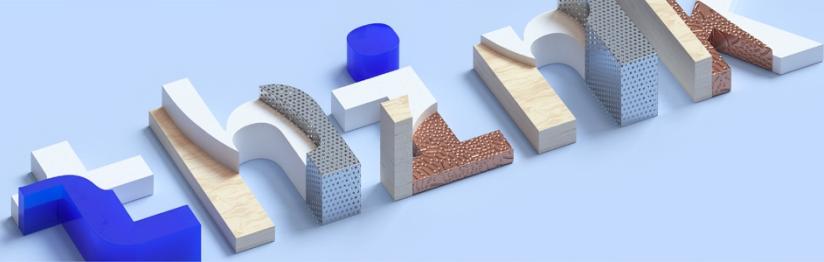


think 2018

IBM



Lab Center – Hands-on Lab

Session 2473

Session Title IBM Connections Customizer : from Zero to Hero

Stefano Pogliani, IBM, stefano.pogliani@fr.ibm.com

Martti Garden, IBM, martti.garden@de.ibm.com

Padraig Edwards, IBM, padraig.edwards@ie.ibm.com

Martin Donnelly, IBM, marti_donnelly@ie.ibm.com

Table of Contents

Disclaimer	4
Introduction	6
Icons	6
Code, Solutions, Examples	6
Documentation.....	7
Lab 1 The typical “Hello World” script	8
What we want to achieve	8
Preparation Activities	9
The Greeting.....	9
The Description	9
IBM Connections page loading mechanism	10
Preparing the Environment.....	12
Creating your script.....	13
Creating your extension.....	19
Making your extension available	27
Activating the extension.....	28
Lab 2 Playing with CSS	31
What we want to achieve	31
Two different approaches	32
The CSS File.....	32
Using Javascript injection	33
Preparation Activities	33
Creating your script.....	35
Creating your extension.....	40
Making your extension available	46
Activating the extension.....	47
Using CSS Declaration.....	50
Preparation Activities	50

Making your extension available	58
Activating the extension.....	59
Lab 3 Reducing the complexity of Connections UI	62
What we want to achieve	62
How we did ?.....	62
The Extension ?.....	65
Lab 4 Using JQuery	67
What we want to achieve	67
How is it done ?.....	69
We Value Your Feedback!	75

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.

In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may

need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

© 2018 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Introduction

This workshop was prepared as step by step instruction. It should lead you through IBM Connections Customizer components and its capabilities. All included exercises are going to support you in creating your first IBM Connections Customizer scripts.

Information



Information about user and password, method of access etc are going to be provided by your teacher.

All exercises are fulfilled with screen shots for easier navigation. However, there may be some small differences based on your environment, browser version, language or user account used.

Icons

The following symbols appear in this document at places where additional guidance is available.

Icon	Purpose	Explanation
	Important!	This symbol calls attention to a particular step or command. For example, it might alert you to type a command carefully because it is case sensitive.
	Information	This symbol indicates information that might not be necessary to complete a step but is helpful or good to know.
	Trouble-shooting	This symbol indicates that you can fix a specific problem by completing the associated troubleshooting information.

Code, Solutions, Examples

A special Github repository has been created where solutions, examples and the code or the graphics to be used are made available to all the students. We will reference this Github repository throughout this document as “**Reference Repository**”.

This Github repository is available at this address:

<https://github.com/stefanopog/CustomizerThinkLab>

The screenshot shows a GitHub repository page for 'CustomizerThinkLab' owned by 'stefanopog'. The repository has 6 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was made a minute ago. The repository contains files like README.md, Docs, Lab1, and Lab2. A section titled 'CustomizerThinkLab' is present.

IBM Connections Customizer Labs for Think2018

Add topics

6 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Commit Message	Time
Docs	Create README.md	a minute ago
Lab1	ok	3 days ago
Lab2	ok	4 days ago
README.md	Initial commit	4 days ago

CustomizerThinkLab

You may want to open a tab of your browser on this address in order to use the material. Feel free to clone this repository for better analyzing the code and the examples.

Documentation

The text of these exercises, together with other useful documentation, is stored in the Docs folder of the Github repository mentioned above.

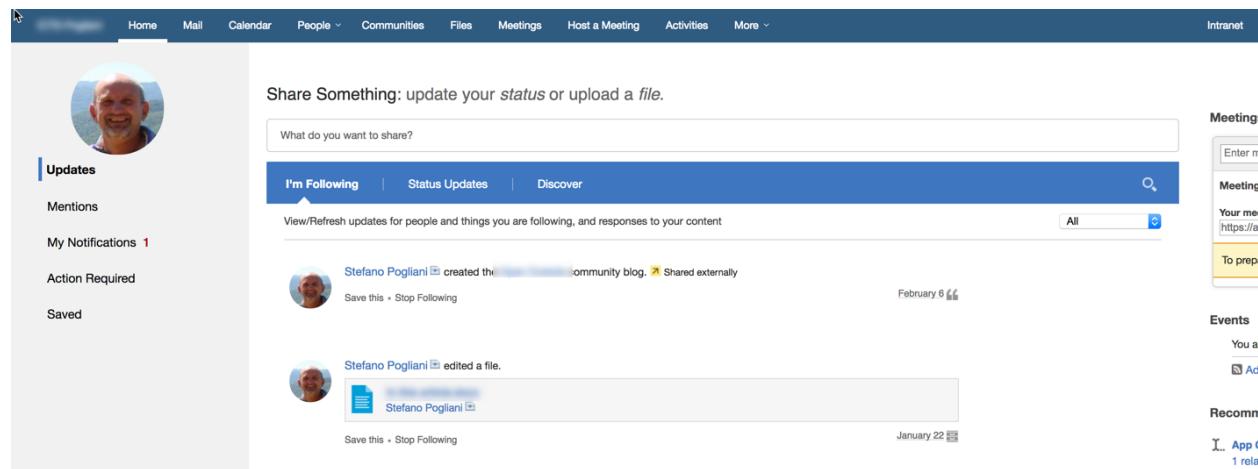
Lab 1 The typical “Hello World” script

This is a very simple exercise, very similar in the approach to the typical “Hello World” exercises you have certainly encountered in other labs.

The main goal we want to reach through this exercise is to guide you through the process of creating an IBM Connections Customizer extension; the extension itself is very simple because the focus will be on the process.

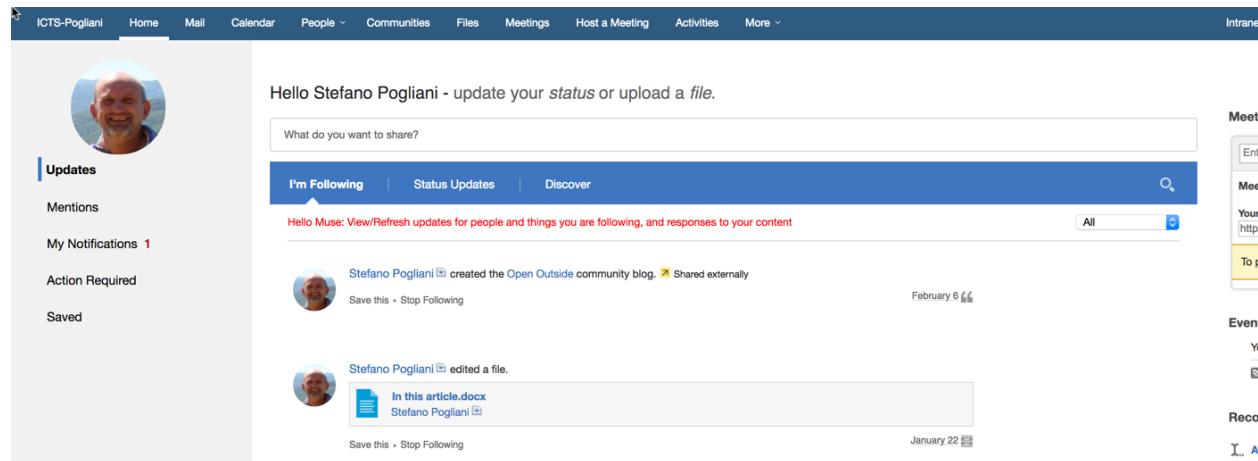
What we want to achieve

When accessing the traditional IBM Connections Cloud Homepage, you are likely to see something like this:



The screenshot shows the IBM Connections homepage with a dark blue header bar containing links for Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, and More. On the right side of the header is the word "Intranet". Below the header is a large search bar with the placeholder "Share Something: update your status or upload a file.". To the left of the search bar is a user profile picture of a man with a beard. Below the profile picture is a sidebar with sections for Updates, Mentions, My Notifications (1), Action Required, and Saved. The main content area displays activity items from Stefano Pogliani. The first item is a status update: "Stefano Pogliani created the [redacted] community blog. Shared externally" (Save this • Stop Following, February 6). The second item is a file edit: "Stefano Pogliani edited a file." (Save this • Stop Following, January 22). To the right of the main content area is a sidebar titled "Meeting:" with "Enter m", "Meeting", "Your me [https://a]", and "To prep". Below that is "Events" with "You ar", "Ad", and "Events". At the bottom is "Recomm" with "App C" and "1 rela".

In our exercise, we will modify the look-and-feel of this page to be the following one:



The screenshot shows the same homepage structure as the previous one, but with a few key differences. The search bar now says "Hello Stefano Pogliani - update your status or upload a file.". The activity items remain the same: the status update from Stefano Pogliani and the file edit. The sidebar on the right has been updated to reflect the changes: it now says "Meet:" instead of "Meeting:", and it includes "Ent", "Mee", "Your [http://]", and "To p". The "Events" section also reflects the changes, and the "Recomm" section is no longer visible.

So, we simply changed the “Share Something” string with the greeting “Hello Stefano Pogliani” and we changed the color of the font for the string below the blue header of the Activity Stream (and added a little text to it).

Preparation Activities

In order to do those modifications, we first need to understand which DOM elements we need to modify.

Our best friends here are the “Developer Tools” you find in your browser.

The Greeting

Which is the DOM element in the page holding the string “Share Something” ?

The screenshot shows a web application interface with a navigation bar at the top. Below the navigation bar is a profile picture and a "Updates" sidebar on the left. The main content area features a blue header bar with the text "Share Something: update your *status* or upload a *file*". A red arrow points from the text "Share Something" in the header down to the corresponding span element in the DOM tree below. The DOM tree is displayed in the developer tools' Inspector tab, showing the HTML structure of the page. The span element with the class "shareSome-title" is highlighted in blue, matching the color of the text in the header. The developer tools also show other tabs like Inspector, Console, Debugger, Style Editor, Canvas, Performance, Memory, Network, Storage, and Scratchpad.

```
<div id="lotusContent" class="lotusContent lotusBoard" role="main">
  <span style="position: absolute;"></span>
  <div id="hppPlaceholder_updates_content_top"></div>
  <div id="messageContainer" widgetid="messageContainer"></div>
  <div id="hppPlaceholder_updates_content_aboveStreamHeader"></div>
  <div id="activityStreamHeader" class="lotusHeader" widgetid="activityStreamHeader">
    <div class="lotusDetails lotusHidden" dojoattachpoint="asDesc"></div>
    <div class="shareSome-desc">
      <span class="shareSome-title">
        Share Something
        <span class="shareSome-titleSep">:</span>
        update your
        <i>status</i>
        or upload a
        <i>file</i>
      </span>
    </div>
  </div>
</div>
```

Using the Browser Developer Tools, we quickly find that the “Share Something” string is the content of a `` element whose class is “**shareSome-title**”.

The Description

The description string below the Blue Bar on the top of the Activity Stream is easily found also:

The screenshot shows the IBM Connections 'Updates' page. On the left, there's a sidebar with 'Updates' selected. The main area displays a timeline of activity. A red arrow points from the page content to the browser's developer tools, specifically the DOM inspector. In the DOM inspector, a span element with id='asDesc' is highlighted. This span contains the text 'View/Refresh updates for people and things you are following, and responses to your content'.

Using the Browser Developer Tools, we quickly find that the string is the content of a `` element whose id is “`asDesc`”.

IBM Connections page loading mechanism

IBM Connections Customizer injects into the page one or more scripts according to the definition of the extension(s). The IBM Connections Customizer scripts that are loaded, are declared as `<script>` elements at the bottom of the HTML code of the page that is rendered.

```

B06
B07
B08
B09 <script type="text/javascript">window.NREUM||(NREUM={});NREUM.info={"errorBeacon":"bam.nr-data.net","licenseKey":"0e69123577","agent":"","beacon":"bam.nr-data.net","appl
B10
B11
B12
B13 </div></div><script type='text/javascript' src='/files/customizer/'></script><script type='text/javascript' src='/files/customizer/'>
B14
B15

```

If the script will execute immediately, it is likely that it would execute before the page is fully built in your browser.

This means that we cannot assume that the HTML elements that we need to modify are already rendered when the script executes (during the load of the page). Thus, we need to implement a strategy

that would allow the real changes made by the script to happen once all the required elements in the page are there.

We think that it is fair enough to assume that we could start modifying the two **** elements introduced in the previous sections when the ActivityStream starts loading: at that point, the two **** elements are likely to be there:

The screenshot shows a user profile on the left and an activity feed on the right. The feed lists three items: 1) Stefano Poglian created the 'Open Outside' community blog, 2) Stefano Poglian edited a file named 'In this article.docx', and 3) Stefano Poglian edited a file. A red arrow points from the bottom of the screenshot to the 'dojoattachpoint' attribute in the DOM inspector, which is highlighted in blue. The DOM inspector shows the following code snippet:

```
View/Refresh updates for people and things you are following, and responses to your content
</span>
</div>
</div id="connectViews" class="lotusHidden" dojoattachpoint="feedTabsNode"></div>
<div class="lotusStreamTopLoading" dojoattachpoint="loadingNode">
  <div id="com_ibm_social_as_loading_Loader_0" class="loaderMain lotusHidden" dojoattachpoint="asLoadingNode" widgetid="com_ibm_social_as_loading_Loader_0">
    
      Loading...
  </div>
</div>
```

So, our strategy is going to be the following:

- Wait until the page will start loading the items in the ActivityStream
This happens once the **<div>** element whose class is “**loaderMain lotusHidden**” is rendered on the page.
In order to be sure we target that very element, we can say that the **<div>** element must be a child of the other **<div>** element whose class is “**lotusStreamTopLoading**”.
The “**dojo**” query for this is

```
dojo.query(".lotusStreamTopLoading div.loaderMain.lotusHidden")
```

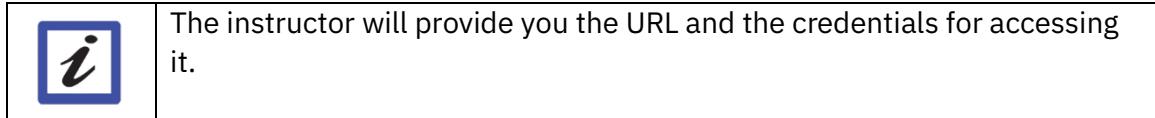


We could have used the “**id**” property of the child **<div>** element in order to be reasonably sure that we would have targeted the one we wanted. Unfortunately, you will see that the “**id**” ends with a “**_0**”. This implies that the value of the “**id**” property is calculated at runtime by IBM Connections and we cannot be sure there will not be one ending with “**_1**” or other strings.

- Once that element is rendered, we can modify the “textContent” property of the two `` elements and the font color of the second one.

Preparing the Environment

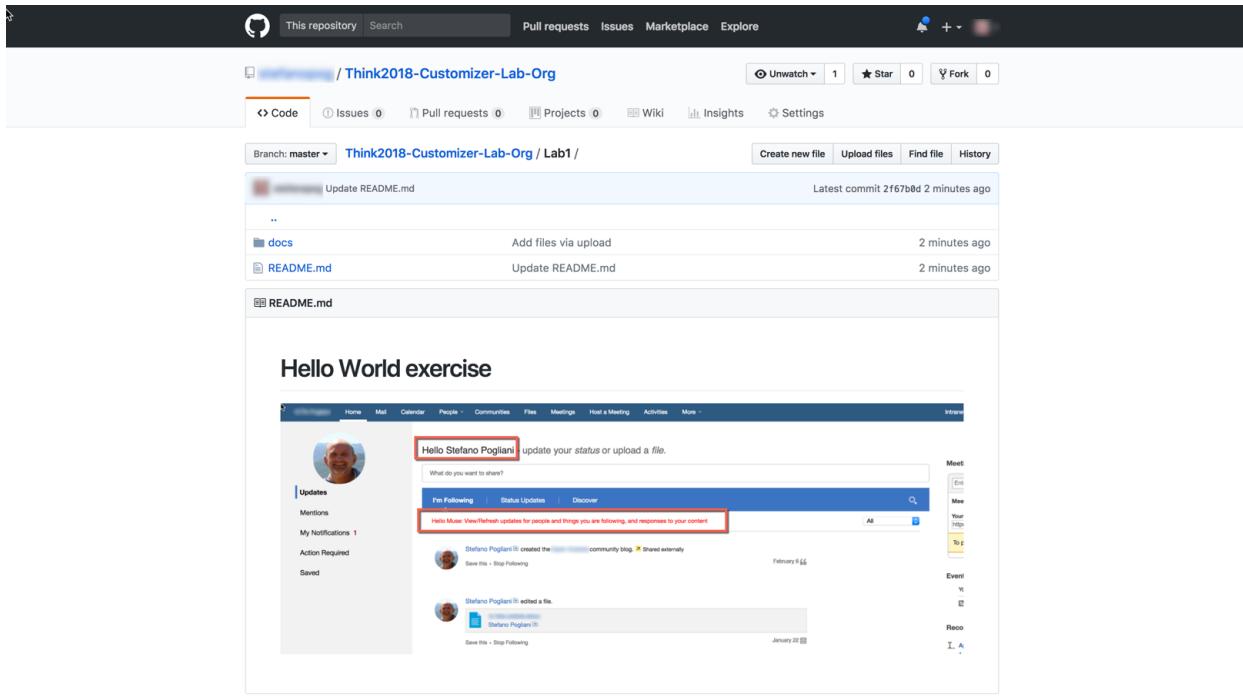
You have been invited as a Contributor to the Github repository associated to this organization.



You will arrive to a page similar to this one.

This screenshot shows a GitHub repository page for 'Exercise'. The repository has 11 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was made just now. The repository contains files named Lab1, Lab2, Lab3, Lab4, and README.md. The README.md file contains the text '123456789' and '123456789 - Think2018 Customizer Lab Organization'. A note at the bottom states: 'All the scripts that will be activated for the Think2018 IBM Connections Customizer Organization'.

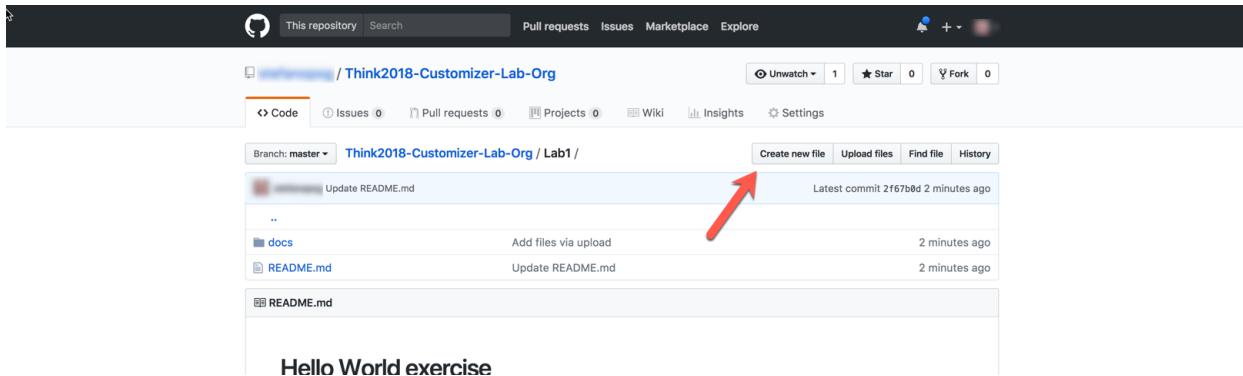
Click on the “**Lab1**” item and you will access this page:



Now you are ready to code!

