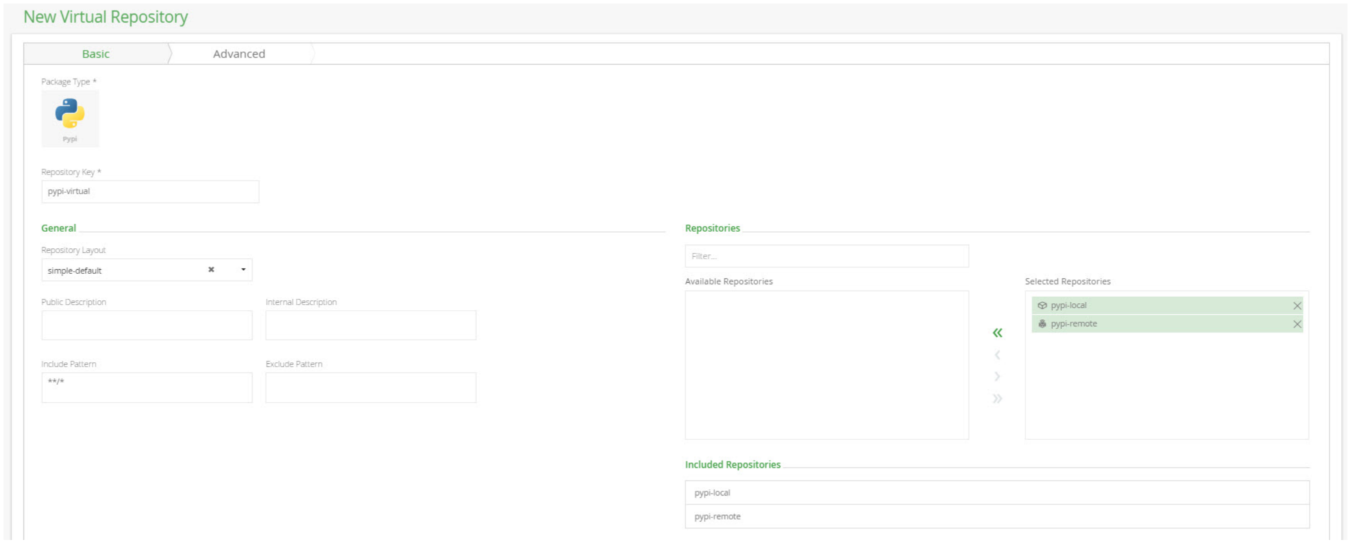
   > **Registry URL field is version specific**
   >
   > The `Registry URL` field is only available in Artifactory 5.10.3 and above.
   >
   > For patch versions provided for Artifactory 5.8.9 and above and Artifactory 5.9.5 and above, to support the new PyPI structure, the only change you need to make is to set the URL field to https://pypi.org. The https://files.pythonhosted.org is automatically extracted from a system property, `artifactory.pypi.default.download.url` which is set to this value by default.

4. Click "Save & Finish"

   > **Remote Artifactory**
   >
   > If the remote repository is also managed by an Artifactory server, then you need to point to its PyPI API URL in both **URL** field and **Registry URL** field. For example `http://my.remote.artifactory/artifactory/api/pypi/python-project`

## Virtual Repositories

A Virtual Repository defined in Artifactory aggregates packages from both local and remote repositories.
This allows you to access both locally hosted PyPI packages and remote proxied PyPI repositories from a single URL defined for the virtual repository.
To define a virtual PyPI repository, create virtual repository, set its **Package Type** to be PyPI, select the underlying local and remote PyPI repositories to include in the **Basic** settings tab, click "Save & Finish".



