使用Ajax局部刷新页面

I recently discovered a neat way of displaying data retrieved using jQuery AJAX in concert with Django’s template engine. You can create a view in Django which simply uses the [render\_to\_response](http://docs.djangoproject.com/en/dev/topics/http/shortcuts/#render-to-response) shortcut function to render the results server-side and then just use [jquery.load](http://docs.jquery.com/Ajax/load#urldatacallback) to dynamically fetch the results.

Eventhough, returning some raw JSON data is much more efficient as far as bandwidth is concerned, this method is a lot simpler.

I have been using jQuery for over a year now. I have found that its built-in DOM manipulation features are a bit limited, especially for manipulating tables (e.g., adding rows dynamically). This method is much cleaner than doing all that DOM manipulation.

Here is all the jQuery code to handle the search and AJAX spinner display:

[view source](http://www.nomadjourney.com/2009/01/using-django-templates-with-jquery-ajax/#viewSource)[print](http://www.nomadjourney.com/2009/01/using-django-templates-with-jquery-ajax/#printSource)[?](http://www.nomadjourney.com/2009/01/using-django-templates-with-jquery-ajax/#about)

01.$(document).ready(function() {

02.$('#searchSubmit').click(function() {

03.q = $('#q').val();

04.$('#results').html('&nbsp;').load('{% url demo\_user\_search %}?q=' + q);

05.});

06.});

07.

08.$(document).ajaxStart(function() {

09.$('#spinner').show();

10.}).ajaxStop(function() {

11.$('#spinner').hide();

12.});

Here is the Django view function that does the heavy lifting on the server-side:

[view source](http://www.nomadjourney.com/2009/01/using-django-templates-with-jquery-ajax/#viewSource)[print](http://www.nomadjourney.com/2009/01/using-django-templates-with-jquery-ajax/#printSource)[?](http://www.nomadjourney.com/2009/01/using-django-templates-with-jquery-ajax/#about)

01.def ajax\_user\_search(request):

02.if request.is\_ajax():

03.q = request.GET.get('q')

04.if q is not None:

05.results = User.objects.filter(

06.Q(first\_name\_\_contains = q) |

07.Q(last\_name\_\_contains = q) |

08.Q(username\_\_contains = q)).order\_by('username')

09.

10.template = 'results.html'

11.data = {

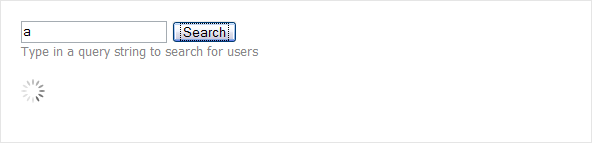
12.'results': results,

13.}

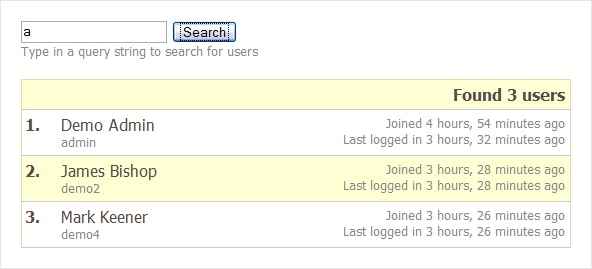
14.return render\_to\_response(template, data,

15.context\_instance = RequestContext(request))

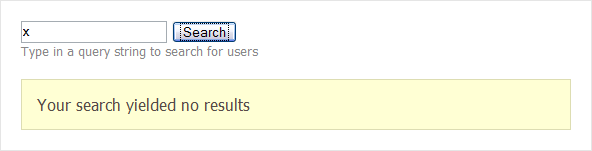
Here are some screenshots of the results:

[](http://www.nomadjourney.com/wp-content/uploads/2009/01/2_ajax1.png)

*AJAX operation in progress*

[](http://www.nomadjourney.com/wp-content/uploads/2009/01/3_results1.png)

*Returned results*

[](http://www.nomadjourney.com/wp-content/uploads/2009/01/1_noresults1.png)

*No results*

The sample Django project is included for your perusal and is released under the MIT license. I used the excellent[Aptana Studio](http://www.aptana.com/) IDE when working on this demo so it can be imported straight into it as an Aptana project.

[Download ajax\_user\_list.zip](http://www.nomadjourney.com/wp-content/uploads/2009/01/ajax_user_list.zip)