Joining REST API Call with AJAX within Inputbox to Reduce the Time to Load and Populate the Box as Well as Responds to User's Input

How to make Inputboxes with large amount of data more accessible

Tomasz Wrona, Splunk Developer

September 9, 2018 | Final

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf18

Introduction

Few words about my person

Tomasz Wrona

Splunk Developer

- Working in IT since 2000
- Experience in programming, data analysis, application integration and operations
- First contact with Splunk in 2011
- Since 2015 working with Splunk only for Vodafone Group, Vodafone Germany and German Railways

- Contact: https://www.linkedin.com/in/tomasz-wrona-freelance/

Input fields in Splunk

Multiselect and dropdown inputboxes

splunk> .conf18

Input fields in Splunk

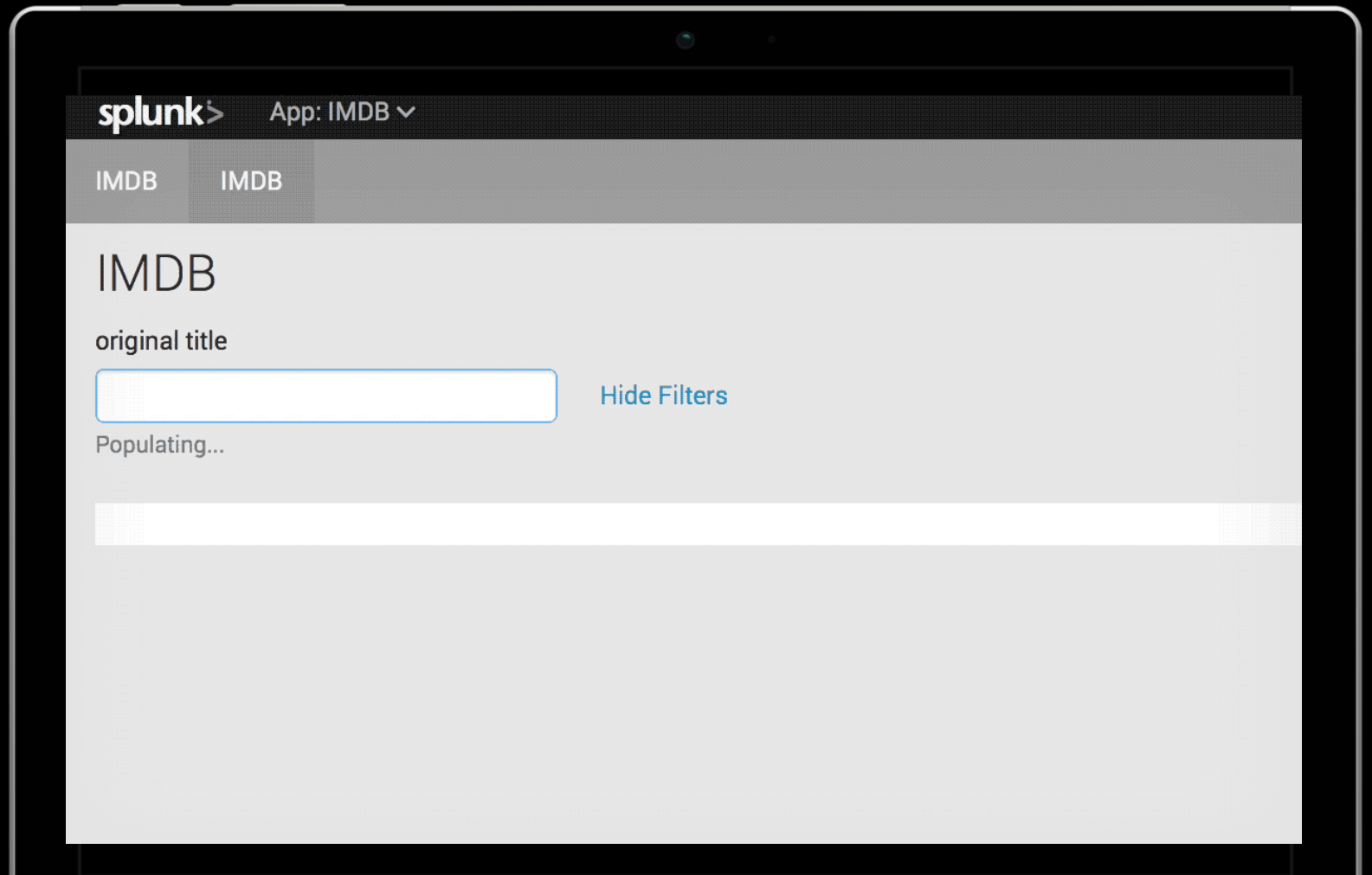
Multiselect and dropdown inputboxes

You all know how default input fields are behaving in Splunk's Simple XML:

