Zope Project and Community

Sep 21, 2023

Contents

Zope is a free and open source web application server written in the object-oriented programming language Python. Since its release in 1998, Zope continues to grow into many distinct applications, frameworks, libraries and tools. *The World of Zope* highlights the most important components.

Zope community projects are hosted in the Zope Foundation organization on GitHub.

CHAPTER 1

Overview

What is Zope and its related projects?

1.1 The World of Zope

A Bit of History

In 1996 Jim Fulton, now Zope Corporation CTO, was drafted to teach a class on common gateway interface (CGI) programming, despite not knowing very much about the subject. CGI programming is a commonly-used web development model that allows developers to construct dynamic web sites. Traveling to the class, Jim studied all the existing documentation on CGI. On the way back, Jim considered what he didn't like about traditional CGI-based programming environments. From these initial musings the core of Zope was written while flying back from the CGI class.

Zope Corporation (then known as Digital Creations) went on to release three open-source software packages to support web publishing: Bobo, Document Template, and BoboPOS. These packages were written in a language called Python, and provided a web publishing facility, text templating, and an object database, respectively. Digital Creations developed a commercial application server based on these components and called it Principia. In November of 1998, investor Hadar Pedhazur convinced Digital Creations to open source Principia, thereby creating the foundation for the Zope application server.

In 2001, the Zope community began working on a component architecture for Zope, but after several years they ended up with something much more: Zope 3. While Zope 2 was powerful and popular, Zope 3 was designed to bring web application development to the next level.

During more than a decade Zope Corp. and the Zope Community have grown an outstanding set of products and technologies, influencing the general development of Python based Web application servers and tools.

1.1.1 Frameworks

ZCA The Zope Component Architecture provides facilities for defining, registering and looking up components. It's perfect for building enterprise applications based on loosely coupled components.

More information at the zope.component documentation and zope.interface documentation.

ZTK The Zope Toolkit (ZTK) is a set of libraries intended for reuse by projects to develop web applications or web frameworks. The ZCA is part of it.

More information at the Zopetoolkit documentation.

ZPT Zope Page Templates is Zope's templating mechanism.

More information at the https://zope.readthedocs.io/en/latest/zopebook/AppendixC.html. An alternative implementation is provided by Chameleon.

CMF The Content Management Framework (CMF) for Zope provides libraries for building content management applications together with the Zope Application Server.

Repoze Repoze integrates Zope technologies with WSGI and reusable Python middleware.

More information at https://repoze.org

1.1.2 Databases

ZODB The Z Object Database (ZODB) is a native object database, that stores your objects while allowing you to work with any paradigms that can be expressed in Python.

More information at https://zodb.org

1.1.3 Application Servers

Zope Zope is a Python-based application server for building secure and highly scalable web applications.

More information at https://zope.readthedocs.io

1.1.4 Tools

Buildout Buildout is a Python-based build system for creating, assembling and deploying applications from multiple parts, some of which may be non-Python-based.

More information at https://buildout.org

CHAPTER 2

Documentation

An overview over the most important documentation resources.

2.1 Resources

This site links to various documentation sources for Zope and related software. All Zope software is maintained on GitHub at https://github.com/zopefoundation.

2.1.1 Zope

The Zope web application server, formerly called Zope 2, is under active development. The latest major release was version 5.

- Zope documentation
- Zope Change log
- Zope source code
- *Zope release schedule*
- The Zope Book
- The Zope Developers Guide
- zope_secrets/index (written for Zope 2)

2.1.2 ZODB

The ZODB object database is the default storage engine for Zope.

- ZODB documentation
- ZODB Change log

- ZODB source code
- ZODB tutorial
- ZODB/ZEO programming guide
- API documentation
- ZODB articles

2.1.3 Zope Toolkit (ZTK)

The Zope Toolkit represents the various library packages that were created for Zope 3/Bluebream. Many of them live on in Zope itself.

- Zope Toolkit documentation

CHAPTER 3

Community

The Zope community is one of the largest and most professional open-source communities worldwide.

