Web Development Using Application Frameworks
Coding Assignment: Generic Class-Based Views
Instructions

Overview

The Generic Class-Based Views coding assignment is the next in a series of assignments in which we will be developing the EZ University database system, a full C-R-U-D database application for simplified university record keeping. In this assignment, we refactor our views.py code to take advantage of generic class-based views (GCBVs).

Tools

I am expecting you to use the tools that are demonstrated in the tutorial videos: Anaconda and PyCharm.

Tool Versions

Use the versions of PyCharm Professional, Anaconda, and Python that we installed during Week 1 of the course when we created the *e4_trainor_django_course* virtual env. These versions are documented in *Instructions for Tool Versions, Installation, and Virtual Environments*.

Tutorial Parts

This is a seven-part tutorial.

Part 1 – RedirectView and TemplateView

In this part of the tutorial, we refactor the code that redirects the empty URL string so that it is implemented with the RedirectView GCBV. Additionally, we work together to add an *About* page to our Web application and implement it using the TemplateView GCBV.

While accomplishing this goal, we will use the starter files provided in:

- gcbv_starter_files_parts_1_through_3.zip

Part 2 – Paginated ListViews

In this part of the tutorial, we begin by work together to refactor the InstructorList view so that it uses the ListView GCBV. Some code needed for this part of the tutorial is provided in:

- gcbv_starter_files_parts_1_through_3.zip

After we have worked together to code and test our refactoring of InstructorList, you will complete this part by doing a similar refactoring of the StudentList view on your own.

Part 3 – Non-Paginated ListViews

In this part of the tutorial, we begin by working together to refactor the SectionList view so that it uses the ListView GCBV.

After we have worked together to code and test our refactoring of SectionList, you will complete this part by doing a similar refactoring the following views on your own:

- CourseList
- SemesterList
- RegistrationList

Part 4 – DetailViews

In this part of the tutorial, we begin by working together to refactor the following views so that they use the DetailView GCBV:

- InstructorDetail
- SectionDetail

After we have worked together to code and test our refactoring of the views above, you will complete this part by doing a similar refactoring of the following views on your own:

- CourseDetail
- SemesterDetail
- StudentDetail
- RegistrationDetail

Part 5 – CreateViews

In this part of the tutorial, we begin by working together to refactor the following views so that they use the CreateView GCBV:

- InstructorCreate
- SectionCreate

After we have worked together to code and test our refactoring of the views above, you will complete this part by doing a similar refactoring of the following views on your own:

- CourseCreate
- SemesterCreate
- StudentCreate
- RegistrationCreate

Part 6 – UpdateViews

In this part of the tutorial, we begin by working together to refactor the following views so that they use the UpdateView GCBV:

- InstructorUpdate
- SectionUpdate

After we have worked together to code and test our refactoring of the views above, you will complete this part by doing a similar refactoring of the following views on your own:

- CourseUpdate
- SemesterUpdate
- StudentUpdate
- RegistrationUpdate

Part 7 – DeleteViews

In this part of the tutorial, we begin by working together to refactor the following views so that they use the DeleteView GCBV:

- RegistrationDelete
- InstructorDelete

After we have worked together to code and test our refactoring of the views above, you will complete this part by doing a similar refactoring of the following views on your own:

- SectionDelete
- CourseDelete
- SemesterDelete
- StudentDelete

Exercises

1. **Exercise 1 (Regular)**
   Follow Parts 1 through 7 of the tutorial instructions exactly.

2. **Exercise 2 (Challenge)**
   The challenge exercises for all of the EZU-based coding assignments will be based on finding ways to maximize the usefulness of the Django Admin app in the activities of the current assignment. When working on these challenge exercises, you may consult any resources that you may find helpful. I am offering the following resources as a starting point:

   1. Django Software Foundation (2025). *The Django admin site* in Django Documentation.

   2. Real Python (2025). *Customize the Django Admin With Python*.

   3. Tomazic, N. (2024). *Customizing the Django Admin* in TestDriven.io.

   This week's challenge exercise is to do the following:

   1. Write a short report that identifies ways in which we might customize the Django Admin app to help with the current week's work of writing and testing the code for this assignment.

   2. Identify 1 or 2 customizations. For each customization briefly explain how that customization might make our work easier or might make us more effective. Also, identify any coding work that would be required to implement the customization.

   3. Place your report in a text file named *gcbv_assignment_challenge.txt*. Place that text file directly into your Django Project directory.

   4. Format your work as a business report with proper paragraphs and sentences. Pay attention to grammar, choice of words, and spelling. You do not need to cite sources.

Code and Document Deliverables

You are expected to submit one properly organized PyCharm Django project that is ready to be tested using PyCharm. Please refer to my tutorial video for details. Even if you have decided to do Exercise 2, just submit one Django project.

Non-Code Deliverables

Please be sure that the project you submit includes the following:

1. A test user (username = "tester", password = "{iSchoolUI}"
2. Sufficient test data present in the database to allow for testing all functions

Submission Method

Follow the process that I demonstrated in the tutorial video on submitting your work. This involves:
- Locating the properly named directory associated with your project in the file system.
- Compressing that directory into a single .ZIP file using a utility program.
- Submitting the properly named zip file to the submission activity for this assignment.

File and Directory Naming

Please use the following naming scheme for naming your PyCharm project:

surname_givenname_ezu

If this were my own project, I would name my PyCharm project as follows:

trainor_kevin_ezu

Use a zip utility to create one zip file that contain the PyCharm project directory. The zip file should be named according to the following scheme:

surname_givenname_ezu.zip

If this were my own project, I would name the zip file as follows:

trainor_kevin_ezu.zip

PLEASE NOTE: All file and directory names must be in lower case. Deductions will be made for submissions that do not conform to this standard.

Due Date

Please see the Weekly Schedule for the date and time when this assignment is due.

Last Revised

2025-03-06