How it Works using jQuery!

AJAX has now become an important part of web applications. AJAX also helped Django a lot to increase efficiency and save the time of users. All the popular services out there use it in some way or the other.

Applications like Facebook, Twitter, Instagram, Gmail, Google Maps, Spotify, etc cannot function without AJAX.

You will have to deal with AJAX code at some

point. Even Google search uses AJAX. You must have seen those predictions down the search bar while typing, those are AJAX generated results. We are going to learn the implementation of AJAX in Django in this tutorial.

This tutorial will become easy to understand if you know JavaScript. It will be amazing if you also have some experience with **DOM (Document**

Object Model).

AJAX in Django

We talked about AJAX and how it is present in our web pages. Now, let's learn why we need AJAX in Django. AJAX is an acronym of **Asynchronous JavaScript and XML**.

It is a mixture of technologies implemented to solve a very serious problem. It utilizes *XMLHttpRequest objects* to transmit and receive data.

Stay updated with latest technology trends
Join DataFlair on Telegram!!

The Problem that AJAX Solved

Imagine that you are reading your emails. Now, think that for every new mail received, you have to reload the entire page which involves some steps:

- Request from your browser

to reload the page.

- The server receives and responses accordingly with new mails added.
- The process continues every time you want to read a new mail.

This approach is very time-consuming and unproductive. The problem

becomes more serious when urgent replies are demanded by the recipient. Apparently, it was a real situation for websites before 2003.

In late 2003, all the browsers solved the above problem by using XHR. XHR is acronym for **XMLHttpRequest** Objects. Browsers use the XHR object to communicate with the server. This transmission happens without reloading the whole page.

XMLHttpRequest API is behind the scenes handling the XHR requests.

Now, almost all browsers have XMLHttpRequest APIs. This made applications like Gmail, Google Maps become a reality.

How AJAX works in Django

AJAX is nothing but a combination of JavaScript and XHR object. The concept is simple:

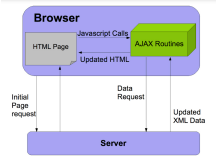
- **JavaScript Code on the client** – Side/ browser makes a request when an event occurs on the web page. The JavaScript code will generate an XHR object and is sent as a request object to the server. The XHR object contains some JavaScript data/ object.
- The XHR object also contains the URL

or name of the call-back function on server.

- **The request is handled by the server with a callback function** – The appropriate view function or callback function handles the request. It will send a success or failure response. Since the request is asynchronous, the rest of the code

executes without interruptions.

- At that time, the server processes the request.



- **The response is received as success and failure** – The success response can contain some server data in multiple formats like:

1. Text Format

Html Format (This contains HTML elements).

2. JSON Format

JSONP Format (It is JSON but only when the response comes from a different domain).

Script (This contains some JavaScript to be changed in page).

3. XML Format

Similarly, a failure response can be formatted.

- **The JavaScript will execute according to the response received** – The AJAX routine will now execute according to data from the server. We can change the HTML elements or run some JavaScript. Many things are possible using it.

The best example is Facebook's like button.

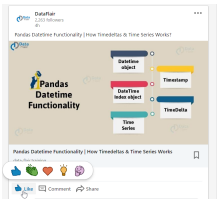
AJAX has multiple implementations but this is the common flow.

Example of AJAX in Real World

AJAX is very common in web applications. For example – when you like a post on LinkedIn, there are certain actions performed at the back-end but at the front-end, the color of

Like button changes.

When you clicked that button, an XHR object was sent to the server. The server receives the request and registers a like for that post. Then, the server sends a success response and the **Like** button turns to **Blue**



Using jQuery for AJAX in Django

jQuery is one of the most popular libraries of JavaScript. Many advanced frameworks like *Angular JS, React,* etc are based on jQuery. It is a very useful library providing functions with DOM of web pages. To use bootstrap's JavaScript part, jQuery is essential.

We are using jQuery for the following reasons:

- **AJAX implementation is different on multiple browsers**

There are many browsers which use different XHR APIs. These APIs parse request and response in a slightly different manner. Keeping your code up-to-date while also making it platform independent can be difficult. jQuery provides you with a built-in solution and takes care of all these things for you.

Since jQuery is actively developed thus, the code is kept updated frequently.

- **jQuery provides a vast range of methods using DOM**

jQuery contains many DOM objects and related methods. These objects can be used as per your need and jQuery

makes it more convenient to use.

- **jQuery is very popular**

jQuery is heavily used in the IT industry. For any web developer, they will deal with jQuery at some point. jQuery has many features and it will be an important skill for you in the long run.

- **jQuery is basic for many popular frameworks**

There are many popular frameworks

like Angular Js, react js based on jQuery. Bootstrap JavaScript is also based on jQuery. jQuery is basically everywhere on JavaScript frameworks.

These reasons are good enough for anyone to learn and use jQuery. There are more ways by which we can use AJAX in Django like, dajax, dajaxice, etc. These applications are made such that there is minimal

JavaScript for AJAX.

