Django ORM Model Field Choices

Django Field Choices

Django provides a clean way to define **enumerated values** using `choices` on model fields. This helps keep your database values consistent while making your code more readable.

---

1. Basic `choices` Syntax

```
class MyModel(models.Model):
    STATUS_CHOICES = [
        ('draft', 'Draft'),
        ('published', 'Published'),
    ]

    status = models.CharField(max_length=20, choices=STATUS_CHOICES)
```

- The **first item** is the *actual database value*
- The **second item** is the *human-readable label*

---

2. Using **TextChoices**

Django's `TextChoices` makes enumerations cleaner and provides autocompletion.

```
from django.db import models

class Status(models.TextChoices):
    DRAFT = "draft", "Draft"
    PUBLISHED = "published", "Published"
    ARCHIVED = "archived", "Archived"


class Article(models.Model):
    status = models.CharField(
        max_length=20,
        choices=Status.choices,
        default=Status.DRAFT
    )
```

Benefits:

- Avoid typos (`Status.DRAFT` instead of `"draft"`)
- Easy comparisons

```
if article.status == Status.PUBLISHED:
    ...
```

---

3. Using IntegerChoices

Useful when you want efficient storage (integers) but readable code.

```
class Priority(models.IntegerChoices):
    LOW = 1, "Low"
    MEDIUM = 2, "Medium"
    HIGH = 3, "High"


class Task(models.Model):
    priority = models.IntegerField(
        choices=Priority.choices,
        default=Priority.MEDIUM
    )
```

Usage

```
if task.priority == Priority.HIGH:
    ...
```

---

4. Accessing Labels

Django automatically provides a `get_<field>_display()` helper.

```
task.get_priority_display()  # → "High"
article.get_status_display()  # → "Draft"
```

Summary

Type	Values Stored	Recommended When
Regular `choices`	Strings/ints	Simple use cases
`TextChoices`	Strings	Readability + safety
`IntegerChoices`	Integers	Performance, compact storage

Looping Through Choices Manually in a Template

If you want full control:

```
<select name="priority">
    {% for value, label in model.priority.field.choices %}
        <option value="{{ value }}" {% if model.priority.value
== value %}selected{% endif %}>
            {{ label }}
        </option>
    {% endfor %}
</select>
```

Displaying the Human-Readable Value in Templates

Use:

```
{{ task.get_priority_display }}
```

Example:

```
<p>Priority: {{ task.get_priority_display }}</p>
```