



Summer 2023

Build Your Own Analytical App

A Short Course on Tableau Embedded Analytics



Alex Cortez

PRINCIPAL SOLUTION ENGINEER - TABLEAU

Tim Payne

PRINCIPAL SOLUTION ENGINEER - TABLEAU

Course Agenda

Getting Started	2
Initial Setup	3
Prep for the tutorial	3
Test your prep	3
Lesson 1: Simple Embedding	4
Get familiar with your TC app files	4
Get familiar with TC application	4
Complete Exercise 1	6
Lesson 2: Connected Apps	7
Create a Connected App	7
Configure the JWT	8
Verify CA works	9
Complete Exercise 2	10
Lesson 3: Advanced Embedding API v3	11
Complete Exercise 3.1 - Viz Switching	11
Complete Exercise 3.2.1 - Static Filters	13
Complete Exercise 3.2.2 - Dynamic Filters	15
Complete Exercise 3.3 - Mark Selections	17
Complete Exercise 3.4 - Event Listeners & Get Marks	19
Lesson 4: User Attribute Functions	21
Load Dashboard without filtering	21
Configure User Attribute Function in Workbook	21
Pass User Attributes into JWT	21
Test Dashboard with filtering	22
Lesson 5: Complete Solution	24
Solution Code Snippets	25

Getting Started

Welcome to this course on Tableau Embedded Analytics!

You'll learn how to leverage Tableau features to embed AND integrate analytics into external applications.

Specifically, this course will walk through:

- Leverage Tableau's Embedding API v3 to interact with and embed visualizations & dashboards into a web application
- Configure a Connected App to enable single-sign-on authentication for external users
- Configure User Attribute Functions to pass in data authorizations at runtime

Plan on following along the step-by-step instructions. And we encourage you to attempt the exercises to get a well-rounded understanding of existing Embedded Analytics capabilities.

Each lesson will cover a broad topic with exercise(s) to follow. Solution code snippets are provided at the end of this course document to check your work.

Click on an Agenda item to navigate to that section.

Helpful links for additional reading & to consult during exercises:

- [Embedding API v3 docs](#)
- [Connected Apps docs](#)
- [User Attribute Functions docs](#)
- [Tableau Embedding Playbook](#)

Initial Setup

The three places you will spend your time in during this course: (1) An IDE for code development, (2) the TC Web Portal for app testing, and (3) your Tableau Cloud Site for tableau configuration. There are several pieces you need to install/configure first before we can run through the course.

Prep for the tutorial

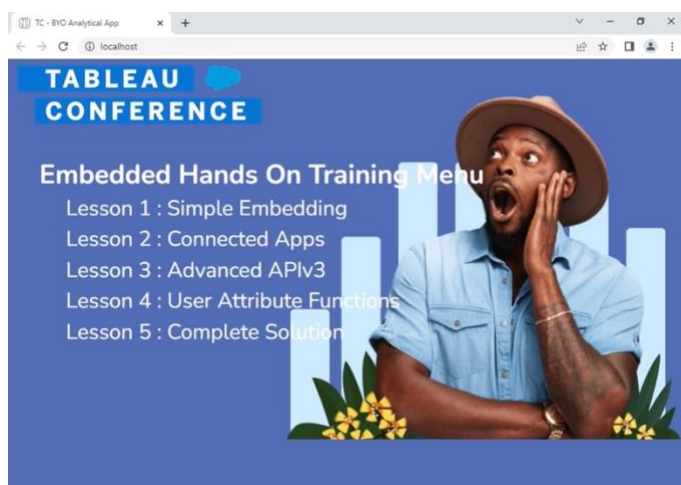
To prepare for the rest of the course, complete the following:

- Install Python
- `pip install -r requirements.txt`
- Install Visual Studio Code (or any other IDE you're familiar with) and open this repository
- Launch an incognito browser and open a Tableau Cloud Site where you are an admin (you can get a new developer Site by joining the [Tableau Developer Program](#))
- Create a top-level Project in Tableau Cloud called "hotembed"
- Rename your local workbooks in the resources folder: `workbook_userxxx` to `workbook_hotembed` & `redirect_userxxx` to `redirect_hotembed`
- Upload `workbook_hotembed.twbx` into Project `hotembed`
- Upload `redirect_hotembed.twbx` into Project `hotembed`
- Create a new Viewer user, this can be any email you want to use, just remember it for logging in with Connected Apps later → Assign this new user viewing permissions for the content above
- Switch to `EmbeddedPortal.py` in your IDE and change the following variable: `hotUser = "hotembed"` in line 16
- Set the URL of your Tableau Cloud site and Site Name in lines 27 and 28
- Launch a terminal window and run `EmbedPortal.py` (The exact command may depend on which terminal is used)

Test your prep

Now, let's test your work:

- Launch a standard browser and point to "localhost"
- If successful, you should see the TC app menu screen with links to different lessons!!!



Lesson 1: Simple Embedding

In the previous section, you published content to your Tableau Cloud Site and you started running the TC application locally. Let's now get more familiar with this application and the files behind it.

Get familiar with your TC app files

Your IDE is where our application code is stored and where we'll manipulate the code in the exercises.

Switch to your IDE. Open the static/js & templates folders so you can see the relevant HTML & JavaScript files on the left-hand side. They are numbered to coincide with the lesson/exercise we are on.

