Getting the Most out of Django's User Model

JULIA LOONEY

The User Model

django.contrib.auth.models.User

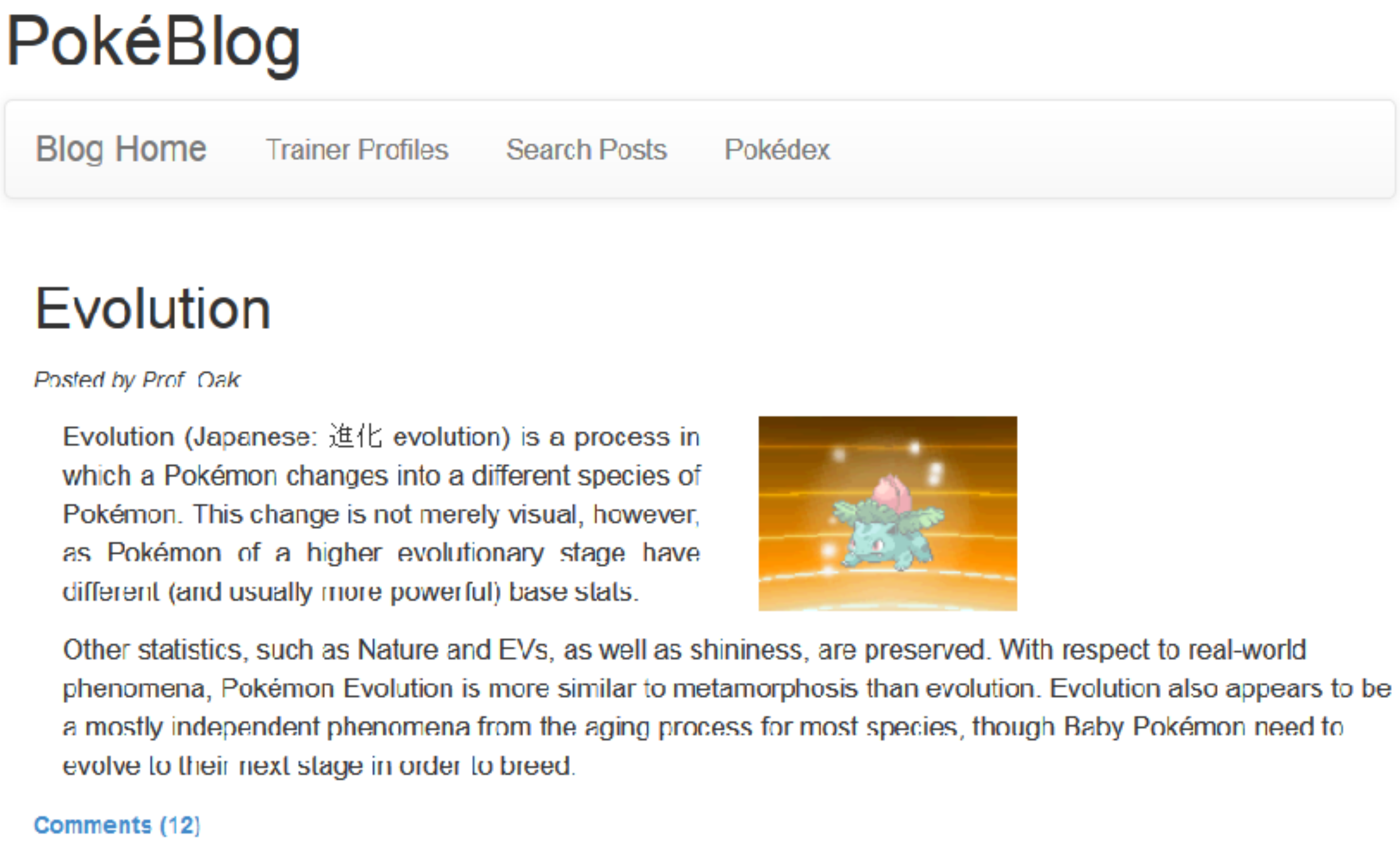
The User Model

Attributes:

- username (primary key)
- password
- email
- first_name
- last_name

What if we want more?

- Custom fields
- Custom permissions
- Custom model manager
- Alternative identifier to ‘username’



PokéBlog

Evolution

Posted by Prof. Oak

Evolution (Japanese: 進化 evolution) is a process in which a Pokémon changes into a different species of Pokémon. This change is not merely visual, however, as Pokémon of a higher evolutionary stage have different (and usually more powerful) base stats.