- Behind each input field (multiselect or dropdown) there is are search which provide the box with the data
- All of the data is loaded and populated into the DOM structure of the page
- In case the search returned large amount of data, loading and populating take some time
- You always get all the results – making the use of the input not really comfortable

Input fields in Splunk

Multiselect and dropdown inputboxes



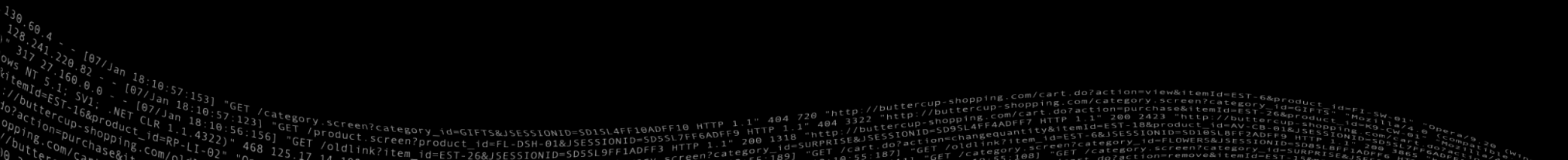
Display of multiselect where search returns 15,000 entries

Input fields in Splunk

Multiselect and dropdown inputboxes

What I needed was something similar to google.com input field:

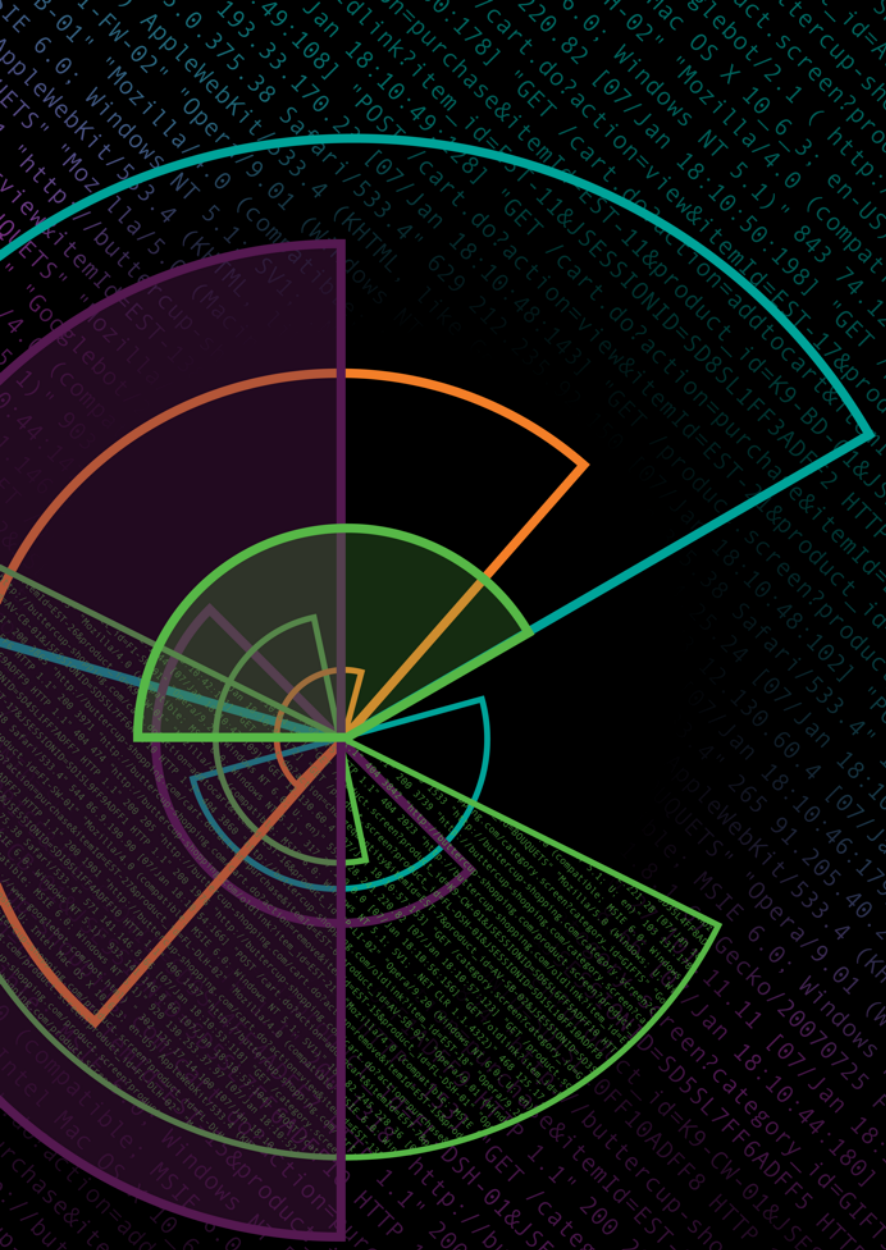
- Very fast loading regardless of data amount
- Autocompletion without reload
- Limited number of entries
- Tokens should be still functioning