3.1 Zope release schedule

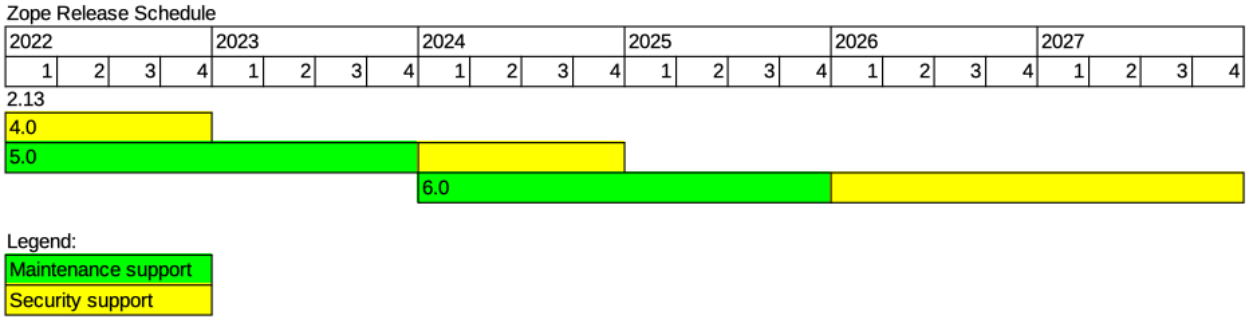
The Zope development and support roadmap. **Last updated: September 2023**

If you use Plone, please visit the Plone release schedule for all Plone-specific details. Support for Zope releases aims to track support for the respective Plone version(s) built on top of them.

Zope currently has two release managers, Michael Howitz and Jens Vagelpohl.

Table of contents:

3.1.1 Support schedule



Series	Latest release	Python versions	Status	First release	End support: maintenance	End support: security
2.13	2.13.30	2.7	end of life	2010-11-05	2019-02-09	2020-12-31
4.0	4.8.7	2.7, 3.5-3.8	end of life	2019-05-10	2021-09-30	2022-12-31
5.0	5.8.5	3.7-3.11[1][2][3]	maintenance	2020-10-08	2024-12-31[4]	2025-12-31[4]
6.0	-	TBD	unreleased	TBD[5]	2026-12-31[6]	2027-12-31[6]

3.1.2 Definitions

On this page, we use these terms:

- *End of Life*: No further development is done and no releases are published.
- *Maintenance support*: security and other bug fixes and small new features are added. Around the end of maintenance support, a last release will be done.
- *Security support*: security fixes will be made available for this series.
- *Major* version: 4.0, 5.0
- *Minor* version: 4.5, 5.7
- *Bugfix* version: 4.4.1, 5.6.1

3.1.3 Release cadence

This is the expected release cadence for Zope:

- We aim to release a new minor version roughly every 2-6 months.
- We aim to release a new major version roughly every 2-3 years.
- Planning for a new minor or major version usually starts after a release has been done. Currently, there is no clear calendar with expected dates.

[1] Python 3.10 support added with release 5.4.
[2] Python 3.11 support added with release 5.7.
[3] Python 3.6 support removed with release 5.8.
[4] Tentative: Final support deadlines for the Zope 5 series will be set when Zope 6 is released. Zope 5 will enjoy full maintenance support at least until Zope 6 is released and Security support for at least one year after that.
[5] There is no release date for Zope 6 yet because there hasn't been any backwards-incompatible changes warranting a new major release yet.
[6] Purely speculative and subject to change after a release date has been set for Zope 6.

3.1.4 Maintenance policy

These are the general rules for maintenance support for Zope:

- Each major release series gets at least two years of maintenance support, most likely more.
- Maintenance support is extended at least until a new major release is published.
- If a new major release is published after the two year maintenance support window, the previous release series may be downgraded to Security support soon thereafter.

3.1.5 Security policy

These are the general rules for security support for Zope:

- Each major release gets at least three years of security support.
- There is a security support overlap of at least one year between major versions. This gives you time to migrate to a new major version.
- You have to be on the last minor release to have the full period of security support.
- Each minor release gets at least one year of security support.
- Note that underlying versions of Python and other dependencies may be out of security support sooner.

3.1.6 Versioning policy

Zope and Zope ecosystem packages follow semantic versioning: a new feature means a minor release, a breaking change means a major release for a package.