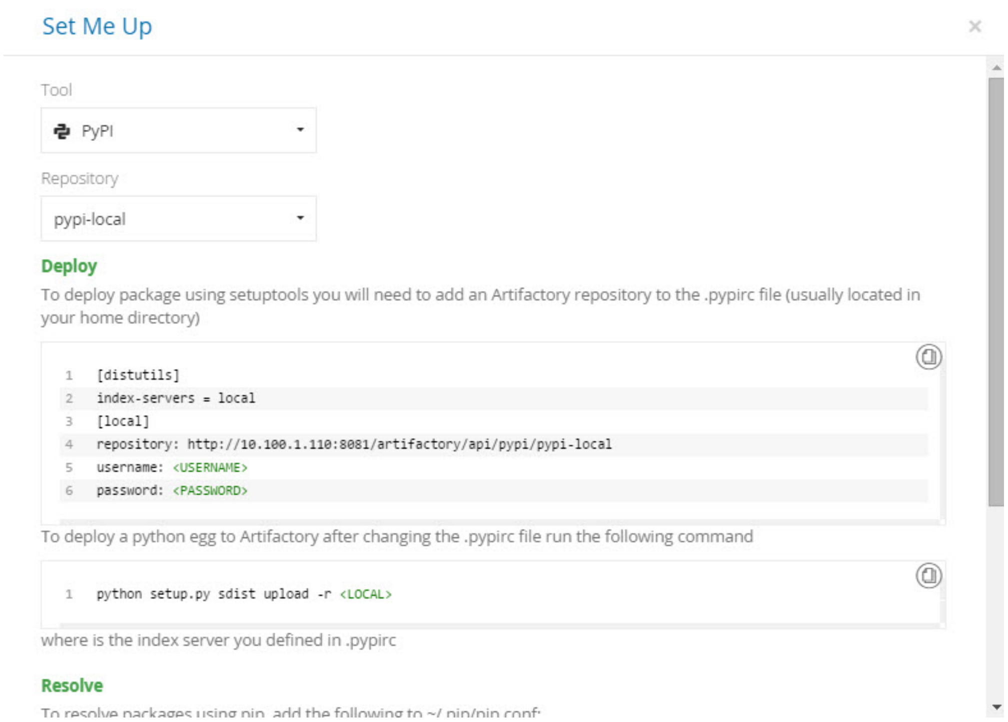
Repository Key * pypi-virtual
Repository Layout simple-default
Public Description
Internal Description
Include Pattern **/*
Exclude Pattern
Selected Repositories: pypi-local, pypi-remote
Included Repositories: pypi-local, pypi-remote

---

## Resolving from Artifactory Using PIP

To install the `pip` command line tool refer to pip documentation pages. We recommend using virtualenv to separate your environment when installing PIP.

To display code snippets you can use to configure `pip` and `setup.py` to use your PyPI repository, select the repository and then click **Set Me Up.**



To deploy package using setuptools you will need to add an Artifactory repository to the .pypirc file (usually located in your home directory)

```
[distutils]
index-servers = local
[local]
repository: http://10.100.1.110:8081/artifactory/api/pypi/pypi-local
username: <USERNAME>
password: <PASSWORD>
```

To deploy a python egg to Artifactory after changing the .pypirc file run the following command

```
python setup.py sdist upload -r <LOCAL>
```

where is the index server you defined in .pypirc

To resolve packages using pip, add the following to ~/.pip/pip.conf:

### Specifying the Repository on the Command Line

**Index URL**

When accessing a PyPI repository through Artifactory, the repository URL must be prefixed with **api/pypi** in the path. This applies to all `pip` commands and `distutils` URLs including `pip install`.

When using `pip` to resolve PyPI packages it must point to *`<Artifactory URL>/api/pypi/<repository key>/`**`simple`*** .