splunk> .conf18

All of this can be achieved with the use of REST API and AJAX

splunk> .conf18



Ideas

Where to look for the solution...

Ideas

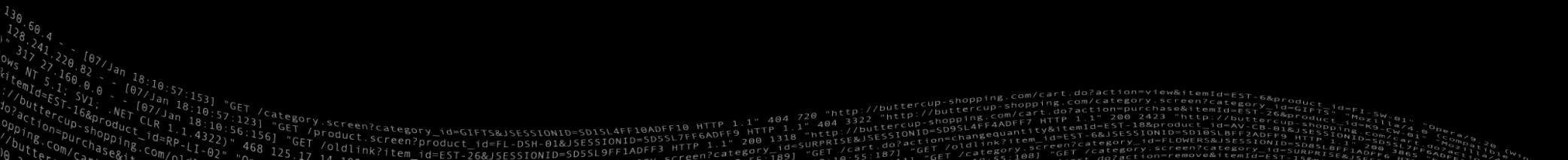
Where to look for the solution...

All Splunk input fields are based on select2

Check select2 documentation for possible solutions

https://select2.org/data-sources/ajax

Adapt the solution to select2 version available in Splunk

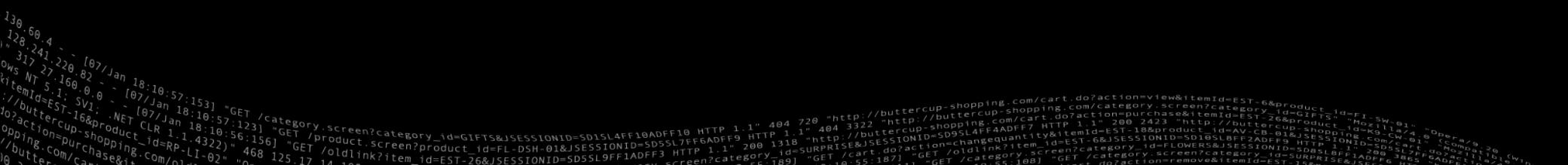


splunk> .conf18

Ideas

Where to look for the solution...

```
$(".js-example-data-ajax").select2({
  ajax: {
    url: "https://api.github.com/search/repositories", dataType: 'json', delay: 250,
    data: function (params) { return { q: params.term, page: params.page }; },
    processResults: function (data, params) { return { results: data.items }; },
    minimumInputLength: 1,
    templateResult: formatRepo,
    templateSelection: formatRepoSelection
});
```



Ideas

Where to look for the solution...

Things to be done:

- Extend the dashboard with javascript, put a multiselect field on our dashboard without a search
- Create a saved search for the input field and call it from a javascript
- Get search results, limit their number, update them based on user input
- Put it all together

Implementation

Just do it...

Implementation

Just do it...

For some results to be seen I have created IMDB app with free IMDB data extract taken from https://datasets.imdbws.com/title.basics.tsv.gz

I have also created a datamodel and my saved search is based on this datamodel. Moreover one dashboard imdb.xml has also been created.

