MicroStrategy | 2021

Embedding Analytics with MicroStrategy REST API

Core learning for: Developers

CONTENTS

© 2021 MicroStrategy, Inc.

© 2021 MicroStrategy, Inc.

© 2021 MicroStrategy, Inc.

© 2021 MicroStrategy, Inc.

1

PRE-COURSE SETUP

MicroStrategy Cloud environment

We begin the course by setting up the virtual MicroStrategy Cloud environment to support the exercises you will complete throughout the course.

Exercise 1.1: Access the environment

To prepare for the course, you access your MicroStrategy Cloud environment. The login credentials and other information you need to access your environment are included in the MicroStrategy Cloud email that you received. You will also learn to access MicroStrategy Developer and MicroStrategy Web in your MicroStrategy Cloud environment.

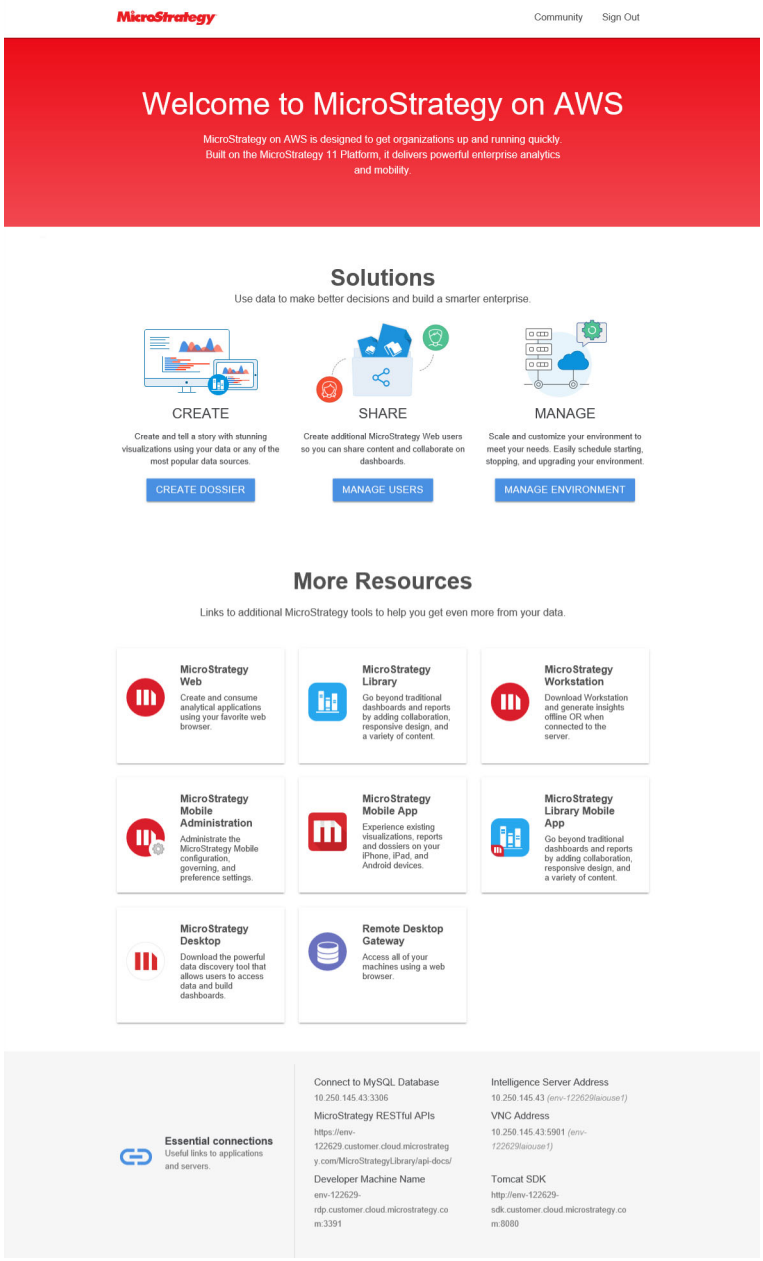
Access the MicroStrategy platform

1 On your local machine, in the MicroStrategy Cloud email, click **Access MicroStrategy Platform**.

© 2021 MicroStrategy, Inc.