- 5.0.1 and 5.0.2 are examples of bugfix releases. These contain bug fixes. They may contain new code that should not affect stability, for example updated dependency versions. Add-ons should not need changes.
- 5.1 and 5.2 are minor releases. These contain larger changes or new backwards-compatible features. Occasionally a minor release may drop support for a Python version, if this version is no longer supported by the Python community. Minor releases may also add support for newer Python versions. Both dropped and added Python version support will be clearly visible in the support table above. Add-ons may need to be changed to use the new features or Python version. It should be easy to make the same version of an add-on work in all minor releases.
- 4 and 5 are major releases. These contain breaking changes. Add-ons are likely to need changes. It may be hard to have one version of an add-on work across two major versions.

Note that Zope is both a framework and a product, and you can use a lot of community add-ons and own code on top of it. This means that even a seemingly innocent fix like adding a class in HTML could break something for someone. We cannot foresee everything. Please be a bit forgiving.

3.1.7 General advice for all Zope versions

- Zope 4 has reached end-of-life status. Migrate to Zope 5 as soon as you can.
- Use the **highest Python version** that is supported by your Zope version. For release schedules of core Python, see https://www.python.org/downloads/
- Zope 4 and Zope 5 users should upgrade to at least Python 3.7 **as soon as possible** to mitigate an unfixed security issue in the waitress WSGI server.

- Regularly check the Zope release page at https://pypi.org/project/Zope/ to see if any security fixes are available for your Zope version.

3.1.8 Supported Zope versions

Zope 5

- First official release: 5.0, October 2020
- Current release: 5.8.3, June 2023
- Next release expected: early 2023, roughly every 2-6 months.
- Supports Python 3.7, 3.8, 3.9, 3.10 and 3.11.
  - Python 3.6 support was removed in release 5.8.
  - Python 3.10 support was added in release 5.4.
  - Python 3.11 support was added in release 5.7.
- Used by Plone 6.0
- Maintenance support until at least December 31, 2023.
- Security support until at least December 31, 2024.

3.1.9 No longer supported Zope versions

Zope 4

Zope 4 supports Python 2 and Python 3. It is meant to act as a bridge for those upgrading applications from Zope 2. Once you are on Zope 4 and Python 3 the next step to Zope 5 is painless and you should migrate **immediately**.

- First official release: 4.0, May 2019
- Current release: 4.8.7, January 2023
- Next release expected: Zope 4 has reached end-of-life. There **may** be sporadic releases to fix urgent issues. Please move to Zope 5.
- Supports Python 2.7, 3.5, 3.6, 3.7 and 3.8.
  - Please note that Python 2.7, 3.5 and 3.6 have reached end of life, you should use Python 3.7 at least.
- Used by Plone 5.2
- Maintenance support has ended September 30, 2021
- Security support has ended on December 31, 2022.

Zope 2.13

- First official release: 2.13.0, November 2010
- Last release: 2.13.30, February 2020
- Supports Python 2.7
- Used by Plone 4.3, 5.0 and 5.1
- Maintenance support has ended on February 9, 2019

- Security support has ended on December 31, 2020

3.2 Community

The Zope community is one of the Largest and most professional open-source communities worldwide.

3.2.1 Getting in touch

Security To report a security problem, please contact the Plone Security Team via email (security@plone.org).

E-Mail Several Zope-related email lists are maintained on the Zope community list server.

Web sites and forums The Plone community forums have a lively Zope discussion category.

All projects under the Zope umbrella are maintained under the GitHub zopefoundation organization. Each project has an issue tracker for questions, feature requests and bug reports.

IRC freenode.net hosts lots of Zope and Zope products / application related IRC channels. Visit irc.freenode.net and try one of the Following channels: #zope, #zope.de, #plone, #grok

Planets News collections from different Zope related blogs, like Planet Plone and Planet Python .

3.2.2 The Zope Foundation

The Zope Foundation had the goal to promote, maintain, and develop the Zope platform, which it did successfully for many years. It is merging with the Plone Foundation and passed its responsibilities to the Plone Foundation in 2018.

3.2.3 The Plone Foundation

Our community includes the open source community of contributors to the Zope software, contributors to the documentation and web infrastructure, as well as the community of businesses and organizations that use Zope.