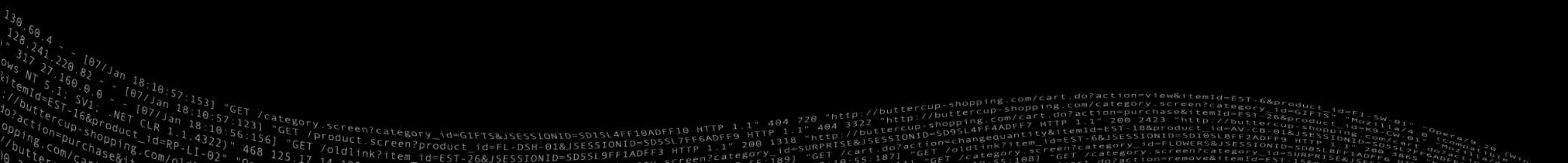
As all of this has been done on my laptop I have reduced the amount of records for the input field to 250.000.

Implementation

Just do it...

Things to be done:

- Extend the dashboard with javascript, put a multiselect field on our dashboard without a search
- Create a saved search for the input field and call it from a javascript
- Get search results, limit their number, update them based on user input
- Put it all together



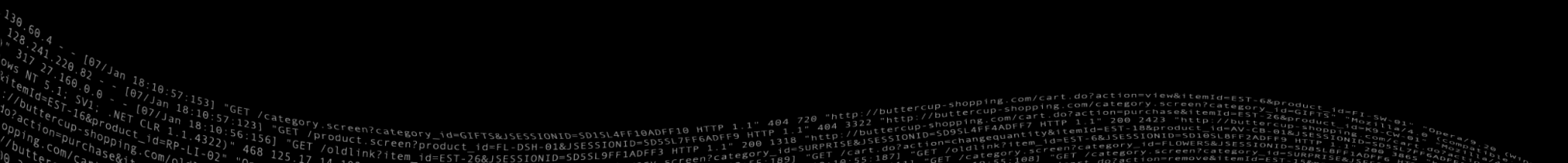
Implementation

Just do it...

- Extend the dashboard with javascript

First step is very easy, just put script parameter with script name into first line of the dashboard. The script must located in `SPLUNK_HOME/etc/apps/IMDB/appserver/static`:

```
<form script="restApiAndAjax.js">
```



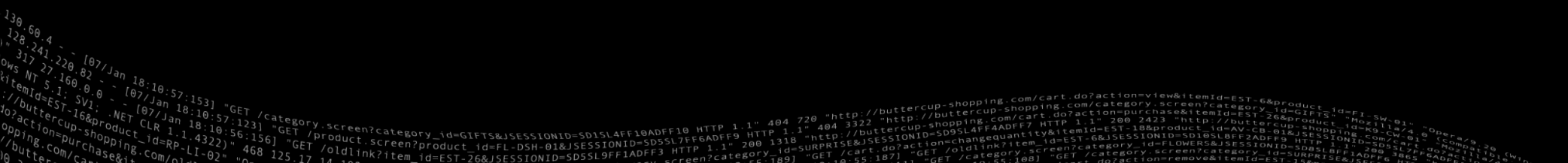
splunk> .conf18

Implementation

Just do it...

- Put a multiselect field on our dashboard without a search

`SPLUNK_HOME/etc/apps/IMDB/local/data/ui/views`



splunk> .conf18

Implementation

Just do it...

```
<input id="in_originalTitleTok" type="multiselect"
token=“originalTitleTok" searchWhenChanged="false">
  <label>Title</label>
  <valuePrefix>originalTitle="</valuePrefix>
  <valueSuffix>"</valueSuffix>
  <delimiter> OR </delimiter>
  <fieldForLabel>label</fieldForLabel> <!– saved search must
return those two values -->
  <fieldForValue>value</fieldForValue> <!– saved search must
return those two values -->
<allowCustomValues>true</allowCustomValues>
</input>
```

Implementation

Just do it...

- Create a saved search for the input field and call it from a javascript

Second step is not much more complicated then the previous one.

restApiAndAjax.js should look like this

Implementation

Just do it...