Creating your script

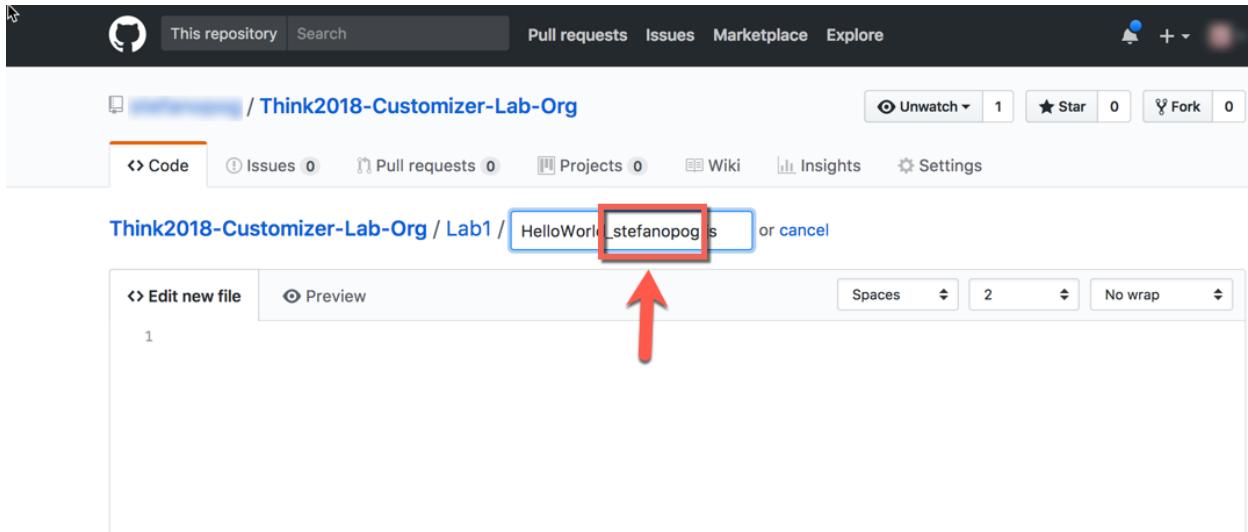
Create the Javascript file that will contain the code for your extension. Click on “Create File” as shown in the following image:



Now, you have to enter the name of the file. Since there are many people doing the exercise at the same time, we need to avoid conflict in filenames. For this reason, enter the following for the filename:

HelloWorld_<identifier>.js

<identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab).



Let's now proceed step-by-step in building our code together.

- Changing the Greeting.

As we [previously said](#), we need to modify the content of a `` element whose class is `"shareSome-title"`

The code that we need to use is the following:

```
//  
// Change the Greetings String  
// Note that the username of the currently Logged in user can be obtained by means of the  
// global variable "lconn.homepage.userName". This variable is defined by the  
// IBM Connections Home page  
//  
var thisUser = lconn.homepage.userName;  
dojo.query("span.shareSome-title")[0].textContent = "Hello " + thisUser + " -";
```

We use the “`dojo.query`” function to retrieve the `` element. Since the “`query`” returns an array, we need to select the very first element of this array.



In production code, we would certainly need to validate that the “`query`” actually returned the result that we were looking for.

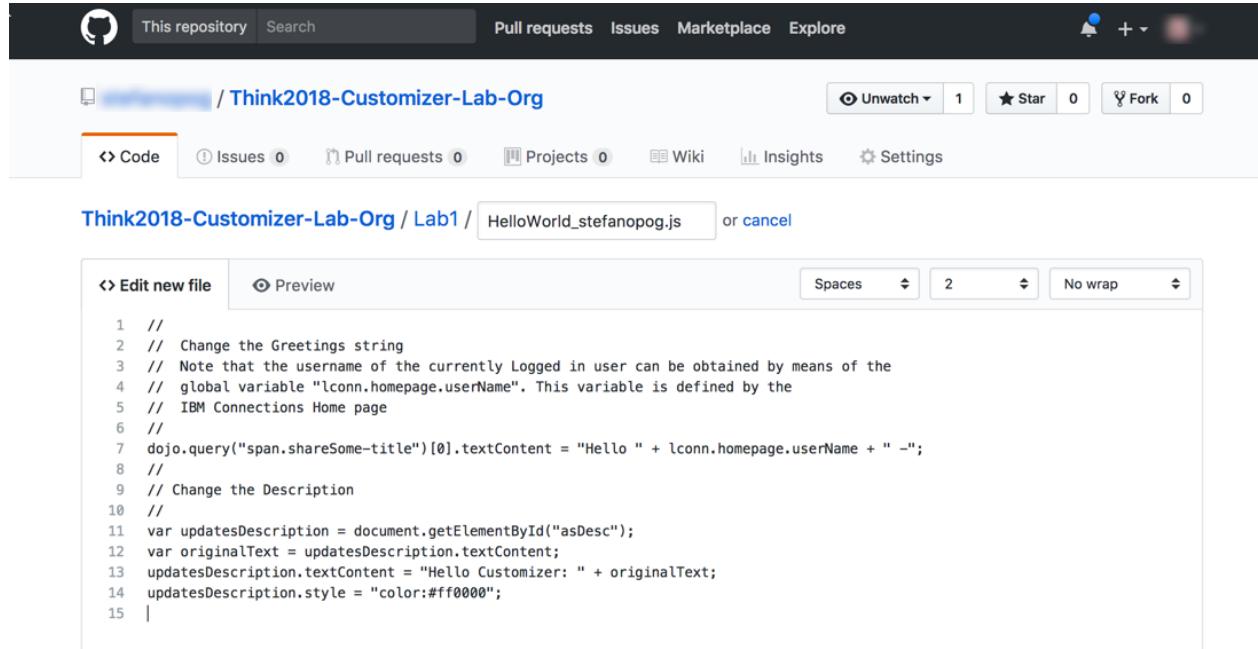
- Changing the Description.

As we [previously said](#), we need to modify the content of a `` element whose id is `“asDesc”`. The code that we need to use is the following:

```
//  
// Change the Description  
//  
var updatesDescription = document.getElementById("asDesc");  
var originalText = updatesDescription.textContent;  
updatesDescription.textContent = "Hello Customizer: " + originalText;  
updatesDescription.style = "color:#ff0000";
```

In this case we just used the “document.getElementById” function to retrieve the whose id is “asDesc”. We could have used the “dojo.byId” function.

- At this point our script looks like this:



The screenshot shows a GitHub repository interface. The repository name is "Think2018-Customizer-Lab-Org". The file being edited is "HelloWorld_stefanopog.js". The code in the editor is as follows:

```
1 //  
2 // Change the Greetings string  
3 // Note that the username of the currently Logged in user can be obtained by means of the  
4 // global variable "lconn.homepage.userName". This variable is defined by the  
5 // IBM Connections Home page  
6 //  
7 dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + "-";  
8 //  
9 // Change the Description  
10 //  
11 var updatesDescription = document.getElementById("asDesc");  
12 var originalText = updatesDescription.textContent;  
13 updatesDescription.textContent = "Hello Customizer: " + originalText;  
14 updatesDescription.style = "color:#ff0000";  
15 |
```

We have now to make sure that the code we just entered would be executed when the elements we are modifying are properly rendered (as [mentioned previously](#)).

To implement this behavior, let's suppose that we have a function, called “waitFor”, which waits for the condition to be met and, then, will execute our code.

The way we would describe this in Javascript is by using the callback mechanism: the code we just entered becomes the body of a callback function executed by our “waitFor” once the condition is met.

At high level :

```
//  
// wait for the required elements are rendered  
//  
waitFor(function() {  
    .... Our code here ....  
}, ".lotusStreamTopLoading div.loadMain.lotusHidden");
```

So, change the code in your editor to look like what is shown in the following picture:

```

1 waitFor(function(){
2     //
3     // Change the Greetings string
4     // Note that the username of the currently Logged in user can be obtained by means of the
5     // global variable "lconn.homepage.userName". This variable is defined by the
6     // IBM Connections Home page
7     //
8     dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + " -";
9     //
10    // Change the Description
11    //
12    var updatesDescription = document.getElementById("asDesc");
13    var originalText = updatesDescription.textContent;
14    updatesDescription.textContent = "Hello Customizer: " + originalText;
15    updatesDescription.style = "color:#ff0000";
16    updatesDescription.innerHTML = originalText;
17 }, ".lotusStreamTopLoading div.loaderMain.lotusHidden");

```

- Of course, we now need to provide the implementation of our “waitFor” function, right ?
You can copy the code here and paste before the first line of the script:

```

var waitFor = function(callback, elXpath, elXPathRoot, maxInter, waitTime) {
    If (!elXPathRoot) var elXPathRoot = dojo.body();
    If (!maxInter) var maxInter = 10000; // number of intervals before expiring
    If (!waitTime) var waitTime = 1; // 1000=1 second
    If (!elxpath) return;

    var waitInter = 0; // current interval
    var intId = setInterval( function(){
        if ( ++waitInter < maxInter && !dojo.query(elXpath,elXPathRoot).length) return;
        clearInterval(intId);
        if ( waitInter >= maxInter ) {
            console.log("**** WAITFOR [" + elXpath + "] WATCH EXPIRED!!! interval
" + waitInter + " (max:" + maxInter + ")");
        } else {
            console.log("**** WAITFOR [" + elXpath + "] WATCH TRIPPED AT interval
" + waitInter + " (max:" + maxInter + ")");
            callback();
        }
    }, waitTime);
}

```

This code simply implements a polling mechanism using the Javascript “setInterval” function. We do not use the last 3 parameters of the function and the code defaults them to some value. What is to be noticed is the way in which “waitFor” uses the **callback** function (which is actually the code we wrote in the previous part of the exercise).

At the end, the code in your editor should look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab1'. The code editor displays a file named 'HelloWorld_stefanopog.js'. The code is a JavaScript function named 'waitFor' that uses the Dojo toolkit's 'dojo.query' and 'setInterval' methods to poll for the presence of an element. It includes comments explaining its purpose and handling of expiration intervals.

```

1 var waitFor = function(callback, elXPath, elXPathRoot, maxInter, waitTime) {
2     if (!elXPathRoot) var elXPathRoot = dojo.body();
3     if (!maxInter) var maxInter = 10000; // number of intervals before expiring
4     if (!waitTime) var waitTime = 1; // 1000=1 second
5     if (!elXPath) return;
6     var waitInter = 0; // current interval
7     var intId = setInterval(function(){
8         if (++waitInter < maxInter && !dojo.query(elXPath,elXPathRoot).length) return;
9
10        clearInterval(intId);
11        if (waitInter >= maxInter) {
12            console.log("**** WAITFOR [" + elXPath + "] WATCH EXPIRED!!! interval " + waitInter + " (max:" + maxInter);
13        } else {
14            console.log("**** WAITFOR [" + elXPath + "] WATCH TRIPPED AT interval " + waitInter + " (max:" + maxInter);
15            callback();
16        }
17    }, waitTime);
18 };
19 waitFor(function(){
20     //
21     // Change the Greetings string
22     // Note that the username of the currently Logged in user can be obtained by means of the
23     // global variable "lconn.homepage.userName". This variable is defined by the
24     // IBM Connections Home page
25     //
26     dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + "-";
27     //
28     // Change the Description
29     //
30     var updatesDescription = document.getElementById("asDesc");
31     var originalText = updatesDescription.textContent;
32     updatesDescription.textContent = "Hello Customizer: " + originalText;
33     updatesDescription.style = "color:#ff0000";
34 }, ".lotusStreamTopLoading div.loaderMain.lotusHidden");
35

```

- We are still missing one point though.

In the code we wrote we assumed that the **dojo toolkit** would be available in the page in order for your script to use it. We are confident that the IBM Connections page will load the **dojo toolkit** at a certain point, but we cannot assume that it will be loaded by the time we first use it (look at the line 8 in the previous picture: our “waitFor” function uses the **dojo toolkit** to poll the presence of the element...).

We certainly do not want our script to load the **dojo toolkit** another time.... So we need to wait for it to be ready on the page.

Fortunately the **dojo toolkit** provides a standard mechanism to deal with this point: the “dojo/domReady!” plugin (described here <https://dojotoolkit.org/reference-guide/1.10/dojo/domReady.html>).

So, we will simply need to wrap the code we wrote up until now in this way:

```

if (typeof(dojo) != "undefined") {
    require(["dojo/domReady!"], function () {
        try {
            ... all our previous code ....
        } catch(e) {
            alert('exception occurred in HelloWorld : ' + e);
        }
    });
}

```

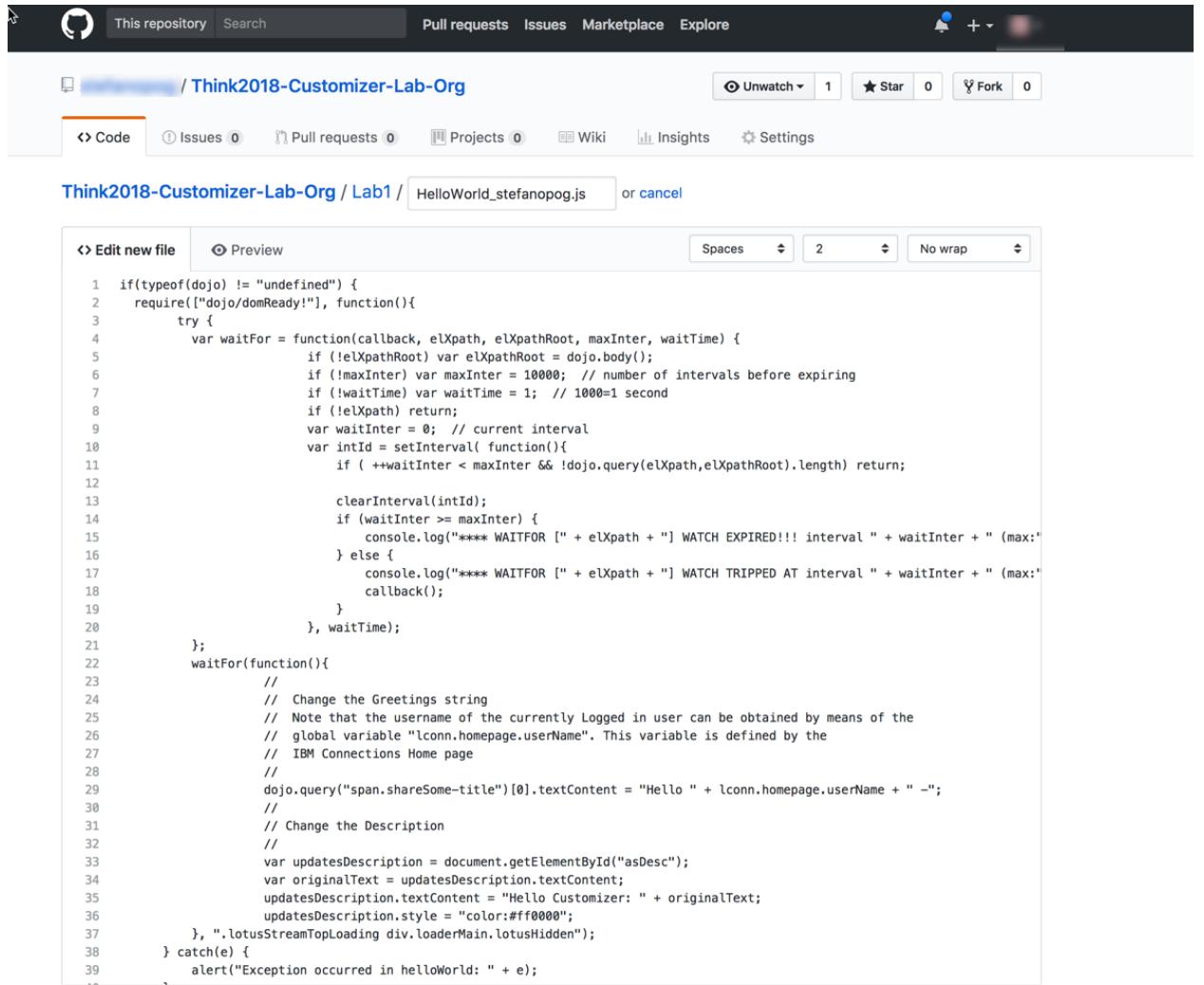
```

        }
    });
}

```

We have introduced a “try... catch” statement to globally catch any exception we may encounter.

At the end, the code in your editor should look like this:

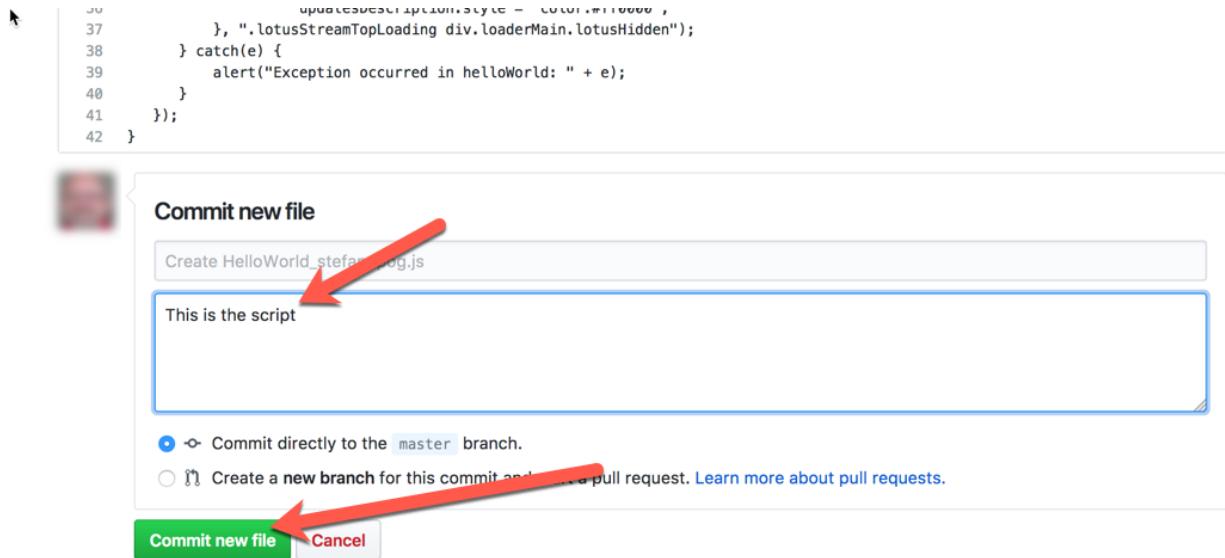


```

1 if(typeof(dojo) != "undefined") {
2     require(["dojo/domReady!"], function(){
3         try {
4             var waitFor = function(callback, elXPath, elXPathRoot, maxInter, waitTime) {
5                 if (!elXPathRoot) var elXPathRoot = dojo.body();
6                 if (!maxInter) var maxInter = 10000; // number of intervals before expiring
7                 if (!waitTime) var waitTime = 1; // 1000=1 second
8                 if (!elXPath) return;
9                 var waitInter = 0; // current interval
10                var intId = setInterval(function(){
11                    if ( ++waitInter < maxInter && !dojo.query(elXPath,elXPathRoot).length) return;
12
13                    clearInterval(intId);
14                    if (waitInter >= maxInter) {
15                        console.log("**** WAITFOR [" + elXPath + "] WATCH EXPIRED!!! interval " + waitInter + " (max:"+
16                    } else {
17                        console.log("**** WAITFOR [" + elXPath + "] WATCH TRIPPED AT interval " + waitInter + " (max:"+
18                            callback();
19                    }
20                }, waitTime);
21            };
22            waitFor(function(){
23                //
24                // Change the Greetings string
25                // Note that the username of the currently Logged in user can be obtained by means of the
26                // global variable "lconn.homepage.userName". This variable is defined by the
27                // IBM Connections Home page
28                //
29                dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + " -";
30                //
31                // Change the Description
32                //
33                var updatesDescription = document.getElementById("asDesc");
34                var originalText = updatesDescription.textContent;
35                updatesDescription.textContent = "Hello Customizer: " + originalText;
36                updatesDescription.style = "color:#ff0000";
37            }, ".lotusStreamTopLoading div.loaderMain.lotusHidden");
38        } catch(e) {
39            alert("Exception occurred in helloWorld: " + e);
        }
    });
}

```

- You can now commit your code to the repository. It is always good practice to add, as a comment, what you did.