For example, if you are using Artifactory standalone or as a local service, you would access your PyPI repositories using the following URL:

*`http://localhost:8081/artifactory/`**`api/pypi/`**`<repository key>/`**`simple`***

Or, if you are using Artifactory SaaS, the URL would be:

*`https://<server name>.jfrog.io/<server name>/`**`api/pypi/`**`<repository key>/`**`simple`***

Once pip is installed, it can be used to specify the URL of the repository from which to resolve:

**Installing with full repository URL**

```
$ pip install frog-bar -i http://localhost:8081/artifactory/api/pypi/pypi-local/simple
```

### Using Credentials

Due to it's design, pip does not support reading credentials from a file. Credentials can be supplied as part of the URL, for example *`http://`**`<username>:<password>`**`@localhost:8081/artifactory/api/pypi/pypi-local/simple`*. The password can be omitted (with the preceding colon), and in this case, the user will be prompted to enter credentials interactively.

## Using a Configuration File

Aliases for different repositories can be specified through a pip configuration file, *~/.pip/pip.conf* . The file contains configuration parameters per repository, for example:

**~/.pip/pip.conf**

```
[global]
index-url = http://user:password@localhost:8081/artifactory/api/pypi/pypi-virtual/simple
```

For more information, please refer to PIP User Guide.

## Using a Requirements File

A requirements file contains a list of packages to install. Usually these are dependencies for the current package. It can be created manually or using the *`pip freeze`* command. The index URL can be specified in the first line of the file, For example:

**requirements.txt**

```
--index-url http://localhost:8081/artifactory/api/pypi/pypi-local/simple
PyYAML==3.11
argparse==1.2.1
frog-bar==0.2
frog-fu==0.2a
nltk==2.0.4
wsgiref==0.1.2
```

---

## Publishing to Artifactory

## Using distutils or setuptools

**setuptools vs. distutils and python versions**

Artifactory is agnostic to whether you use `setuptools` or `distutils`, and also to the version or implementation of Python your project uses.

The following instruction were written for Python 2.7 and `setuptools` in mind. Using different version of Python, or different tools such `zest, distutils` and others may require minor modification to the instructions below.

Uploading to Artifactory using a *setup.py* script is supported in a similar way to uploading to PyPI. First, you need to add Artifactory as an index server for your user.

For instructions on using *setuptools* to package Python projects and create a *setup.py* script, please refer to the  setuptools documentation and this tutorial project.

### Create the $HOME/.pypirc File

To upload to Artifactory, an entry for each repository needs to be made in *$HOME/.pypirc* as follows:

```
[distutils]
index-servers =
    local
    pypi

[pypi]
repository: https://pypi.org/pypi
username: mrBagthrope
password: notToBeSeen

[local]
repository: http://localhost:8081/artifactory/api/pypi/pypi-local
username: admin
password: password
```

Notice that the URL does not end with `/simple`.

**The HOME environment variable**

*`setuptools`* requires that the `.pypirc` file be found under *`$HOME/.pypirc,`* using the `HOME` environment variable.

On unix-like systems this is usually set by your system to */home/yourusername/* but in certain environments such as build servers you will have to set it manually.

On Windows it must be set manually.

### Uploading

After creating a *.pypirc* file and a *setup.py* script at the root of your project, you can upload your egg (tar.gz) packages as follows:

```
~/python_project $ python setup.py sdist upload -r local
```

If you are using wheel (whl) you can upload your packaged as follows:

```
~/python_project $ python setup.py bdist_wheel upload -r local
```

Or if you wish to use both egg (tar.gz) and wheel (whl), you can upload them as follows:

```
~/python_project $ python setup.py sdist bdist_wheel upload -r local
```

Where **local** is the name of the section in your *.pypirc* file that points to your Artifactory PyPI repository.

**Default upload**

By default, both `setuptools` and `distutils` will upload to *`https://pypi.org/pypi`* if no repository is specified.

**The 'register' command should be omitted**

When uploading directly to *pypi.org,* the documentation states that your package must first be registered by calling `python setup.py` **`register.`**

When uploading to Artifactory this is neither required nor supported and **should be omitted**.

### Publishing Manually Using the Web UI or REST

PyPI packages can also be uploaded manually using the Web UI or the Artifactory REST API. For Artifactory to handle those packages correctly as PyPI packages they must be uploaded with **pypi.name** and **pypi.version** **Properties** .