Other statistics, such as Nature and EVs, as well as shininess, are preserved. With respect to real-world phenomena, Pokémon Evolution is more similar to metamorphosis than evolution. Evolution also appears to be a mostly independent phenomena from the aging process for most species, though Baby Pokémon need to evolve to their next stage in order to breed.

Comments (12)

Option 1: Proxy model

- Requires the least amount of steps
- Good for when you only need to customize a few things

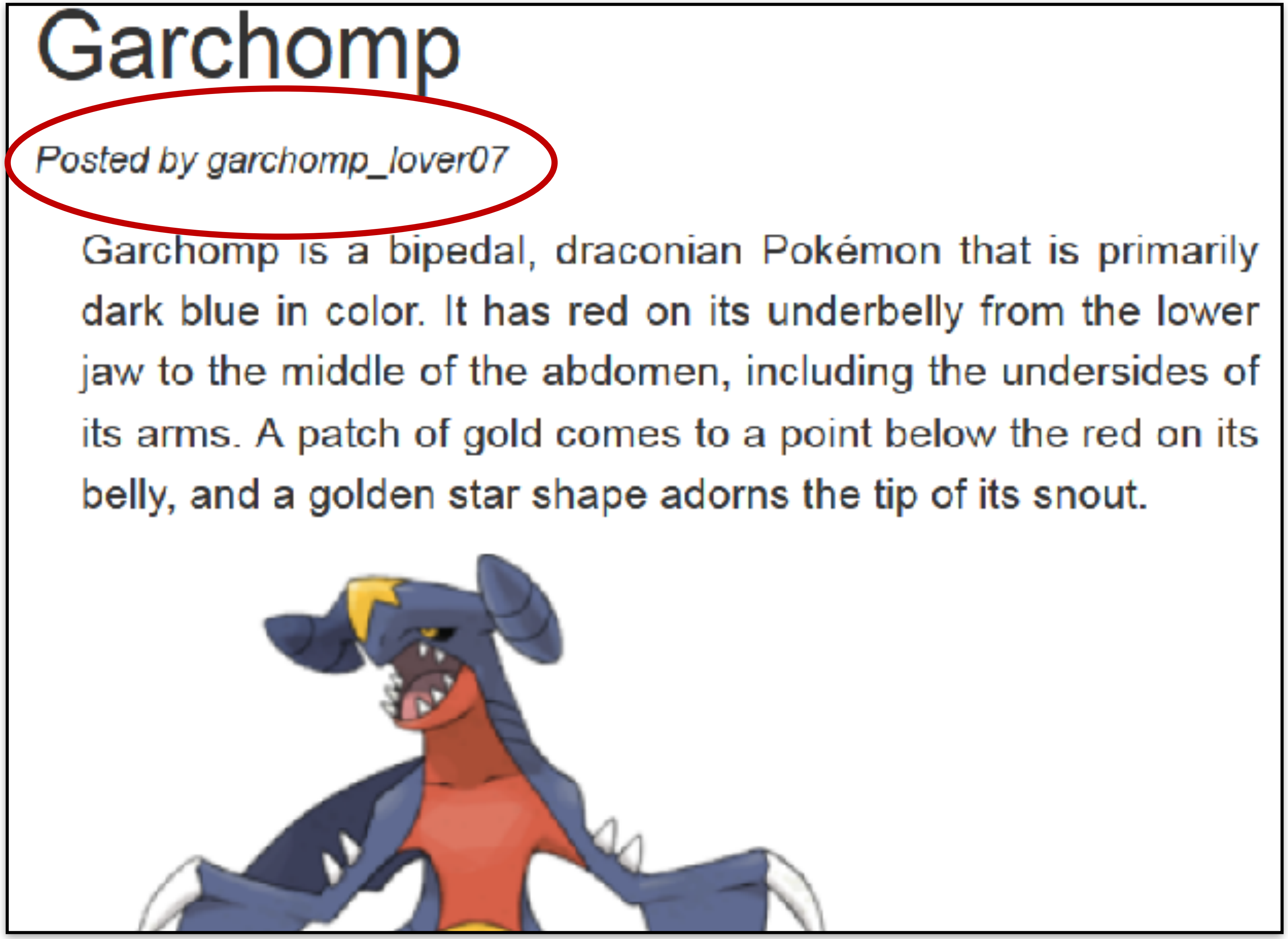
Option 2: One-to-one Relationship

- Requires only a few more steps
- Good for when you need more attributes

Option 3: Custom User Model

- More steps, but more freedom
- Good for when:
  - you need different required fields
  - you need custom permissions
  - you want to add validators

Scenario 1



Garchomp

Posted by garchomp_lover07

Garchomp is a bipedal, draconian Pokémon that is primarily dark blue in color. It has red on its underbelly from the lower jaw to the middle of the abdomen, including the undersides of its arms. A patch of gold comes to a point below the red on its belly, and a golden star shape adorns the tip of its snout.