- Your script is now safely recorded in the Github repository.

A screenshot of a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Think2018-Customizer-Lab-Org / Lab1'. The 'Code' tab is selected. A red arrow points from the text input field in the commit dialog to the file listing in the repository. The file 'HelloWorld_stefanopog.js' is listed under the 'docs' directory, with a timestamp of 'just now'.

File	Action	Timestamp
docs/HelloWorld_stefanopog.js	Create HelloWorld_stefanopog.js	just now
docs/README.md	Update README.md	an hour ago
README.md		an hour ago

Below the repository listing, there is a section titled 'Hello World exercise'.

A screenshot of an IBM Connections profile page for 'Hello Stefano Pogliani'. The status update text is 'Hello Muse: View/Refresh updates for people and things you are following, and responses to your content'. A red box highlights the status update text, and another red box highlights the 'Hello Muse' link.

Creating your extension

Whilst the extension can be directly created in the IBM Connections Cloud Administration panel, we suggest that you save the extension definition as a JSON file in the same Github repository.



In this way, the extension may be used in other situations and may become part of the overall documentation of the project

In order to create the extension, we need to create a new file inside our Github repository. The file will follow this naming convention

HelloWorld_<identifier>.json

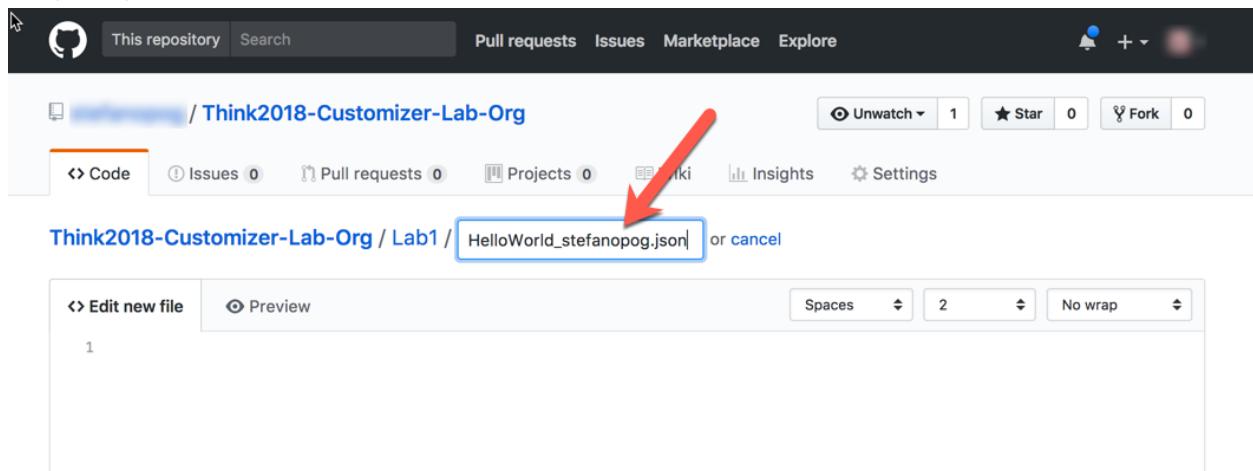
Where <identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab). In this way, everybody will immediately understand that this is the extension definition associated with the HelloWorld_<identifier> script.

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org / Lab1'. At the top right, there is a 'Create new file' button. A large red arrow points from the bottom left towards this button. Below the button, there is a list of files: 'Create HelloWord_stefanopog.js', 'docs', 'HelloWorld_stefanopog.js', and 'README.md'. The 'HelloWorld_stefanopog.js' file was created just now. The 'Create HelloWord_stefanopog.js' entry has a timestamp of 'just now'. The 'docs' folder was added via upload an hour ago. The 'README.md' file was updated an hour ago.

Hello World exercise

The screenshot shows the IBM Connections dashboard. On the left, there is a sidebar with 'Updates' selected. In the center, there is a status update from 'Hello Stefano Poglian'. The update says: 'Hello Stefano Poglian - update your status or upload a file.' Below this, there is a text input field with placeholder text 'What do you want to share?'. At the bottom of the status update, there is a link 'Hello Muse: View/Refresh updates for people and things you are following, and responses to your content'. The right side of the screen shows a sidebar with 'Meet', 'Ent', 'Mee', 'Your', 'http', and 'To p'.

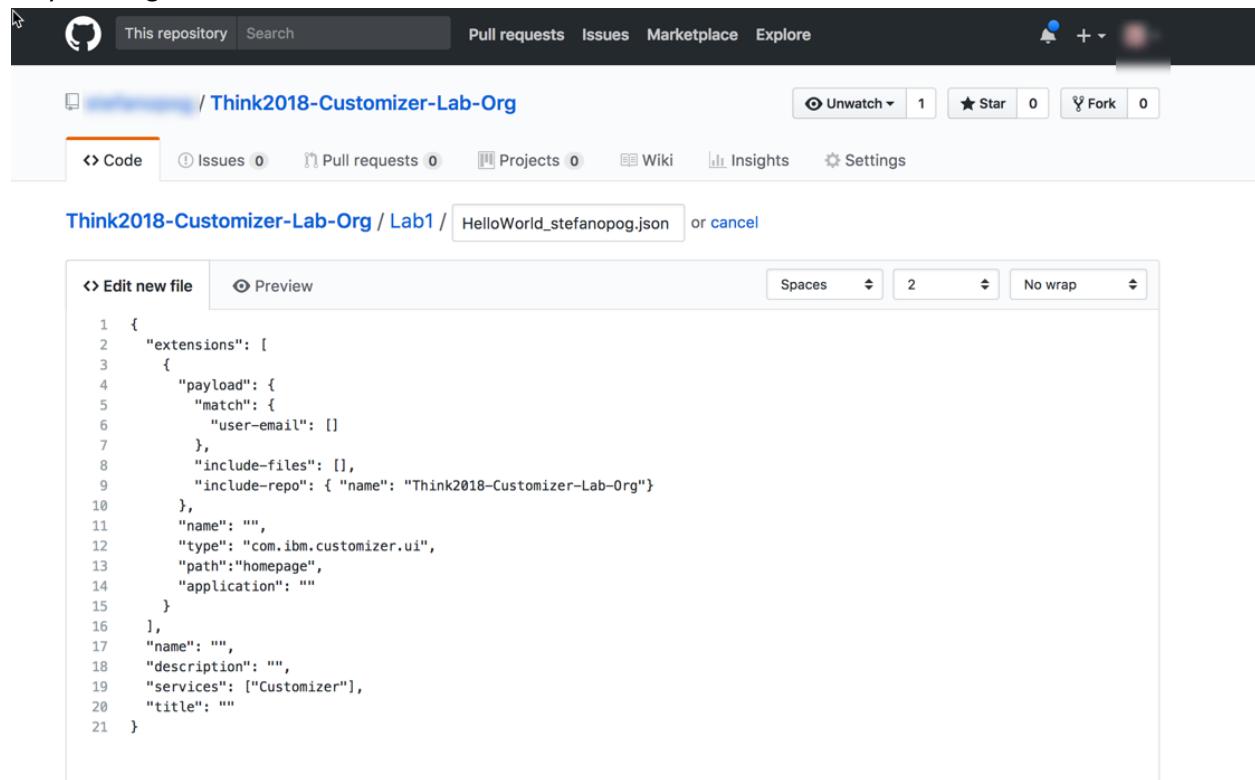
And, then,



Let's first copy this empty template inside our new json file (a copy of this template is available as the Lab1/emptyTemplate.json under "**Reference Repository**"):

```
{  
  "extensions": [ {  
      "payload": {  
          "match": {  
              "user-email": []  
          },  
          "include-files": [],  
          "include-repo": {  
              "name": "Think2018-Customizer-Lab-Org"  
          }  
      },  
      "name": "",  
      "type": "com.ibm.customizer.ui",  
      "path": "homepage",  
      "application": ""  
    } ],  
  "name": "",  
  "description": "",  
  "services": ["Customizer"],  
  "title": ""  
}
```

So you will get this result:



The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab1'. A file named 'HelloWorld_stefanopog.json' is being edited. The code editor displays the following JSON content:

```
1  {
2    "extensions": [
3      {
4        "payload": {
5          "match": {
6            "user-email": []
7          },
8          "include-files": [],
9          "include-repo": { "name": "Think2018-Customizer-Lab-Org" }
10         },
11        "name": "",
12        "type": "com.ibm.customizer.ui",
13        "path": "homepage",
14        "application": ""
15      }
16    ],
17    "name": "",
18    "description": "",
19    "services": ["Customizer"],
20    "title": ""
21 }
```

Some of the values of the JSON descriptor are already provided. A complete documentation for the attributes is available at this url :

<https://github.com/ibmcnxdev/customizer/blob/master/docs/IBMConnectionsCustomizer.pdf>

- “include-repo”: (line 8)
This MUST be the name of the Github repository that we are using. It is the “root” directory that the IBM Connections Customizer engine will use in order to retrieve the scripts that the extension uses
- “type”: (line 13)
This is a string whose value “com.ibm.customizer.ui” will tell IBM Connections Cloud which type of IBM Connections Customizer service will be used by the extension
- “path”: (line 14)
The value of “homepage” will instruct the IBM Connections Customizer engine to add the scripts referenced by the extension only to the “homepage” service of IBM Connections
- “services”: (line 19)
This is a string whose value “Customizer” will tell IBM Connections Cloud which type extension it needs to register in the Application Registry.

Let's now fill the other values that are required:

- “user-email”: (line 5)
We can restrict the injection of the scripts associated to the extension only to users whose email address will match the string.
Since we do not want to interfere with other people working at the same time on the same IBM

Connections Cloud organization, we need to provide a value for this attribute. The value needs to be the email address you have been provided by the instructor to log into IBM Connections Cloud (enclosed in double-quotes). So something like :

```
"user-email": ["stefanopog@think2018-Customizer.com"]
```



In real life, you may not need to use this attribute if the extension you are creating will be used by **all the people** in your organization. See the [documentation](#) for discovering all the possibilities associated to the "match" clause.

- "include-files": (line 7)

This is where you will actually reference the script that you created earlier in the exercise. The path to reference the script is relative to the Github repository you are using (see the "include-repo" clause described above).

So in your situation you need to provide the following value :

```
"include-files": ["Lab1>HelloWorld_<identifier>.js"]
```

- "name": (line 12)

this is the name of the Extension.

You can use the following:

```
"name": "HelloWorld Ext Lab1 <identifier>"
```

- "name": (line 17)

This is the name of the Application we are going to declare to IBM Connections. Typically, an "Application" can declare one or more extensions (in our example, we have one extension only associated with the application).

This name **must be unique** for the IBM Connections Organization, since it is the name that will be recognized by the IBM Connections Application registry. You can use the following name:

```
"name": "HelloWorld App Lab1 <identifier>"
```

- "application": (line 15)

This entry (which is within the "extension") actually needs to reference the correct "Application"). So, the value you will provide for this attribute **MUST BE the same** as the value you provided for the "name" attribute of the Application (line 17). You can use the following name:

```
"application": "HelloWorld App Lab1 <identifier>"
```

- "description": (line 18)

This is a free text description of what the Application (and its Extensions) provide. You can use the following:

```
"description": "Hello World Homepage Customization from <identifier>"
```

- “title”: (line 20)

This is the title that will appear in the IBM Connections Cloud application registry. You can use the following title:

```
"title": "<identifier> Hello World"
```

Once the Organization admin will use your JSON file to declare the extension, this is how it will appear in the IBM Connections Cloud Application Registry:

The screenshot shows the IBM Connections Cloud application registry interface. On the left, there's a sidebar with 'Administration' selected. Under 'Organization Extensions', 'Hello World' is listed. The main area shows a grid of applications. One application in the grid has a blue icon with a white 'C' and the title 'stefanopog Hello World'. Below the title, it says 'Service: Customizer' and 'Hello World Homepage Customization from stefanopo'. A red arrow points to the 'title' field, and another red arrow points to the 'description' field.

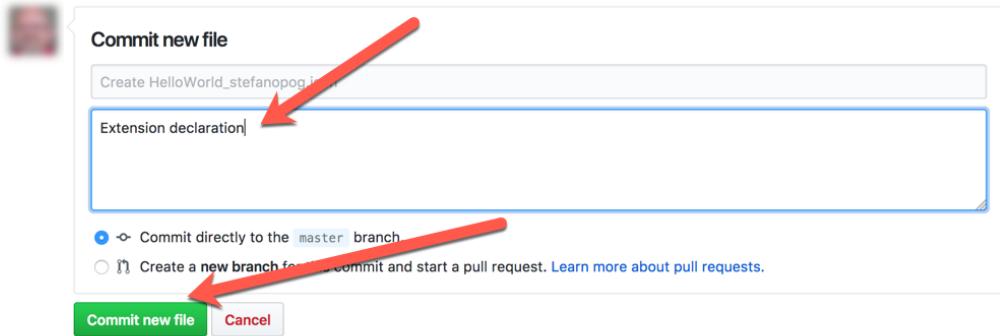
At this point, the JSON file you are editing in Github will look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 unwatched star and 0 forks. The 'Code' tab is selected, showing the file 'HelloWorld_stefanopog.json'. The file contains the following JSON code:

```
1 {
2   "extensions": [
3     {
4       "payload": {
5         "match": {
6           "user-email": ["stefanopog@think2018-Customizer.com"]
7         },
8         "include-files": ["Lab1/Helloworld_stefanopog.js"],
9         "include-repo": { "name": "Think2018-Customizer-Lab-Org" }
10      },
11      "name": "HelloWorld Ext Lab1 stefanopog",
12      "type": "com.ibm.customizer.ui",
13      "path": "homepage",
14      "application": "HelloWorld App Lab1 stefanopog"
15    }
16  ],
17  "name": "HelloWorld App Lab1 stefanopog",
18  "description": "Hello World Homepage Customization from stefanopog",
19  "services": ["Customizer"],
20  "title": "stefanopog Hello World"
21 }
```

You can Commit the change to the Github repository:

```
16 ],
17 "name": "HelloWorld App Lab1 stefanopog",
18 "description": "Hello World Homepage Customization from stefanopog",
19 "services": ["Customizer"],
20 "title": "stefanopog Hello World"
21 }
```



And your Repository will look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. A red arrow points to the file 'HelloWorld_stefanopog.json' in the file list.

File List:

- docs
- HelloWorld_stefanopog.js
- HelloWorld_stefanopog.json** (highlighted)
- README.md

Commit History:

- Create HelloWorld_stefanopog.json ... (just now)
- Add files via upload (3 hours ago)
- Create HelloWorld_stefanopog.js (2 hours ago)
- Create HelloWorld_stefanopog.json (just now)
- Update README.md (3 hours ago)

Latest commit: 3df2c12 just now

The screenshot shows an IBM Intranet profile page for 'Hello Muse'. It displays a status update from 'Hello Muse' with the message: 'Hello Muse: View/Refresh updates for people and things you are following, and responses to your content'.

Making your extension available

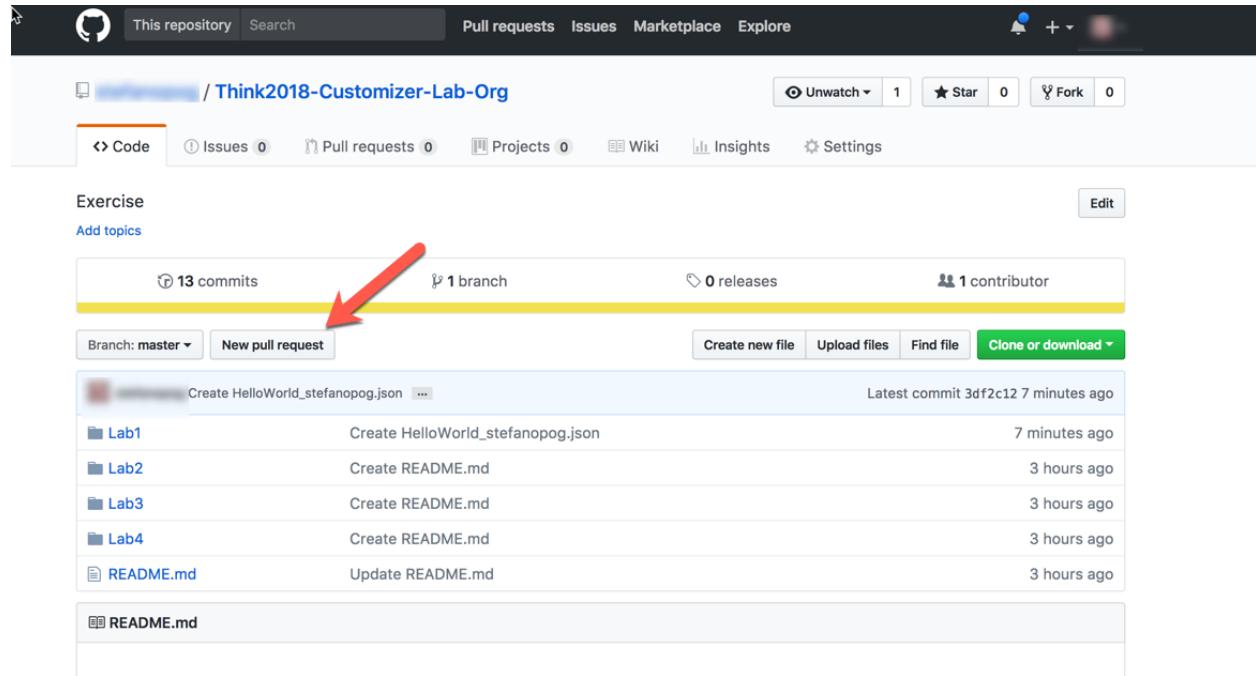
At this point, everything is ready to be put in production. This is what the [documentation](#) states:

- Share your repo with IBM – [add "ibmcndev" as a collaborator](#)
- IBM ([ibmcnxdev](#)) then creates a fork of your repository under [github.com/ibmcnxdev](#) and grants you read access by default.
- You can continue to work on your extension using your original repo for your source code activity, but once you are ready to deliver to IBM Cloud you must issue a [pull request](#) to IBM.
- IBM merges your pull request once acceptance criteria are met.
- Upon merge, the repo files are automatically pushed to IBM Customizer via a webhook.
- Rinse & repeat starting at Step (c) for extension updates.

Steps **a.** and **b.** have already been done for you before you started the exercise.

Now, you need to execute **Step c.**

Go back to the root directory of your Github repository and click on the “Pull Request” button:



Activating the extension

Now your extension has been validated. The validation process (from **IBMCNDEV**), if successful, will make the files you created in this exercise available to IBM Connections Cloud and to IBM Connections Customizer.

The only thing that is missing is to actually define the Extension in the IBM Connections Cloud Application Registry. This is the step that informs IBM Connections Customizer of what to do with the scripts that you just created.

The IBM Connections Cloud organization administrator needs to be given the JSON file which contains the extension (the one you created [here](#)).

He will go to the IBM Connections Application Registry and will declare a new extension:

The screenshot shows the IBM Connections Cloud Administration interface. On the left, there's a sidebar with sections like 'Personal', 'User Accounts', 'Organization Account Settings', 'Partitions', 'Subscriptions', 'Announcements', 'Internal Apps', 'Order History', and 'Organization Extensions'. The 'Organization Extensions' section is currently selected. The main area has a heading 'Info: To create classic Organization Extensions, click here' and tabs 'Apps' and 'All Apps'. A red arrow points to the 'New App' button. Below it, several app cards are visible, including 'Connections Mobile App Management', 'Chat and Meetings', and 'IBM SmartCloud Notes'.

