**Inserting data in Django via AJAX without a page refresh.**

[[biswajit panda](https://medium.com/@biswajitpanda973?source=post_page-----be50e65bd9d2--------------------------------)](https://medium.com/@biswajitpanda973?source=post_page-----be50e65bd9d2--------------------------------)

[biswajit panda](https://medium.com/@biswajitpanda973?source=post_page-----be50e65bd9d2--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F2cd894ffbe14&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40biswajitpanda973%2Finserting-data-in-django-via-ajax-without-a-page-refresh-be50e65bd9d2&user=biswajit+panda&userId=2cd894ffbe14&source=post_page-2cd894ffbe14----be50e65bd9d2---------------------post_header-----------)

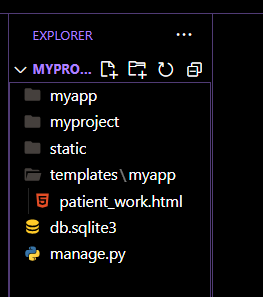
7 min read

·

Dec 11, 2023

5

In simple terms, AJAX (Asynchronous JavaScript and XML) is a technology that allows web pages to update or fetch information from a server without needing to reload the entire page. It’s like having a conversation with a server in the background while you interact with a webpage. Imagine you’re on a social media site and you want to post a comment — with AJAX, you can submit that comment without having to refresh the entire page. It makes web pages feel more dynamic and responsive, providing a smoother user experience by handling data behind the scenes without disrupting what you see on the screen.



this is my folder structure

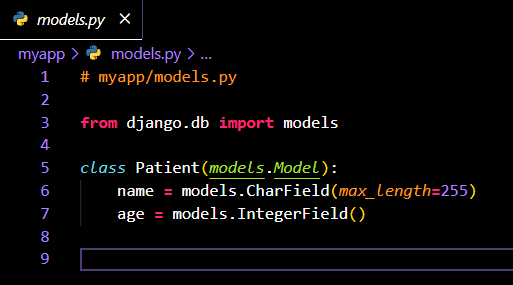
You can read this article for more details about folder structure

**[Render your first html page with django](https://medium.com/@biswajitpanda973/render-your-first-html-page-with-django-3e96721b180d?source=post_page-----be50e65bd9d2--------------------------------" \t "_blank)**

[in this tutorial from scratch we will setup django in our system and we will render one html page](https://medium.com/@biswajitpanda973/render-your-first-html-page-with-django-3e96721b180d?source=post_page-----be50e65bd9d2--------------------------------" \t "_blank)

[medium.com](https://medium.com/@biswajitpanda973/render-your-first-html-page-with-django-3e96721b180d?source=post_page-----be50e65bd9d2--------------------------------" \t "_blank)

In the provided example, we used AJAX (Asynchronous JavaScript and XML) to handle form submissions without refreshing the entire web page.

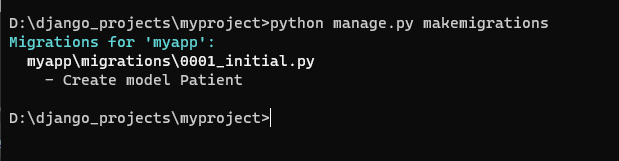


In models.py, we define a Django model named Patient with fields name and age. Using Django's ORM, it creates a database table (myapp\_patient) to store patient information, facilitating Python-based database interactions in the Django app.

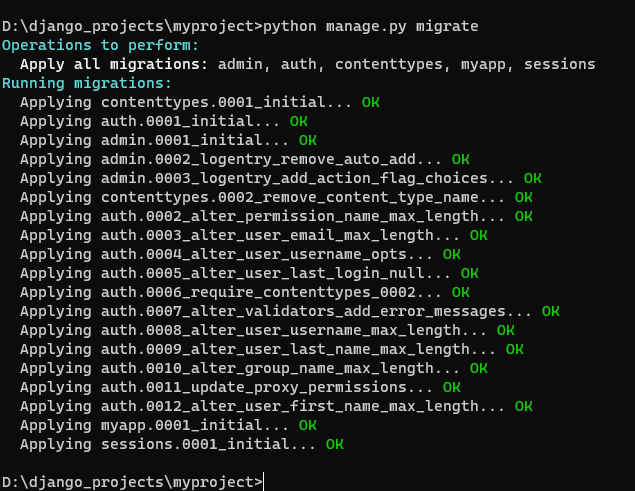
To apply the changes defined in your models.py file to the database, you need to use Django's management commands. Here are the typical commands you would run:

This command is used to generate new migration files based on the changes in your models.