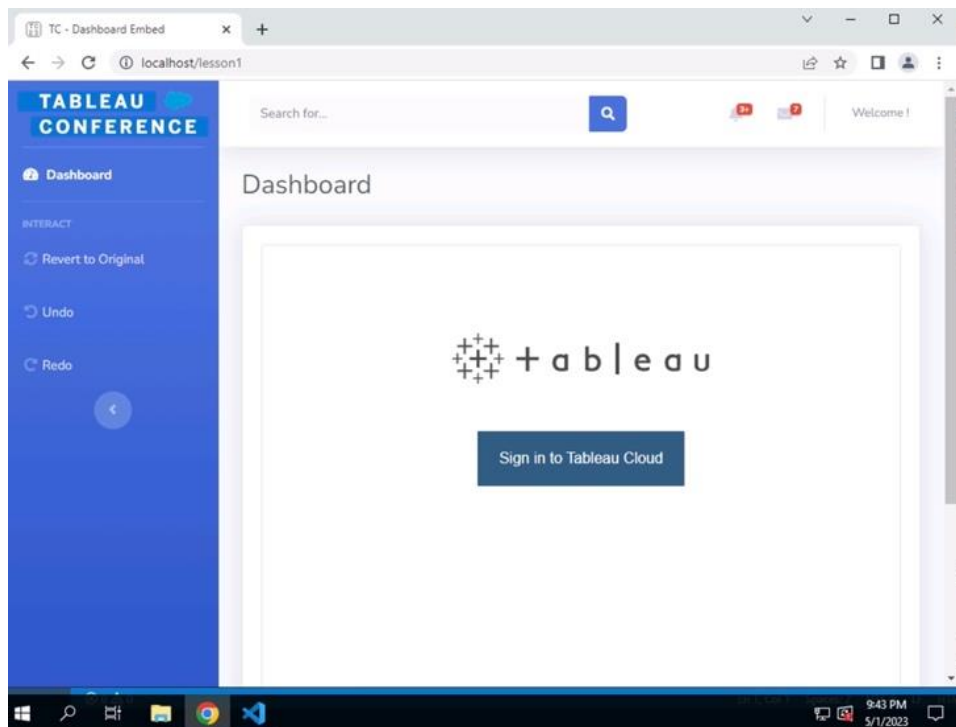
The web application we are using today to embed Tableau analytics into was made with Flask. This app (*EmbedPortal.py*) is already running on your machine, which you can confirm by looking at your terminal.

In the terminal, you'll see something like:

```
127.0.0.1 - - [timestamp] "GET /static/..." -
```

Get familiar with TC application

Switch to your TC application and click on *Lesson 1: Simple Embedding*



You should now be in the app and see a Tableau Sign-in page embedded in the middle. This is what we call a 'Simple Embed' where there's no Single Sign-On component to this. Upon entering the TC application, the user was not signed into Tableau yet.

Go ahead and click on the "Sign into Tableau Cloud" button and use your admin credentials to sign into your Site.

By signing in, you now have a session with Tableau and can interact with this visualization and others too. If you're curious, check your cookies in the browser and look for a *workgroup_session_id*. In Lesson 2, we'll learn how to do this sign-in behind the scenes for the end user, so it feels seamless.

Now, notice the toolbar at the top of the dashboard and all the buttons it includes. Just below the toolbar, use the REGION filter dropdown to switch the region twice (for ex, West & then East). Notice how each time we select a region, the entire dashboard filters to that region too. Then, use the first three buttons on the toolbar to Undo, Redo, and then Reset view. Notice the buttons may be grayed out until you've made a selection.

These are useful actions a user may want to take on the dashboard, but aside from the first three buttons on the toolbar, we don't need the rest of these. Let's learn how to remove the toolbar and wire up our custom html buttons on the left-hand side to keep the functionality we do want.

Complete Exercise 1

For the first exercise, we're going to have you:

- Hide the Tableau toolbar from the dashboard
- Wire up the Revert to Original, Undo, and Redo buttons on the left-hand side

Switch to your IDE and look at the *lesson1.html* and *embed1.js* files

Lines of code to note in *lesson1.html*:

- Link to the Embedding API library (Ctrl+F for 'module')
- 3 html buttons that will interact with the tableau dashboard (Ctrl+F for 'onclick')
- Tableau viz web component with configuration parameters (Ctrl+F for 'tableau-viz')

To remove the toolbar, [look at this doc](#) and make the relevant change to *lesson1.html*. (Ctrl+F for 'toolbar' or 'hidden')

To wire up the Revert to Original, Undo, and Redo buttons, first you need to create the Viz object by grabbing the *tableauViz* div from the html document.

Then, [look at the Viz object methods here](#) and make the relevant changes to *embed1.js*. (Ctrl+F for 'redoAsync()' & 'undoAsync()' & 'revertAllAsync()')

And don't forget those methods must be used on a Viz object!

When you're done making your changes to *lesson1.html* & *embed1.js*, save your changes and refresh your TC application in the browser. Check that the toolbar is hidden. And switch between a couple of regions in the dropdown filter on the dashboard so that you can then test the custom buttons left of the dashboard.

If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Lesson 2: Connected Apps

In the previous section, you demonstrated a simple embed without Single-Sign-On. Let's now pivot to setting up SSO for seamless authentication for external users. We will do this via Connected Apps.

Create a Connected App

Connected Apps (CA) is an authentication mechanism that allows you to establish a session for an end user in an embedded workflow. Check the attached slide at the end of this document to understand the auth flow; essentially, your application (or Identity Provider) will create a JSON Web Token (JWT) for Tableau, pass in the correct claims including the username of the tableau user & the appropriate scopes, and your app will sign the JWT to establish trust between itself (or IDP) and Tableau Cloud. Check the link in the Getting Started section for more details.

Log out of the TC app (Click *Welcome!* in top right corner >> *Logout*), then click on *Lesson 2: Connected Apps*

Notice that your TC app presents you with a sign-in page for this exercise. When a user signs into the TC app, they should not have to sign in anywhere again, including Tableau. To set this up, we first need to create and configure a Connected App in Tableau Cloud.

Switch to your incognito browser and make sure you're logged into Tableau Cloud with your admin credentials.

We're looking for your external-user account (the Viewer you created in the Initial Setup section). Go to your Tableau Homepage and click on the Users tab. Find your external-user and make note of their username.