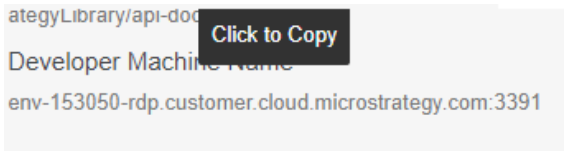
2 In the **User name** and **Password** boxes, enter the login credentials provided in the MicroStrategy Cloud email. Click **Login**.

The MicroStrategy Cloud landing page is displayed.



 © 2021 MicroStrategy, Inc.

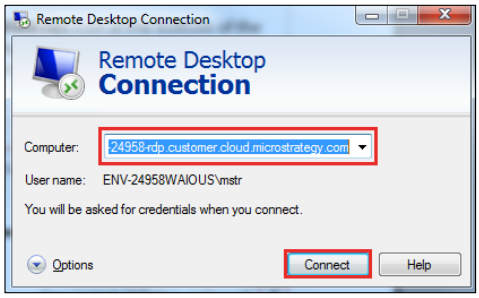
3 In the Essential Connections area, hover over **Developer Machine Name** and click **Copy** to copy the address to your clipboard. For example, the address is similar to: **env-123456-rdp.customer.cloud.microstrategy.com:3391**.



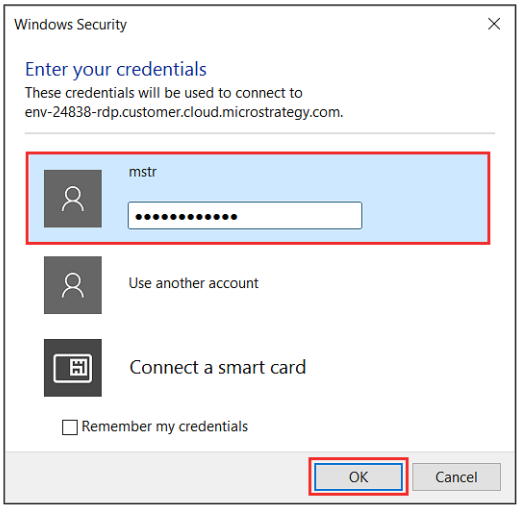
4 On your local Windows machine, from the taskbar, search for **Remote Desktop**. From the search results, select **Remote Desktop Connection**.

This step may differ, depending on the version of Windows operating system on your computer.

5 In the Remote Desktop Connection window, in the **Computer** box, paste the machine name, and click **Connect**.



6 In the Windows Security window, type the username and password from the MicroStrategy Cloud email, and click **OK**.



If an identity verification message is displayed, click **Yes**.

You are now connected to the Windows machine in your MicroStrategy Cloud environment.

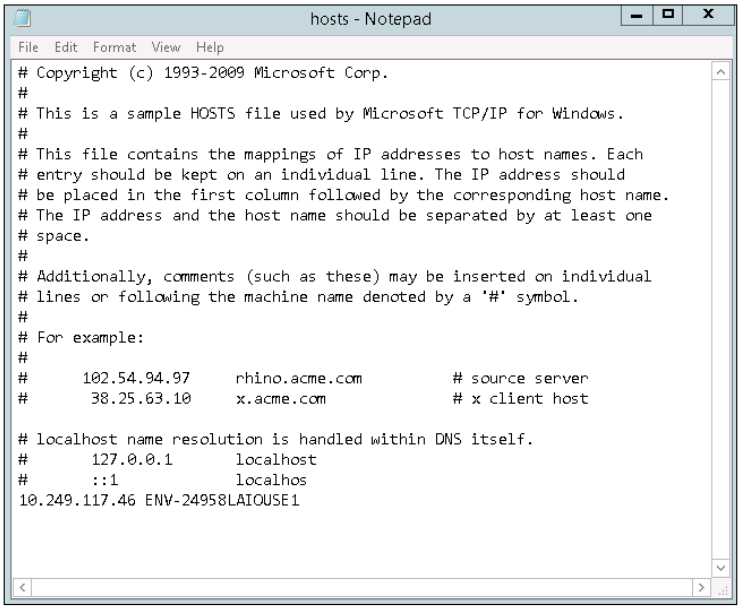
Review the Windows hosts file

The hosts file cross-references specific IP addresses with user-friendly names, just like a Domain Name Server (DNS) that translates IP addresses to human-readable website names as you browse the Internet. DNS translation enable you to navigate to www.microstrategy.com instead of a forgettable series of numbers like 104.121.85.249.

