Contributing to Django: A Beginner's Guide

This guide will walk you through the process of contributing to Django, one of the most popular Python web frameworks. We'll use a real beginner-friendly ticket as an example to demonstrate the entire workflow.

Table of Contents

Understanding Open Source Contribution

Contributing to open source projects like Django is a great way to improve your coding skills, learn from experienced developers, and give back to the community. The process typically involves:

1. Finding an issue to work on
2. Setting up your development environment
3. Making code changes
4. Testing your changes
5. Submitting a pull request
6. Responding to feedback and making revisions

Let's go through each step using a real Django ticket.

Setting Up Your Development Environment

Before you can contribute to Django, you need to set up your development environment:

1. Fork the Django Repository

Go to the Django GitHub repository and click the "Fork" button in the top-right corner. This creates your own copy of the repository.

2. Clone Your Fork

```
# Replace YOUR-USERNAME with your GitHub username
git clone https://github.com/YOUR-USERNAME/django.git
cd django
```

3. Set Up the Upstream Remote

```
git remote add upstream https://github.com/django/django.git

# check upstream in the repo
git remote -v
```

4. Create a Virtual Environment

```
python -m venv django-dev
source django-dev/bin/activate  # On Windows: django-dev\Scripts\activate
```

5. Install Development Dependencies

```
pip install -e .
pip install -r requirements/testing.txt
```

6. Run Initial Tests

```
python tests/runtests.py
```

Finding a Beginner-Friendly Ticket

Django uses a Trac instance for issue tracking. To find beginner-friendly tickets:

1. Go to Django's Trac
2. Filter for tickets with "Easy pickings" set to "yes"
3. Look for tickets that are not assigned or have been inactive

For this guide, we'll use ticket #35924: FilteredSelectMultiple widget choose/remove all button arrows not responsive.

Understanding the Issue

Let's break down our example ticket:

Ticket #35924: FilteredSelectMultiple widget choose/remove all button arrows not responsive

Description:

> The arrow buttons to choose or remove selected items correctly change direction when in a mobile view to be up/down rather than left/right. However, in the choose all/remove all buttons the arrow icons are always left/right.

> This needs either fixing or we should just remove these arrow icons.

Key Information:

- **Component:** contrib.admin
- **Type:** Bug
- **Triage Stage:** Accepted
- **Has patch:** yes (but needs improvement)
- **Easy pickings:** yes
- **UI/UX:** yes

Refresher: Django Admin and FilteredSelectMultiple Widget

The Django Admin site provides an interface for managing your application's data. The `FilteredSelectMultiple` widget is used in the admin interface for selecting multiple items from a list, with filtering capabilities.

```
# Example of how FilteredSelectMultiple is used in Django admin
from django.contrib import admin
from django.db import models
from django.forms import ModelForm
from django.contrib.admin.widgets import FilteredSelectMultiple

class MyModelAdmin(admin.ModelAdmin):
    formfield_overrides = {
        models.ManyToManyField: {'widget': FilteredSelectMultiple('items',
False)},
    }
```

Creating Your Development Branch

Always create a new branch for each contribution:

```
# Make sure you're on the main branch and up-to-date
git checkout main
git pull upstream main

# Create a new branch with a descriptive name
git checkout -b fix-35924-filteredselectmultiple-arrows
```

Making the Code Changes

For our example ticket, we need to fix the arrow icons in the choose all/remove all buttons of the FilteredSelectMultiple widget.