Now onto configuring a Connected App: Click on the Settings tab >> Click on the Connected Apps tab

Click New Connected App >> Click Direct Trust

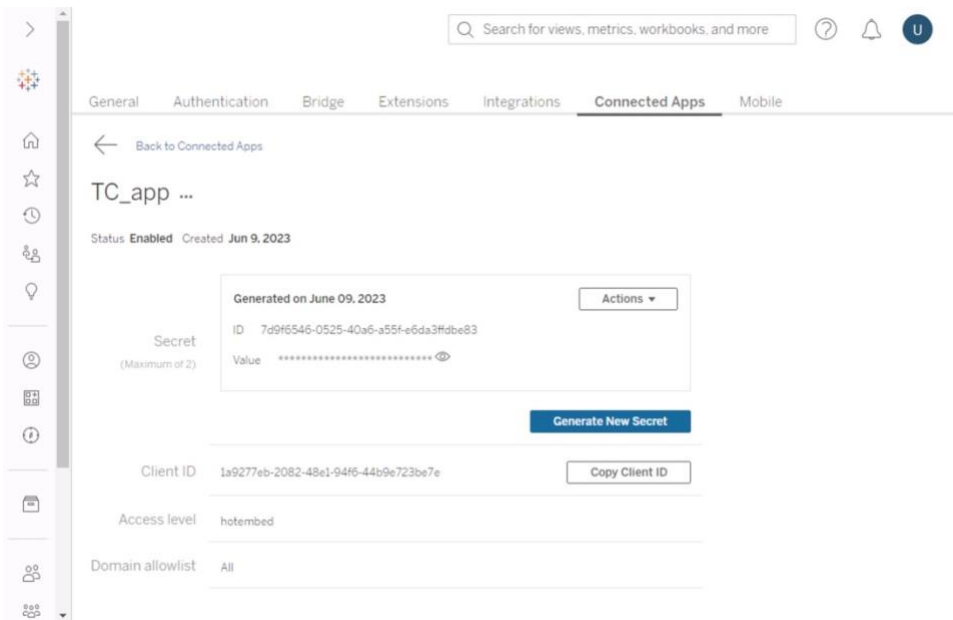
Choose the following settings:

- Connected app name = *TC_app*
- Access level: Applies to = *Only one project*
- Access level: Project name = *hotembed*
- Domain allowlist: *All domains*

Click Create >> Click Generate New Secret

Click 3 dots next to Connected App Name >> Click Enable

Below the name of the CA, you should see **Status Enabled**



Configure the JWT

After creating the CA in Tableau Cloud, we need to enable our app to send a valid JSON Web Token. JWT is a standard used to securely transfer information between two parties. The JWT is signed by our TC app to securely send info to Tableau Cloud.

In your IDE, open *EmbedPortal.py* >> Find the *getJWT()* function (Ctrl+F for 'getJWT')

There are several placeholder values in this function we need to replace with values we get from Tableau Cloud.


HINT: It's easy to mistakenly copy & paste the wrong values into the wrong place during this step. Proceed carefully!

Copy *Client ID* from Tableau Cloud CA configuration page >> Replace *connectedAppClientID* value (2 times)


Copy *Secret ID* >> Replace *connectedAppSecretID* value (1 time)

Copy *Secret Value* >> Replace *connectedAppSecretKey* (above Algorithm)

Double-check that you have copied and pasted the right values into the right place. Then, save your changes to the *EmbedPortal.py* file. You should see the terminal update.



DebuggerLibrariesIntroductionAsk

Crafted by

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsImIzcyI6IjFhOTI3N2VlTiIwODItNDhmMS05NGY2LTQ0YjllNmZIZmY0ZS3ZImltpZCI6IjdkOWY2NTQ2LTA1MjUtNDZhNi1hNTVmLUWU2ZGEzMkYmUmYyIsInR5cCI6IkpXVCJ9.eyJpc3MiOiIxYTkyNzdlY10yMDgyLTQ4ZTctOTRmMT08NGE1STcyMjJlN2UiLCJleHAiOjEyE2ODYzMjkxNTYsImp0eSI6IjkwMDIzODBhLTRhOTAtNDlmNSIiwuWU2LTK3ZWJkZGQyNDgwNCIsImF1ZC6InRhYmxlYXU6IjZldWIiOiJlc2VyME42Xzh0QG15dGMymy5jb20iLCJzY3AiOiJsIdGFiGbvHdpT2aWV3czplbWJlZCIsInR5bmxiYXU6bWV0cmllZjZlbnWJlZCjdGf.AWnrJ87oIAYGdf-k6lAatJkJnBjaxQEKnDu265PJg
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "iss": "1a9277eb-2082-48e1-94f6-44b9e723be7e",  
  "kid": "7d9f6546-9525-40a6-a55f-e6da3ffdb83",  
  "typ": "JWT"  
}
```

PAYLOAD: DATA