The hosts file serves a similar purpose, but it operates locally on a specific computer. We use the hosts file to map the Intelligence Server IP address to the readable machine name, as in the following example:

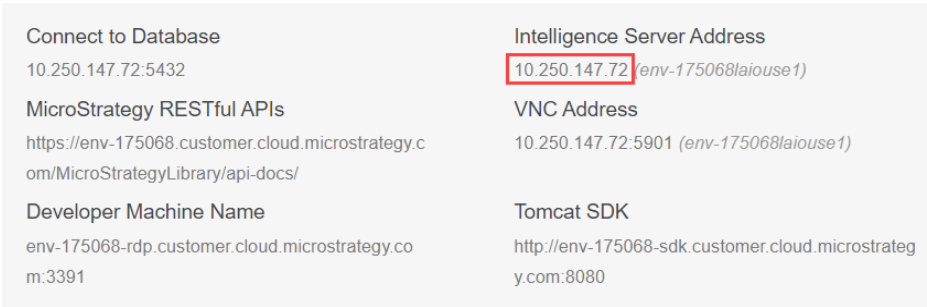
© 2021 MicroStrategy, Inc.

10.249.117.46 ENV-24958LAIOUSE1



In the hosts file, the IP address is mapped to the machine name using the following information:

- **XXX.XXX.XX.XXX**: The IP address of the Intelligence Server located in the Essential Connections area of the MicroStrategy Cloud landing page, as shown in the image below. To copy the address, hover over it and click **Copy**.



- **TAB**: A tabulation character must separate the first and third parts of the line. A space will not work.
- **ENV-XXXXXXLAIOUSE**: The Intelligence Server machine name, where XXXXXX is your environment's unique ID number located in the Essential

Connections area of the MicroStrategy Cloud landing page, as shown in the image below.

Your cloud environment, explained

The MicroStrategy Cloud environment is composed of two virtual servers. In this section, we explore the environment configuration to identify component locations and connections.

Linux Server

The first server in this environment has a Linux operating system. It serves many purposes, including supporting the following:

- Intelligence Server
- Application/web server (Tomcat)
- Mobile Server

Browsing to MicroStrategy Web

To access MicroStrategy Web, you use an address with the following format:

https://env-XXXXX.customer.cloud.microstrategy.com/MicroStrategy/servlet/mstrWeb

where XXXXX represents the environment number, as indicated in the MicroStrategy Cloud email. Because the address includes a “servlet”, you can deduce that you are accessing a Tomcat web server.

Various requests are processed within the same physical machine using logical components like the web server and Intelligence Server.

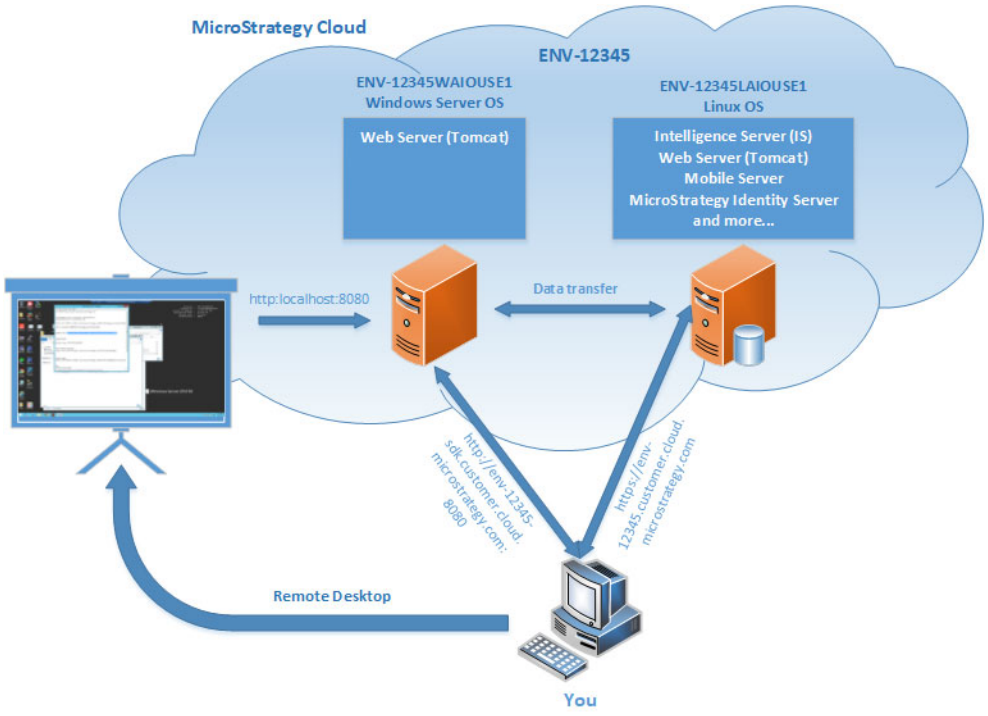
© 2021 MicroStrategy, Inc.

