SPRINGTAB

Guide to SpringTab integration

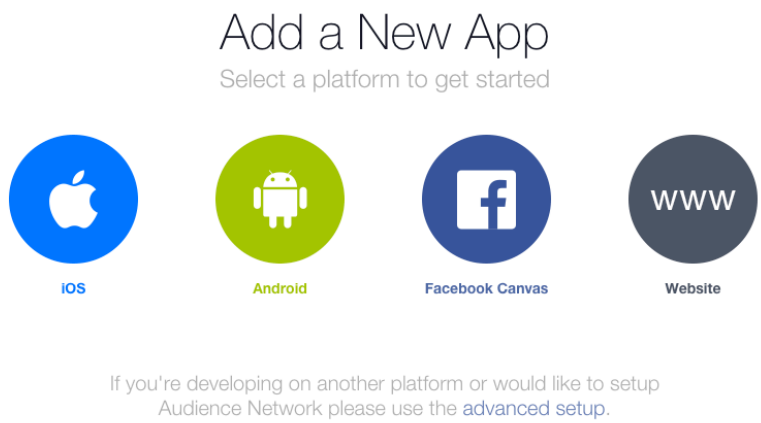
1. Data export integration

We are going through the SpringTab integration process on an existing website (without FB client side integration).

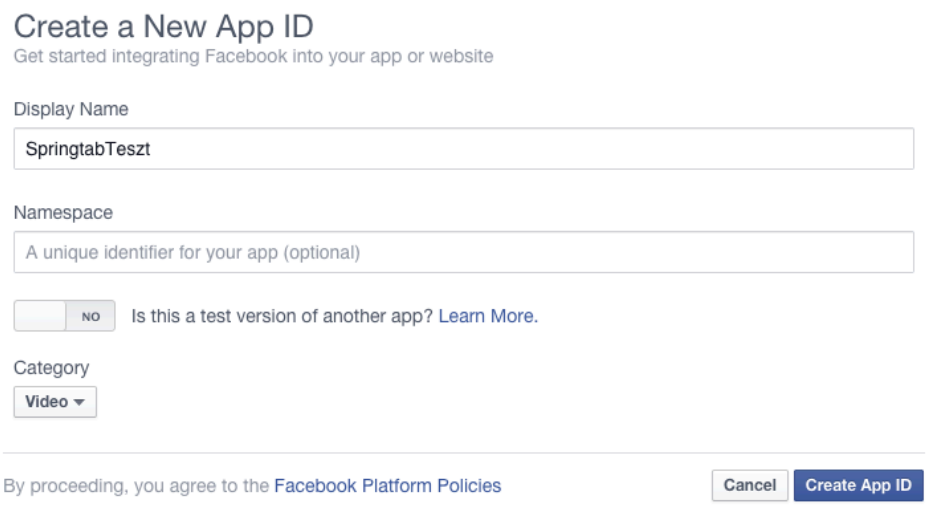
1.1 Creating a Facebook application

If there is an existing app for the website, then the process continues directly from step 1.3.

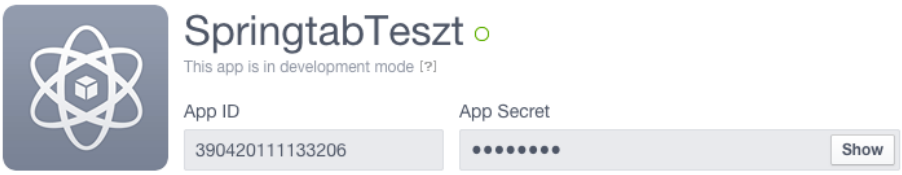
Create a new app at https://developers.facebook.com/



Choose the „advanced setup" option and specify a Display name and category. The Namespace field is optional.