The Plone Foundation is the copyright holder of the Zope software and many extensions and associated software maintained in the *zopefoundation* GitHub organization at https://www.github.com/zopefoundation.

For more information, visit https://plone.org/foundation

3.3 Developer Information

This guide is for developers who are working **on** the various Zope-related software projects, rather than for developers who work **with** those projects' released software to build applications. Developers in the latter category should visit the general *Resources* section instead.

The projects under the zopefoundation GitHub organization are open source and welcome contributions in different forms:

- bug reports
- code improvements and bug fixes
- documentation improvements
- pull request reviews

For any changes in the repository besides trivial typo fixes you are required to sign the contributor agreement. See *Becoming a Contributor* for details.

Visit https://github.com/zopefoundation to see and browse all repositories maintained by the Zope developer community.

The overview at https://zope3.pov.lt/py3/gha.html lists the supported Python versions and the build status for all Zope Foundation projects (the overview is not guaranteed to be complete and accurate).

3.3.1 Reporting Bugs

The Zope developer community uses the GitHub zopefoundation organization to host all projects. Each repository has its own issue tracker for bug reports, feature requests or questions. For example, Zope developers track bugs within Zope using the `Zope` project on GitHub:

> https://github.com/zopefoundation/Zope/issues

How to write a good bug report

Here's how you get help most effectively:

- Before filing a bug report, make sure you can reproduce the issue on a plain vanilla Zope installation. If you cannot, then it's probably not a Zope bug.
- Please provide as many details as possible, such as...
  - Zope version
  - Python version
  - Operating system
  - Full Python tracebacks if you have them
- It's helpful to describe bugs in terms of expected and actual outcomes, that makes it easier to follow along and reproduce it: *In the Zope ZMI I was clicking on the Interfaces tab of a DTML Method, and instead of seeing the Interfaces tab I saw the following error output: ...*
- If the developers need more information and follow up with questions, please make sure to answer in a timely manner, especially if the issue report is marked with the feedback required tag. If we don't hear from you after 2-3 weeks the issue may be closed.

3.3.2 Mailing Lists

Active Zope and Zope Toolkit developers should subscribe to the zope-dev mailing list.

Depending on your interest you can subscribe to other Zope mailing lists as well or visit discontinued and archived list archives:

> https://mail.zope.dev/

3.3.3 Becoming a Contributor

Applying for Committer Status

Detailed instructions for gaining commit rights to the Zope repositories are at https://plone.org/foundation/contributors-agreement/contributors-agreement-explained.

Note: Exisiting Plone contributors are required to submit the contributor agreement again to gain access to the Zope repositories.

1. Print, fill out and sign the Plone Contributor Agreement. On page 2 enter *Zope* in the field for *program*.
2. Scan the agreement and submit it by email to agreements@plone.org
3. You will be added to the GitHub repository shortly.

Subscribe to the Mailing Lists

Please subscribe to the *Mailing Lists* which correspond to your area(s) of interest.

3.3.4 Developer guidelines

Note: Any code contributions that are more than trivial typo fixes require a signed contributor agreement. Please see *Becoming a Contributor* for details.

Pull requests

Pull requests for bug fixes, features or documentation are always welcome. Here's how you can make it easy to accept your contribution:

- Please create branches in the package's repository instead of forking the package. That way other contributors can help easily.
- Respect the conventions you find in the package you're contributing to. This includes code style, tools used to run and configure tests, the package structure and its management. Coordinate with other developers for that package before changing any of these.
- Keep cosmetic and code changes apart. If necessary, put them in separate pull requests.
- If your code is fixing a bug it should have unit tests that exercise the bug and pass with your fix.
- Make sure all unit and functional tests pass before creating a pull request. Most repositories will have a `tox.ini` configuration file and you can run all tests at once using `tox`.
- "vendoring in" of third-party code is not allowed unless all contributors to that code have also signed a contributor agreement.

Please wait for approval from at least one other contributor before merging your code, which you can do yourself.

Coding Standards

As a general rule, projects in our repositories abide by the following standards:

- Code in Zope-related projects should generally conform to PEP 8 coding style. In particular, *Python code should never exceed 80 columns*. Existing code should be updated to this standard only conservatively, to ease integration of patches made against older releases.
- Project *master* branches and all branches that are officially supported should be kept in "ready-to-release" state: all unit tests pass, changelogs are kept updated, etc.