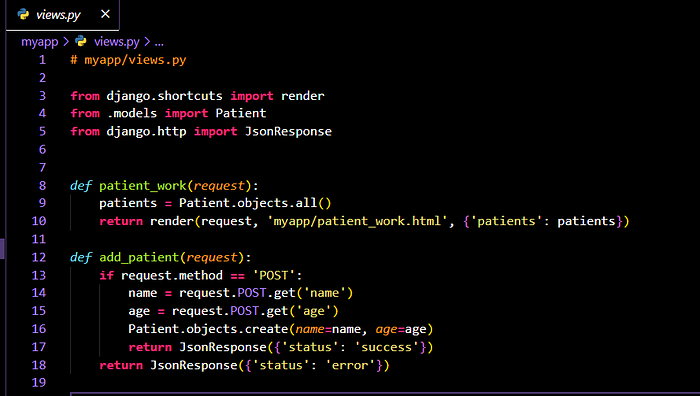
python manage.py makemigrations



python manage.py migrate



After running these commands, Django will create the corresponding database table (myapp\_patient) with the specified fields (name and age). These commands should be executed whenever you make changes to your models and want to update the database schema accordingly.



# myapp/views.py  
  
from django.shortcuts import render, JsonResponse  
from .models import Patient  
from django.views.decorators.csrf import csrf\_exempt  
from django.http import HttpResponse  
  
def patient\_work(request):  
 patients = Patient.objects.all()  
 return render(request, 'myapp/patient\_work.html', {'patients': patients})  
  
@csrf\_exempt  
def add\_patient(request):  
 if request.method == 'POST':  
 name = request.POST.get('name')  
 age = request.POST.get('age')  
 Patient.objects.create(name=name, age=age)  
 return JsonResponse({'status': 'success'})  
 return JsonResponse({'status': 'error'})

The JsonResponse class in Django is used to send JSON-formatted responses from the server to the client.

* Success Indication

The status field can indicate whether a requested operation (in this case, adding a patient) was successful. For example, {'status': 'success'} signals that the patient was added without any issues.

* Error Indication

Conversely, the status field can be used to indicate an error or failure in the operation. For example, {'status': 'error'} signals that there was an issue while processing the request.

* The add\_patient view function in views.py handles the AJAX request when submitting the form, adding a new patient to the database.
* The patient\_work.html template is used to display the current list of patients and contains a form to add new patients. The template uses Django template syntax to loop through existing patient data received from the server.

<!-- myapp/templates/myapp/patient\_work.html -->  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Patient Work</title>  
 <!-- Include jQuery -->  
 <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>  
</head>  
<body>  
  
<div>  
 <!-- Entry Form -->  
 <form id="submitForm">  
 {% csrf\_token %}  
 <label for="name">Patient Name:</label>  
 <input type="text" id="name" name="name" required>  
 <label for="age">Patient Age:</label>  
 <input type="text" id="age" name="age" required>  
 <button type="button" id="addPatient">Add Patient</button>  
 </form>  
</div>  
  
<div>  
 <!-- Table -->  
 <table id="patientTable">  
 <thead>  
 <tr>  
 <th>Name</th>  
 <th>Age</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for patient in patients %}  
 <tr>  
 <td>{{ patient.name }}</td>  
 <td>{{ patient.age }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
</div>  
  
<script>  
 $(document).ready(function () {  
 $('#addPatient').on('click', function () {  
 // Obtain the CSRF token from the hidden input field  
 var csrfToken = $('[name="csrfmiddlewaretoken"]').val();  
  
 // Get the data from the form  
 var formData = {  
 'name': $('#name').val(),  
 'age': $('#age').val()  
 };  
  
 // Send AJAX request  
 $.ajax({  
 type: 'POST',  
 url: '/myapp/add\_patient/', // Adjust the URL as per your project structure  
 data: formData,  
 headers: {  
 'X-CSRFToken': csrfToken  
 },  
 success: function (response) {  
 if (response.status === 'success') {  
 // Clear the form  
 $('#name').val('');  
 $('#age').val('');  
  
 // Update the table  
 $('#patientTable tbody').append(  
 '<tr><td>' + formData.name + '</td><td>' + formData.age + '</td></tr>'  
 );  
 } else {  
 alert('Error adding patient');  
 }  
 },  
 error: function () {  
 alert('Error adding patient');  
 }  
 });  
 });  
 });  
</script>  
  
</body>  
</html>

* <form id="submitForm">: This opens the HTML form element with the ID "submitForm."
* {% csrf\_token %}: This Django template tag inserts a hidden input field containing the CSRF token. It's crucial for protecting against Cross-Site Request Forgery (CSRF) attacks.
* <label> and <input> elements create fields for entering patient information (name and age).
* The required attribute ensures that the fields must be filled out before the form can be submitted.
* <button type="button" id="addPatient">Add Patient</button>: This is a button with the ID "addPatient" and a type of "button" to prevent the default form submission behavior. It triggers the AJAX request.