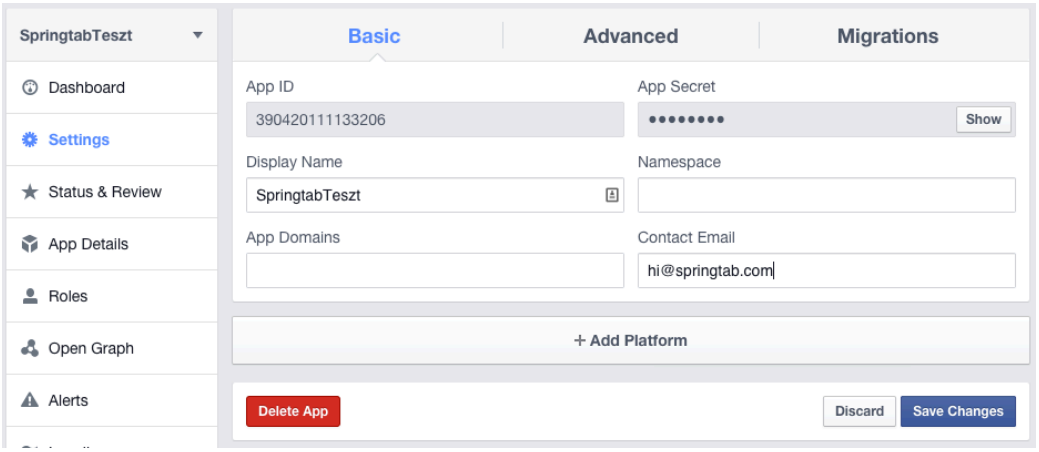
After creating the app, we can see its FB App ID, which is a vital part of further integration. Therefore, we should note it!



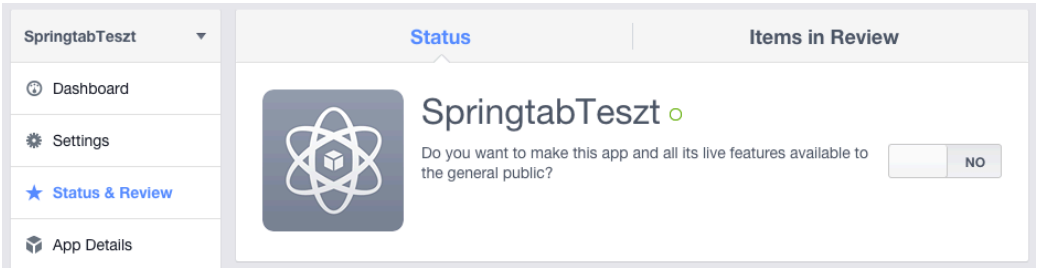
1.2 Publishing Facebook app

The app created so far is available only for developers and testers. In order for everyone to use the app, it needs to be published. The following two steps are required.

An email address needs to be specified under Contact Email tab under Settings.



Then, the app needs to be published under Status & Review.



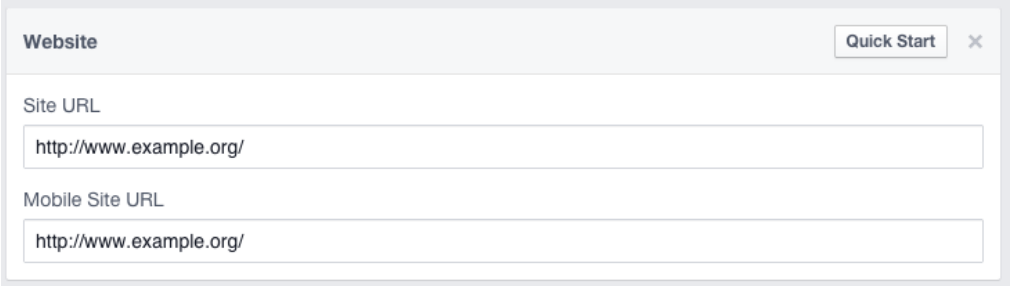
Pursuing this, the app is available for everyone.

1.3a Integration of Facebook app into the website

When integrating into the website, we need a publicly available website (in the example that is http://example.org), and access to the source code of the site.

As a first step, on the online platform, we should create a “Platform”, on which the login process can go smoothly. Go to Settings, click on Add and choose Website.

As site URL, specify the page where we would like to do the integration:



After saving the data, click on Quick start, where we can find the codes to be copied to the website. With the current (2.2 or newer sdk), the code is as follows:

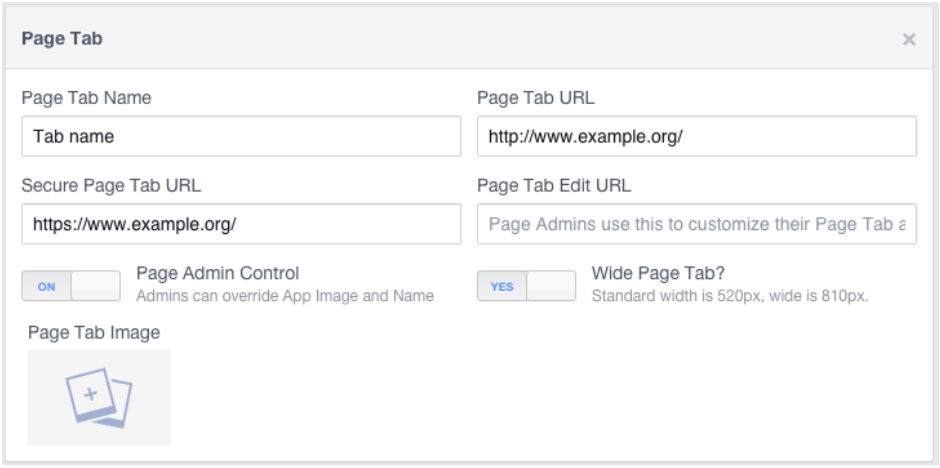
```
<script>
  window.fbAsyncInit = function() {
    FB.init({
      appId      : '390420111133206',
      xfbml      : true,
      version    : 'v2.2'
    });
  };

  (function(d, s, id){
     var js, fjs = d.getElementsByTagName(s)[0];
     if (d.getElementById(id)) {return;}
     js = d.createElement(s); js.id = id;
     js.src = "//connect.facebook.net/en_US/sdk.js";
     fjs.parentNode.insertBefore(js, fjs);
   }(document, 'script', 'facebook-jssdk'));
</script>
```

The code above should be placed directly after the <body> tag. The code may be modified at two points. If we do not want to use the API in English but in another language, such as Hungarian, then we have to specify connect.facebook.net/hu_HU/sdk.js instead of connect.facebook.net/en_US/sdk.js. Also, always check if the appid is the same as our App ID.

1.3b Integration of a Facebook app into a tab app

In the case of a tab app, similarly to 1.3a, we have to create a new platform, but now as Page Tab platform.



In this case, our website needs to have a valid https certificate, which was singed by certifiers, because the Secure Page Tab URL requires this.

The short code pasted into the website is the one that needs to be places, the same way as in 1.3a.

1.4 SpringTab integration

In order to integrate SpringTab's data mining feature, the code from the SpringTab admin panel needs to be pasted into the source code of the website. On the admin panel, there is a Code tab, where the code is available.

```
<script type="text/javascript">
var _springtab=_springtab||[];

(function(){var a=document.createElement("script");a.t
ype="text/javascript";a.async=!0;a.src="//js.springtab
.com/t1.js";var b=document.getElementsByTagName("scrip
t")[0];b.parentNode.insertBefore(a,b)})();
</script>
```

<script type="text/javascript">
var _springtab=_springtab||[];

(function(){var
a=document.createElement("script");a.type="text/javascript";a.async=!0;a.src="//**js.springtab.com/t1.js**";var
b=document.getElementsByTagName("script")[0];b.parentNode.insertBefore(a,b)})();
</script>

The code above needs to be placed before the</head> part. The code is different in every campaign, and the campaign ID is coded into the URL. Thus, we need to check that we use the right campaign ID.

SpringTab data mining only works when FB client side integration is present with the SpringTab code.

2. Integration of personalization

The personalization function of SpringTab requires all steps to be taken of the data mining integration. In the following, we assume these steps are already taken.

The placed SpringTab code loads an external javascript into the website. The code contains the placement of the user into a cluster, further inquiry is not require. We deal with the script in detail in chapter 4.

2.1 Personalization by hiding content

When the user visits the site, the system automatically puts her into the cluster she belongs to, and notifies about the website about the outcome. The basic operation in this case is that among all .springtab class DOM elements, it removes those not containing the spt_<cluster_identification>.

For example:

```
<div class=”springtab spt_2”>…</div>
<div class=”springtab spt_1 spt_2”>…</div>
<div class=”springtab spt_1 spt_3”>…</div>
```

If the user is in cluster 2, then the third div will not show up on the site. If cluster 1, then the first div.

No further javascript code required for this integration.

2.2 Personalization by unique callback

SpringTab enables sophisticated integration with a uniquely defined callback. The function specified in that gets one parameter, which is the cluster number. Deining the callback works as follows.

```
<script>
//...
_springtab.push(['spt_callback', function(clusterid) {
        //your custom JavaScript code
}]);
//...
</script>
```

This code needs to be placed in the SpringTab integration code, where the _springtab object exists, but the asynchron script placement is not done yet (Example 3.2. gives a code).