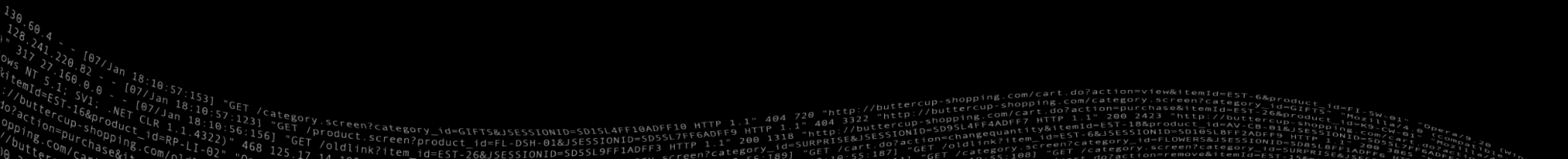
```
require([
  'underscore', 'jquery', 'splunkjs/mvc',
'splunkjs/mvc/searchmanager', 'splunkjs/mvc/simplexml/ready!'
], function (_, $, mvc, SearchManager) {
...
var sm = new SearchManager({
  id: "baseSearch",
  search: '| loadjob savedsearch="admin:imdb:imdb_search_ajax"'
});
...
```

Implementation

Just do it...

admin:imdb:imdb_search_ajax:

```
| tstats summariesonly=true count from datamodel=imdb_dm
by imdb_ds_root.originalTitle | head 250000
| rename imdb_ds_root.originalTitle as originalTitle
| eval value = originalTitle, label = originalTitle
| fields value label | sort limit=0 label
```



splunk> .conf18

Implementation

Just do it...

Things to be done:

- Extend the dashboard with javascript, put a multiselect field on our dashboard without a search
- Create a saved search for the input field and call it from a javascript
- Get search results, limit their number, update them based on user input
- Put it all together

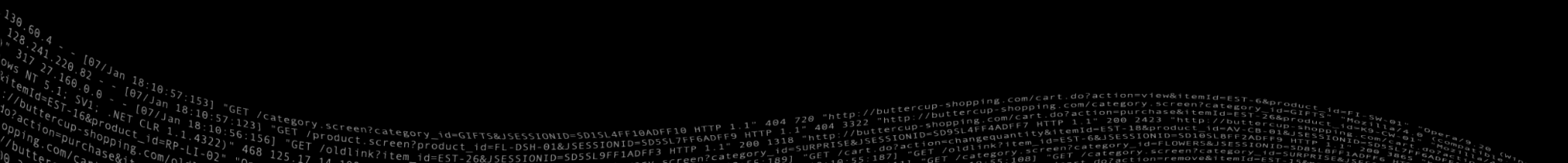
Implementation

Just do it...

- Get search results, limit their number, update them based on user input

Third step took me to Splunk REST API documentation:

http://docs.splunk.com/Documentation/Splunk/7.1.2/RESTREF/RESTsearch#search.2Fjobs.2F.7Bsearch_id.7D.2Fresults



Implementation

Just do it...

- Get search results, limit their number, update them based on user input

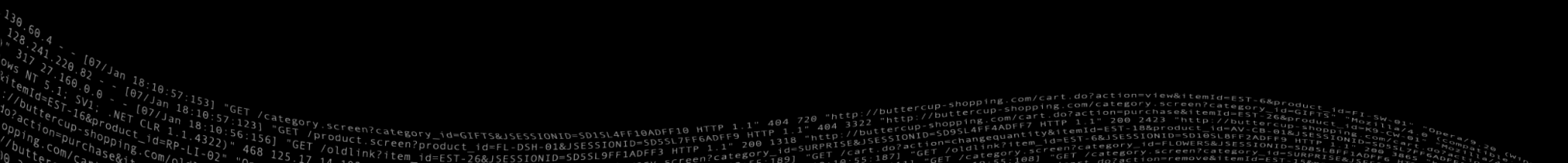
From there I got following result:

http://IP:PORT/de-DE/splunkd/__raw/services/search/jobs/SEARCH_SID/results

?search=SUBSEARCH → update search results with user input

&output_mode=json → needs to be json for select2 to work

&count=10 → limit for number of records



Implementation

Just do it...

