Commit

✕ [3.12] gh-123270: Replaced SanitizedNames with a more surgical fix. (G...

Browse files

…H-123354) (#123411)

gh-123270: Replaced SanitizedNames with a more surgical fix. (GH-123354)

Applies changes from zipp 3.20.1 and jaraco/zippGH-124
(cherry picked from commit 2231286)

Co-authored-by: Jason R. Coombs <jaraco@jaraco.com>

3.12 (#123411)

miss-islington and **jaraco** committed yesterday Verified 1 parent 514787a commit 95b073b

Showing **3 changed files** with **87 additions** and **71 deletions**.

Whitesp... Ignore whitesp... S... Uni...

Filter changed files

- Lib
  - test/test_zipfile/_path
    - test_path.py
  - zipfile/_path
    - __init__.py
- Misc/NEWS.d/next/Library
  - 2024-08-26-13-45-2...

73 Lib/test/test_zipfile/_path/test_path.py

```
 4   4    import pathlib
 5   5    import pickle
 6   6    import sys
     7  + import time
 7   8    import unittest
 8   9    import zipfile
 9  10
592 593
593 594        def test_malformed_paths(self):
594 595            """
595     -          Path should handle malformed paths.
     596  +          Path should handle malformed paths
         gracefully.
     597  +
     598  +          Paths with leading slashes are not
         visible.
     599  +
     600  +          Paths with dots are treated like
         regular files.
596 601            """
597 602            data = io.BytesIO()
598 603            zf = zipfile.ZipFile(data, "w")
601 606            zf.writestr("../parent.txt",
         b"content")
602 607            zf.filename = ''
603 608            root = zipfile.Path(zf)
```

```
604      -         assert list(map(str, root.iterdir()))
           == [
605      -             'one-slash.txt',
606      -             'two-slash.txt',
607      -             'parent.txt',
608      -         ]
609      +         assert list(map(str, root.iterdir()))
           == ['../']
610      +         assert
           root.joinpath('..').joinpath('parent.txt').read
           _bytes() == b'content'
611      +
612      +     def test_unsupported_names(self):
613      +         """
614      +         Path segments with special characters
           are readable.
615      +
616      +         On some platforms or file systems,
           characters like
617      +         ``:`` and ``?`` are not allowed, but
           they are valid
618      +         in the zip file.
619      +         """
620      +         data = io.BytesIO()
621      +         zf = zipfile.ZipFile(data, "w")
622      +         zf.writestr("path?", b"content")
623      +         zf.writestr("V: NMS.flac", b"fLaC...")
624      +         zf.filename = ''
625      +         root = zipfile.Path(zf)
626      +         contents = root.iterdir()
627      +         assert next(contents).name == 'path?'
628      +         assert next(contents).name == 'V:
           NMS.flac'
629      +         assert root.joinpath('V:
           NMS.flac').read_bytes() == b"fLaC..."
630      +
631      +     def test_backslash_not_separator(self):
632      +         """
633      +         In a zip file, backslashes are not
           separators.
634      +         """
635      +         data = io.BytesIO()
636      +         zf = zipfile.ZipFile(data, "w")
637      +
           zf.writestr(DirtyZipInfo.for_name("foo\\bar",
           zf), b"content")
638      +         zf.filename = ''
639      +         root = zipfile.Path(zf)
640      +         (first,) = root.iterdir()
641      +         assert not first.is_dir()
642      +         assert first.name == 'foo\\bar'
609      643
610      644      @pass_alpharep
611      645      def test_interface(self, alpharep):
612      646          from importlib.resources.abc import
           Traversable
```

```
613  647
614  648          zf = zipfile.Path(alpharep)
615  649          assert isinstance(zf, Traversable)
     650  +
     651  +
     652  + class DirtyZipInfo(zipfile.ZipInfo):
     653  +     """
     654  +     Bypass name sanitization.
     655  +     """
     656  +
     657  +     def __init__(self, filename, *args,
          **kwargs):
     658  +         super().__init__(filename, *args,
          **kwargs)
     659  +         self.filename = filename
     660  +
     661  +     @classmethod
     662  +     def for_name(cls, name, archive):
     663  +         """
     664  +         Construct the same way that
          ZipFile.writestr does.
     665  +
     666  +         TODO: extract this functionality and
          re-use
     667  +         """
     668  +         self = cls(filename=name,
          date_time=time.localtime(time.time())[:6])
     669  +         self.compress_type =
          archive.compression
     670  +         self.compress_level =
          archive.compresslevel
     671  +         if self.filename.endswith('/'):  #
          pragma: no cover
     672  +             self.external_attr = 0o40775 << 16
          # drwxrwxr-x
     673  +             self.external_attr |= 0x10  # MS-
          DOS directory flag
     674  +         else:
     675  +             self.external_attr = 0o600 << 16  #
          ?rw-------
     676  +         return self
```

