# Jekyll-Scholar Compatibility with Ruby 3.3: Technical Resolution

When attempting to build a Jekyll site with the jekyll-scholar plugin using Ruby 3.3.0, the following error occurs:

```
Liquid Exception: private method `new' called for class Jekyll::Scholar::URL in
publications-bytype.html
C:/Ruby33-x64/lib/ruby/gems/3.3.0/gems/jekyll-scholar-
7.2.0/lib/jekyll/scholar/utilities.rb:664:in `details_path_for': private method `new'
called for class Jekyll::Scholar::URL (NoMethodError)

        URL.new(
           ^^^^
```

This error prevents the site from building successfully, specifically affecting pages that use the bibliography tag:

```
liquid

{% capture refs %}{% bibliography --group_by type --group_order ascending %}{% endcapture %
```

## Environment Details

- **Ruby Version**: 3.3.0
- **Jekyll Version**: 4.3.4
- **Jekyll-Scholar Version**: Initially 7.2.0, later downgraded to 7.0.0
- **OS**: Windows (running on Windows with Ruby33-x64)

## Root Cause Analysis

There are two underlying issues preventing jekyll-scholar from working with Ruby 3.3.0:

1. **Constructor Access Issue**: In the jekyll-scholar gem, the `URL` class has a private constructor (`new` method), but the code in `details_path_for` method attempts to call it directly with `URL.new(...)`. In Ruby 3.3.0, this access restriction is more strictly enforced.
2. **Deprecated Method Usage**: The gem uses the deprecated `File.exists?` method instead of the current `File.exist?` method, which is required in Ruby 3.3.0.

## Attempted Solutions

## 1. Creating a Jekyll Plugin to Override URL Class

**File**: `_plugins/00_fix_scholar_url.rb`

```ruby
Jekyll::Hooks.register :site, :after_init do |site|
  module Jekyll
    module Scholar
      module Utilities
        # Override the problematic method directly
        def details_path_for(entry, base = config['details_dir'])
          # Skip using the URL.new call and implement the functionality directly
          name = entry.key.to_s.dup
          name.gsub!(/[:\s]+/, '_')
          File.join(base, name)
        end
      end
    end
  end
end
```

**Result**: This didn't work because Jekyll plugin hooks run after the gem is already loaded and initialized.

## 2. Alternative Plugin Approach

**File**: `_plugins/00_monkey_patch.rb`

ruby

```ruby
# Force open the jekyll-scholar gem and fix it directly
Jekyll::Hooks.register :site, :after_init do |site|
  # Load the actual file from the gem
  require 'jekyll/scholar/utilities'

  # Open the Jekyll::Scholar module
  module Jekyll
    module Scholar
      # Redefine the URL class completely
      class URL
        attr_reader :entry, :base

        def initialize(entry, base = '')
          @entry = entry
          @base = base
        end

        def to_s
          name = entry.key.to_s.dup
          name.gsub!(/[:\s]+/, '_')
          File.join(base, name)
        end
      end
    end
  end
end
```

**Result**: This approach also failed because the gem code had already been loaded and executed.

### 3. Fixing the File.exists? Method

**File**: `_plugins/01_fix_file_exists.rb`

ruby

```ruby
Jekyll::Hooks.register :site, :after_init do |site|
  require 'jekyll/scholar/utilities'

  module Jekyll
    module Scholar
      module Utilities
        # Override the problematic method
        def extend_path(name, base = source)
          p = File.join(base, name)
          # Change File.exists? to File.exist?
          name << '.bib' if File.extname(name).empty? && !File.exist?(name)
          p
        end
      end
    end
  end
end
```

**Result**: Still encountered the same error, as the plugin system couldn't effectively override the methods.

### 4. Adding "exists?" Method to File Class

**File**: `_plugins/00_file_patch.rb`

ruby

```ruby
# Add an alias for File.exists? to point to File.exist?
# This must load before any other plugins
class File
  class << self
    # Add the exists? method as an alias for exist?
    alias_method :exists?, :exist? unless respond_to?(:exists?)
  end
end
```

**Result**: This plugin also didn't load early enough to affect the Jekyll::Scholar initialization.

### 5. Configuration-Based Solution Attempt

**File**: `_config.yml` (relevant section)

```yaml
scholar:
  # ... other settings ...
  details_dir: ''
  details_layout: ''
  details_link: false
```

**Result**: This helped prevent the details feature from being used, but the code was still being called during initialization.

## Successful Solution

The solution that finally worked was directly modifying the jekyll-scholar gem files:

### Step 1: Locate the Gem Files

```bash
bundle show jekyll-scholar
# Output: C:/Ruby33-x64/lib/ruby/gems/3.3.0/gems/jekyll-scholar-7.0.0/
```

### Step 2: Fix the File.exists? Issue

**File**: `C:/Ruby33-x64/lib/ruby/gems/3.3.0/gems/jekyll-scholar-7.0.0/lib/jekyll/scholar/utilities.rb`

**Important Note**: There were multiple occurrences of `File.exists?` throughout the utilities.rb file. All instances needed to be changed to `File.exist?`. Below is one example of the change:

**Original Code** (around line 463):

```ruby
def extend_path(name, base = source)
  p = File.join(base, name)
  name << '.bib' if File.extname(name).empty? && !File.exists?(name)
  p
end
```

**Modified Code**:

ruby

```ruby
def extend_path(name, base = source)
  p = File.join(base, name)
  name << '.bib' if File.extname(name).empty? && !File.exist?(name)
  p
end
```

You should search through the entire file for all occurrences of `File.exists?` and replace each one with `File.exist?`.