In the real world, developers prefer jQuery, therefore, we are going to make an application based on that.

Simple Post App and Like Button

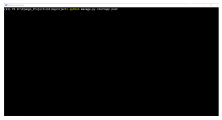
We are going to create a simple project where users can like a post. For this, we will be adding posts through admin model. Let's follow these steps:

- **Make a new app 'post'**

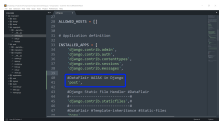
We all know how to ***create a new application in Django***. In your PowerShell/ Terminal, execute this command:

```
1. p
   ma
   st
   po
```

Code Display:



Now, install the application in settings.py of project. Add the name of the application in INSTALLED_APPS list.



- **Creating models – ‘Post’ and ‘Like’**

Models are the tables in our databases. Paste this code in post/ models.py file.

Code:

```
1. f
mo
2.
3.  #
#D
Po
po
mo
=
4.
mo
5.
mo
=
'a
6.
7.
8.  c
9.  p
mo
on
```

Output:

Here, Post and Like are two models. We are using the **Foreign Key Field**. It is used to define many-to-one relationships between models. The post variable in Like model will contain the primary key of the post that we added. Thus, we can then refer to that post's data.

We are not using it in this example though.

- **Creating view functions – 'index' and 'like'**

Open post/view.py file and paste this code in it.

Code:

```
1.  f
    im
2.  f
    Li
3.  f
    Ht
4.  #
5.
6.  #
7.  d
8.
    Po
9.
    'p
    'p
10.
11. d
12.
    'G
13.
    re
14.
    Po
    po
15.
    po
16.
17.
    Ht
18.
19.
    Ht
```

Output:

We created two view functions. The first one is to render all the posts and the second one is for the like function.

The like function exists to handle the AJAX request from like button. As you can see, it is similar to the function that we use in forms.

The like function will first check for the request. If it is the GET method, then it will read the data. The data is the id of the post we liked. Then, we create a Like class object and save its id equal to data. Then, we store the object in our database.

At last, we are returning an HttpResponse. This response will do nothing in our code as we are not printing the same. We could have just passed a script. These

can be utilized according to one's use case.

- **Creating templates – 'index.html'**

Here comes the main part, templates, as AJAX is a front-end technology. To implement this, create some directories.

1. Inside post directory, create templates directory.
2. Inside the templates directory, create post directory.

3. Create index.html inside post/templates/post directory/folder.

Now, paste the html in index.html

Code:

```
<
<
<

hr
in
gg
cr
<
Wp
<
<

AP


</
<

}}
hr
<
<
{

<
$
v
i
$
{


}
s
{
```

```
34.
35.
36.   <
37.   <
38.   <
```

Output:

There are two important parts in this HTML file.

- **Importing jQuery and Bootstrap**

Since jQuery is a JavaScript library, so it is essentially a static file for the server. It can be imported

the same as bootstrap and other files. We chose to use it directly from CDN. The link can be found on jQuery's official website.

Similarly, we used the link tag with bootstrap.

- **Generating AJAX request from JavaScript**

Here, is the important

part. If you have no idea of DOM, ***JavaScript Objects***, then this might be difficult to understand. We will try our best to make it relatable.

Focus on <script> </script> at the end of document. We have put our AJAX code there.

We are using some of the properties of jQuery to call the object. The like button is the class of button that we are going to click.

When the button is clicked, the function inside it is executed. That function has some objects in it.

This is an **XHR** object. Actually, this is the speciality of **$.ajax() method**. It is what will make our code run on multiple browsers. Inside that, there is a JavaScript Object Datatype.

JavaScript Object is similar to dictionaries in Python. This is for Pythonistas who don't know JavaScript.

Now, you can relate it easily.

1. Type key: It takes in the method of request that we will send. Here, XHR is sent over the GET request.

2. URL key: This is the URL of the call back function on the server. Remember the view function we created? This code will call that function

without reloading the page.

3. Data key: This key contains the data that we are sending to call back function. In this example, it contains the id of the post. It can be an object and can contain multiple values. You can also pass it in a JSON. It depends on your use case.

4. Success method: JavaScript Objects have methods too. You can define methods like this easily. The methods

are called properties of Objects. Success is a built-in response for **$.ajax() method**. This method will be executed when the AJAX request returns with a success response. It is one of the conveniences that jQuery provides to programs.

5. Inside Success: There are two methods that we are executing inside success. The first one will remove the class inside the parenthesis

(). Similarly, the other one will add the class.

It is as simple as that. No, worries if you didn't get it right the first time.

- **Connecting it with urls-config**

Now, comes the Python part. You just created a very beautiful app. Now we have to link all the components together.

Make a new python file in post directory.

Inside post/urls.py, paste this Code:

```
1.  f
    dj
    im
2.  f
    dj
    im
3.  f
    vi
4.  u
5.  #
    #A
6.
    pa
    vi
    na
7.
    pa
    vi
    na
```

Output:

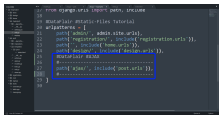
Now, configure your main urls.py file.