**Automatic extraction of properties**

While indexing the newly uploaded packages Artifactory will automatically try to extract required properties from the package metadata saved in the file. Note that not all supported files can be extracted.

Currently, only *zip, tar, tgz, tar.gz, tar.bz2, egg* and *whl* files can be extracted for metadata.

In addition, indexing starts after a 60 second quiet period, counting from the last upload to the current repository.

## Searching for PyPI Packages

### Using PIP

Artifactory supports search using *pip*'s *search* command in local, remote and virtual repositories. For example:

**pip search**

```
$ pip search frog-fu --index http://localhost:8081/artifactory/api/pypi/pypi-virtual/
frog-fu                   - 0.2a
  INSTALLED: 0.2a (latest)

$ pip search irbench --index http://localhost:8081/artifactory/api/pypi/pypi-virtual/
irbench                   - Image Retrieval Benchmark.
```

In this example **frog-fu** is a locally installed package, while **irbench** is found at *pypi.org*, both repositories aggregated by the *pypi-virtual* repository.

**Specifying the index**

When using the search command, the index should be specified explicitly (without the `/simple` at the end), as pip will ignore the **index-url** variable in its *pip.conf* file.

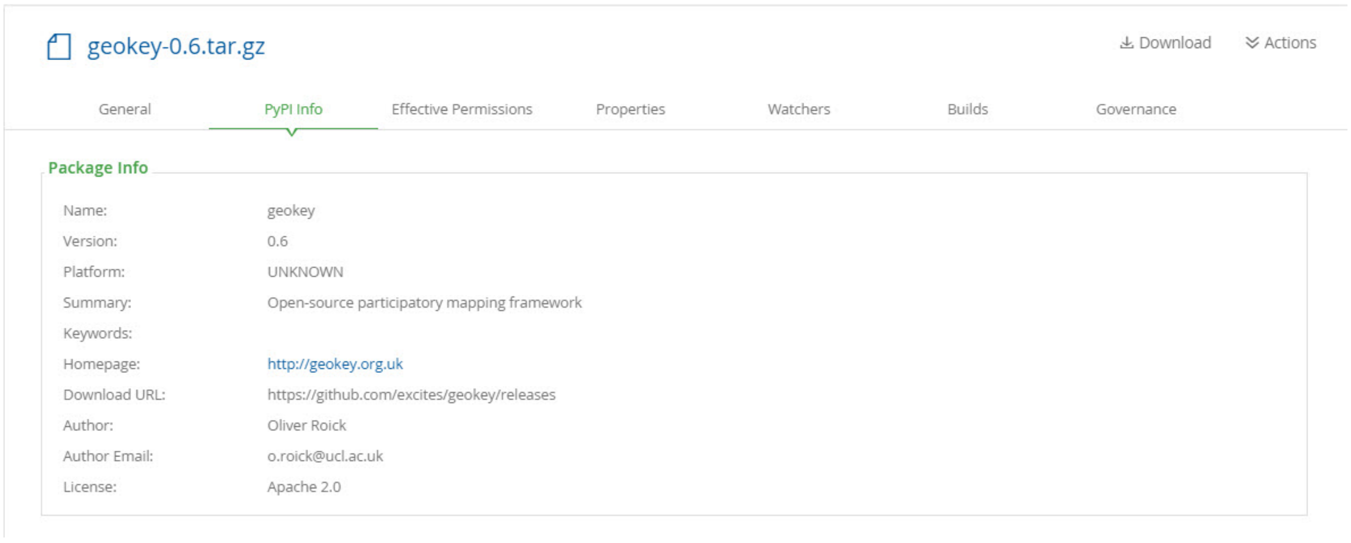
### Artifactory Search

PyPI packages can also be searched for using Artifactory's Property Search. All PyPI packages have the properties **pypi.name**, **pypi.version** and **pypi.summary** set by the uploading client, or later during indexing for supported file types.

