# Updating and Deleting Records

**Objective**: Learn how to implement update and delete functionality in Django, handling HTTP methods for CRUD operations in the TODO App.

## Implementing Update and Delete Functionality

Updating and deleting records are common tasks in web applications. Django provides built-in views and methods to simplify these operations.

## Update View

An update view allows users to edit existing records. For the TODO App, this view will allow users to update task details.

**Function-Based Update View Example:**

```python
# views.py
from django.shortcuts import render, get_object_or_404, redirect
from .forms import TaskForm
from .models import Task

def update_task(request, pk):
    task = get_object_or_404(Task, pk=pk)
    if request.method == 'POST':
        form = TaskForm(request.POST, instance=task)
        if form.is_valid():
            form.save()
            return redirect('task_list')
    else:
        form = TaskForm(instance=task)
    return render(request, 'update_task.html', {'form': form})
```

## Template Example:

```
<!-- update_task.html -->
<form method="post">
  {% csrf_token %} {{ form.as_p }}
  <button type="submit">Update Task</button>
</form>
```

## Delete View

A delete view allows users to remove records. For the TODO App, this view will allow users to delete tasks.

**Function-Based Delete View Example:**

```python
# views.py
from django.shortcuts import render, get_object_or_404, redirect
from .models import Task

def delete_task(request, pk):
    task = get_object_or_404(Task, pk=pk)
    if request.method == 'POST':
        task.delete()
        return redirect('task_list')
    return render(request, 'delete_task.html', {'task': task})
```

## Template Example:

```
<!-- delete_task.html -->
<form method="post">
  {% csrf_token %}
  <p>Are you sure you want to delete "{{ task.title }}"?</p>
  <button type="submit">Yes, delete</button>
</form>
```

## Handling HTTP Methods (GET, POST) for CRUD Operations

Django views can handle different HTTP methods to perform CRUD operations. For example, a create view handles POST requests to create new records, while an update view handles POST requests to update existing records.

## Function-Based View Example for Update:

```python
# views.py
from django.shortcuts import render, get_object_or_404, redirect
from .models import Task
from .forms import TaskForm

def update_task(request, pk):
    task = get_object_or_404(Task, pk=pk)
    if request.method == 'POST':
        form = TaskForm(request.POST, instance=task)
        if form.is_valid():
            form.save()
            return redirect('task_list')
    else:
        form = TaskForm(instance=task)
    return render(request, 'update_task.html', {'form': form})
```

## Function-Based View Example for Delete:

```python
# views.py
from django.shortcuts import render, get_object_or_404, redirect
from .models import Task

def delete_task(request, pk):
    task = get_object_or_404(Task, pk=pk)
    if request.method == 'POST':
        task.delete()
        return redirect('task_list')
    return render(request, 'delete_task.html', {'task': task})
```

## Summary

Implementing update and delete functionality allows users to manage records efficiently. By handling HTTP methods appropriately, you can perform CRUD operations and ensure data integrity in your Django application.