Option 1: Using a Proxy Model

Option 1: Using a Proxy Model

Good for:

- Using a custom model manager
- Using custom model methods

Limitations:

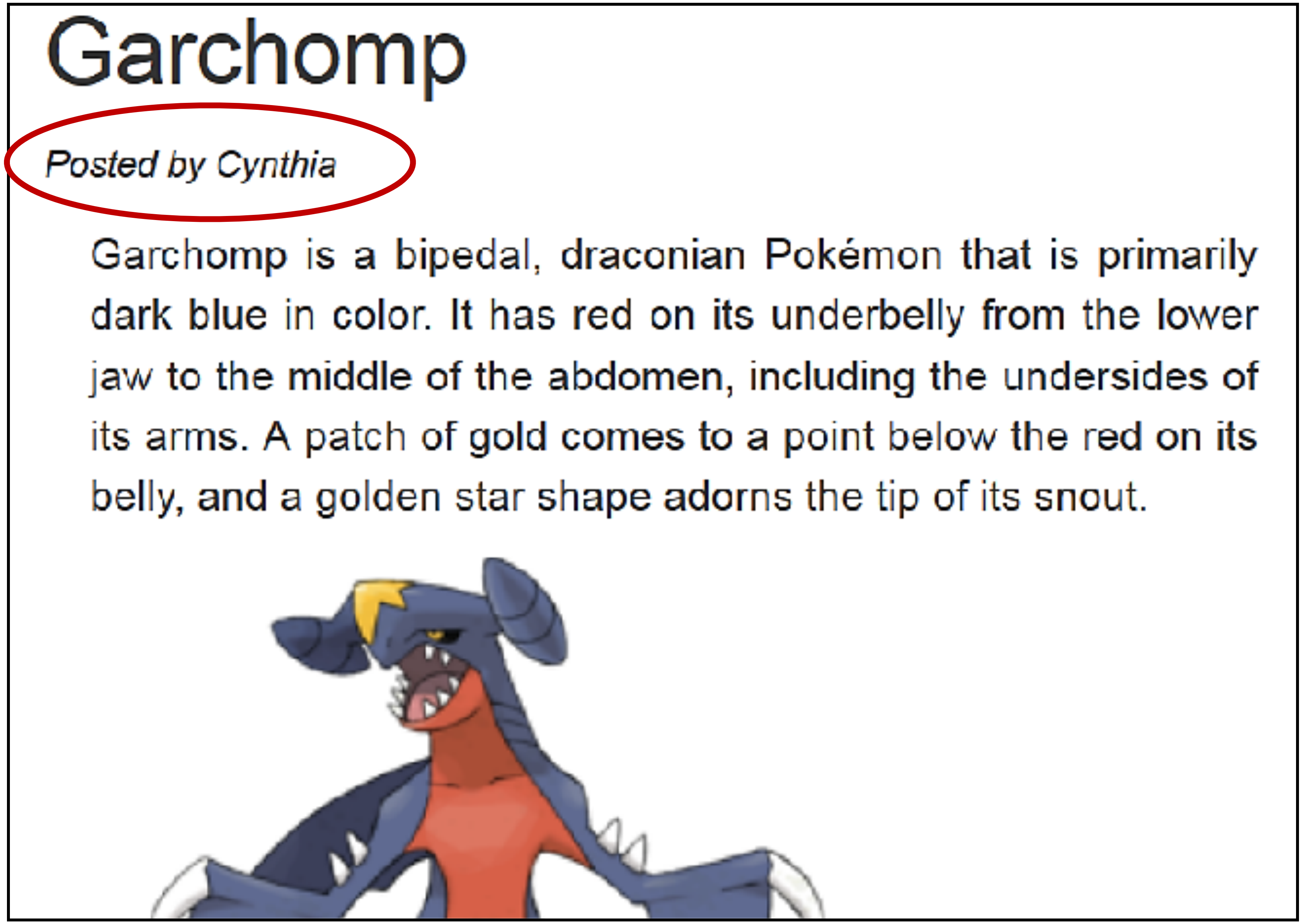
- Can’t add additional attributes
- Can’t change the database