First, let's locate the relevant files:

```
# Search for FilteredSelectMultiple in the codebase
grep -r "FilteredSelectMultiple" django/
```

The main files we need to examine are:

- `django/contrib/admin/widgets.py` - Contains the widget definition
- `django/contrib/admin/static/admin/js/SelectFilter2.js` - Contains the JavaScript functionality
- `django/contrib/admin/static/admin/css/widgets.css` - Contains the styling

Understanding the Code

Let's look at how the arrows are currently implemented:

```
// In SelectFilter2.js (simplified)
quickElement(
  "a",
  quickElement("div", selector_chosen_box, "selector-chooseall"),
  gettext("Choose all"),
  "href",
  "javascript:void(0);",
  "title",
  interpolate(gettext("Choose all of the %s at once."), [field_name]),
  "id",
  field_id + "_add_all_link"
);
quickElement(
  "a",
  quickElement("div", selector_chosen_box, "selector-clearall"),
  gettext("Remove all"),
  "href",
  "javascript:void(0);",
  "title",
  interpolate(gettext("Remove all chosen %s at once."), [field_name]),
  "id",
  field_id + "_remove_all_link"
);
```

The issue is that these buttons don't adjust their arrow direction on mobile views like the individual item buttons do.

Implementing the Fix

Here's how we might fix this issue:

1. Modify the CSS to make the arrows responsive:

```
/* In widgets.css */
@media (max-width: 767px) {
  .selector .selector-chooseall:before,
  .selector .selector-clearall:before {
    content: "\2191"; /* Up arrow instead of left arrow */
  }
  .selector .selector-chooseall:after,
  .selector .selector-clearall:after {
    content: "\2193"; /* Down arrow instead of right arrow */
  }
}
```

2. Or alternatively, remove the arrows completely if they're not needed:

```
/* In widgets.css */
.selector .selector-chooseall:before,
.selector .selector-chooseall:after,
.selector .selector-clearall:before,
.selector .selector-clearall:after {
  content: none;
}
```

Running Tests

Before submitting your changes, you need to run tests to ensure nothing breaks:

```
# Run the specific tests for the admin app
python tests/runtests.py admin_widgets

# Run the full test suite
python tests/runtests.py
```

You should also manually test the widget in the Django admin interface:

```
# Create a test project
django-admin startproject testproject
cd testproject

# Create a test app with models using ManyToManyField
python manage.py startapp testapp
```

Then configure the app to use the FilteredSelectMultiple widget and test on both desktop and mobile views.

Submitting Your Pull Request

Once your changes are working and tested:

```
# Commit your changes with a descriptive message
git add django/contrib/admin/static/admin/css/widgets.css
git commit -m "Fixed #35924: Made FilteredSelectMultiple choose/remove all
button arrows responsive"

# Push to your fork
git push origin fix-35924-filteredselectmultiple-arrows
```

Then:

1. Go to your fork on GitHub
2. Click "Pull Request"
3. Set the base repository to django/django and the base branch to main
4. Set your repository and branch as the compare
5. Add a descriptive title and description, referencing the ticket number
6. Submit the pull request

Example Pull Request Description

```
Fixed #35924: Made FilteredSelectMultiple choose/remove all button arrows
responsive

The arrow buttons to choose or remove selected items correctly change
direction
when in a mobile view to be up/down rather than left/right. However, in the
choose all/remove all buttons the arrow icons were always left/right.

This PR fixes the issue by making the arrows in the choose all/remove all
buttons
responsive to match the behavior of the individual item buttons.

Ticket: https://code.djangoproject.com/ticket/35924
```

Responding to Feedback

After submitting your PR, Django maintainers will review your code and may request changes. This is a normal part of the process:

1. Read the feedback carefully
2. Make the requested changes
3. Push new commits to your branch
4. Respond to the comments

The PR might go through several rounds of review before being accepted.

Additional Resources

- Django's Contributing Guide
- Django's Coding Style
- Django's Git Workflow

Conclusion

Contributing to Django is a rewarding experience that helps you grow as a developer while improving a tool used by millions. By starting with beginner-friendly tickets like the one we explored, you can gradually build confidence and expertise to tackle more complex issues.

Remember that everyone in the Django community was once a first-time contributor. Don't be afraid to ask questions and learn from the feedback you receive.

Happy contributing!