Things to be done:

- Extend the dashboard with javascript, put a multiselect field on our dashboard without a search
- Create a saved search for the input field and call it from a javascript
- Get search results, limit their number, update them based on user input
- Put it all together

Implementation

Just do it...

- Put it all together - dashboard

```
<form script="ajax.js"><label>IMDB</label><fieldset submitButton="false"
autoRun="false"><input id="in_originalTitleTok" type="multiselect" (or dropdown)
token="originalTitleTok" searchWhenChanged="false">
<label>Title</label><valuePrefix>originalTitle="</valuePrefix>
<valueSuffix>"</valueSuffix><delimiter> OR
</delimiter><fieldForLabel>label</fieldForLabel>
<fieldForValue>value</fieldForValue><default>*</default>
<allowCustomValues>true</allowCustomValues></input></fieldset><row>
<panel><html><div id="title"></div></html></panel></row></form>
```

Implementation

Just do it...

- Put it all together – saved search (semicolons only for one page view)

[imdb_search_ajax]

action.email.useNSSubject = 1; alert.digest_mode = True

alert.suppress = 0; alert.track = 0

auto_summarize.dispatch.earliest_time = -30d@h; cron_schedule = */30 * * * *

dispatch.earliest_time = -60d; enableSched = 1

search = | tstats summariesonly=true count from datamodel=imdb_dm by imdb_ds_root.originalTitle | head 250000 | rename imdb_ds_root.originalTitle as originalTitle | eval value = originalTitle, label = originalTitle | fields value label | sort limit=0 label

Implementation

Just do it...

- Put it all together – javascript

```
require(['underscore','jquery','splunkjs/mvc','splunkjs/mvc/searchmanager','splunkjs/mvc/simplexml/ready!'],
function (_, $, mvc, SearchManager) {
var defaultTokens = mvc.Components.getInstance("default");
var sm = new SearchManager({
  id: "baseSearch", search: '| loadjob savedsearch="admin:imdb:imdb_search_ajax"' });
var tmp = $('#in_originalTitleTok').find('input').select2({
  multiple: 'true', (for dropdown: multiple: 'false')
```

Implementation

Just do it...

- Put it all together – javascript

```
minimumInputLength: 1,
ajax: {
  url: function () {return
        Splunk.util.make_url('splunkd/__raw/services/search/jobs/' +
        sm.query.attributes.data.sid + '/results');},
  dataType: 'json',
```

Implementation

Just do it...

- Put it all together – javascript

```
data: function (params) {
  var tok = defaultTokens.get("originalTitleTok"); var originalTitleTok = "";
  if (tok != null && tok != "" && tok != '(originalTitle="*")') {
      originalTitleTok = tok.replace(/originalTitle=/g, "value!="); originalTitleTok =
      originalTitleTok.replace(/OR/g, "AND"); }
  return { search: '| search value="*' + params + '*" ' + originalTitleTok,
      output_mode: "json", count: "10"};
},
```

Implementation

Just do it...

- Put it all together – javascript

```
results: function (data) { var tmp;
  if (data.results.length) {
    tmp = _.each(data.results, function (result) {
      result.id = result.value; result.text = result.label;
    });
  } else { tmp = []; }
return { results: tmp }; } }, });
```