Callback call happens after receiving the cluster number. If it is before the onready, then the callback is recalled upon onready event.

It is vital to check in callback that the first call happens or not.

By defining the callback, the function described in 2.1. is turned off. Naturally, within the callback, this function can be defined, for example with this function.

```
function clusterDomHide(clusterid)
{
   var trDoms = document.getElementsByClassName("springtab")
   for (var i = 0; i < trDoms.length; i++)
   {
         if (trDoms[i].className.indexOf("spt_" + clusterid) == -1)
      {
         trDoms[i].parentNode.removeChild(trDoms[i]);
          }
   }
}
```

3. Integration how-to

We present some simple integration, showing the nuances of usage. Sources are simplified in every case and they are not 3C validated.

3.1 Data mining integration

Content of test site pre-integration:

```
<html>
<head>
</head>
```

```
<body>
Lorem ipsum
</body>
</html>
```

Simple client side Facebook script integration with Facebook login:

```
<html>
<head>
</head>
<body>
<script>
  window.fbAsyncInit = function() {
    FB.init({
      appId      : '390420111133206',
      xfbml      : true,
      version    : 'v2.2'
    });
  };

  (function(d, s, id){
     var js, fjs = d.getElementsByTagName(s)[0];
     if (d.getElementById(id)) {return;}
     js = d.createElement(s); js.id = id;
     js.src = "//connect.facebook.net/en_US/sdk.js";
     fjs.parentNode.insertBefore(js, fjs);
   }(document, 'script', 'facebook-jssdk'));

function fblogin(){
FB.login(function(response) {},{
          scope:
'email,user_friends,publish_actions,user_likes,user_relationships,user_location'
});
}
</script>
Lorem ipsum
<a onclick="fblogin()">Facebook login</a>
</body>
</html>
```

This code can be complemented with SpringTab integration, using a simple copy-paste.

```
<html>
<head>
<script type="text/javascript">
var _springtab=_springtab||[];

(function(){var
a=document.createElement("script");a.type="text/javascript";a.async=!0;a.src="//js.s
pringtab.com/t1.js";var
b=document.getElementsByTagName("script")[0];b.parentNode.insertBefore(a,b)})();
</script>
```

```
</head>
<body>
<script>
  window.fbAsyncInit = function() {
    FB.init({
      appId      : '390420111133206',
      xfbml      : true,
      version    : 'v2.2'
    });
  };

  (function(d, s, id){
     var js, fjs = d.getElementsByTagName(s)[0];
     if (d.getElementById(id)) {return;}
     js = d.createElement(s); js.id = id;
     js.src = "//connect.facebook.net/en_US/sdk.js";
     fjs.parentNode.insertBefore(js, fjs);
   }(document, 'script', 'facebook-jssdk'));

function fblogin(){
FB.login(function(response) {},{
          scope:
'email,user_friends,publish_actions,user_likes,user_relationships,user_location'
});
}
</script>
Lorem ipsum
<a onclick="fblogin()">Facebook login</a>
</body>
</html>
```

3.2 Defining test callback

For the sake of simplicity, we exclude parts belonging to Facebook from the integration described in 3.1 and only keep SpringTab integration. This way, our code looks like this:

```
<html>
<head>
<script type="text/javascript">
var _springtab=_springtab||[];

(function(){var
a=document.createElement("script");a.type="text/javascript";a.async=!0;a.src="//js.s
pringtab.com/t1.js";var
b=document.getElementsByTagName("script")[0];b.parentNode.insertBefore(a,b)})();
</script>
<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
</head>
<body>
<div id="maincontent">Lorem ipsum</div>
</body>
</html>
```

Again, for the sake of simplicity, we pursue the dom manipulation with jquery. We place the cluster number in #maincontent div.

The complemented code:

```
<html>
<head>

<script type="text/javascript">
var _springtab=_springtab||[];

_springtab.push(['spt_callback', function(clusterid) {
        jQuery(function(){
                $("#maincontent").text("Your "+clusterid+". cluster you are in");
        });
}]);

(function(){var
a=document.createElement("script");a.type="text/javascript";a.async=!0;a.src="//js.s
pringtab.com/t1.js";var
b=document.getElementsByTagName("script")[0];b.parentNode.insertBefore(a,b)})();

</script>
<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
</head>
<body>
<div id="maincontent">Lorem ipsum</div>
</body>
</html>
```