- All features and APIs should be fully documented using Sphinx.
- Solid release management, including releases to PyPI corresponding to "pristine" tags, detailed change logs, etc.

While some older projects may not be completely in line with this culture, we are committed to moving them all closer with any change. As a corollary: if you are submitting a patch to a project in this repository, and you want to expedite its acceptance, ensure that your patch maintains or improves the target project's conformance to these goals.

Package layout

Each project should consist of a single, top-level project folder in GitHub. Because we are mostly working on Python code here, projects are normally arranged as a `distutils` project, e.g.:

```
<directory>
- setup.py
- README.txt
- CHANGES.txt
+ docs/
  - Makefile
  - conf.py
  - index.rst
  - api.rst
  + .static/
  + .build/
  + .templates/
+ package/
  - __init__.py
  + subpacakge/
    - __init__.py
    - module.py
    + tests/
      - __init__.py
      - test_module.py
    + templates/
      - template.pt
```

Many packages place the first-level `package` directory in a `src` subdirectory inside the checkout.

Running Tests using `zc.buildout`

Most projects in the Zope repository are already configured to support building in-place and running tests using `zc.buildout`.

```
$ git clone https://github.com/zopefoundation/Zope.git
$ cd Zope
$ python3 -m venv .
$ bin/pip install -U pip wheel zc.buildout
$ bin/buildout
...
$ bin/test
...
```

3.3.5 Documenting code with Sphinx

Note: This outline needs fleshing out.

One of the most valuable contributions that non-core developers make is helping keep a project's documentation up-to-date and useful.

Each Zope packages should have its own documentation, managed within the project itself. This documentation is maintained as a set of files in the `docs` subdirectory of the project, containing source files in "restructured text" format (see the reStructuredText Refererence) as well as control files which convert those source texts into HTML, Latex, PDF, etc., using Sphinx (see the Sphinx manual).

The top-level document is conventionally `docs/index.rst`. This document should explain the purpose of the project, and will often link to other documents outlining usage, APIs, etc.

Building the HTML Documentation

If you have Sphinx installed somewhere on your system, the `docs/Makefile` can be used to build various flavors of documentation. Assuming that the Sphinx scripts are installed in `/opt/Sphinx/bin`:

```
$ cd /tmp
$ git clone https://github.com/zopefoundation/Zope.git
$ cd Zope/docs
$ PATH=/opt/Sphinx/bin:$PATH make html
```

After running that command, you can view the generated docs in your browser, just open `/tmp/Zope/docs/_build/html/index.html`

Re-running the `make html` command after making changes to the docs lets you see the rendered HTML corresponding to your updates.

You can also use the Sphinx `doctest` extension to test example code snippets in the documentation:

```
$ PATH=/opt/Sphinx/bin:$PATH make doctest
```

3.3.6 Python 2 support

Note: This document is relevant for packages that are either direct dependencies of Zope version 4 or popular add-ons for Zope version 4 and their dependencies.

Zope 4 will retain full Python 2 (and Python 3.5) compatibility throughout its lifetime. That means all its direct dependencies and (ideally) many popular add-on packages should also continue supporting Python 2 and Python 3.5 until Zope 4 reaches end-of-life status.

When to drop support

Now it the time to drop support for Python 2.7 up to 3.6 as they are all no longer maintained by the Python community and the maintencance burden has become too heavy.

How to drop support

When dropping Python 2 support you should also "upgrade" the code to support more modern practices, like using f-strings.

Prerequisites

- Installed packages the code examples below expect it to be on `$PATH`.
  - `pyupgrade`
  - `check-python-versions`
  - `zest.releaser`
- Repository is under control of `meta/config` thus it has a `.meta.toml` file.