Implementation

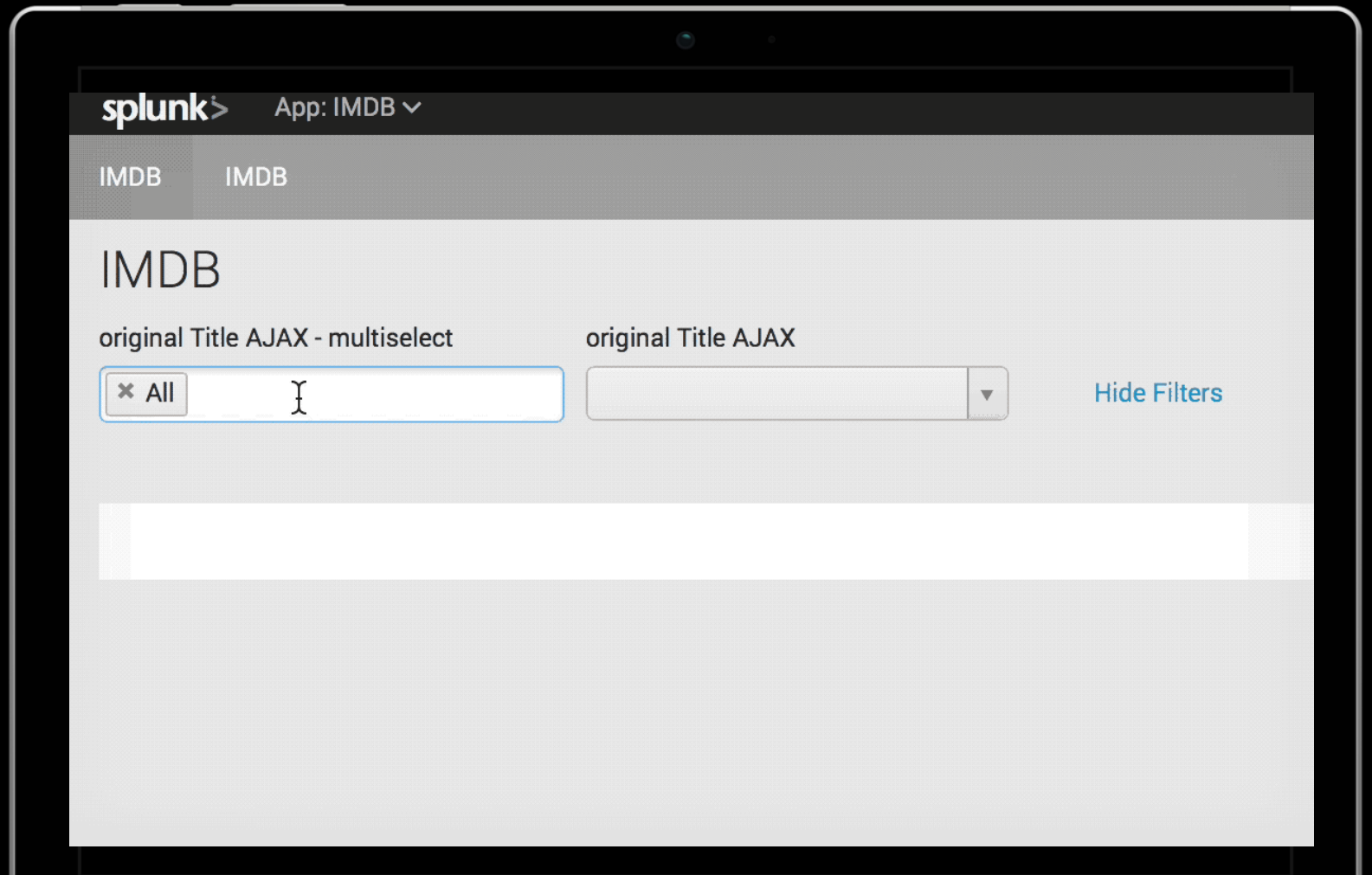
Just do it...

- Put it all together – javascript

```
defaultTokens.on("change:originalTitleTok",
  function (newIndexName, tokValue, options) {
    var tokenVal = defaultTokens.get("originalTitleTok");
    $('#title').text(tokenVal);
  });
});
```

Implementation

Just do it...



- Display of new multiselect/dropdown implementation where search returns 250,000 entries

Implementation

Just do it...