A new screen is shown:

The screenshot shows the 'App Editor' screen. The sidebar remains the same as the previous screen. The main area has a heading 'App Editor' with a 'Back to Apps' link. It includes a warning message: 'Warning: Import will overwrite editor contents.' Below this is a code editor window containing a JSON file. The JSON content is as follows:

```
1 {  
2   "name": "",  
3   "title": "",  
4   "description": "",  
5   "services": [],  
6   "extensions": []  
7 }
```

A red arrow labeled '1' points to the JSON code area. Another red arrow labeled '2' points to the 'Save' button at the bottom of the code editor.

The IBM Connections Cloud organization admin will replace the content pointed by the right arrow (1) with the content of the JSON file you created and, then, **saves** (2) the extension.

You can now go to the homepage and see the final result:

The screenshot shows a Microsoft SharePoint intranet homepage. At the top, there is a navigation bar with links: Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, and Intranet. On the left, there is a sidebar with sections for Updates, Mentions, My Notifications (1), Action Required, and Saved. The main content area displays updates from followed users. A red box highlights the status update from "Hello Muse". Another red box highlights the notification for "Hello Muse". The updates listed are:

- Stefano Pogliani created the community blog. Shared externally. (February 6)
- Stefano Pogliani edited a file. (January 22)

This ends Lab 1.
Congratulations!

Lab 2 Playing with CSS

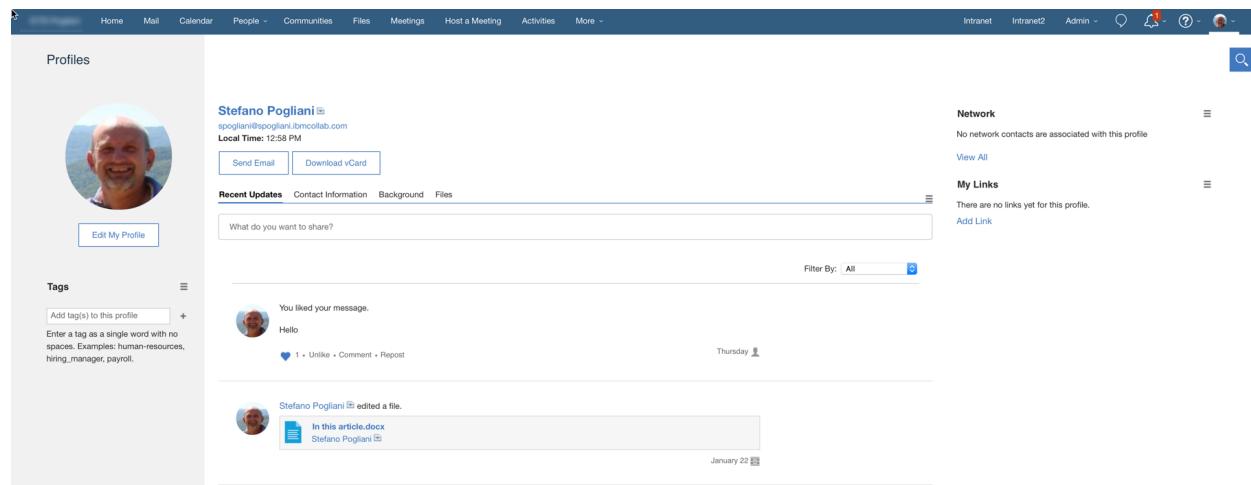
In the previous exercise we played with some DOM Elements using IBM Connections Customizer. We just saw how, programmatically, we could modify the style of an HTML element (when we colored the “Description” in red).

What if we would like to apply extensive changes to the CSS of one or more IBM Connections pages? Probably using Javascript code to wrap CSS directives is going to be tedious and error-prone. It would be much easier if we could compile all of our style changes into a CSS file and have IBM Connections Customizer apply that style with a single command.

This is the goal of this second exercise. Once again, the extension is really simple by itself, and the focus is really on showing how a CSS file can be used by an IBM Connections Customizer extension.

What we want to achieve

When accessing the traditional IBM Connections Cloud Profile page, you are likely to see something similar to this:

A screenshot of the IBM Connections Cloud Profile page for Stefano Poglian. The page has a dark header with navigation links like Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, Intranet, Intranet2, Admin, and a search bar. The main content area shows a profile picture of Stefano Poglian, his name, email, and local time. Below that are sections for Recent Updates, Network (with no contacts), and My Links (with no links). On the left, there's a Tags section where users can add tags to the profile. The right side shows a timeline of recent activity, including a message from Stefano and a file edit.

In our exercise, we will modify the look-and-feel of this page to be the following one:

The screenshot shows the IBM Connections Profile page for Stefano Poglianini. At the top, there's a navigation bar with links like Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, Intranet, Intranet2, Admin, and a search bar. Below the header is a decorative banner with a network graph and tropical leaves. The main content area starts with a profile section featuring a circular photo of Stefano, his name, and a 'Edit My Profile' button. It also shows the local time as 12:46 PM. Below this are tabs for Recent Updates, Contact Information, Background, and Files. The 'Recent Updates' tab is active, displaying a list of recent activities:

- You liked your message. (Hello)
- Stefano Poglianini edited a file. (In this article.docx)
- Stefano Poglianini edited a file.

On the right side, there are two more sections: 'Network' (which shows no network contacts) and 'My Links' (which shows no links yet). Buttons for 'Send Email' and 'Download vCard' are located at the top right of the main content area.

We are not going to modify any functionality from the standard Profile page, but simply changing the way in which the standard information is shown.

Two different approaches

We will present two ways to this usecase:

1. The first one makes use some Javascript code which will inject the CSS file containing the stylesheet we want to apply
2. The second uses a new functionality, recently introduced in IBM Connections Customizer, which allows the developer to directly reference CSS files in the Extension definition.

We thought that, whilst the second one is, by far, the quickest and the most efficient from a performance perspective, the first would be a useful exercise that would allow students to further understand the way in which IBM Connections Customizer works.

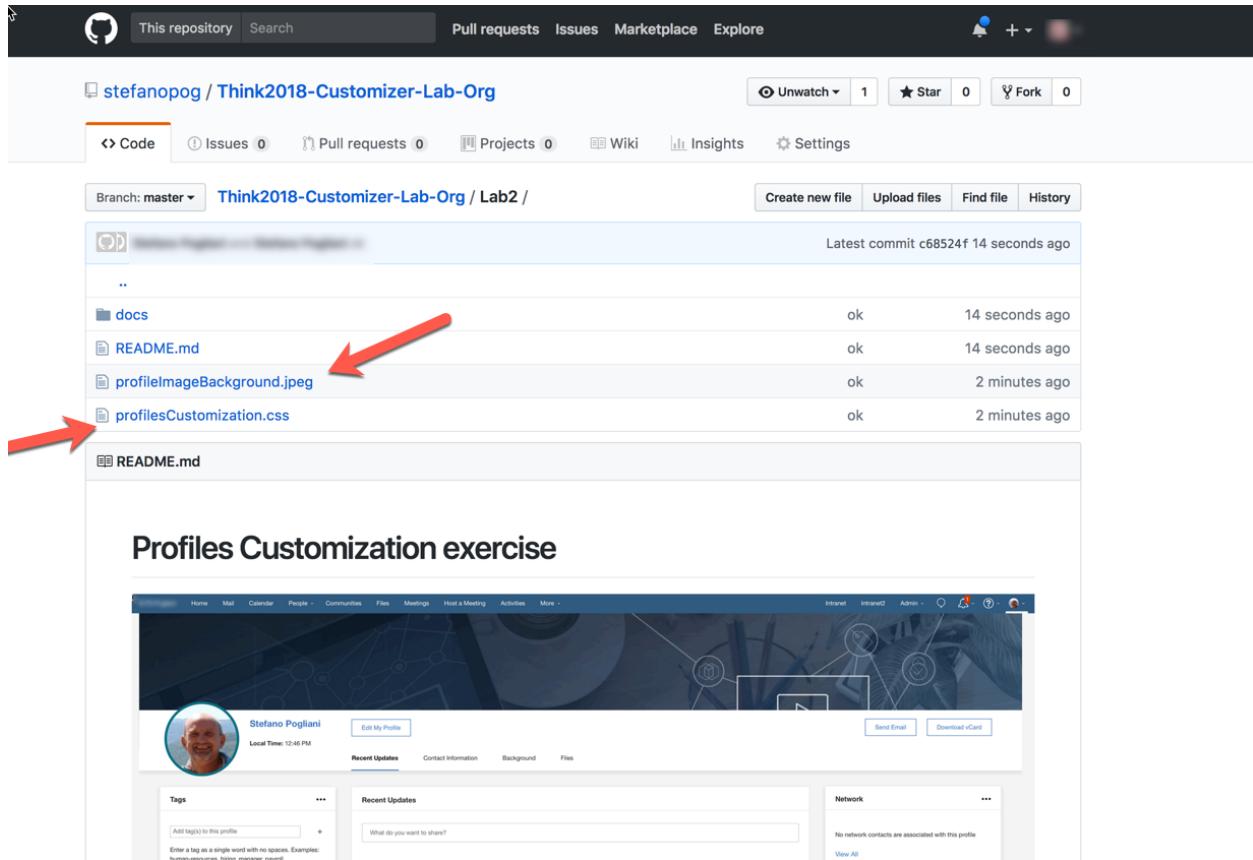
The CSS File

Both approaches would require the definition of the CSS file.

	<p>The objective here is not to explain the Stylesheet itself. You may experiment with the CSS file which is provided as an example in order to better understand the way the CSS directives have been applied in order to produce the desired result.</p>
--	--

The CSS file that we will be using, and the associated image used as a background are already available under the Lab2 directory of the Github repository associated with the IBM Connections Cloud organization we are using.

Please do not touch nor modify those two files as they will be used by the code we are going to write.



The screenshot shows a GitHub repository page for 'stefanopog / Think2018-Customizer-Lab-Org'. The 'Code' tab is selected. Under the 'Branch: master' dropdown, 'Think2018-Customizer-Lab-Org / Lab2 /' is selected. The repository has 1 unwatched star and 0 forks. The file list shows:

- docs (ok, 14 seconds ago)
- README.md (ok, 14 seconds ago)
- profileImageBackground.jpeg (ok, 2 minutes ago) - highlighted with a red arrow
- profilesCustomization.css (ok, 2 minutes ago)

Below the file list, there is a preview of a profile customization exercise interface.



For your own convenience, the CSS file and the associated background image are also available under the Lab2 directory of the “**Reference Repository**”

Using Javascript injection

We have seen [in our first Lab](#) how the IBM Connections Customizer injection engine works. So, we know that if we create a Javascript script, a reference to that script will be placed at the end of the HTML code that displays the page.

If the script will contain executable statements, they will then be executed immediately as the HTML finishes loading.

Preparation Activities

You have been invited as a Contributor to the Github repository associated to this organization.



The instructor will provide you the URL and the credentials for accessing it.

You will arrive to a page similar to this one.

This repository page for 'Think2018-Customizer-Lab-Org' displays the following information:

- 11 commits
- 1 branch
- 0 releases
- 1 contributor

Commit history:

Author	File	Message	Time
Lab1	Update README.md	Update README.md	just now
Lab2	Create README.md	Create README.md	9 minutes ago
Lab3	Create README.md	Create README.md	8 minutes ago
Lab4	Create README.md	Create README.md	8 minutes ago
README.md	Update README.md	Update README.md	10 minutes ago

Large text area:

123456789

123456789 - Think2018 Customizer Lab Organization

All the scripts that will be activated for the Think2018 IBM Connections Customizer Organization

Click on the “**Lab2**” item and you will access this page:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab2'. The file list includes 'docs', 'README.md', 'profileImageBackground.jpeg', and 'profilesCustomization.css'. A preview window shows a user profile with the name 'Stefano Poglianini' and a message box. The top navigation bar has links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'.

Now you are ready to code!

Creating your script

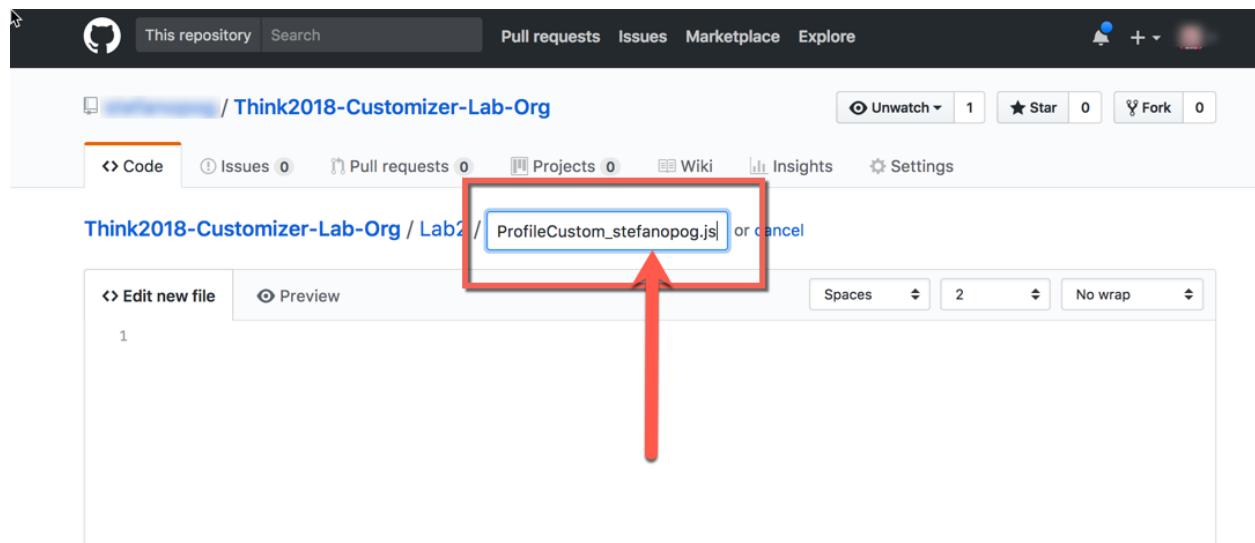
Create the Javascript file that will contain the code for your extension. Click on “Create File” as shown in the following image:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab2'. The file list includes 'docs', 'README.md', 'profileImageBackground.jpeg', and 'profilesCustomization.css'. A red arrow points to the 'Create new file' button in the top right corner of the repository header. The top navigation bar has links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'.

Now, you have to enter the name of the file. Since there are many people doing the exercise at the same time, we need to avoid conflict in filenames. For this reason, enter the following for the filename:

ProfileCustom_<identifier>.js

<identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab).



Let's now proceed step-by-step in building our code together.

- The script will need to upload a CSS file.
The CSS will need to be placed in the <head> element of the HTML page so that the browser will load it immediately and will apply it to the page.
The dojo way to do it is the following:

```
dojo.place('your HTML element', dojo.doc.head, 'last');
```

where:

- “your HTML element” will be the <link> element referencing the script.
In our case it will be :

```
'<link rel="stylesheet" type="text/css"  
src="/files/customizer/Lab2/profilesCustomization.css">'
```

Note the path for the “src” qualifier: “/files/customizer/Lab2” is the URL where the artefacts we create in our Github repository will be stored after the validation phase.



Note the use of the file “profileCustomization.css”.
As previously stated this file has already been placed in your Directory.
It contains the CSS classes that will build the new User Interface for the Profiles page.
Feel free to explore it in order to understand how the CSS classes have been used, but **please do not modify** it.

- dojo.doc.head is the handle to the <head> element provided by dojo
- “last” informs the dojo.place to place the new element as the last child of the <head> element

The final form of our statement will be:

```
dojo.place('<link rel="stylesheet" type="text/css"  
src="/files/customizer/Lab2/profilesCustomization.css">', dojo.doc.head, 'last');
```

- At this point, your script looks like this:

A screenshot of a GitHub repository page for "Think2018-Customizer-Lab-Org". The repository name is at the top. Below it, there's a navigation bar with links for "Code", "Issues 0", "Pull requests 0", "Projects 0", "Wiki", "Insights", and "Settings". The "Code" tab is selected. In the main area, there's a code editor with the following content:

```
1  dojo.place('<link rel="stylesheet" type="text/css" src="/files/customizer/Lab2/profilesCustomization.css">', dojo.doc.head, 'las
```

- We are still missing one point though.
In the code we wrote we assumed that the dojo toolkit would be available for the page in order for your script to use it. We are confident that the IBM Connections page will load the dojo toolkit at a certain point, but we cannot assume that it will be loaded by the time we first use it
We certainly do not want our script to load the dojo toolkit another time.... So we need to wait for it to be ready on the page.

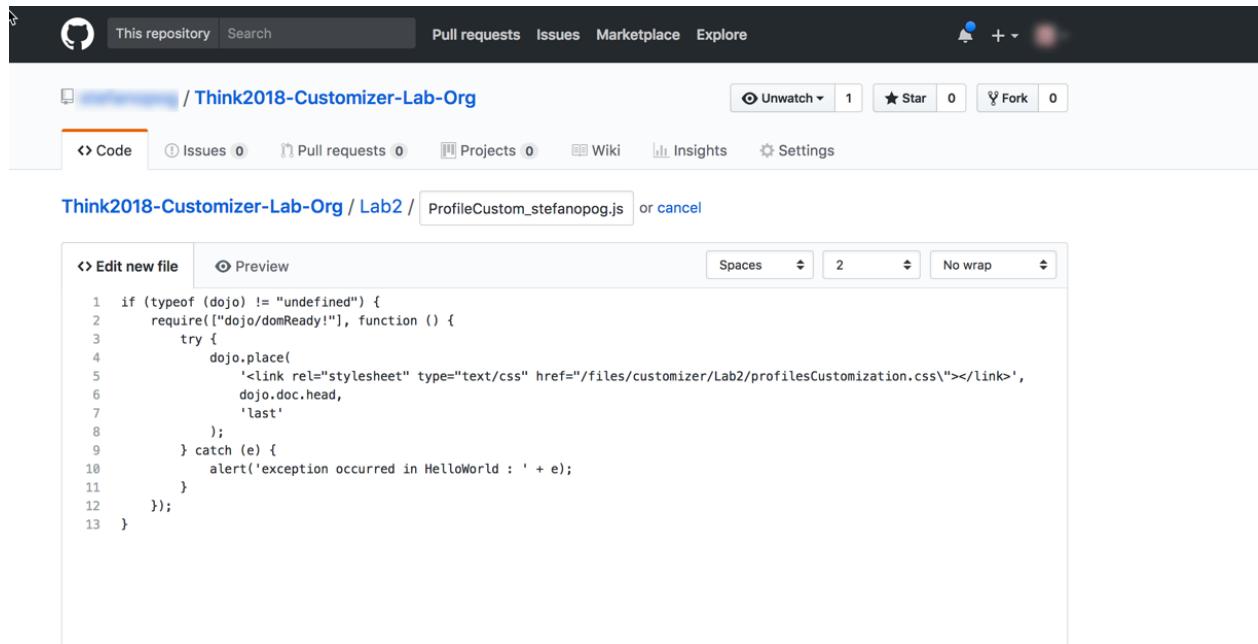
Fortunately the dojo toolkit provides a standard mechanism to deal with this point: the “dojo/domReady!” plugin (described here <https://dojotoolkit.org/reference-guide/1.10/dojo/domReady.html>).

So, we will simply need to wrap the code we wrote up until now in this way:

```
if (typeof(dojo) != "undefined") {
    require(["dojo/domReady!"], function () {
        try {
            ... all our previous code ...
        } catch(e) {
            alert('exception occurred in HelloWorld : ' + e);
        }
    });
}
```

We have introduced a “try... catch” statement to globally catch any exception we may encounter.