```
{  
  "iss": "1a9277eb-2082-48e1-94f6-44b9e723be7e",  
  "exp": 1686329156,  
  "iat": "9002380a-4a90-49f5-b9e3-97ebddd24804",  
  "aud": "tableau",  
  "sub": "user218ext@mytc23.com",  
  "scp": [  
    "tableau:views:embed",  
    "tableau:metrics:embed"  
  ]  
}
```

VERIFY SIGNATURE

Complete Exercise 2

For this exercise, you are going to:

- Switch out the dashboard that is embedded for a different dashboard in a different Tableau Project
- Configure the CA to work with the new dashboard

Switch to your IDE and take note of the dashboard URL path in *lesson1.html* (Ctrl+F for 'MyTCDash')

In your incognito window, switch to Tableau Cloud and navigate to a different dashboard in a different Project

Look for a similar URL path you saw in *lesson1.html* and replace the default path in your IDE with the new URL pattern >> Save

You've just switched out the original dashboard URL for a different dashboard URL.

You can test your changes by logging out of the TC app, clicking on Lesson 2, and going through the login process again.

You'll notice that the JWT seems to be created successfully, but the dashboard won't load. Why? Is your JWT configured correctly? (Hint: Tweak your CA configuration and then refresh the app to test)

If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Lesson 3: Advanced Embedding API v3

In the previous section, you configured SSO for a simple embed. Let's now pivot to some more advanced features of the Embedding API. We will cover switching between different vizs, static & dynamic filters, selecting marks, detecting events, and getting data.

Complete Exercise 3.1- Viz Switching

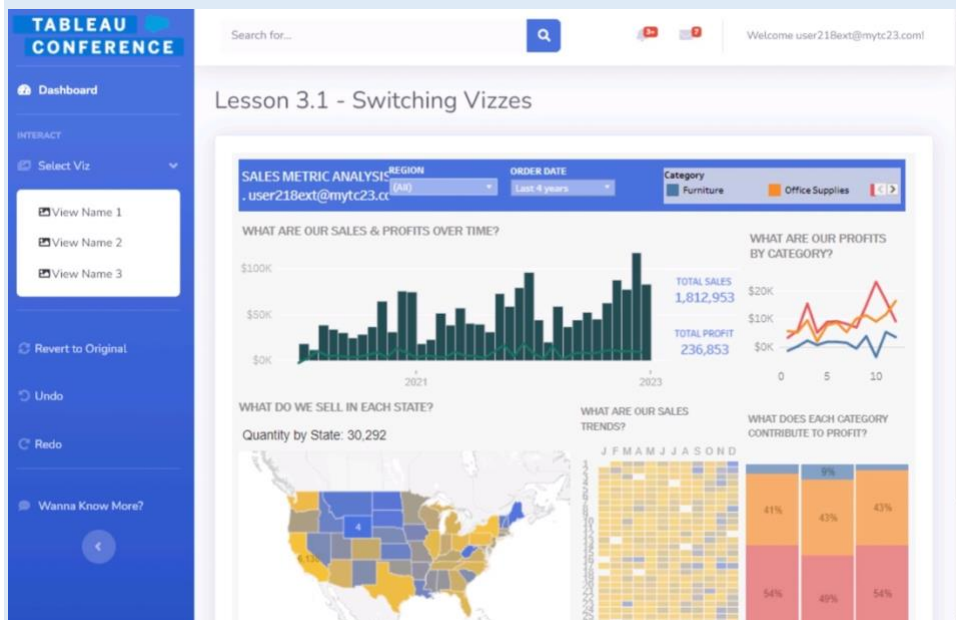
For this exercise, you are going to:

- Configure html buttons to switch between 3 different vizs

To get started:

- Switch to your TC application >> Logout and click on *Lesson 3: Advanced APIv3*
- Login with external-user Username >> Click *Lesson 3_1: Multi Viz* link

Expand the Select Viz button on the left-hand side. This reveals three individual buttons that should each load a different viz. In this exercise, you will configure these buttons to work.



Here are the steps you need to take:

In your IDE, close the tabs from the previous exercise >> Open *lesson3_1.html*

Identify buttons to configure (Ctrl+F for 'onclick')

Notice there are two things you need to add:

When clicked, the *change_viz* function is empty right now. You need to add the correct Tableau Cloud viz URL to the function

If you scroll to the right, the View Name within the span element needs to be updated to the correct viz name

Identify two additional vizzes on your Cloud Site to use for this exercise >> Make changes to three buttons above >> Save your changes

Switch to *embed3_1.js* file >> Declare the viz object and set it to the document element with id 'tableauViz' (look at *embed1.js* for example)

Configure the *change_viz(url)* function to set the viz source property to url of the new viz (the url passed into the function)

Save your changes and reload TC app in your non-incognito browser >> Test out buttons to switch vizzes

If you've done every step in the manual thus far, the buttons should work. If you didn't complete exercise 2, you may notice that your other dashboards do not load. If this happens to you, perhaps you need to check the scopes on your CA in Tableau Cloud (hint: this is the same issue encountered in exercise 2 and was resolved by changing the CA project scope)

If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Complete Exercise 3.2.1- Static Filters

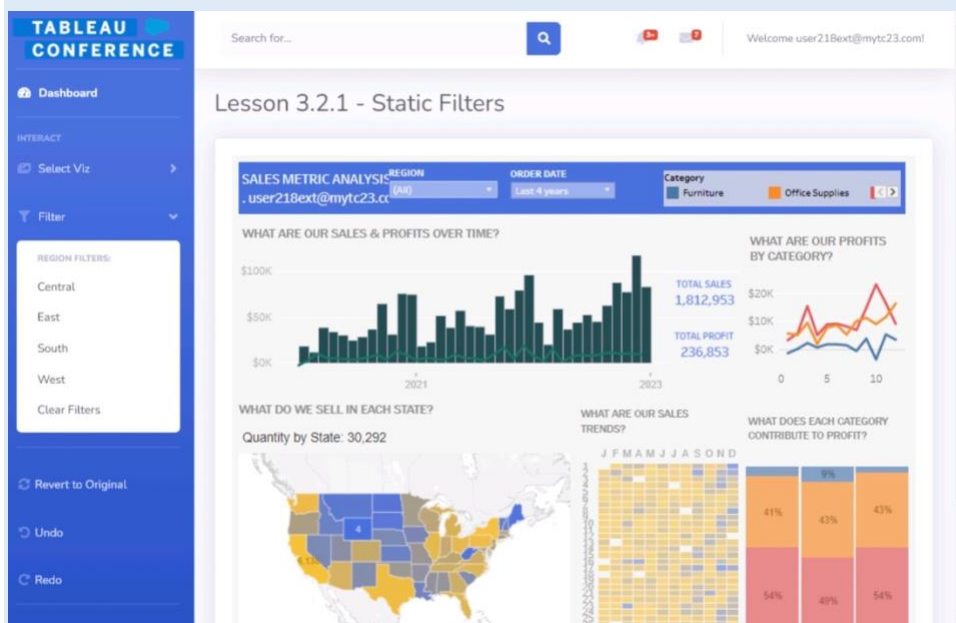
For this exercise, you are going to:

- Review the html for a static categorical filter
- Call the right js methods to apply filter behavior

To get started:

- Switch to your TC application >> Logout and click on *Lesson 3: Advanced API v3*
- Login with external-user Username >> Click *Lesson 3_2_1: Static Filters* link

Expand the Filter button on the left-hand side. This reveals five individual buttons that should filter (or clear) the embedded dashboard. In this exercise, you will configure these buttons to work.