Option 1: Using a Proxy Model

```
Author(User):

    class Meta:
        proxy = True
```

Option 1: Using a Proxy Model

```
Author(User):

    class Meta:
                proxy = True

    def __str__(self):
        return self.first_name
```

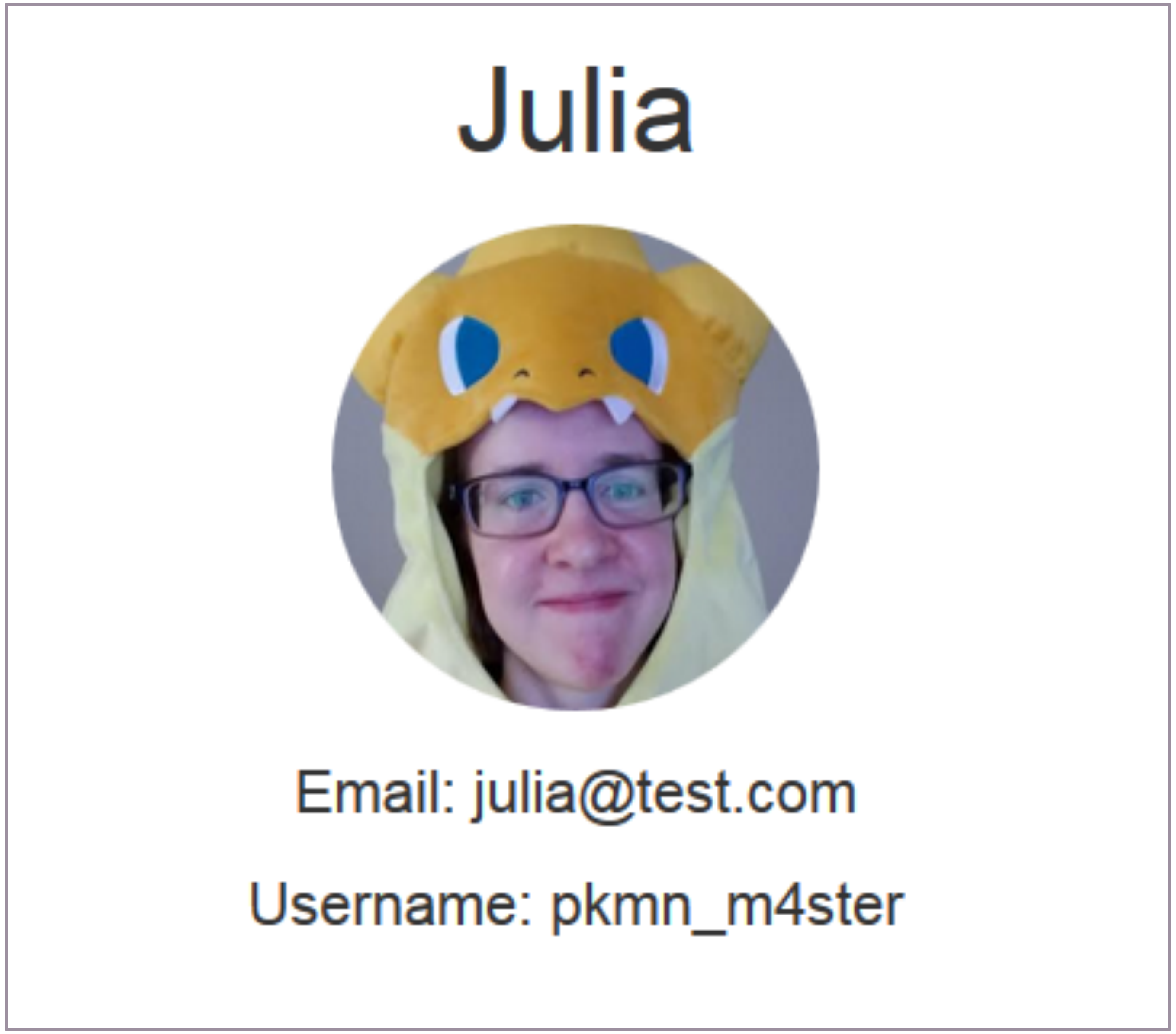


Garchomp

Posted by Cynthia

Garchomp is a bipedal, draconian Pokémon that is primarily dark blue in color. It has red on its underbelly from the lower jaw to the middle of the abdomen, including the undersides of its arms. A patch of gold comes to a point below the red on its belly, and a golden star shape adorns the tip of its snout.