## Viewing Metadata of PyPI Packages

Artifactory lets you view selected metadata of a PyPI package directly from the UI.

In the **Artifacts** module **Tree Browser,** drill down to select the file you want to inspect. The metadata is displayed in the **PyPI Info** tab.


geokey-0.6.tar.gz

Package Info

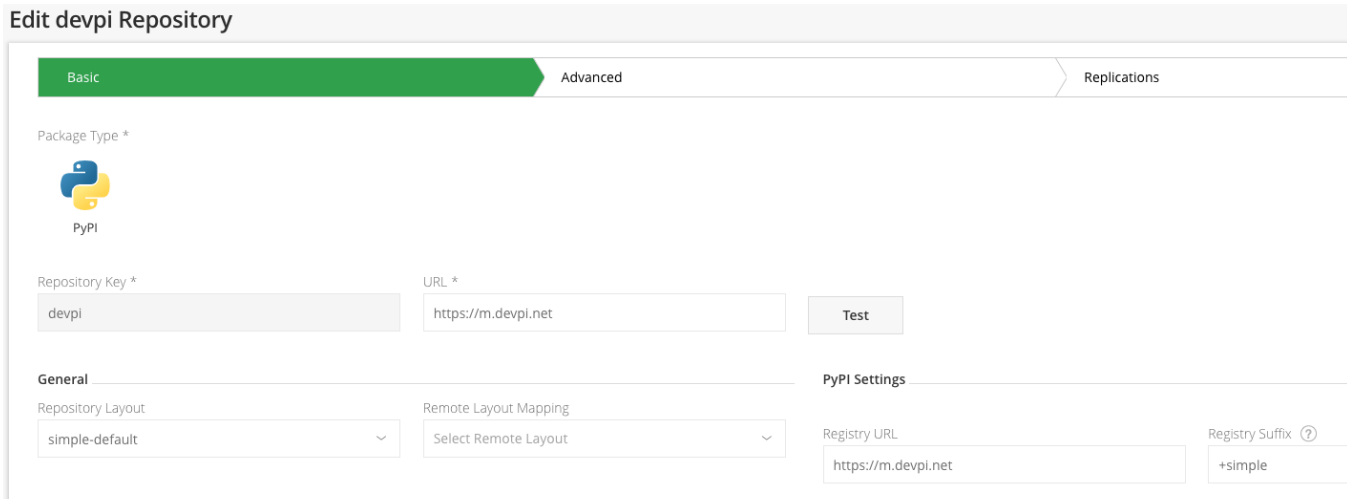
| | |
|---|---|
| Name: | geokey |
| Version: | 0.6 |
| Platform: | UNKNOWN |
| Summary: | Open-source participatory mapping framework |
| Keywords: | |
| Homepage: | http://geokey.org.uk |
| Download URL: | https://github.com/excites/geokey/releases |
| Author: | Oliver Roick |
| Author Email: | o.roick@ucl.ac.uk |
| License: | Apache 2.0 |

## Working with Remote Repositories with the Custom Registry Suffix

From Artifactory 6.10.3, you can set a custom suffix instead of the default simple like in cases of DevPi.

To set the devpi registry suffix to the server suffix:

Use the root URL in the URL and Registry URL. For example: http://m.devpi.net.


Edit devpi Repository

Package Type *: PyPI

Repository Key *: devpi

URL *: https://m.devpi.net

General

Repository Layout: simple-default

Remote Layout Mapping: Select Remote Layout

PyPI Settings

Registry URL: https://m.devpi.net

Registry Suffix: +simple

In order to search, include the required scope in the index URL (as in the devpi example, it could be root/pypi).

```
$ pip search frog-fu --index http://localhost:8081/artifactory/api/pypi/devpi/root/pypi/
```

To install, include the desired scope in the index url (like in devpi example, it could be root/pypi)

```
$ pip install frog-bar -i http://localhost:8081/artifactory/api/pypi/devpi/root/pypi/simple
```

## Watch the Screencast