DJ4E (https://www.dj4e.com)

Login / Autos CRUD

This assignment is to build a fully working CRUD (Create, Read, Update, and Delete) application to manage automobiles and their makes (i.e. Ford, Hyundai, Toyota, Tata, Audi, etc.).

This application will be based on this folder in the samples repo:

https://github.com/csev/dj4e-samples/tree/master/autos

Do not clone this repository for this assignment. You will make a new project and application in your `django_projects` folder and use this application as *sample code*.

This application will be similar to:

https://crud.dj4e.com

The login information is as follows:

```
Account: dj4e-crud
Password: dj4e_nn_!
```

The 'nn' is a 2-digit number that by now, you should be able to easily guess.

Making a New Application

Activate any virtual environment you need (if any) and go into your `django_projects` folder and start a new application in your `mysite` project (this project already should have the 'hello' application from a previous assignment (dj4e_hello.md)).

```
workon django3  # as needed
cd ~/django_projects/mysite
python manage.py startapp autos
```

Extending the home (i.e. main) page

Since we will build a number of applications in this project, we will use the `home` application to provide

Select Language ▼

convienent urls to switch between applications.

And you should have a file `mysite/home/templates/home/main.html` that has the text for the top-level page. You can keep the "Hello World" text in the page somewhere.

Add a link to the "/autos" url in `mysite/home/templates/home/main.html` and anything else the autograder needs:

```
<li><a href="/autos">Autos CRUD</a>
```

It is a list because we will be adding more applications in future assignments. :)

Building the Autos Application

The essense of this task is to adapt the code from:

https://github.com/csev/dj4e-samples/tree/master/autos

and make it work in your `autos` project. As always there is a lot of code in `dj4e-samples` - be careful copying - and only copy code when you know why you are copying it. Go slowly.

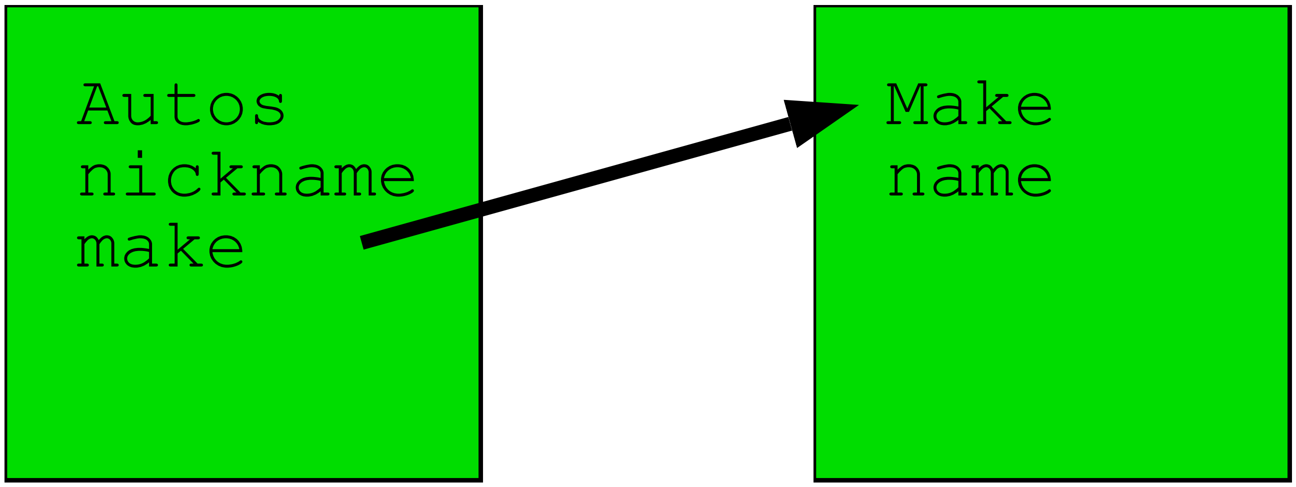
Here are some tasks:

- Go into your `dj4e-samples` folder and do a `git pull` to get the latest version of the samples code.
- Create `mysite/home/templates` and `mysite/home/templates/registration` folders using `mkdir` and copy the (login.html) template from `dj4e-samples` into `mysite/home/templates/registration/login.html` .
- Copy the file from `dj4e-samples/home/templates/base_bootstrap.html` into your `mysite/home/templates` - this will be used in your `autos/templates` and make our HTML look better by applying the Bootstrap and other styling libraries.
- Edit `mysite/mysite/settings.py` add the autos application to the list of `INSTALLED_APPS` . You can follow the pattern of the `HomeConfig` line in that file.
- Edit `mysite/mysite/urls.py` and add the `accounts/` path so you can use the Django built in login features. (Authentication Views). Also edit `mysite/mysite/urls.py` to route `autos/` urls to `autos/urls.py` file.

```
path('accounts/', include('django.contrib.auth.urls')),  # Add
path('autos/', include('autos.urls')),                   # Add
```

Select Language ▼

- Edit the `autos/urls.py` file to add routes for the list, edit, and delete pages for both autos and makes
- Edit the `autos/views.py` file to add views for the list, edit, and delete pages for both autos and makes. It will make things a lot simpler in the long run if you convert the Make views to the shorter form like the Auto views. (Example)
- In your `views.py` file, you should *not* simply use the code for the `Make` views. You should rewrite the `Make` views using the same patterns as the `Auto` views. If you don't use the generic edit views on your `Make` views you will need to define the appropriate `MakeForm` in your `forms.py` just like in the sample code. The best approach is to build your `views.py` *without* using a `forms.py` - but you can do it either way.
- Run the `python manage.py check` until you see no errors
- Edit the `autos/models.py` file to add Auto and Makes models with a foreign key from Autos to Makes.



- Run the `python manage.py makemigrations` until it has no errors. If you find that the migrations have partially completed and seems ot be confused, you can always start the migrations over by typing:

```
rm */migrations/00*.py
```

- Run the `python manage.py migrate` to create the database.
- Edit `autos/admin.py` to add the Auto and Make models to the Django administration interface.
- Run the `python manage.py check` until you see no errors
- Create a superuser so you can test the admin interface and log in to the application.
- Create the necessary template files in `autos/templates/autos` to support your views. Note that the the

Select Language ▼

second sub folder under `templates` is there to make sure that templates are not inadvertently shared across multiple applications within a Django project.

- Find the line in your `base_bootstrap.html` that looks like this:

```
<meta name="dj4e-code" content="99999999">
```

  and change the `9999999` to be "42161041685418"

Make sure to check the autograder for additional markup requirements.

Things that can go wrong

If you ever get a 405 error on a Django page it probably means that you have defined a view class that does not have a `get()` method. For example if you meant to say this:

```
class AutoUpdate(LoginRequiredMixin, UpdateView):
    model = Auto
    fields = '__all__'
    success_url = reverse_lazy('autos:all')
```

But instead you did:

```
class AutoUpdate(LoginRequiredMixin, View):
    model = Auto
    fields = '__all__'
    success_url = reverse_lazy('autos:all')
```

(i.e. you extended `View` instead of `UpdateView`) - the result is that there is no `def get(self, request):` in your view. So you get the 405 HTTP status code (invalid method) when you navigate to the URL that forwards to the view.

Manual Testing

It is always a good idea to manually test your application before submitting it for grading. Here is a rough outline of the steps that the autograder will take to grade your application. You should run them by hand before running the autograder and make sure they work without error.

- Log in
- Add a make
- Add an auto selecting the make

Select Language ▼

- View the makes
- Update the make and press 'Cancel' - Should go to the auto list list
- Update the make, make a change and save it - Should go to the auto list
- The make for the auto in the list should be automatically updated
- Update the auto, change its nickname and save it - it should be updated in the auto list
- Add a second make
- Add a second auto using the second make
- View the autos - make sure the second car and make are present
- Delete the second make - the auto corresponding to the make should go away automaticallr (Cascade in action)
- Delete the first auto

References

- Autos CRUD Sample Code
- Installing Django Locally
- Using ngrok to turn in your assignments
- Embedding SVG in Markdown

Select Language ▼