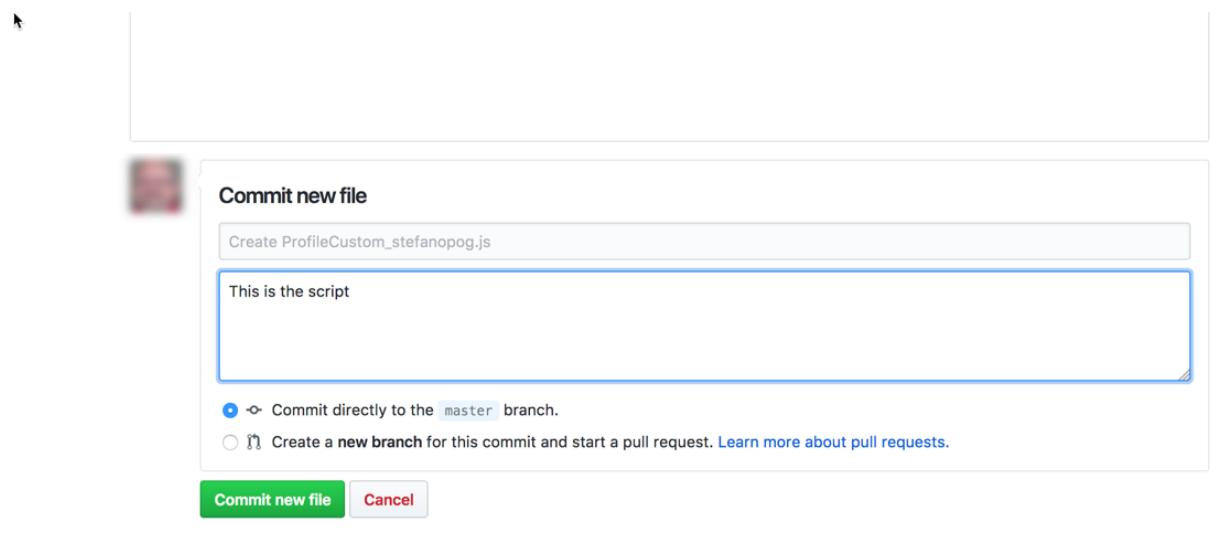
At the end, the code in your editor should look like this:



The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below that is a header for the repository "Think2018-Customizer-Lab-Org". The main area is a code editor for the file "ProfileCustom_stefanopog.js". The code in the editor is:

```
1 if (typeof(dojo) != "undefined") {
2     require(["dojo/domReady!"], function () {
3         try {
4             dojo.place(
5                 '<link rel="stylesheet" type="text/css" href="/files/customizer/Lab2/profilesCustomization.css"></link>',
6                 dojo.doc.head,
7                 'last'
8             );
9         } catch (e) {
10             alert('exception occurred in HelloWorld : ' + e);
11         }
12     });
13 }
```

- You can now commit your code to the repository. It is always good practice to add, as a comment, what you did.



- Your script is now safely recorded in the Github repository.

Branch: master [Create new file](#) [Upload files](#) [Find file](#) [History](#)

File	Message	Time
docs	ok	an hour ago
ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js	just now
README.md	ok	an hour ago
profileImageBackground.jpeg	ok	an hour ago
profilesCustomization.css	ok	an hour ago

Profiles Customization exercise

Creating your extension

Whilst the extension can be directly created in the IBM Connections Cloud Administration panel, we suggest that you save the extension definition as a JSON file in the same Github repository.



In this way, the extension may be used in other situations and may become part of the overall documentation of the project

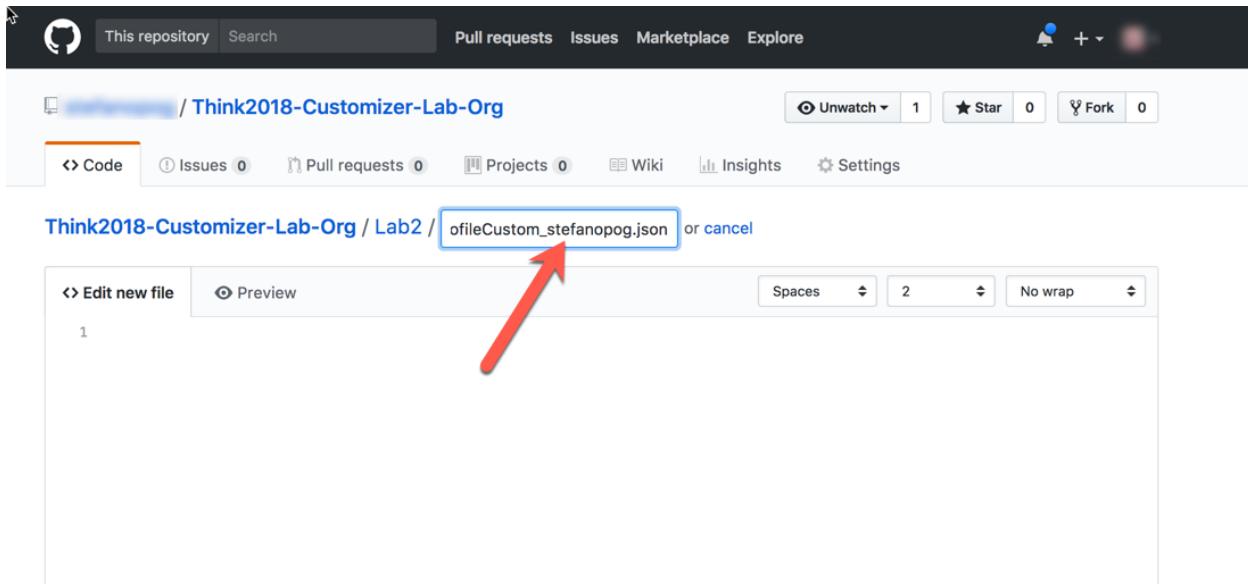
In order to create the extension, we need to create a new file inside our Github repository. The file will follow this naming convention

`ProfileCustom_<identifier>.json`

Where `<identifier>` is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab). In this way, everybody will immediately understand that this is the extension definition associated with the `ProfileCustom_<identifier>` script.

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links for 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the repository name is 'Think2018-Customizer-Lab-Org' and the branch is 'master'. A red arrow points from the text 'Create ProfileCustom_stefanopog.js' in the commit message to the 'Create new file' button in the toolbar above the file list. The commit message also includes 'Latest commit 937d9fc just now'. The file list shows several files: 'Create ProfileCustom_stefanopog.js', 'docs', 'ProfileCustom_stefanopog.js', 'README.md', 'profileImageBackground.jpeg', and 'profilesCustomization.css'. All these files were committed 'just now'. Below the file list, there's a section titled 'Profiles Customization exercise' which displays a screenshot of the IBM Connections profile customization interface. The interface shows a user profile with a photo, a 'Recent Updates' section, and a 'Network' section indicating no network contacts.

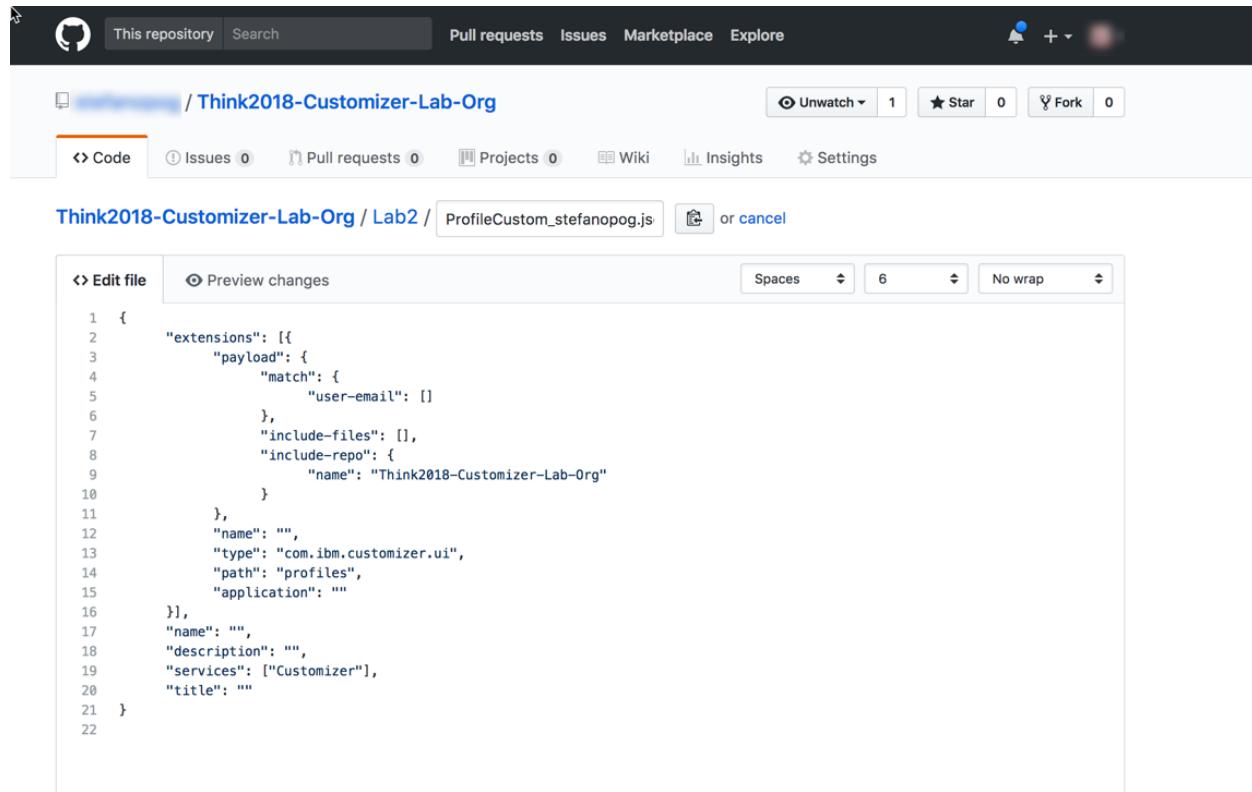
And, then,



Let's first copy this empty template inside our new json file (a copy of this template is available as the `Lab2/emptyTemplate.json` under "Reference Repository"):

```
{  
    "extensions": [ {  
        "payload": {  
            "match": {  
                "user-email": []  
            },  
            "include-files": [],  
            "include-repo": {  
                "name": "Think2018-Customizer-Lab-Org"  
            }  
        },  
        "name": "",  
        "type": "com.ibm.customizer.ui",  
        "path": "profiles",  
        "application": ""  
    }],  
    "name": "",  
    "description": "",  
    "services": ["Customizer"],  
    "title": ""  
}
```

So you will get this result:



The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 unwatched pull request, 0 stars, and 0 forks. The 'Code' tab is selected, showing a file named 'ProfileCustom_stefanopog.js'. The code content is as follows:

```
1  {
2      "extensions": [
3          {
4              "payload": {
5                  "match": {
6                      "user-email": []
7                  },
8                  "include-files": [],
9                  "include-repo": {
10                     "name": "Think2018-Customizer-Lab-Org"
11                 }
12             },
13             "name": "",
14             "type": "com.ibm.customizer.ui",
15             "path": "profiles",
16             "application": ""
17         },
18         {
19             "name": "",
20             "description": "",
21             "services": ["Customizer"],
22             "title": ""
23     }
24 }
```

Some of the values of the JSON descriptor are already provided. A complete documentation for the attributes is available at this URL :

<https://github.com/ibmcnxdev/customizer/blob/master/docs/IBMConnectionsCustomizer.pdf>

- “include-repo”: (line 8)
This MUST be the name of the Github repository that we are using. It is the “root” directory that the IBM Connections Customizer engine will use in order to retrieve the scripts that the extension uses
- “type”: (line 13)
This is a string whose value “com.ibm.customizer.ui” will tell IBM Connections Cloud which type of IBM Connections Customizer service will be used by the extension
- “path”: (line 14)
The value of “profiles” will instruct the IBM Connections Customizer engine to add the scripts referenced by the extension only to the “profiles” service of IBM Connections
- “services”: (line 19)
This is a string whose value “Customizer” will tell IBM Connections Cloud which type extension it needs to register in the Application Registry.

Let's now fill the other values that are required:

- “user-email”: (line 5)
We can restrict the injection of the scripts associated to the extension only to users

whose email address will match the string.

Since we do not want to interfere with other people working at the same time on the same IBM Connections Cloud organization, we need to provide a value for this attribute. The value needs to be the email address you have been provided by the instructor to log into IBM Connections Cloud (enclosed in double-quotes). So, something like:

```
"user-email": ["stefanopog@think2018-Customizer.com"]
```



In real life, you may not need to use this attribute if the extension you are creating will be used by **all the people** in your organization. See the [documentation](#) for discovering all the possibilities associated to the “match” clause.

- “include-files”: (line 7)

This is where you will actually reference the script that you created earlier in the exercise.

The path to reference the script is relative to the Github repository you are using (see the “include-repo” clause described above).

So in your situation you need to provide the following value :

```
"include-files": ["Lab2/ProfileCustom_<identifier>.js"]
```

- “name”: (line 12)

this is the name of the Extension.

You can use the following:

```
"name": "Profile Ext Lab2 <identifier>"
```

- “name”: (line 17)

This is the name of the Application we are going to declare to IBM Connections.

Typically, an “Application” can declare one or more extensions (in our example, we have one extension only associated with the application).

This name must be unique for the IBM Connections Organization, since it is the name that will be recognized by the IBM Connections Application registry. You can use the following name:

```
"name": "Profile App Lab2 <identifier>"
```

- “application”: (line 14)

This entry (which is within the “extension”) actually needs to reference the correct “Application”). So, the value you will provide for this attribute MUST BE the same as the value you provided for the “name” attribute of the Application (line 17). You can use the following name:

```
"application": "Profile App Lab2 <identifier>"
```

- “description”: (line 18)

This is a free text description of what the Application (and its Extensions) provide. You can use the following:

```
"description": "Profile Customization from <identifier>"
```

- “title”: (line 20)

This is the title that will appear in the IBM Connections Cloud application registry. You can use the following title:

```
"title": "<identifier> Profile Customization"
```

Once the Organization admin will use your JSON file to declare the extension, this is how it will appear in the IBM Connections Cloud Application Registry:

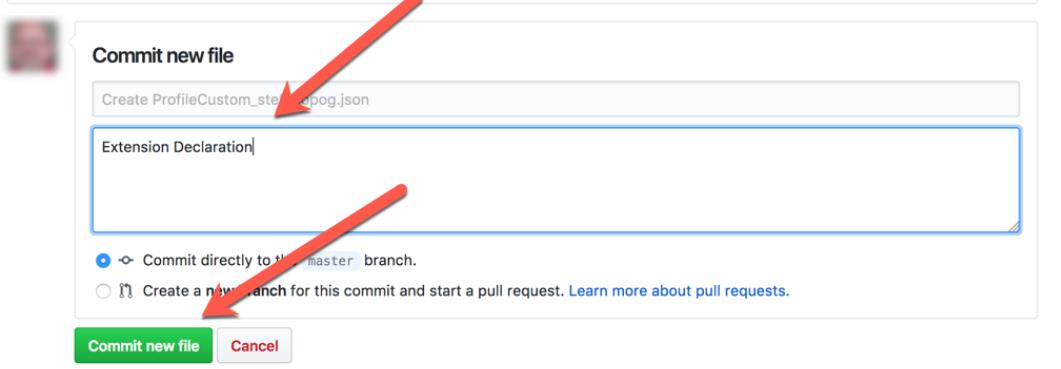
The screenshot shows the IBM Connections Cloud application registry interface. On the left, there's a navigation sidebar with sections like Personal, Organization Extensions (which is currently selected), and System Settings. The main area is titled 'Administration' and shows a list of apps under 'All Apps'. One specific app card is highlighted: 'stefanopog Profile Customiz...', which is a 'Service: Customizer'. The card also indicates that the 'Profile Customization from stefanopog' extension is 'Enabled'. Two red arrows point to the app card: one labeled 'title' pointing to the app name, and another labeled 'description' pointing to the service name and description below it.

At this point, the JSON file you are editing in Github will look like this:

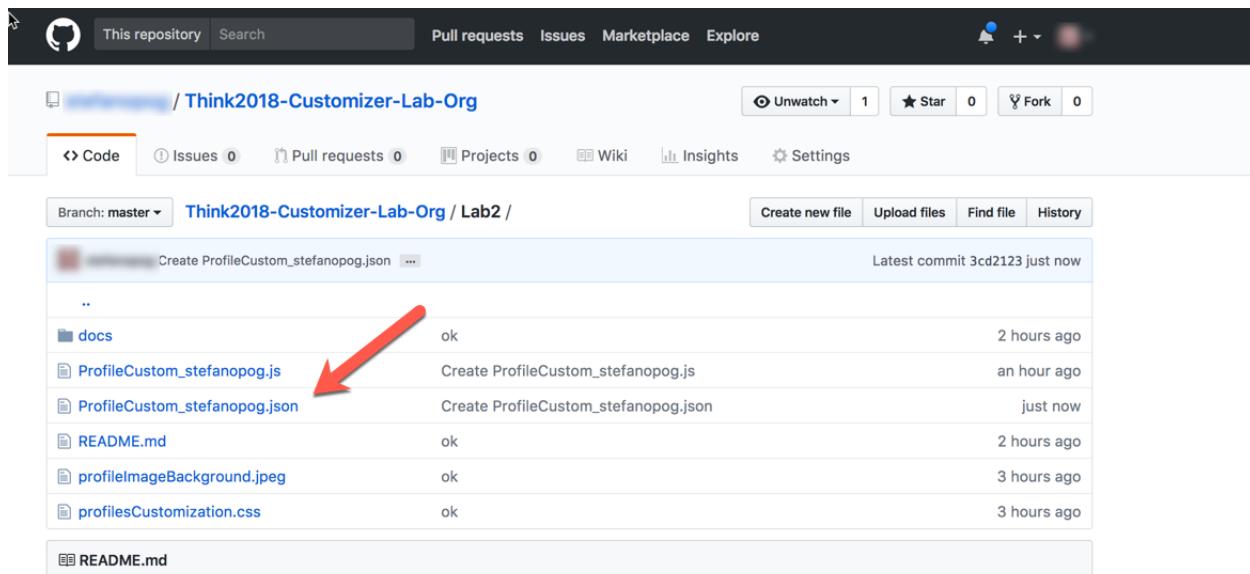
```
1  {
2      "extensions": [
3          {
4              "payload": {
5                  "match": {
6                      "user-email": ["stefanopog@think2018-Customizer.com"]
7                  },
8                  "include-files": ["Lab2/Profilecustom_stefanopog.js"],
9                  "include-repo": {
10                      "name": "Think2018-Customizer-Lab-Org"
11                  }
12              },
13              "name": "Profile Ext Lab2 stefanopog",
14              "type": "com.ibm.customizer.ui",
15              "path": "profiles",
16              "application": "Profile App Lab2 stefanopog"
17          },
18          {
19              "name": "Profile App Lab2 stefanopog",
20              "description": "Profile Customization from stefanopog",
21              "services": ["Customizer"],
22              "title": "stefanopog Profile Customization"
23          }
24      ]
25  }
```

You can Commit the change to the Github repository:

```
15      "application": "Profile App Lab2 stefanopog"
16  ],
17  "name": "Profile App Lab2 stefanopog",
18  "description": "Profile Customization from stefanopog",
19  "services": ["Customizer"],
20  "title": "stefanopog Profile Customization"
21 }
```

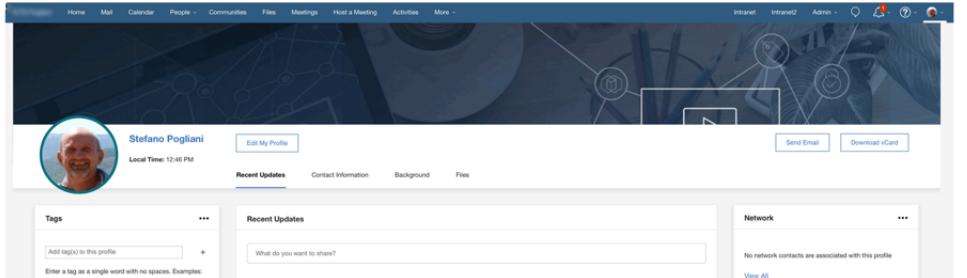


And your Repository will look like this:



The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Think2018-Customizer-Lab-Org' and the branch is 'master'. The 'Code' tab is selected. A red arrow points to the file 'ProfileCustom_stefanopog.json' in the file list. The file was created just now and has a status of 'ok'. Other files listed include 'ProfileCustom_stefanopog.js', 'docs', 'README.md', 'profileImageBackground.jpeg', and 'profilesCustomization.css'. Below the file list, there is a preview of a profile customization exercise.