Add this to your

urlpatterns list:

```
1.  p
in
```

Output:



- ### Populating the database via admin

Now, open post/admin.py file.

Paste this code:

```
1.  f
im
2.  f
3.  #
he
4.  #
5.  a
)
6.  a
)
```

Output:

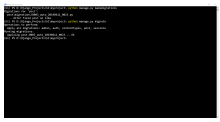
We are just registering our models

in Django admin.

Now, open console/ PowerShell/ Terminal and run these commands in order.

```
1. p
   ma
   ma
2.  p
   ma
   mi
```

Code Display:



Start your server and add some posts from admin.

URL:

http://localhost:8000/admin/

Populate your database with some posts.

- **Running server**

We already started the server. Now, search for **URL**:

http://localhost:8000/ajax/

Isn't it great? We just created a very beautiful web page with AJAX

functionality. Try clicking on one of the like buttons.

Awesome, it turned to green. Now, you can actually build something amazing with Django which also looks good. Our Like button works. !! Awesome !!

Summary

I appreciate that you completed the tutorial. There were some parts which I think can be difficult for people who don't know JavaScript. In this tutorial, we took a brief look at AJAX in Django. We have observed AJAX real life problems, solutions and also how it works in Jquery.

Do let us know your opinion on this tutorial through comments.

Did you like our efforts? If Yes, please give DataFlair 5 Stars on Google | Facebook

Tags:

AJAX in Django

AJAX Real world Example

AJAX working in Django

Using jQuery for AJAX in Django

12 RESPONSES

Comments 12

Pingbacks 0

BALA SUBRAMANYAM.A

January 23, 2020 at 9:15 am

Thank you so much to data-flair team for nice and clear explanation on ajax in django.

I tried the tutorial. It is working fine with small issues.

at the
end of
saving
the
FORM
data to
database,
I am
trying
to
redirect
to
“HOME”
page
through
“views.py”
(using
return
redirect(reverse(‘home’))
But,
redirection
is not
working.
Form
data is
saving
to the
database.

It is
always
redirecting
to the
same
template
from
which
ajax
request
is
called.

Please guide me how to avoid “Redirect” is AJAX.

Thanks and regards Bala Subramanyam

Reply

DataFlair Team

January 28, 2020 at 3:05 pm

Hey Bala,

Check out our blog on Django redirection. Also, use redirect() without reverse(). redirect takes you to

the URL directly without adding to the page.

Thank You

Reply

Surendar

February 24, 2020 at 12:09 pm

can I use ajax script as an external link? (style.js)

because I tried this. it perfectly works for me. if I use ajax script as an external link it shows the

below
error

POST
403
(Forbidden)

please
help
me to
solve
this.

Thanks
in
advance.

Reply

Karan Mittal

December 21, 2020 at 7:08 pm

Yes, You can write AJAX Scripts in separate Js files and link them at the end of the page.

Although, if you are loading the script before the elements are fetched that might be giving the error. Also, you will have to configure your static URLs settings to serve the JS file. Check the static file handling section of dataflair

tutorials. That will be a great help.

Reply

Satya

April 15, 2020 at 11:18 am

Thanks for the help. I searched a lot of places for this AJAX implementation and only here, I found help.

Reply

Mrunal Mestry

May 4, 2020 at 10:32 am

It is not working me

getting
HTTP
404
error –
like?
post_id=1
HTTP/1.1″
404
2697
Please
Help
Me

Reply

Mrunal Mestry

May 4, 2020 at 11:48 am

It
Worked
well ..
Tried
Again
..
Thank
You

Reply

Joe Wilson

May 13, 2020 at 9:38 pm

Hei..
why
did the
error
on_delete

must
be
callable
when I
makemigrations...
is it
delete_on
=
'CASCADE'
still
valid?
or any
other
new
version
of
delete_on?
Please
help...
thank
you

Reply

Karan Mittal

December 21, 2020 at 7:05 pm

You can also use on_delete=models.CASCADE.

Reply

Riddhi Mistry

June 21, 2020 at

11:59
am
I was
getting
the
same
issue
this
code
solved
it:
post =
models.ForeignKey(Post,
on_delete=models.CASCADE)

Reply

Pierre

November
16, 2020
at 2:02
am
I was
able to
correct
this
one by
changing
to :

class
Like(models.Model):
post =
models.ForeignKey(Post,
on_delete
=
models.CASCADE)

Reply

Jenny
Damcevska

March

1, 2021
at 7:20
am

Thank
you
very
much
for the
following
tutorial.
I have
searched
for
quite
some
time
regarding
this
subject
and
this
was
one of
the
most
clear
tutorials
to
understand
regarding
Django
and
Ajax,
so
thank
you
very
much!
I
wanted
to ask

how it is possible to also change the text on the button, would this also be performed in the success method? Thank you

Reply

LEAVE A REPLY

Comment

Name **Email**

* *

Post Comment