### Step 3: Fix the URL Constructor Issue

**File**: `C:/Ruby33-x64/lib/ruby/gems/3.3.0/gems/jekyll-scholar-7.0.0/lib/jekyll/scholar/utilities.rb`

**Original Code** (around line 652):

ruby

```ruby
def details_path_for(entry, base = config['details_dir'])
  URL.new(
    template: config['details_permalink'],
    placeholders: url_placeholders,
  ).to_s
end
```

**Modified Code**:

ruby

```ruby
def details_path_for(entry, base = config['details_dir'])
  name = entry.key.to_s.dup
  name.gsub!(/[:\s]+/, '_')

  # If there's a permalink template, use a simplified version
  if config['details_permalink']
    template = config['details_permalink'].dup
    template.gsub!(/:([a-z_]+)/) do
      case $1
      when 'key'
        name
      when 'details_dir'
        base
      else
        **Modified Code**:
```ruby
def details_path_for(entry, base = config['details_dir'])
  name = entry.key.to_s.dup
  name.gsub!(/[:\s]+/, '_')
  File.join(base, name)
end


  end
end
return template
```

end

## Otherwise just join the base and name

File.join(base, name)

end
```

```
This change bypasses the `URL` class entirely by implementing the path generation logic
directly in the method.

## Verification

After making these changes to the gem files, running:

```bash
bundle exec jekyll build --trace
```

The build completed successfully with some Sass deprecation warnings but no Jekyll-Scholar errors:

```
done in 20.208 seconds.
Auto-regeneration: disabled. Use --watch to enable.
```

## Future-Proofing Recommendations

1. **Permanent Solution Options**: a. **Leverage Your Existing Docker Setup**:
   - You already have a Docker configuration (`Dockerfile_antunesweb`) which can be modified to avoid these issues
   - Update your existing Dockerfile to specify Ruby 3.2 instead of using the default Ruby version:
```

dockerfile

```
# Modify your existing Dockerfile_antunesweb
FROM ubuntu:kinetic

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get -y install \
        # Keep your existing packages
        # ...
        # Specify Ruby 3.2 explicitly
        software-properties-common

# Add Ruby 3.2 repository
RUN apt-add-repository ppa:brightbox/ruby-ng && \
    apt-get update && \
    apt-get install -y ruby3.2 ruby3.2-dev

# Install your gems
RUN gem install jekyll-scholar minimal-mistakes-jekyll nokogiri faraday-retry jemoji

# Continue with the rest of your Dockerfile
```

Then build and run with your existing command:

bash

```
docker build -t antunesweb2 -f Dockerfile_antunesweb .
docker run -it -v $(pwd):/home -p 4000:4000 antunesweb2
```

b. **Simpler Ruby Docker Alternative**:

- Alternatively, use the official Ruby image for a simpler setup:

dockerfile

```
FROM ruby:3.2
WORKDIR /site
COPY Gemfile* ./
RUN bundle install
VOLUME /site
EXPOSE 4000
CMD ["bundle", "exec", "jekyll", "serve", "--host", "0.0.0.0"]
```

c. **Fork jekyll-scholar**:

- Create a fork of the jekyll-scholar gem on GitHub

- Implement these fixes properly
- Use your fork in your Gemfile:

```ruby
gem "jekyll-scholar", github: "yourusername/jekyll-scholar", branch: "ruby-3.3-fix"
```

d. **Ruby Version Manager**:

- Use a Ruby version manager to run Jekyll with Ruby 3.2.x
- Example with rbenv:

```bash
rbenv install 3.2.3
rbenv local 3.2.3
gem install bundler
bundle install
```

2. **Ongoing Maintenance**: When updating jekyll-scholar in the future, remember that:
   - Your direct modifications to the gem files will be lost
   - You'll need to reapply the changes or switch to one of the permanent solutions above

   Always test your site after gem updates, as compatibility issues might recur.

## Technical Details for Reference

### Jekyll-Scholar Class Structure

The issue stems from how Jekyll-Scholar defines its classes:

- `Jekyll::Scholar` (main module)
  - `Jekyll::Scholar::Utilities` (module with helper methods)
  - `Jekyll::Scholar::URL` (class with a private constructor)

The URL class appears to be implemented with a private constructor but is called directly from within the utilities module, which would normally work in older Ruby versions but breaks in Ruby 3.3.0 due to stricter access controls.

### Ruby 3.3.0 Changes

Ruby 3.3.0 includes several changes that affect gem compatibility:

1. Stricter enforcement of method visibility
2. Removal of deprecated methods like `File.exists?` in favor of `File.exist?`

3. Changes to how modules and classes are accessed

These changes improve the language but can break older gems that rely on the previous behavior.