Profiles Customization exercise



The screenshot shows a profile customization interface. It features a profile picture of Stefano Pogliani, a local time of 12:00 PM, and a 'Edit My Profile' button. Below the profile are sections for 'Recent Updates', 'Contact Information', 'Background', and 'Files'. The 'Recent Updates' section includes a text input field for sharing updates. To the right, there is a 'Network' section showing no network contacts. The interface has a dark blue background with various UI elements.

Making your extension available

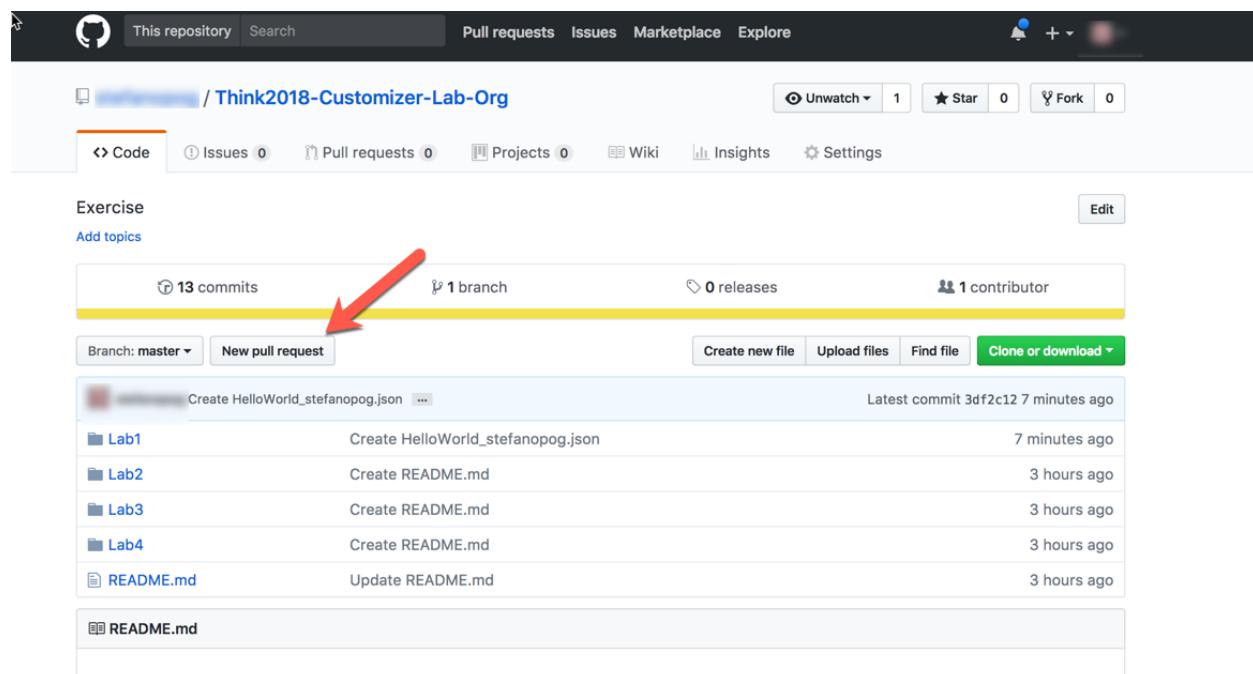
At this point, everything is ready to be put in production. This is what the [documentation](#) states:

- a. Share your repo with IBM – [add "ibmcndev" as a collaborator](#)
- b. IBM (`ibmcnxdev`) then creates a fork of your repository under `github.com/ibmcnxdev` and grants you read access by default.
- c. You can continue to work on your extension using your original repo for your source code activity, but once you are ready to deliver to IBM Cloud you must issue a [pull request](#) to IBM.
- d. IBM merges your pull request once acceptance criteria are met.
- e. Upon merge, the repo files are automatically pushed to IBM Customizer via a webhook.
- f. Rinse & repeat starting at Step (c) for extension updates.

Steps **a.** and **b.** have already been done for you before you started the exercise.

Now, you need to execute Step **c.**

Go back to the root directory of your Github repository and click on the “Pull Request” button:



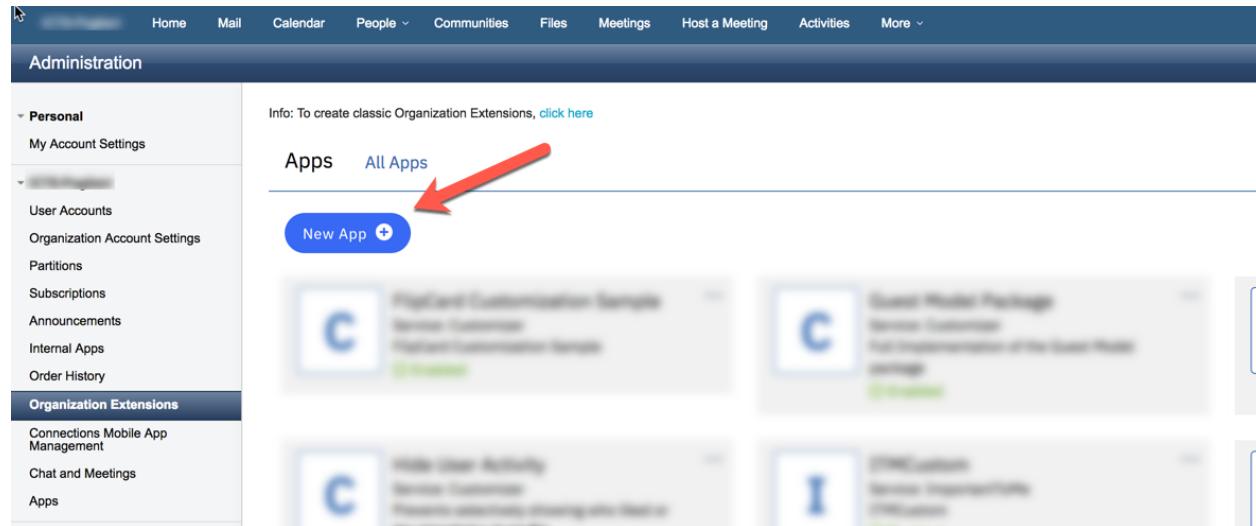
Activating the extension

Now your extension has been validated. The validation process (from **IBMCNDEV**), if successful, will make the files you created in this exercise available to IBM Connections Cloud and to IBM Connections Customizer.

The only thing that is missing is to actually define the Extension in the IBM Connections Cloud Application Registry. This is the step that informs IBM Connections Customizer of what to do with the scripts that you just created.

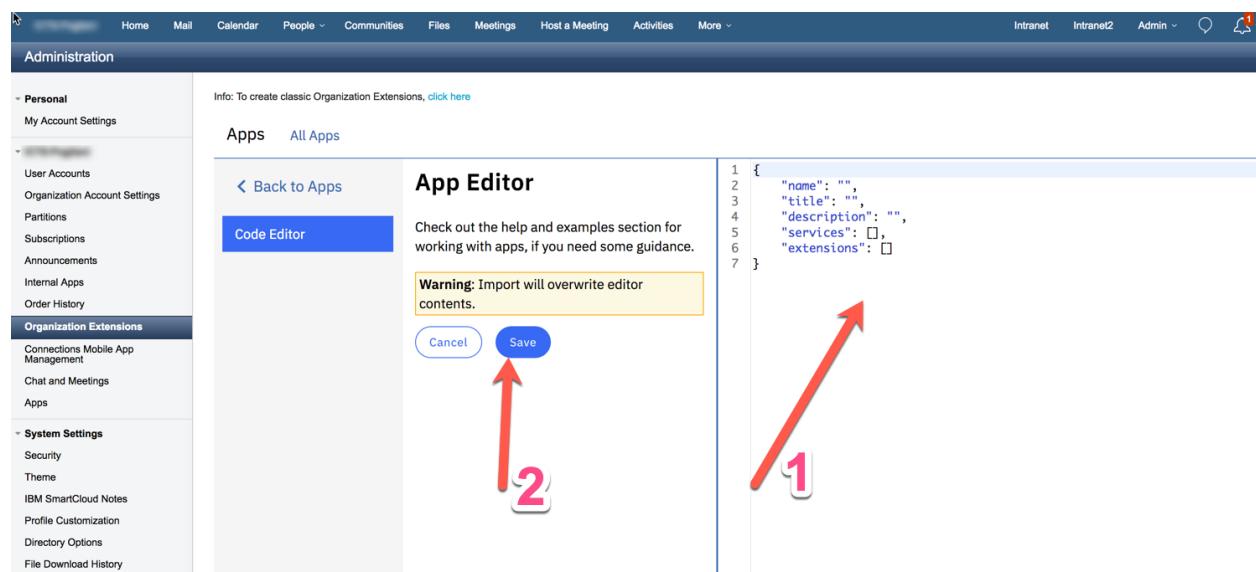
The IBM Connections Cloud organization administrator needs to be given the JSON file which contains the extension (the one you created [here](#)).

He will go to the IBM Connections Application Registry and will declare a new extension:



A screenshot of the IBM Connections Application Registry interface. The top navigation bar includes Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, and More. Below this is a dark blue header bar with the word 'Administration'. On the left, a sidebar menu shows Personal (My Account Settings), User Accounts, Organization Account Settings, Partitions, Subscriptions, Announcements, Internal Apps, Order History, and Organization Extensions (Connections Mobile App Management, Chat and Meetings, Apps). The main content area has a heading 'Info: To create classic Organization Extensions, click here'. Below it are tabs for 'Apps' and 'All Apps', with 'All Apps' selected. A red arrow points to a blue 'New App' button. Several blurred app cards are visible in the background.

A new screen is shown:



A screenshot of the 'App Editor' screen. The top navigation bar and sidebar are identical to the previous screenshot. The main content area has a heading 'Info: To create classic Organization Extensions, click here'. Below it are tabs for 'Apps' and 'All Apps', with 'All Apps' selected. A red arrow points to the 'Code Editor' tab. The right side of the screen shows a code editor with the following JSON content:

```
1 {  
2   "name": "",  
3   "title": "",  
4   "description": "",  
5   "services": [],  
6   "extensions": []  
7 }
```

Below the code editor is a warning message: 'Warning: Import will overwrite editor contents.' with a 'Cancel' and 'Save' button. A red arrow labeled '1' points to the JSON code area, and another red arrow labeled '2' points to the 'Save' button.

The IBM Connections Cloud organization admin will replace the content pointed by the right arrow (1) with the content of the JSON file you created and, then, saves (2) the extension.

You can now go to the homepage and see the final result:

The screenshot shows a user profile page for Stefano Poglianì. At the top, there's a navigation bar with links for Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, Intranet, Intranet2, Admin, and a search bar. Below the header is a decorative banner featuring a network graph and tropical leaves.

The main content area includes:

- User Profile:** A circular profile picture of Stefano Poglianì, his name, and the local time (12:46 PM).
- Profile Actions:** Buttons for "Edit My Profile", "Send Email", and "Download vCard".
- Recent Updates:** A section where users can share messages. One entry shows a message from Stefano: "Hello" (Thursday, January 22). Another entry shows he edited a file named "In this article.docx".
- Tags:** A section for adding tags to the profile.
- Network:** A section showing that no network contacts are associated with this profile, with a "View All" link.
- My Links:** A section showing that there are no links yet for this profile, with a "Add Link" button.

Using CSS Declaration

What we have been describing in the previous section ([Using Javascript injection](#)) is the approach which has Javascript to load the CSS file.

Recently, the IBM Connections Customizer engine has been improved in order to directly accept the declaration of CSS files within the Extension declaration. Roughly, you will be able to use a direct reference to the CSS files you may want to apply to a given IBM Connections page from the “include-files” clause of the Extension declaration.



Obviously, the rendering of the “customized” IBM Connections page will be much quicker and effective

Let's see together how we can modify what we have previously done in order to make a proper use of this new functionality.

Preparation Activities

What we need to do is, simply, to modify the Extension declaration in order to reference the CSS file instead of the Javascript file.

Go to the Github directory associated with the second Lab (the one we just worked on in the [first part of this lab](#)).

The screenshot shows a GitHub repository interface for 'Think2018-Customizer-Lab-Org'. A red arrow points to the 'profileCustomization.css' file in the 'Lab2' folder. The file was updated by Stefano Poglian on May 22, 2018, at 10:45 AM. Below the repository view, there is a preview of a web page titled 'Profiles Customization exercise'.

Branch: master / Think2018-Customizer-Lab-Org / Lab2 / profileCustomization.css

Update ProfileCustom_stefanopog.json

ProfileCustom_stefanopog.js

ProfileCustom_stefanopog.json

README.md

profileImageBackground.jpeg

profileCustomization.css

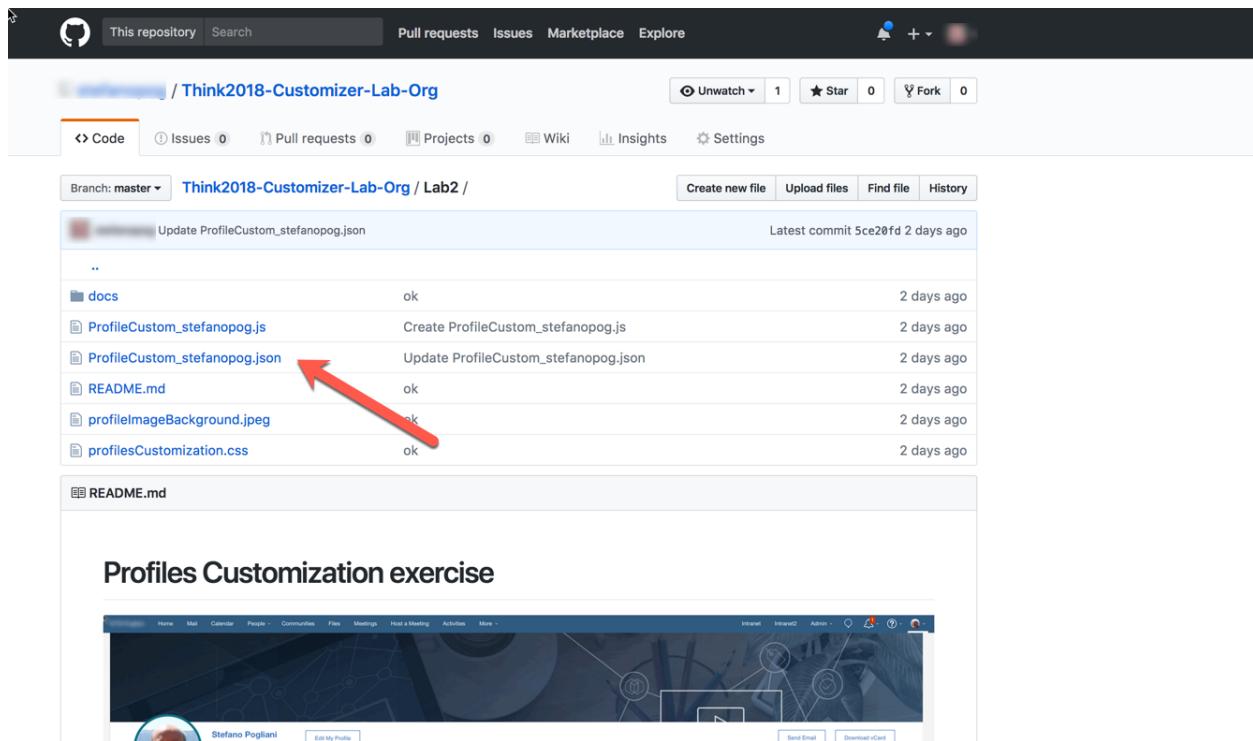
CREATE NEW FILE UPLOAD FILES FIND FILE HISTORY

Unwatch 1 Star 0 Fork 0

Profiles Customization exercise

Home Mail Calendar People Communities File Meetings Host a Meeting Activities More Intranet Intranet Admin Send Email Download vCard

Click on the JSON Extension file your previously created:



The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Think2018-Customizer-Lab-Org / Lab2 /'. The 'Code' tab is selected. The file list includes:

File	Commit Message	Last Commit
..		2 days ago
docs	ok	2 days ago
ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js	2 days ago
ProfileCustom_stefanopog.json	Update ProfileCustom_stefanopog.json	2 days ago
README.md	ok	2 days ago
profileImageBackground.jpeg	ok	2 days ago
profilesCustomization.css	ok	2 days ago

A red arrow points to the 'ProfileCustom_stefanopog.json' file. Below the file list is a preview of a profile page titled 'Profiles Customization exercise'.

Profiles Customization exercise

Stefano Pogliani

When the “view” on this file opens,

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links for 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. On the right side of the header, there are buttons for 'Unwatch' (with a count of 1), 'Star' (0), 'Fork' (0), and a profile icon.

The main content area shows a file path: '/ Think2018-Customizer-Lab-Org'. Below this, there are tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. A dropdown menu indicates the branch is 'master'. The file name is 'Think2018-Customizer-Lab-Org / Lab2 / ProfileCustom_stefanopog.json'. There are buttons for 'Find file' and 'Copy path'.

The file content is displayed as follows:

```
22 lines (21 sloc) | 773 Bytes
Raw Blame History ⌂ ⌐ ⌁
```

```
1 {
2     "extensions": [
3         "payload": {
4             "match": {
5                 "user-email": ["stefanopog@think2018-Customizer.com"]
6             },
7             "include-files": ["Lab2/Profilecustom_stefanopog.js"],
8             "include-repo": {
9                 "name": "Think2018-Customizer-Lab-Org"
10            }
11        },
12        "name": "Profile Ext Lab2 stefanopog",
13        "type": "com.ibm.customizer.ui",
14        "path": "profiles",
15        "application": "Profile App Lab2 stefanopog"
16    ],
17    "name": "Profile App Lab2 stefanopog",
18    "description": "Profile Customization from stefanopog",
19    "services": ["Customizer"],
20    "title": "stefanopog Profile Customization"
21 }
```

by means of the mouse, select the whole content of the file and, then, issue CTRL-C to copy the selection into the computer clipboard:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 unwatched pull request, 0 stars, and 0 forks. The 'Code' tab is selected, showing a single commit titled 'Update ProfileCustom_stefanopog.json' made by 'stefanopog' 2 days ago. The commit message is '5ce20fd 2 days ago'. The JSON file content is displayed below:

```
1 {
2     "extensions": [
3         {
4             "payload": {
5                 "match": {
6                     "user-email": ["stefanopog@think2018-Customizer.com"]
7                 },
8                 "include-files": ["Lab2/Profilecustom_stefanopog.js"],
9                 "include-repo": {
10                     "name": "Think2018-Customizer-Lab-Org"
11                 }
12             },
13             "name": "Profile Ext Lab2 stefanopog",
14             "type": "com.ibm.customizer.ui",
15             "path": "profiles",
16             "application": "Profile App Lab2 stefanopog"
17         ],
18         "name": "Profile App Lab2 stefanopog",
19         "description": "Profile Customization from stefanopog",
20         "services": ["Customizer"],
21         "title": "stefanopog Profile Customization"
22 }
```

Now go back to the Lab2 Directory clicking on the “Lab2” link shown in the picture below:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 0 pull requests, 0 issues, 0 projects, 0 wiki pages, 0 insights, and 0 settings. A red arrow points to the file path 'Think2018-Customizer-Lab-Org / Lab2 / ProfileCustom_stefanopog.json' in the breadcrumb navigation bar. The file was last updated by '5ce20fd' 2 days ago. It contains 22 lines (21 sloc) and 773 Bytes. The code is a JSON object:

```
1 {
2     "extensions": [
3         "payload": {
4             "match": {
5                 "user-email": ["stefanopog@think2018-Customizer.com"]
6             },
7             "include-files": ["Lab2/Profilecustom_stefanopog.js"],
8             "include-repo": {
9                 "name": "Think2018-Customizer-Lab-Org"
10            }
11        },
12        "name": "Profile Ext Lab2 stefanopog",
13        "type": "com.ibm.customizer.ui",
14        "path": "profiles",
15        "application": "Profile App Lab2 stefanopog"
16    ],
17    "name": "Profile App Lab2 stefanopog",
18    "description": "Profile Customization from stefanopog",
19    "services": ["Customizer"],
20    "title": "stefanopog Profile Customization"
21 }
```

Now create a new file in the Lab2 Directory as shown here:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab2'. At the top right, there is a 'Create new file' button. A red arrow points to this button. Below it is a list of files:

Update ProfileCustom_stefanopog.json		Latest commit 5ce20fd 2 days ago
..		
docs	ok	2 days ago
ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js	2 days ago
ProfileCustom_stefanopog.json	Update ProfileCustom_stefanopog.json	2 days ago
README.md	ok	2 days ago
profileImageBackground.jpeg	ok	2 days ago
profilesCustomization.css	ok	2 days ago

Below the file list is a section titled 'Profiles Customization exercise' containing a screenshot of an IBM Connections Cloud profile page.