82 Lib/zipfile/_path/__init__.py

```
...  ...  @@ -1,3 +1,12 @@
          1  + """
          2  + A Path-like interface for zipfiles.
          3  +
          4  + This codebase is shared between zipfile.Path in
          the stdlib
          5  + and zipp in PyPI. See
          6  + https://github.com/python/importlib_metadata/wi
          ki/Development-Methodology
          7  + for more detail.
          8  + """
          9  +
```

```
import io
import posixpath
import zipfile
def _ancestry(path):
    """
    Given a path with elements separated by
-   posixpath.sep, generate all elements of
that path
+   posixpath.sep, generate all elements of
that path.

    >>> list(_ancestry('b/d'))
    ['b/d', 'b']
    ['b']
    >>> list(_ancestry(''))
    []
+
+   Multiple separators are treated like a
single.
+
+   >>> list(_ancestry('//b//d///f//'))
+   ['//b//d///f', '//b//d', '//b']
    """
    path = path.rstrip(posixpath.sep)
-   while path and path != posixpath.sep:
+   while path.rstrip(posixpath.sep):
        yield path
        path, tail = posixpath.split(path)

        super().__init__(*args, **kwargs)


- class SanitizedNames:
-     """
-     ZipFile mix-in to ensure names are
sanitized.
-     """
-
-     def namelist(self):
-         return list(map(self._sanitize,
super().namelist()))
-
-     @staticmethod
-     def _sanitize(name):
-         r"""
-         Ensure a relative path with posix
separators and no dot names.
-
-         Modeled after
-
https://github.com/python/cpython/blob/bcc1be39
cb1d04ad9fc0bd1b9193d3972835a57c/Lib/zipfile/__
init__.py#L1799-L1813
-         but provides consistent cross-platform
behavior.
-
```

```
-         >>> san = SanitizedNames._sanitize
-         >>> san('/foo/bar')
-         'foo/bar'
-         >>> san('//foo.txt')
-         'foo.txt'
-         >>> san('foo/.././bar.txt')
-         'foo/bar.txt'
-         >>> san('foo../.bar.txt')
-         'foo../.bar.txt'
-         >>> san('\\foo\\bar.txt')
-         'foo/bar.txt'
-         >>> san('D:\\foo.txt')
-         'D/foo.txt'
-         >>> san('\\\\server\\share\\file.txt')
-         'server/share/file.txt'
-         >>> san('\\\\?\\GLOBALROOT\\Volume3')
-         '?/GLOBALROOT/Volume3'
-         >>> san('\\\\.\\PhysicalDrive1\\root')
-         'PhysicalDrive1/root'
-
-         Retain any trailing slash.
-         >>> san('abc/')
-         'abc/'
-
-         Raises a ValueError if the result is empty.
-         >>> san('../..')
-         Traceback (most recent call last):
-         ...
-         ValueError: Empty filename
-         """
-
-         def allowed(part):
-             return part and part not in {'..', '.'}
-
-         # Remove the drive letter.
-         # Don't use ntpath.splitdrive, because that also strips UNC paths
-         bare = re.sub('^([A-Z]):', r'\1', name, flags=re.IGNORECASE)
-         clean = bare.replace('\\', '/')
-         parts = clean.split('/')
-         joined = '/'.join(filter(allowed, parts))
-         if not joined:
-             raise ValueError("Empty filename")
-         return joined + '/' * name.endswith('/')
-
-
- class CompleteDirs(InitializedState, SanitizedNames, zipfile.ZipFile):
+ class CompleteDirs(InitializedState, zipfile.ZipFile):
      """
```

```
150  102      A ZipFile subclass that ensures that
                implied directories
151  103      are always included in the namelist.
```

3 ▪▪▪▫▫

Misc/NEWS.d/next/Library/2024-08-26-13-45-20.gh-issue-1232…

```
...  ...   @@ -0,0 +1,3 @@
     1   + Applied a more surgical fix for malformed payloads
           in :class:`zipfile.Path`
     2   + causing infinite loops (gh-122905) without
           breaking contents using
     3   + legitimate characters.
```

0 comments on commit `95b073b`

Please sign in to comment.