Here are the steps you need to take:

In your IDE, close the tabs from the previous exercise >> Open lesson3_2_1.html

Identify buttons to configure (Ctrl+F for 'filterTableau')

Notice the two custom functions *filterTableau* and *filterClear*. Both functions are already configured; take note of what is passed into both functions. In *filterTableau*'s case, it's the name of the field and the categorical values to filter out. In *filterClear*'s case, it's the name of the field to clear.

Switch to *embed3_2_1.js* file >> Scroll down to the for-loop to find code that loops through a dashboard's individual sheets to find the sheet to filter on.

Add the name of the sheet we want to filter on ('Sales Map') to the case statement

Scroll down to the *filterTableau* function and add code to call the *applyFilterAsync()* method on the *activeFilterSheet* object [as shown here](#).

Scroll down to the *filterClear* function and add code to call the *clearFilterAsync()* method on the *activeFilterSheet* object [as shown here](#).

Save your changes and reload the TC app in your non-incognito browser >> Test out filter buttons

If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Complete Exercise 3.2.2- Dynamic Filters

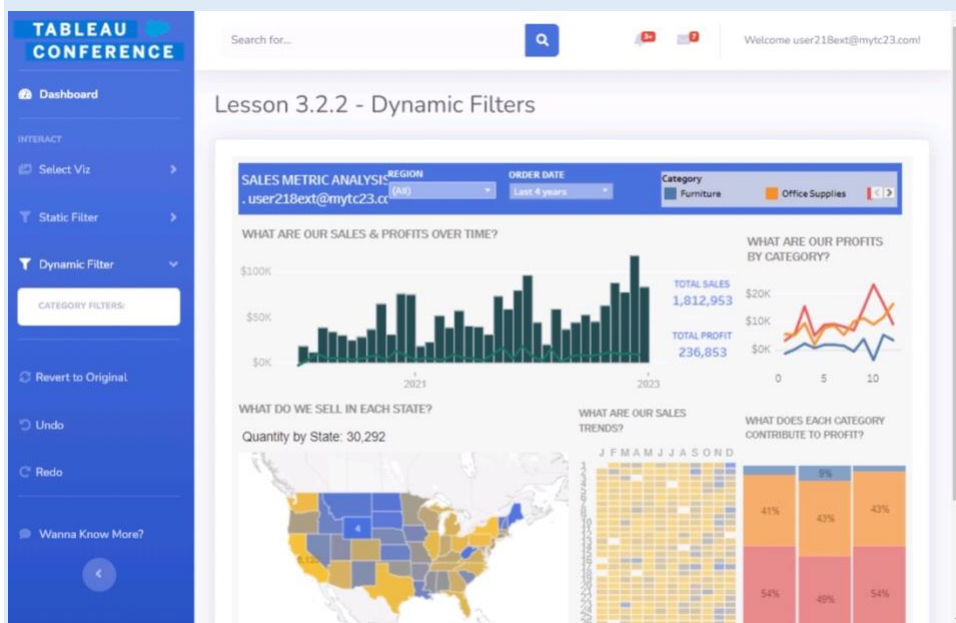
For this exercise, you are going to:

- Review the html for a dynamic categorical filter
- Call the right js methods to apply filter behavior

To get started:

- Switch to your TC application >> Logout and click on *Lesson 3: Advanced API v3*
- Login with external-user Username >> Click *Lesson 3_2_2: Dynamic Filters* link

Expand the Dynamic Filter button on the left-hand side. Right now, the button is empty. In this exercise, you will configure it to work.



Here are the steps you need to take:

Switch to Tableau Cloud in your incognito browser window >> Open the Sales Map sheet and click the Edit button at the top.

Once in edit mode, look in the top left corner to identify the other categorical filter applied to this sheet besides Region. Make note of the name of this filter. We will use it in a later step >> Hit the X to close out web edit mode.

In your IDE, close the tabs from the previous exercise >> Open *lesson3_2_2.html*

Identify button to configure (Ctrl+F for 'dynFilter')

Notice that we have carved out space where we will dynamically populate the html with filter values. We don't have to add anything here for now.

Switch to *embed3_2_2.js* >> Scroll down to the for-loop to find code that loops through a dashboard's individual sheets to find the *Sales Map* sheet and get all the filters applied to this sheet.

Scroll down to the *getFilters* function and add the name of the remaining filter to the blank if-statement.

Review *domain* function to see how we can loop through the array of filter values and input into our html file.

Save your changes and reload the TC app in your non-incognito browser >> Confirm dynamic filter list is populated and buttons work

If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Complete Exercise 3.3- Mark Selections

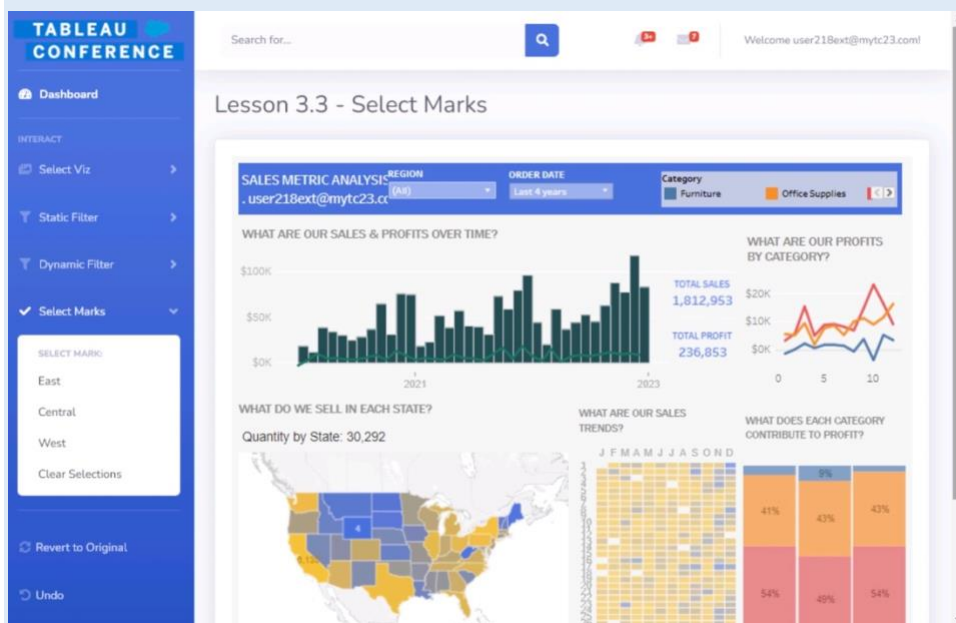
For this exercise, you are going to:

- Configure 3 html buttons to select specific states
- Call the right js method to apply selection

To get started:

- Switch to your TC application >> Logout and click on *Lesson 3: Advanced API v3*
- Login with external-user Username >> Click *Lesson 3_3: Select Marks* link

Expand the Select Marks button on the left-hand side. It expands to 4 individual buttons. In this exercise, you will configure them to work.



Here are the steps you need to take:

In your IDE, close the tabs from the previous exercise >> Open *lesson3_3.html*

Identify buttons to configure (Ctrl+F for 'selectTableau')

For the East, Central, and West buttons, add a couple states to each array >> Save your changes

Switch to *embed3_3.js* >> Identify sheet we will be selecting marks on (Ctrl+F for '3_3')

Find *SelectTableau* function (Ctrl+F for '3_3')