Give the new file the following name:

ProfilCustom_<identifier>_2.json

Where <identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab). Note that we append a “_2” suffix to the file name in order to distinguish it from the other one.

This screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The user is editing the file 'leCustom_stefanopog_2.json'. The file content is currently empty, indicated by the number '1' at the top. A red arrow points from the file name in the path bar to the input field where the file content is being pasted.

Now, using CTRL-V paste in the canvas what you previously copied to the clipboard:

The screenshot shows the same GitHub repository page after the JSON content has been pasted into the code editor. A red box highlights the pasted JSON code, and a red arrow points from the right edge of the highlighted area towards the bottom right corner of the code editor.

```
1 {  
2     "extensions": [{  
3         "payload": {  
4             "match": {  
5                 "user-email": ["stefanopog@think2018-Customizer.com"]  
6             },  
7             "include-files": ["Lab2/ProfileCustom_stefanopog.js"],  
8             "include-repo": {  
9                 "name": "Think2018-Customizer-Lab-Org"  
10            }  
11        },  
12        "name": "Profile Ext Lab2 stefanopog",  
13        "type": "com.ibm.customizer.ui",  
14        "path": "profiles",  
15        "application": "Profile App Lab2 stefanopog"  
16    ]},  
17    "name": "Profile App Lab2 stefanopog",  
18    "description": "Profile Customization from stefanopog",  
19    "services": ["Customizer"],  
20    "title": "stefanopog Profile Customization"  
21 }
```

The only modifications you need to perform are:

- **at line 7.** You will change:

```
"include-files": ["Lab2/ProfileCustom_stefanopog.js"]
```

 with

```
"include-files": ["Lab2/profilesCustomization.css"]
```
- **at line 17.** You will change:

```
"name": "Profile App Lab2 <identifier>"
```

 with

```
"name": "Profile App 2 Lab2 <identifier>"
```
- **at line 15.** You will change:

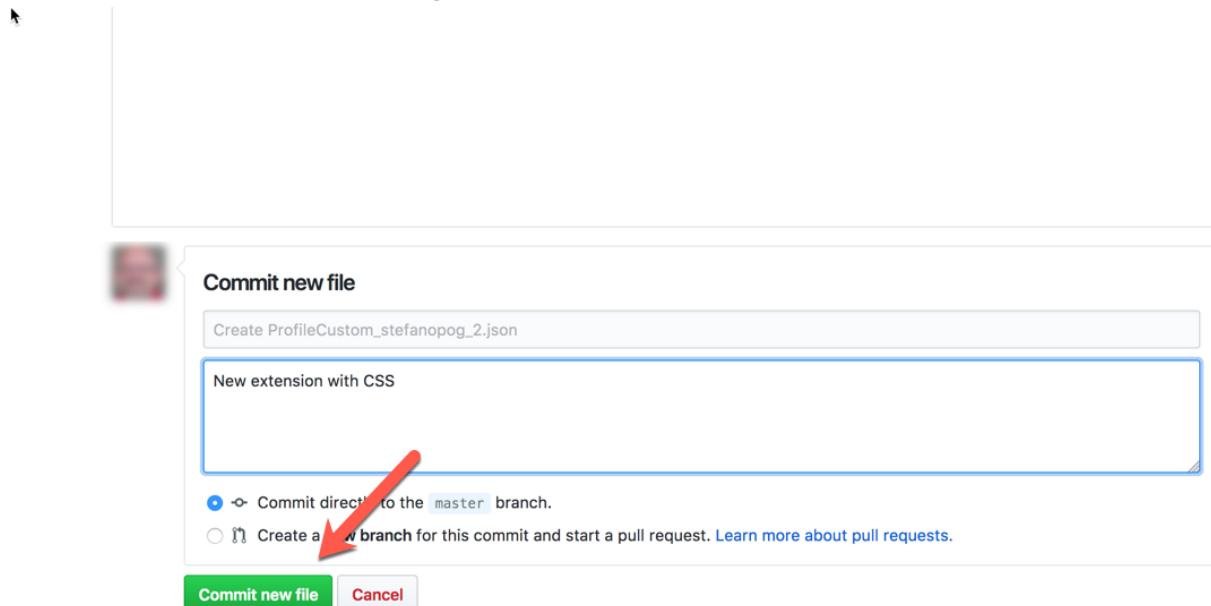
```
"application": "Profile App Lab2 <identifier>"
```

 with

```
"application": "Profile App 2 Lab2 <identifier>"
```

Those changes created a new extension (we added a **2** to the name in order to distinguish it from the one we previously created) which loads the CSS file directly.

Then you can commit your change:



And your Repository will look like this:

Branch: master / Think2018-Customizer-Lab-Org / Lab2 /

	Create ProfileCustom_stefanopog_2.json	Latest commit c3fb247 just now
..		
	docs	ok
	ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js
	ProfileCustom_stefanopog.json	Update ProfileCustom_stefanopog.json
	ProfileCustom_stefanopog_2.json	Create ProfileCustom_stefanopog_2.json
	README.md	ok
	profileImageBackground.jpeg	ok
	profilesCustomization.css	ok

README.md

Profiles Customization exercise

Home Mail Calendar People Communities Files Meetings Host a Meeting Activities More Intranet Intranet2 Admin -

Stefano Pogliani Edit My Profile Send Email Download vCard

Making your extension available

At this point, everything is ready to be put in production. This is what the [documentation](#) states:

- a. Share your repo with IBM – [add "ibmcnxd" as a collaborator](#)
- b. IBM ([ibmcnxdev](#)) then creates a fork of your repository under [github.com/ibmcnxdev](#) and grants you read access by default.
- c. You can continue to work on your extension using your original repo for your source code activity, but once you are ready to deliver to IBM Cloud you must issue a [pull request](#) to IBM.
- d. IBM merges your pull request once acceptance criteria are met.
- e. Upon merge, the repo files are automatically pushed to IBM Customizer via a webhook.
- f. Rinse & repeat starting at Step (c) for extension updates.

Steps **a.** and **b.** have already been done for you before you started the exercise.

Now, you need to execute **Step c.**

Go back to the root directory of your Github repository and click on the “Pull Request” button:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. At the top, there are navigation links: 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the header, the repository name is displayed along with metrics: 13 commits, 1 branch, 0 releases, and 1 contributor. A red arrow points to the 'New pull request' button, which is located in the middle of the page below the repository stats. The main content area shows a list of recent commits, including 'Create HelloWorld_stefanopog.json' by 'Lab1' (7 minutes ago), 'Create README.md' by 'Lab2' (3 hours ago), 'Create README.md' by 'Lab3' (3 hours ago), 'Create README.md' by 'Lab4' (3 hours ago), and 'Update README.md' by 'README.md' (3 hours ago). There is also a section for 'README.md'.

Activating the extension

Now your extension has been validated. The validation process (from **IBMCNDEV**), if successful, will make the files you created in this exercise available to IBM Connections Cloud and to IBM Connections Customizer.

The only thing that is missing is to actually define the Extension in the IBM Connections Cloud Application Registry. This is the step that informs IBM Connections Customizer of what to do with the scripts that you just created.

The IBM Connections Cloud organization administrator needs to be given the JSON file which contains the extension (the one you created [here](#)).

He will go to the IBM Connections Application Registry and will declare a new extension:

The screenshot shows the IBM Connections Cloud Administration interface. On the left, there's a sidebar with sections like 'Personal', 'User Accounts', 'Organization Account Settings', 'Partitions', 'Subscriptions', 'Announcements', 'Internal Apps', 'Order History', and 'Organization Extensions'. Under 'Organization Extensions', there are links for 'Connections Mobile App Management', 'Chat and Meetings', and 'Apps'. The main area has a heading 'Info: To create classic Organization Extensions, click here'. Below it, there are tabs 'Apps' and 'All Apps'. A prominent red arrow points to a blue 'New App' button with a '+' icon. To the right, there are several thumbnail previews of existing organization extensions.

A new screen is shown:

The screenshot shows the 'App Editor' screen. The left sidebar includes 'Personal', 'User Accounts', 'Organization Account Settings', 'Partitions', 'Subscriptions', 'Announcements', 'Internal Apps', 'Order History', 'Organization Extensions' (which is selected), and 'System Settings'. The 'Organization Extensions' section contains 'Connections Mobile App Management', 'Chat and Meetings', and 'Apps'. The main area has a 'Back to Apps' link and a 'Code Editor' tab (which is selected). It displays a JSON code editor with the following content:

```

1 {  
2   "name": "",  
3   "title": "",  
4   "description": "",  
5   "services": [],  
6   "extensions": []  
7 }

```

A red arrow points from the right towards the JSON code area, labeled '1'. Another red arrow points upwards from the bottom towards the 'Save' button, labeled '2'.

The IBM Connections Cloud organization admin will replace the content pointed by the right arrow (1) with the content of the JSON file you created and, then, **saves** (2) the extension.

You can now go to the homepage and see the final result:

Recent Updates

What do you want to share?

You liked your message.

Hello

Thursday 11

Stefano Poglian edited a file.

In this article.docx

Stefano Poglian edited a file.

January 22

Network

No network contacts are associated with this profile

[View All](#)

My Links

There are no links yet for this profile.

[Add Link](#)

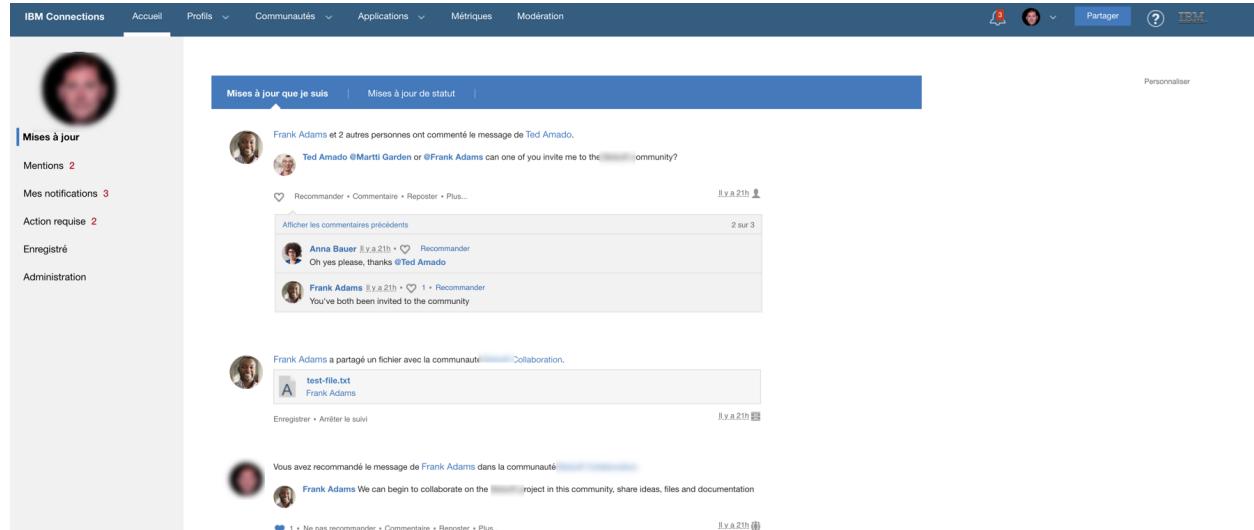
This ends Lab 2.
Congratulations!

Lab 3 Reducing the complexity of Connections UI

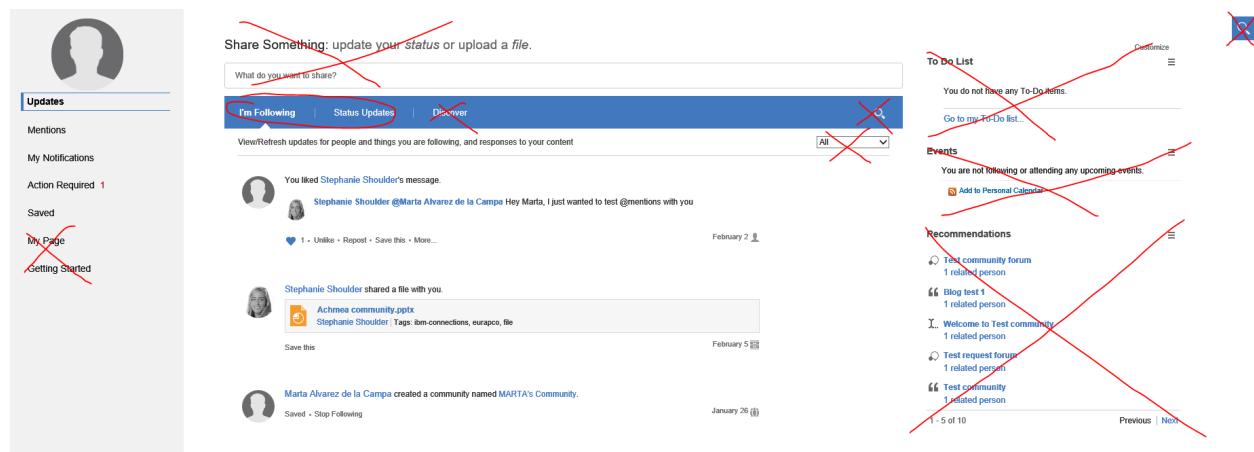
This is a very simple exercise, which continues what we learned in the second part of the [Lab 2 exercise](#).

The main goal here is to reduce the clutter on the Classic IBM Connections Homepage. And we will do using CSS only.

What we want to achieve



Compare the previous image with the default :



How we did ?

So, you will find at this location [Lab3](#) the repository containing the code.

stefanopog / CustomizerThinkLab

Branch: master CustomizerThinkLab / Lab3 /

Stefano Pogliani and Stefano Pogliani lab3 Latest commit 997cdb9 11 minutes ago

..

home.rightcol.widgets.css lab3 11 minutes ago

home.rightcol.widgets.json lab3 11 minutes ago

Let's look at the [CSS file](#) first:

stefanopog / CustomizerThinkLab

Branch: master CustomizerThinkLab / Lab3 / home.rightcol.widgets.css

stefanopog Update home.rightcol.widgets.css ea6ad1a just now

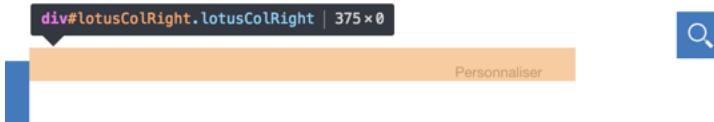
1 contributor

20 lines (19 sloc) | 626 Bytes

```
1 //
2 // Widgets on the right column
3 //
4 #lotusColRight .lotusSection2.homepage-widget {display: none}
5 //
6 // activityStream Header
7 //
8 #activityStreamHeader .shareSome-desc {display: none}
9 //
10 // activityStream corpus
11 #activityStreamMain .streamSharebox {display: none}
12 #activityStreamMain .filterArea {display: none}
13 #activityStreamMain .streamHeaderWrapper #discover {display: none}
14 #activityStreamMain .streamHeaderWrapper span[dojoattachpoint=searchContainerNode] {display: none}
15 //
16 // Links on the left column
17 //
18 #lotusColLeft .lconnHomepageMyPage {display: none}
19 #lotusColLeft .lconnHomepageGettingStarted {display: none}
```

Very few lines of CSS made the magic.

- On Line 4 we addressed the Widgets that appear on the right column:



Using the browser Developers Tools, we find that this area corresponds to :

```
<div id="leftColBgPlaceholder" class="lotusHidden leftColBackgroundPlaceholder">&ampnbsp</div>
:  ▼<div id="lotusColRight" class="lotusColRight">
:    <div id="hplaceholder_right_top"></div>
:      <div id="1" class="dojoDndContainer dojoDndSource dojoDndTarget">
:        ><div class="lotusSection2 homepage-widget" role="complementary" aria-labelledby="a7cffd2b-6
7c1fd3226f52"></div>
:        ><div class="lotusSection2 homepage-widget" role="complementary" aria-labelledby="d56915da-7
d97e58f00b0a"></div>
:        ><div class="lotusSection2 homepage-widget" role="complementary" aria-labelledby="73fb9044-d
13790cf94bd6"></div>
:      </div>
:      <div id="hplaceholder_right_center"></div>
:      <div id="hplaceholder_right_bottom"></div>
:    </div>
:  <div id="hplaceholder_right_center"></div>
:  <div id="hplaceholder_right_bottom"></div>
:  <div id="hplaceholder_right_center"></div>
```

So with the CSS we just said that we want to hide any instance of class “lotusSection2 homepage-widget” which inherits from the widget whose id is “lotusColRight”

- The widget whose id is “activityStreamHeader” controls the header of the ActivityStream
The widget whose class is “shareSome-Desc” controls this area:



- The Widget whose id is “activityStreamMain” actually encompasses all the area shown here below:

The screenshot shows the "activityStreamMain" widget with a blue header bar. Below it is a white section with a blue header containing "Mises à jour que je suis" and "Mises à jour de statut". A post from "Frank Adams" is displayed, followed by two comments from "Anna Bauer" and "Frank Adams". Below this is another post from "Frank Adams" sharing a file. The interface includes navigation links like "Recommander", "Commentaire", "Reposte", and "Plus...". At the bottom, there are buttons for "Enregistrer" and "Arrêter le suivi". A red arrow points to the top of the main content area, labeled "div#activityStreamMain, lotusWidgetBody | 1020 x 1843".

- Line 13 and Line 14 respectively control this :

The screenshot shows a blue navigation bar with three items: "Mises à jour que je suis", "Mises à jour de statut", and "Découvrir". The "Découvrir" item is highlighted with a red arrow pointing to it, labeled "Line 13". To the right of the bar is a search icon.

- Line 11 and Line 12 respectively control this:

The screenshot shows the activity stream interface. At the top, there's a search bar with the placeholder "Que voulez-vous partager ?" and a red arrow pointing to it, labeled "Line 11". Below the search bar is a blue header bar with "Mises à jour que je suis" and "Mises à jour de statut". Underneath is a section titled "Affichez/actualisez les mises à jour des personnes et des choses que vous suivez, ainsi que les réponses à votre contenu" with a red arrow pointing to it, labeled "Line 12". At the bottom, a post from "Frank Adams" is visible.

- Lines 18 and 19 hide the two items in the left column

The Extension ?

At this point, the JSON extension is very simple : it is available [here](#):

The screenshot shows a GitHub repository page for **stefanopog / CustomizerThinkLab**. The repository has 1 unwatched star, 0 forks, and 1 pull request. The user **Stefano Pogliani** made a commit to the **master** branch 40 minutes ago, with the commit hash **997cdb9**. The file **home.rightcol.widgets.json** contains 26 lines (26 sloc) and 697 Bytes. The JSON content is as follows:

```
1  {
2      "name": "Homepage RightCol Widgets",
3      "title": "Homepage RightCol Widgets",
4      "description": "Hides RightCol Widgets",
5      "services": [
6          "Customizer"
7      ],
8      "state": "enabled",
9      "extensions": [
10         {
11             "name": "HomeRightColWidget",
12             "type": "com.ibm.customizer.ui",
13             "payload": {
14                 "include-files": [
15                     "home.rightcol.widgets.css"
16                 ],
17                 "cache-headers": {
18                     "cache-control": "max-age=0"
19                 }
20             },
21             "path": "homepage",
22             "application": "Homepage RightCol Widgets",
23             "state": "enabled"
24         }
25     ]
26 }
```

Congratulations. This ends Lab 3.