**Now, let’s explain the AJAX part using jQuery:**

<script>  
 $(document).ready(function () {  
 $('#addPatient').on('click', function () {  
 // Obtain the CSRF token from the hidden input field  
 var csrfToken = $('[name="csrfmiddlewaretoken"]').val();  
  
 // Get the data from the form  
 var formData = {  
 'name': $('#name').val(),  
 'age': $('#age').val()  
 };  
  
 // Send AJAX request  
 $.ajax({  
 type: 'POST',  
 url: '/myapp/add\_patient/', // Adjust the URL as per your project structure  
 data: formData,  
 headers: {  
 'X-CSRFToken': csrfToken  
 },  
 success: function (response) {  
 if (response.status === 'success') {  
 // Clear the form  
 $('#name').val('');  
 $('#age').val('');  
  
 // Update the table  
 $('#patientTable tbody').append(  
 '<tr><td>' + formData.name + '</td><td>' + formData.age + '</td></tr>'  
 );  
 } else {  
 alert('Error adding patient');  
 }  
 },  
 error: function () {  
 alert('Error adding patient');  
 }  
 });  
 });  
 });  
</script>

the JavaScript code uses jQuery to handle the click event on the “Add Patient” button, collects form data, sends an AJAX request to the server, and updates the table based on the server’s response. The CSRF token is included for security purposes. Adjust the URLs and table IDs based on your project structure

JavaScript (jQuery) Section:

* <script>: This opens a script tag to contain the JavaScript code.
* $(document).ready(function () { ... });: This ensures that the script runs after the document (HTML) has been fully loaded.

Event Binding:

* $('#addPatient').on('click', function () { ... });: This binds a click event to the "Add Patient" button with the ID "addPatient."

Obtaining CSRF Token:

* var csrfToken = $('[name="csrfmiddlewaretoken"]').val();: This retrieves the CSRF token from the hidden input field in the form.

Data Collection:

* var formData = { 'name': $('#name').val(), 'age': $('#age').val() };: This collects the patient's name and age from the form.

— — — — — — — — — — — — — — — — — — — — — — — — -

The $.ajax({ ... }); is a jQuery method used for making AJAX (Asynchronous JavaScript and XML) requests to a server. Here's a detailed breakdown of each parameter within the $.ajax() function:

$.ajax({  
 type: 'POST', // HTTP method (e.g., 'GET' or 'POST')  
 url: '/myapp/add\_patient/', // URL to which the request is sent  
 data: formData, // Data to be sent to the server (in this case, patient information)  
 headers: {  
 'X-CSRFToken': csrfToken // CSRF token included in the request headers for security  
 },  
 success: function (response) { // Callback function executed when the request is successful  
 if (response.status === 'success') {  
 // Code to handle success, such as clearing the form and updating the table  
 } else {  
 // Code to handle an unsuccessful response  
 }  
 },  
 error: function () { // Callback function executed when there is an error with the request  
 // Code to handle errors  
 }  
});

Here’s an explanation of each parameter:

type:

Specifies the HTTP method for the request. In this case, it’s set to 'POST' since the form data is being submitted to the server.

url:

Specifies the URL to which the request is sent. Adjust this URL to match the endpoint in your Django project. In the provided example, it’s set to '/myapp/add\_patient/'.

data:

Specifies the data to be sent to the server. In this case, it’s set to formData, which contains the patient's name and age collected from the form.

headers:

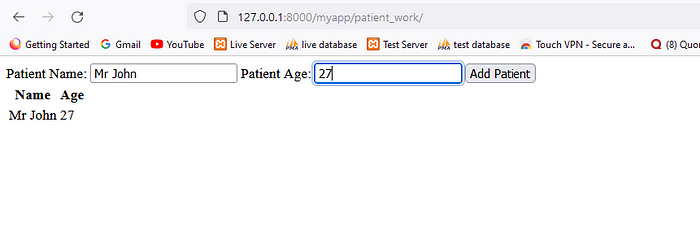
Specifies additional headers to include in the request. In this case, it includes the CSRF token in the headers for security. The CSRF token is obtained from the hidden input field in the form.

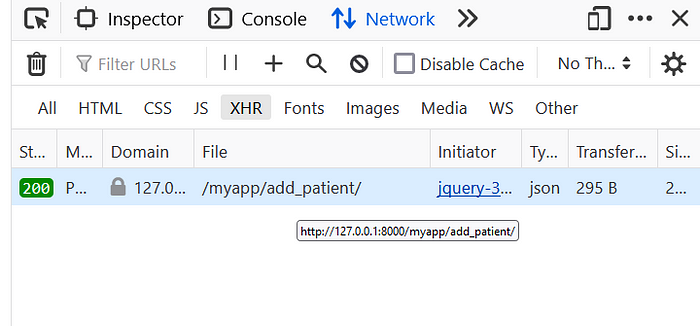
success:

* A callback function that is executed when the request is successful. The response parameter contains the data sent back by the server. In this example, it checks if the status in the response is 'success' and performs actions accordingly.

error:

* A callback function that is executed when there is an error with the request. It provides a way to handle errors gracefully





To monitor AJAX calls in the browser’s Developer Tools, you can use the “Network” tab.

Navigate to the “Network” Tab:

* Within the Developer Tools, find and select the “Network” tab.

Initiate AJAX Request:

* Trigger the AJAX request on your web page (e.g., by clicking a button or submitting a form).

Observe the XHR Entries:

* In the “Network” tab, you should see entries corresponding to each network request, including AJAX (XHR) calls.
* Filter by “XHR” to specifically view AJAX requests.