Call on relevant method to select marks. Find a [useful example here](#). Pay close attention to the syntax.

Also note the *selectClear()* function and its relevant method.

Save your changes and reload TC app in non-incognito browser >> Test buttons to select different states by region

If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Complete Exercise 3.4- Event Listeners & Get Marks

For this exercise, you are going to:

- Wire up an event listener for mark selections
- Grab mark selection info to dynamically open a Wikipedia page

To get started:

- Switch to your TC application >> Logout and click on *Lesson 3: Advanced API v3*
- Login with external-user Username >> Click *Lesson 3_4: Get Marks* link

At the moment, nothing new happens when you select an individual state on the map. It just filters the data in the other charts.

Here are the steps you need to take:

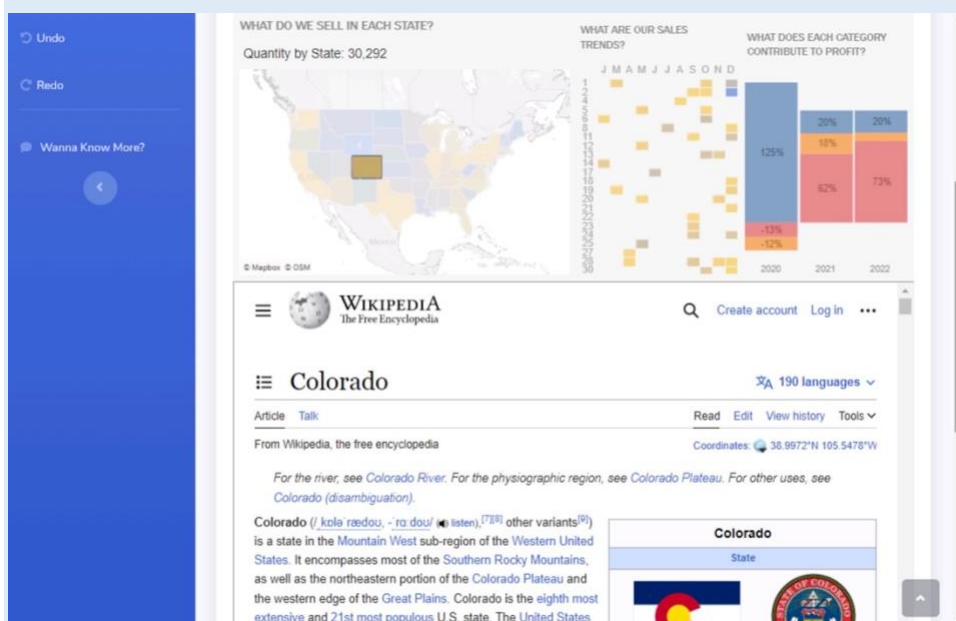
In your IDE, close the tabs from the previous exercise >> Open *lesson3_4.html*

Identify the empty div element to populate with selected mark (Ctrl+F for '3_4')

Switch to *embed3_4.js* >> Under Lesson 3_4 comments, add an event listener which listens for mark selections. Look at the existing event listener located right above it for a working example. (hint: instead of "firstinteractive", this event listener fires on "markselectionchanged")

Scroll to the *getSelectedMarks(marksEvent)* function and follow the flow of the code

Save your changes and reload TC app in non-incognito browser >> Click on a State and see what pops up underneath the dashboard



This exercise demonstrated the ability to add event listeners to our code so we can detect when certain actions take place on the dashboard, in this case a mark selection. Then, once that event has taken place, we were able to “get” those marks so we could do something else with them, in this case we opened a web object and passed the value of the state selected into a Wikipedia URL.

If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Lesson 4: User Attribute Functions

In the previous section, you tested some of the more advanced features of the Embedding API v3. Let's now pivot to using User Attribute Functions to secure the data by passing data authorizations into the JWT at runtime. This will ensure that users see the right rows of data in a multitenant data architecture.

Load Dashboard without filtering

To get started:

- Switch to your TC application >> Logout and click on *Lesson 4: User Attribute Functions*
- Login with external-user Username >> Click Load Your Dashboard with UAF!

Notice the dashboard currently loads all of the data (no filtering).

Configure User Attribute Function in Workbook

The first step is to add the UAF to the workbook itself, and specifically to the embedded data source. Think of this as the plumbing we need to do for the feature to work. It is the mechanism used to apply the filter values specified in the JWT.

Switch to Tableau Cloud in your incognito browser window >> Open *workbook_hotembed*

Click Edit (at the top) >> Switch to *Sales Map*

Click on Analysis (top menu) >> Create Calculated Field >> Name it "UAF Calc"

Enter the following: `USERATTRIBUTEINCLUDES('myuaf', [State])`

Click OK

Now that the calculation has been created, we need to apply it to the data source:

- Click on Data (top left) >> Superstore Data >> Edit Data Source Filters
- Click Add Filter >> UAF Calc >> Exclude selected values >> Select False checkbox >> OK >> OK

You have just secured the User Attribute logic by adding it as a filter to the embedded data source. The viz will now appear blank. **Click the blue Publish button** to apply your changes to the workbook.

Click Go to Workbook link (at the top) after publishing. Notice now there are user filters applied to the thumbnails now.

Pass User Attributes into JWT

Before we can start passing user attributes to our JWT, we need to configure the admin Tableau Cloud Site setting that enables the capture of user attributes during an auth flow.

In Tableau Cloud, click *Settings >> Authentication*