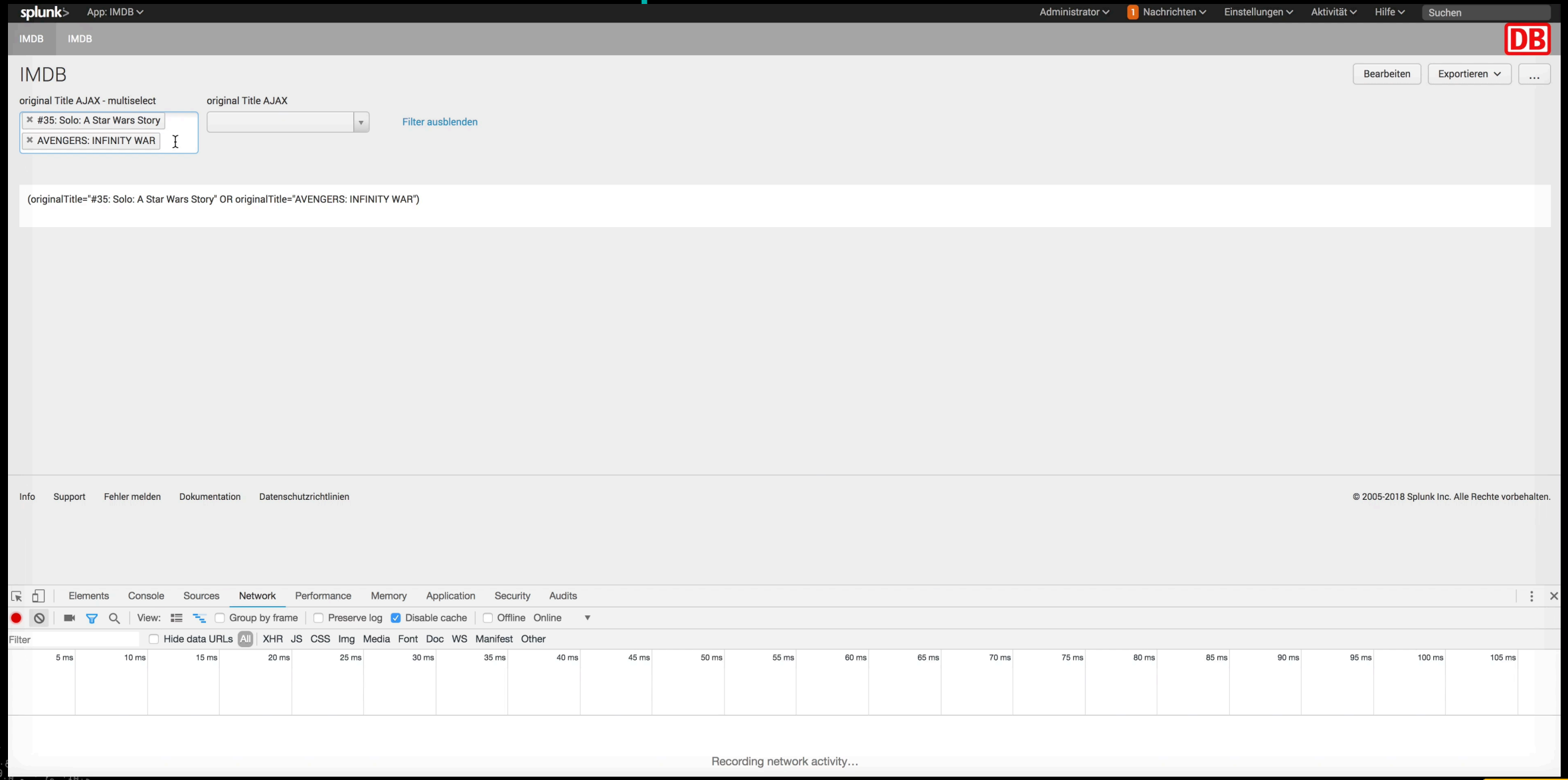
- Debug

In order to debug/check the input field just open developer bar in the browser, switch to network and start typing within the box. With each character the search will be called and can be seen in details.

You can copy the link from network tab and run it as regular URL.

You can also check/edit the URL parameters and see the difference immediately.

Implementation



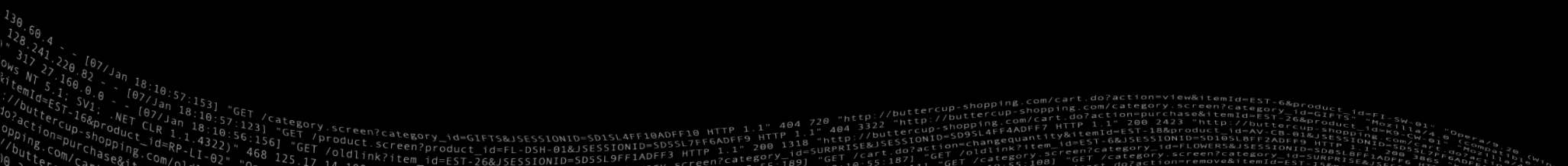
Conclusions

Where to go from here...

Conclusions

Where to go from here...

- Implement filter dependency for filter groups
- Create a global module out of the javascript code
- ...



Thank You!

Don't forget to rate this session in the .conf18 mobile app