Scenario 2



Julia

Email: julia@test.com

Username: pkmn_m4ster

Option 2: Using a 1-1 relationship

Option 2: 1-1 Relationship

Good for:

- Adding custom fields

Down Side:

- Need to keep track of another model
- Requires additional steps

Option 2: 1-1 Relationship

```
class Profile(models.Model):

    user = models.OneToOneField(User)
```

Option 2: 1-1 Relationship

```
class Profile(models.Model):

    user = models.OneToOneField(User)
    location = ...
    bio = ...
    ...
```

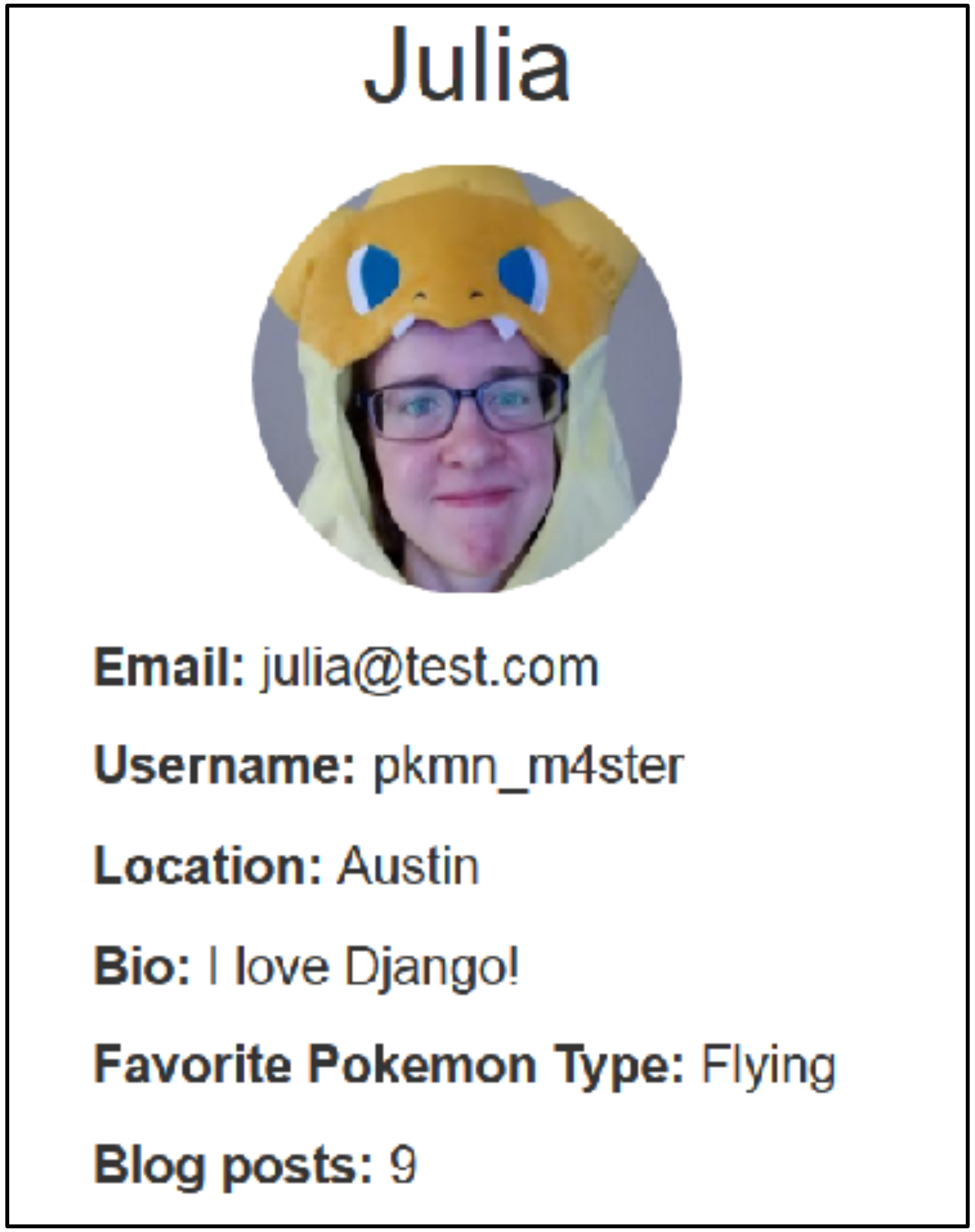
Option 2: 1-1 Relationship

```
print(user.first_name)
>>> Julia

print(user.profile.location)
>>> Austin
```