Under the *Control User Access in Authentication Workflows* heading, select the *Enable capture of user attributes in authentication workflows* checkbox

Ok the plumbing for row level security is done. We now just have to pass the user attribute values directly into a JWT claim correctly.

In your IDE, close the tabs from the previous exercise >> Open *EmbedPortal.py* (We don't need to make any edits to our html/js)

Find the *getJWT()* function (Ctrl+F for 'getJWT') >> Uncomment the "myuaf" claim

On the right-hand side of the colon, enter an array of states, for ex: *["Texas", "New York", "Utah"]*

Make sure there is still a comma at the end of the array (because there is still another claim that follows) >> Save *EmbedPortal.py*

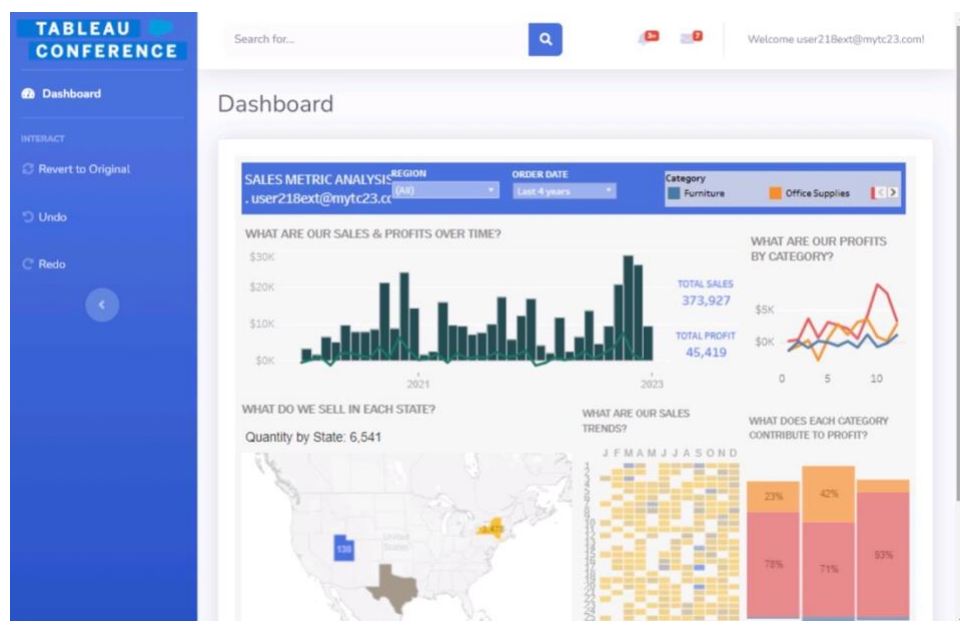
If you're completely stuck, you can see the Solution Code Snippets at the end of this manual.

Test Dashboard with filtering

To test:

- Switch to your TC application >> Logout and click on *Lesson 4: User Attribute Functions*
- Login with external-user Username >> Copy your JWT token for use in a moment
- Click Load Your Dashboard with UAF!

Notice the dashboard now filters the data to only the states you specified in the JWT claim.



Important: If you wish to try a different set of values, make the changes in the JWT, and then Logout of the TC app and log back in to Lesson 4 again.

As we did before, go to jwt.io in your browser >> Scroll down to the Debugger >> Remove the Encoded token on the left >> Paste your token into the text box

On the right, you'll see your decoded JWT which includes all the scopes & claims we passed into it. Notice our data attributes we passed into our JWT are now included here.

The screenshot shows the JWT.io Debugger interface. The top navigation bar includes links for Debugger, Libraries, Introduction, and Ask, along with the 'Crafted by auth0' logo. The interface is split into two main sections: 'Encoded' on the left and 'Decoded' on the right.

Encoded Section: A text area labeled 'PASTE A TOKEN HERE' contains a long, multi-line Base64-encoded JWT token.

Decoded Section: This section displays the decoded structure of the token, divided into three parts:

- HEADER: ALGORITHM & TOKEN TYPE:** A JSON object containing:
 - "alg": "HS256"
 - "iss": "1a9277eb-2082-48e1-94f6-44b9e723be7e"
 - "kid": "7d9f6546-0525-40a6-a55f-e6da3ffdb83"
 - "typ": "JWT"
- PAYLOAD: DATA:** A JSON object containing:
 - "iss": "1a9277eb-2082-48e1-94f6-44b9e723be7e"
 - "exp": 168635965
 - "jti": "e5ac2887-d25c-48dd-abb4-16a18c3547be"
 - "aud": "tableau"
 - "sub": "user218ext@mytc23.com"
 - "myself": {
 - "Texas"
 - "New York"
 - "Utah"
 - "scp": {
 - "tableau:views:embed"
 - "tableau:metrics:embed"
- VERIFY SIGNATURE:** A section at the bottom for verifying the token's signature.

Well done! You delivered SSO authentication & row level security together through the same mechanism (Connected Apps).

Lesson 5: Complete Solution

CONGRATULATIONS!

You've made it until the end. You've officially graduated from embedding bootcamp!



Now you get to try the full solution with all the integrations you worked on.

To test:

- Switch to your TC application >> Logout and click on *Lesson 5: Complete Solution*
- Login with external-user Username

Select a state and make sure the Wikipedia page loads

Run through all the buttons on the left and verify they work

Have a look at *complete.html* and *complete.js* for the entire working code

Which integration do you like the most? Which one do you find the most useful?

Thank you so much for completing this course on embedded analytics with Tableau.

We would appreciate it if you could provide honest feedback by filling out [this short google form](#).

This helps us understand what made this course valuable for you and also how we can improve it.

Thank you!

Solution Code Snippets

Please use this section to review snippets of the completed code. Be careful when you copy/paste bits of code into your own working code. It's very easy to omit a character or two!

Lesson 1