Typical steps

The following steps are necessary to remove support for Python 2.7 up to 3.6 from a package:

- Update version number to next major version:

```
$ bumpversion --breaking
$ addchangelogentry "Drop support for Python 2.7, 3.5, 3.6."
```

- Remove Python 2 support from the Trove classifiers in `setup.py` and update `python_requires`:

```
$ check-python-versions --drop 2.7,3.5,3.6
```

- `.meta.toml`
  - Remove Python 2 specific settings
- Run `meta/config` to remove Python 2 support from other configuration files and create a branch:

```
$ bin/python config-package.py <PATH-TO-REPOS> --without-legacy-python --no-
↪commit --branch=py3-only
```

- `setup.py`
  - Remove `six` from the list of dependencies
  - Remove other things pointing to Python 2 or PyPy2.
- Remove Python 2.7 up to 3.6 support code:
  - Update the code to Python 3 only and update usage of `six`:

```
$ find src -name "*.py" -exec pyupgrade --py3-plus --py37-plus {} \;
```

  - Replace all remaining `six` mentions, find it by running:

```
$ grep -rn six src
```

  - Find any remaining code that may support Python 2:

```
$ egrep -rn "2.7|3.5|3.6|sys.version|PY2|PY3|Py2|Py3|Python 2|Python 3|__
↪unicode__|ImportError" src
```

  - Run the tests with `tox` and fix any problems you encounter.
- Commit the changes to the branch.
- Create pull request against `master` with your changes.

3.3.7 Building binary wheels

Several packages in the Zope Foundation repositories contain C code, which means they should be packaged and published as binary wheels. Building such binary wheels has its pitfalls, though. If not done right they can cause application instability and crashes.

> **Warning:** Publishing binary wheels built in a local build environment is highly discouraged! Every build environment is different, binary wheels made for publication on PyPI should only be built in standardized and predictable environments.

To minimize problems we have automated binary wheel building using CI providers like AppVeyor and GitHub Actions. Wheels are built and uploaded using a special PyPI account named `zope.wheelbuilder`. These configurations and the PyPI account are currently maintained by Marius Gedminas, Michael Howitz and Jens Vagelpohl. You can contact them by posting your question or bug report in the meta repository issue tracker. Here is a short description of the process, please note that some steps can only be performed by the configuration maintainers:

Prerequisites

- Every project publishing binary wheels must add the PyPI account `zope.wheelbuilder` to the list of project maintainers on PyPI.
- Log into the `zope.wheelbuilder` account and create an API token for the project on the "Account settings" page with upload permissions and the project name as scope. Copy the token value - this is the only time you can.

Using Appveyor

- Log into Appveyor at ci.appveyor.com, click on "Account" at the top and then "Encrypt YAML" on the left-hand menu. Paste the copied token into the input field and then copy the encrypted value.
- If the project already uses the standardized parameterized project configuration add the following to the `.meta.toml` file in the `[appveyor]` section, replacing `<USER>` with the Appveyor account used and `<ENCRYPTED TOKEN>` with the encrypted value from the previous step. Then rebuild the configuration with `config-package.py` so the actual `appveyor.yml` file is updated:

```
[appveyor]
global-env-vars = [
    "# Currently the builds use @<USER>'s Appveyor account.  The PyPI token",
    "# belongs to zope.wheelbuilder, managed by @mgedmin and @dataflake.",
    "",
    "global:",
    "  TWINE_USERNAME: __token__",
    "  TWINE_PASSWORD:",
    "    secure: <ENCRYPTED TOKEN>",
    ]
```

- If you are not using the standardized project config you can edit `appveyor.yml` directly:

```
environment:

  global:
    # Currently the builds use @<USER>'s Appveyor account.  The PyPI token
    # belongs to zope.wheelbuilder, managed by @mgedmin and @dataflake.
```

(continues on next page)

(continued from previous page)

```
    TWINE_USERNAME: __token__
    TWINE_PASSWORD:
      secure: <ENCRYPTED TOKEN>

<...>

deploy_script:
  - ps: if ($env:APPVEYOR_REPO_TAG -eq $TRUE) { pip install twine; twine upload --
↪skip-existing dist\*.whl }

deploy: on
```

Using GitHub Actions

- On your project GitHub page go to "Settings" and then click on "Secrets" on the left-hand menu. Create a new repository secret and call it `TWINE_PASSWORD`. Paste the API token value you created for the `zope.wheelbuilder` PyPI account.
- Take a look at a complete GitHub Actions test and wheel building configuration for inspiration, or if you already use the standardized parameterized project configuration simply build your configuration from the template named *c-code*.