Option 2: 1-1 Relationship

Additional Steps:

- Handling User saves & updates
- Handling deletion
- Handling Profile creation upon User creation
- Include in Django Admin



Julia

Email: julia@test.com

Username: pkmn_m4ster

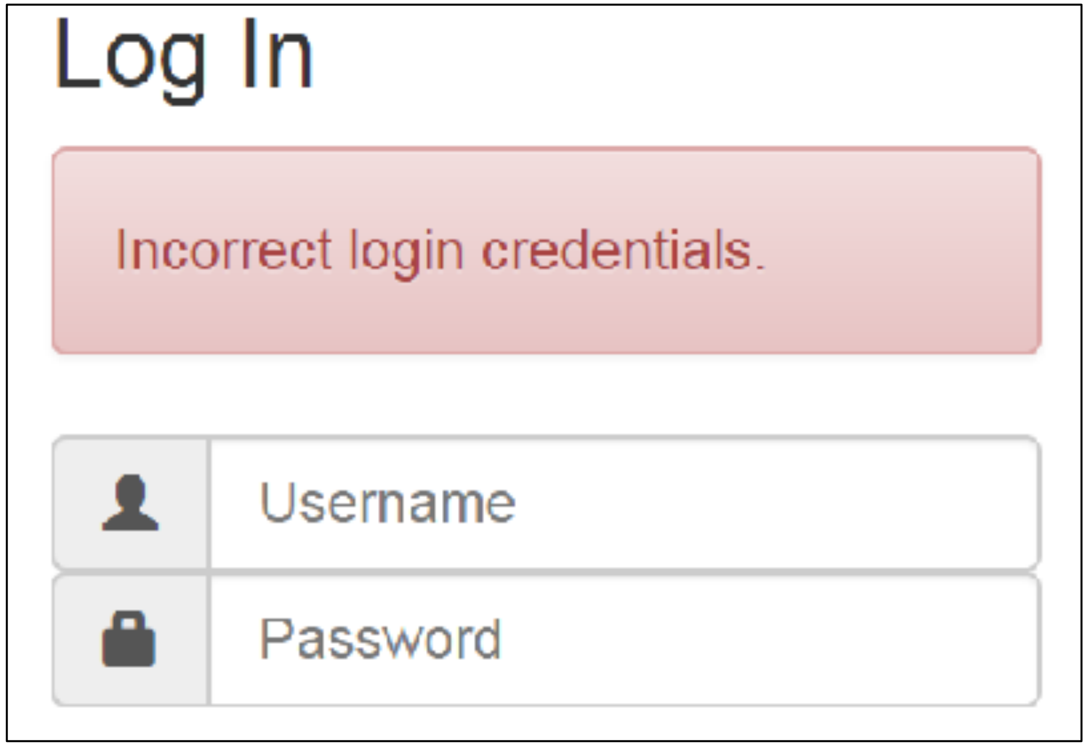
Location: Austin

Bio: I love Django!

Favorite Pokemon Type: Flying

Blog posts: 9

Scenario 3


Log In

Incorrect login credentials.

Option 3: Create a custom User Model

Option 3: Custom User Model

Django’s AbstractBaseUser

- Provides basic User implementation and password handling

Option 3: Custom User Model

```
class MyUser(AbstractBaseUser):
    email = ...
    fav_type = ...
```

Option 3: Custom User Model

```
class MyUser(AbstractBaseUser):
    email = ...
    fav_type = ...

    USERNAME_FIELD = ‘email’
```

Option 3: Custom User Model

```
settings.py

AUTH_USER_MODEL = ‘my_app.MyUser’
```

Option 3: Custom User Model

Additional Steps:

- Set up User Manager
- Set up Django Admin Forms

Recap

- Proxy model for adding methods
- 1-1 Relationship for adding fields
- Custom User for full customization

Resources

https://github.com/jlooney

jlooney@utexas.edu
Or Julia.m.looney@gmail.com

@looneydev