```
// Exercise 1 - Hide the toolbar - Notice that the variables wrapped in {{}} are pulled from the EmbedPortal.py file
```

```
<tableau-viz id="tableauViz"
  src= "{{tabServer}}/t/{{tabSite}}/views/{{tabWorkbook}}/MyTCDash"
  device="mobile" toolbar="hidden" hide-tabs width=100%>
</tableau-viz>
```

```
// Exercise 1 – JavaScript tweaks
```

```
// revertAll function
viz.revertAllAsync();
```

```
// set viz object
viz = document.getElementById('tableauViz');
```

```
// undo function
viz.undoAsync();
```

```
// set viz object
viz = document.getElementById('tableauViz');
```

```
// redo function
viz.redoAsync();
```

Lesson 2

// JWT example - Do not simply copy and paste the python code, it's just an example of what it should look like. Your values will differ.

```
CA_SSO_token = jwt.encode(
    {
        # Lesson 2
        # https://help.tableau.com/current/online/en-us/connected_apps.htm#step-3-configure-the-jwt
        # Set iss (Issuer) to Connected App Client ID
        "iss": '4063aef9-f1ef-4dac-87bc-06a05f21a64b',
        "exp": datetime.datetime.utcnow() + datetime.timedelta(minutes=10),
        "jti": str(uuid.uuid4()),
        "aud": "tableau",
        "sub": username,
        "scp": ["tableau:views:embed", "tableau:metrics:embed"]
    },
    # Set this value to the Connected App Secret Key
    'PBBgl0D9Pe0inqNf+nHy/K6Ytv8mP06bjkcoOtfx6Lk=',
    algorithm="HS256",
    headers={
        # Set kid (Key ID) to Connected App Secret ID
        'kid': 'df4b89a3-7761-4053-9e4f-77958e00e18f',
        # Set iss (Issuer) to Connected App Client ID
        'iss': '4063aef9-f1ef-4dac-87bc-06a05f21a64b'
    }
)
```

// Exercise 2

Look at your CA Scopes in Tableau Cloud.

Did you properly scope the CA to the right project(s)?

Lesson 3.1

```
// Exercise 3.1 – Keep in mind that your URLs within the change_viz function will be different
// depending on the Tableau Cloud pod your Site resides in

<a class="collapse-item" href="#" onclick="change_viz('https://eu-west-1a.online.tableau.com/t/hotembed/views/workbook_hotembed/MyTCDash');"><i class="fas fa-fw fa-image"></i><span>TC Dashboard</span></a>

<a class="collapse-item" href="#" onclick="change_viz('https://eu-west-1a.online.tableau.com/t/hotembed/views/workbook_hotembed/SalesMap');"><i class="fas fa-fw fa-image"></i><span>Sales Map</span></a>

<a class="collapse-item" href="#" onclick="change_viz('https://eu-west-1a.online.tableau.com/t/hotembed/views/{your-workbook-name}/{your-sheet-name}');"><i class="fas fa-fw fa-image"></i><span>{your-dashboard-name}</span></a>
```

```
// Exercise 3.1 – JavaScript tweaks

// window.onload function
viz = document.getElementById('tableauViz');

// change_viz function
viz.src = url;
```

Lesson 3.2.1

```
// Exercise 3.2.1 - JavaScript tweaks

// case statement
case 'Sales Map':



---



// filterTableau function
activeFilterSheet.applyFilterAsync(filterName, value, action);



---



// filterClear function
activeFilterSheet.clearFilterAsync(filterName);
```

Lesson 3.2.2

```
// Exercise 3.2.2 - JavaScript tweaks

// if statement
if (filter.fieldName == "Category")
```

Lesson 3.3

```
// Exercise 3.3 - Choose the States you wish to select

<!-- Example, but you can use any States you like -->
<a class="collapse-item" href="#"
onclick="selectTableau('State',['Virginia','Vermont'])">East</a>
<a class="collapse-item" href="#"
onclick="selectTableau('State',['Wisconsin','Texas'])">Central</a>
<a class="collapse-item" href="#"
onclick="selectTableau('State',['Washington','Utah'])">West</a>
```

```
// Exercise 3.3 - JavaScript tweaks

// function selectTableau()
activeSelectSheet.selectMarksByValueAsync([
  {
    fieldName: fieldName,
    value: value
  }, action );
```

Lesson 3.4

```
// Exercise 3.4 - JavaScript tweaks

// add the event listener
viz.addEventListener("markselectionchanged", getSelectedMarks);
```

Lesson 4

// JWT example - Do not simply copy and paste the python code, it's just an example of what it should look like. Your values will differ.

```
CA_SSO_token = jwt.encode(
    {
        # Lesson 2
        # https://help.tableau.com/current/online/en-us/connected_apps.htm#step-3-configure-the-jwt
        # Set iss (Issuer) to Connected App Client ID
        "iss": '4063aef9-f1ef-4dac-87bc-06a05f21a64b',
        "exp": datetime.datetime.utcnow() + datetime.timedelta(minutes=10),
        "jti": str(uuid.uuid4()),
        "aud": "tableau",
        "sub": username,
        "myuaf": ["Texas", "New York", "Utah"],
        "scp": ["tableau:views:embed", "tableau:metrics:embed"]
    },
    # Set this value to the Connected App Secret Key
    'PBBglOD9Pe0inqNf+nHy/K6Ytv8mP06bjkcoOtfx6Lk=',
    algorithm="HS256",
    headers={
        # Set kid (Key ID) to Connected App Secret ID
        'kid': 'df4b89a3-7761-4053-9e4f-77958e00e18f',
        # Set iss (Issuer) to Connected App Client ID
        'iss': '4063aef9-f1ef-4dac-87bc-06a05f21a64b'
    }
)
```