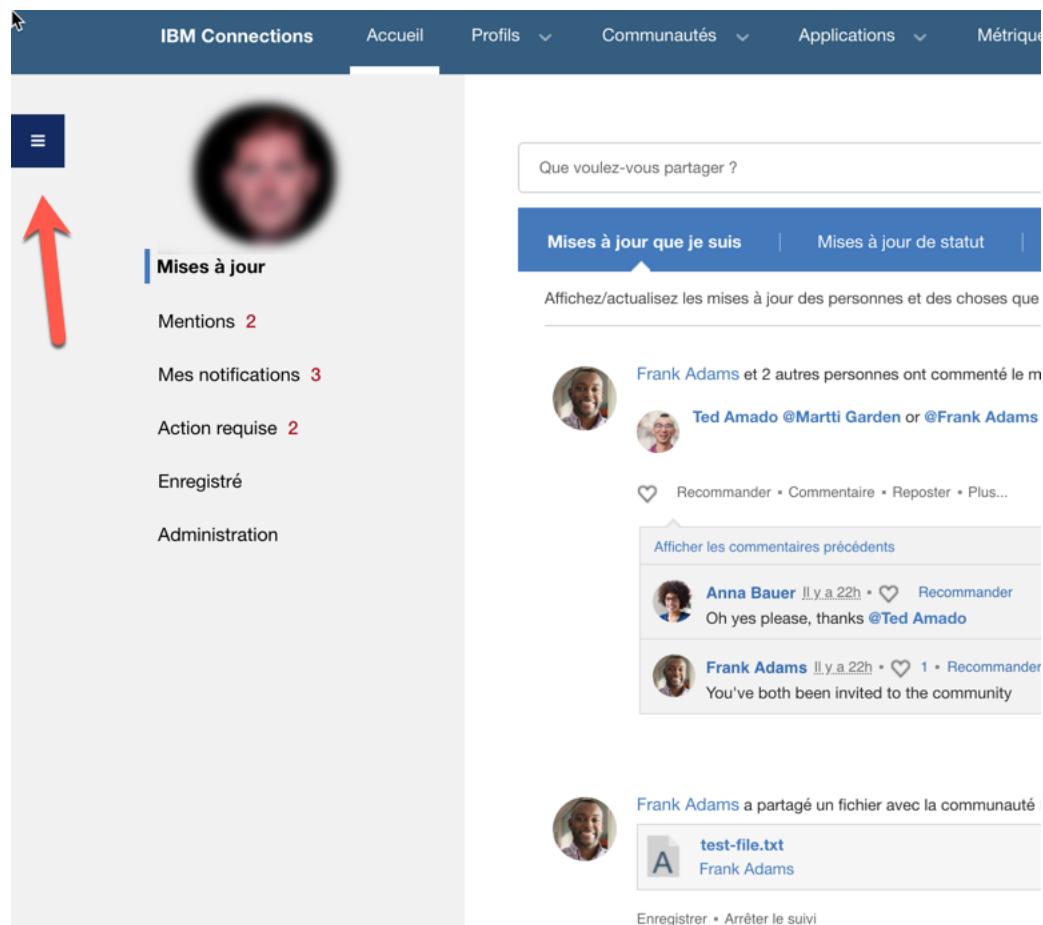
Lab 4 Using JQuery

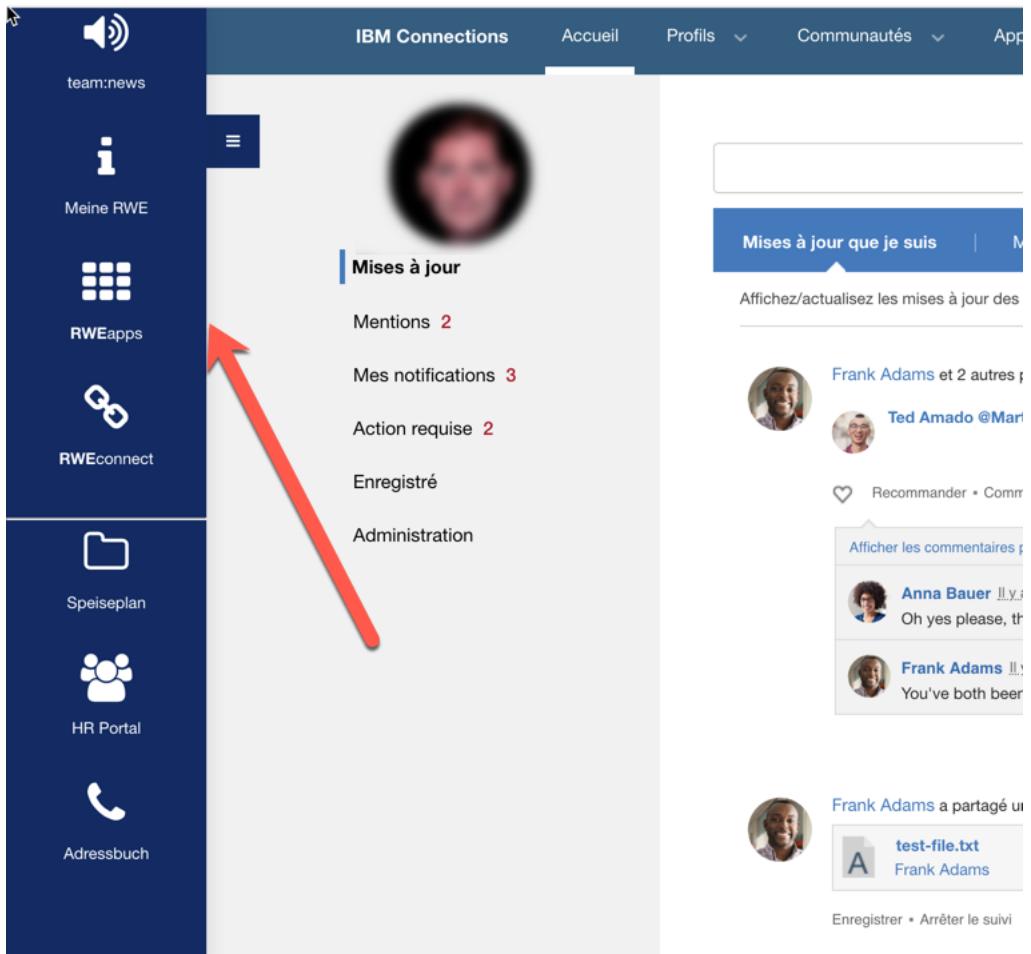
During the initial [Lab1](#) and [Lab2](#), we have started to see the use of the **Dojo Toolkit** to develop dynamic scripts.

Some developers are more familiar with the **JQuery toolkit**. This exercise guides you through the use of a simple **JQuery** script used within IBM Connections Customizer.

What we want to achieve

We want to add a simple flyout menu to all IBM Connections page. The Flyout menu has an animation as shown in the following pictures:



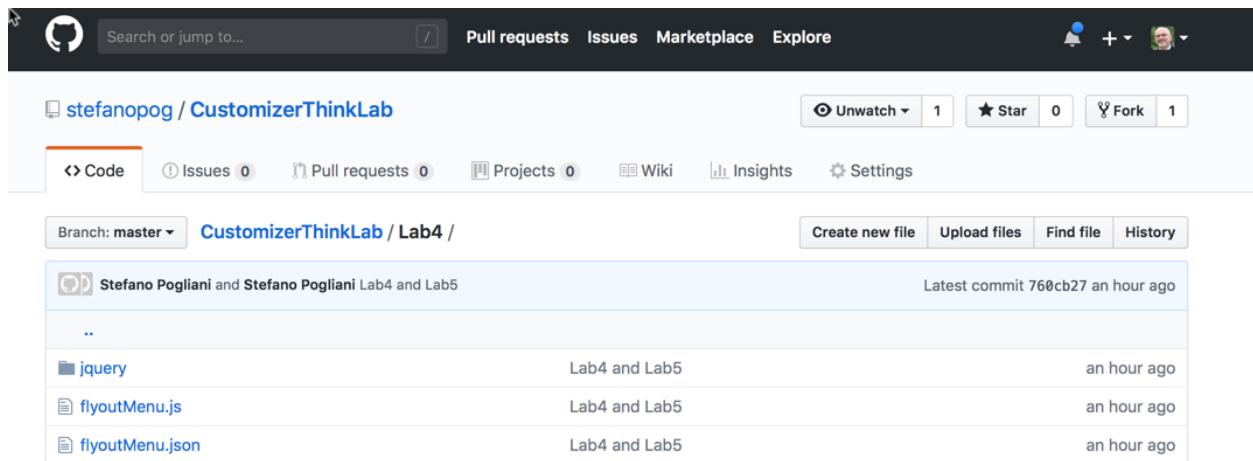


People familiar with the **JQuery** package are, probably, familiar with this kind of development.

The intent of this lab is not to describe in detail the process of developing interfaces using JQuery, but to show how a JQuery-based development can be easily re-used within IBM Connections customizer.

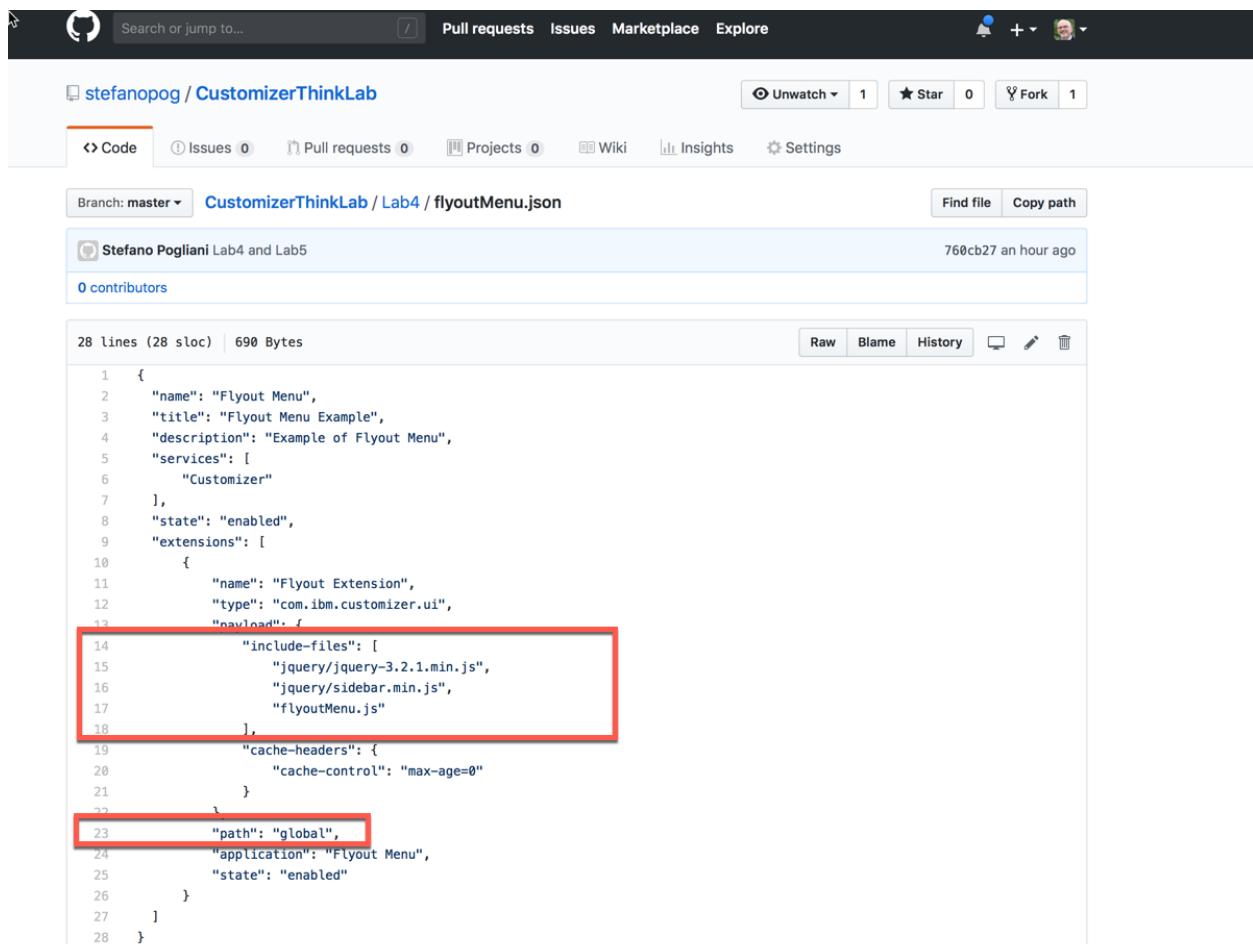
How is it done ?

All the code is available in the [Lab4](#) exercise.



The screenshot shows a GitHub repository page for [stefanopog / CustomizerThinkLab](#). The repository has 1 star, 0 forks, and 1 unwatched pull request. The master branch is selected. In the [Lab4](#) folder, there are four files listed: `jquery`, `flyoutMenu.js`, and `flyoutMenu.json`, all updated an hour ago. The `flyoutMenu.json` file is highlighted.

Let's look at the JSON file first, [flyoutMenu.json](#), this time:



The screenshot shows the `flyoutMenu.json` file content on GitHub. The file contains 28 lines of JSON. A red box highlights the `include-files` section, which lists three files: `jquery/jquery-3.2.1.min.js`, `jquery/sidebar.min.js`, and `flyoutMenu.js`. Another red box highlights the `path` field in the final object, which is set to `"global"`.

```
1  {
2    "name": "Flyout Menu",
3    "title": "Flyout Menu Example",
4    "description": "Example of Flyout Menu",
5    "services": [
6      "Customizer"
7    ],
8    "state": "enabled",
9    "extensions": [
10      {
11        "name": "Flyout Extension",
12        "type": "com.ibm.customizer.ui",
13        "payload": {
14          "include-files": [
15            "jquery/jquery-3.2.1.min.js",
16            "jquery/sidebar.min.js",
17            "flyoutMenu.js"
18          ],
19          "cache-headers": {
20            "cache-control": "max-age=0"
21          }
22        },
23        "path": "global",
24        "application": "Flyout Menu",
25        "state": "enabled"
26      }
27    ]
28 }
```

The important sections are highlighted in the screenshot above and explained here:

- The “`include-files`” section.
It references 3 files. Two of them come from the “`jquery subdirectory`” and the third one is the “`flyoutMenu.js`” that we will analyze later and that contains the code for our extension.
The two scripts contain the runtime of the “**JQuery package**” and of the additional “**Semantic Sidebar**” package.
We upload them in this way as it is very simple.

We will discuss during the class how to eventually load them differently.

- The “`path: ‘global’`” section.
We want here to make the script available on **all Connections pages**.

Now, let’s have a look at the [flyoutMenu.js](#) file, which actually contains the code that makes the magic for us.

- The first thing to notice is that we introduce the script with a function that is declared and executed on the fly.
This function immediately inherits from JQuery (and we see the familiar “`$` notation” used immediately). This means that JQuery is immediately active.
- The lines from 4 to 7 accomplish a basic function: they load the CSS scripts associated to the two packages we previously uploaded.
This could be done statically using the same principle used for the javascript packages

```
1
2
3
4  var $fontAwesomeCSS = $('<link>', {type: 'text/css', rel: 'stylesheet', href: 'https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css'};
5  var $semanticSideBarCSS = $('<link>', {type: 'text/css', rel: 'stylesheet', href: 'https://cdn.jsdelivr.net/npm/semantic-ui@2.4.1/dist/semantic.min.css'};
6  $('head').append($fontAwesomeCSS);
7  $('head').append($semanticSideBarCSS);
8
```

- From that point on, this is mere JQuery programming.

Congratulations. You have finished exercise 4.

Lab 5 Using a Framework

In the previous exercises, we have seen the use of some controls related to the packages that can be used to create your IBM Connections Customizer scripts.

There are also situations in which an IBM Connections page is not rendered statically; in those cases, you may not be able to find the elements you require right away.

There are other situations in which you have to simulate a mouse action in order to, for instance, change the default value/status of a given element; in most situations, changing the visual representation is not enough as you may miss some “side-effects” that properly executing the “mouse action” would apply.

These are some of the reasons why you may benefit from a “framework”. By “framework” here we refer to a collection of tools that would allow you to simplify development and to keep it consistent.

What we want to achieve

This module will introduce you a minimalist set of tools that you can use in writing your IBM Customizer extensions.

Please refer to this link ([Lab5](#)) and to the [commonTools.js](#) file.

The screenshot shows a GitHub repository page for 'stefanopog / CustomizerThinkLab'. The repository has 1 pull request, 0 issues, 0 projects, 0 wiki pages, and 0 insights. It has 1 star, 1 fork, and 0 contributors. The file 'commonTools.js' is displayed, showing 400 lines (389 sloc) and 20.1 KB. The code is a JavaScript file with a copyright notice and some configuration logic for a logger.

```
1  /*
2   * © Copyright IBM Corp. 2017
3   *
4   * Licensed under the Apache License, Version 2.0 (the "License");
5   * you may not use this file except in compliance with the License.
6   * You may obtain a copy of the License at:
7   *
8   * http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in writing, software
11  * distributed under the License is distributed on an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
13  * implied. See the License for the specific language governing
14  * permissions and limitations under the License.
15  */
16  if (__cBill_debug !== undefined) {
17    __cBill_logger('***** commonTools.js already done. SKIPPING *****');
18  } else {
19    //
20    // __cBill_debug needs to be declared NOW otherwise the following "__cBill_logger" statement will not work properly
21    //
22    var __cBill_debug = true;
23    //
24    // __cBill_hideNoDestroy is a configuration that determines if the DOM elements needs to be hidden or destroyed from the UI
25    // It is used to show the possibilities and to provide a late decision point
26    //
27    var __cBill_hideNoDestroy = false;
28    //
29    // Starting ....
30    //
31    __cBill_logger('***** commonTools.js executed *****');
32    //
33    // These are global functions and classes
34
```

The instructor will guide you through the tools included in the framework.

Lab 6 Changing the Behavior of IBM Connections

So far we have been doing very simple labs that, mainly, focused on changing the look and feel of some elements of the IBM Connections user interface.

In this lab, we will be dealing with something more complex: we want to change some default behavior of the interface and we want to provide a different user experience.

What we want to achieve

3.2 Communities - members

Members
View the members in your community, as well as people who are invited but have not yet joined.

Members Invitations

Invite Members Import Members Export Members Find a Member

Add members to automatically include them in the community. Invite members to give them the opportunity to join.

1 - 2 of 2 (2 people)

Sort by: Name Date Added

Marta Alvarez de la Campa Member

Show: 16 32 48 100

Feed for these Members

When mouse over on a person's name:

Stephanie Shoulder

Marketing & Communication Assistant
Office: Eurapco
+41 44 2879599
stephanie.shoulder@eurapco.com

Send Email More Actions Invite to my network

By default show less (delete top menu and two buttons below but KEEP a button with "Invite to my network").

Replace "Send Email" and "More Actions" by simply: "Invite to my network".

Make the email address clickable

Replace "Send Email" and "More Actions" by simply: "Invite to my network".

MARTA's Community

Edit Community

Community Profile **Files** Status Updates Forums Blogs Wiki Gallery

You must save changes on each tab before moving to another tab.

Edit Files Settings

Show by default:

Folders
 Files

Members have the role of:

Editor: (Share, upload, and edit community files and folders)
 Reader: (Share, view files and folders in this community)

Save Save and Close Cancel

The challenges are interesting:

1. We need to change the behavior of the Business Card “in the context of a Connections application, in our case Communities”

The challenge here are the following:

- a. to understand that Business Cards are dynamically generated: so, we need to apply our changes once a given Business Card is generated
 - b. to understand that the default behavior of Connections says that once a user changes some behavior about a Business Card she is viewing, that behavior gets changed for all other Business Cards
 - c. to understand that, sometimes, IBM Connections does not refresh the page (i.e. your Customizer script is not invoked) even if the page visually changes.
2. We need to change the default associated to the Business Card
The challenge here is that the only way to ensure that an “opaque behavior” is honored is, actually, to follow the way an interactive user executes it.
 3. We need to change the default applied to a given community.
The challenge here is catch which panel the user is editing.

We Value Your Feedback!

- Don't forget to submit your Think 2018 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Access the Think 2018 agenda tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.