Remotely accessing the Linux server

You can access the Linux server with tools like VNC, Putty, or WinSCP, depending on your need for a full remote desktop experience or a simple file transfer interface.

The diagram below shows the remote connections in your MicroStrategy Cloud environment.

Accessing the MicroStrategy Cloud environment



Windows server

For specific classes in the MicroStrategy Education catalog (mainly the Administrative and SDK classes), a second server hosting a Windows operating system is provided for the following purposes:

- Web server (local 2nd instance, and sometimes 3rd instance of Tomcat)
- MicroStrategy Web Server (through the 2nd instance of Tomcat)
- MicroStrategy Mobile Server (through the 2nd instance of Tomcat)
- Java applet server (through the local Tomcat)
- Liferay portal host (for specific classes, using the 3rd instance of Tomcat)

© 2021 MicroStrategy, Inc.

- Access to MicroStrategy Developer
- Access to MicroStrategy Workstation
- Access to R tools to integrate in MicroStrategy reports

Accessing MicroStrategy Web

You can access MicroStrategy Web using the following address:

https://env-XXXXX.customer.cloud.microstrategy.com/MicroStrategy/servlet/mstrWeb

where XXXXX represents the environment number, as indicated in the MicroStrategy Cloud email.

You can access MicroStrategy data through the second Tomcat web server located on the Windows server using the following address:

http://env-XXXXX-sdk.customer.cloud.microstrategy.com:8080/MicroStrategy/servlet/mstrWeb

where XXXXX represents the environment number, as indicated in the MicroStrategy Cloud email you received. Notice the -sdk portion of the web address, which indicates that you are accessing MicroStrategy Intelligence Server through the Windows machine's Tomcat instance. Notice also that we are using a different port (8080) for this web server, which is a common Tomcat practice.

You can also access MicroStrategy Web from a browser on the Windows Developer Machine, using the following address:

http://localhost:8080/MicroStrategy/servlet/mstrWeb

Again, port 8080 is used by the Tomcat web server. You must use a browser on the Windows Developer Machine for this address to function. To access the web server on the Windows machine from a different location, use the -sdk address.

Accessing the Windows server

Remote access is required to access the applications or file system on the Windows machine through one of the following options:

- a browser-based remote access application available from the Cloud landing page
- a dedicated remote desktop application, available for Windows, MacOS, and Linux

© 2021 MicroStrategy, Inc.

Refer to the diagram in the previous section for a visual representation of the MicroStrategy Cloud environment and the various access scenarios.

The Education scenario presented includes two servers. However, a typical implementation of the MicroStrategy platform may include a single server where all MicroStrategy components are installed, or multiple servers that each house distinct MicroStrategy components. The implementation in your organizations depends on traffic volume considerations and specific needs like security requirements or deployment conditions.

Preparing the environment

Now that you have configured the Remote Desktop application and have successfully connected to your Windows machine, you are ready to prepare the environment.

Exercise 1.2: Close Eclipse and stop Tomcat

To complete the software installation for this class, you must first close the Eclipse IDE and stop the Tomcat web server on the Windows machine. To ensure proper performance on the Tomcat server, you will also verify that an appropriate amount of memory is allocated to the Tomcat server's Java Virtual Machine.

Close Eclipse and stop Tomcat

1. On the Windows desktop, right-click **Tomcat_Stop** and select **Run as Administrator.**

2. In the User Account window, click **Yes**.

3. If the Eclipse environment is open, save your work and close it.

Exercise 1.3: Run the final installation script

The exercises in this class require additional software installation and configuration. In this section, you will run an automated script that performs the following tasks:

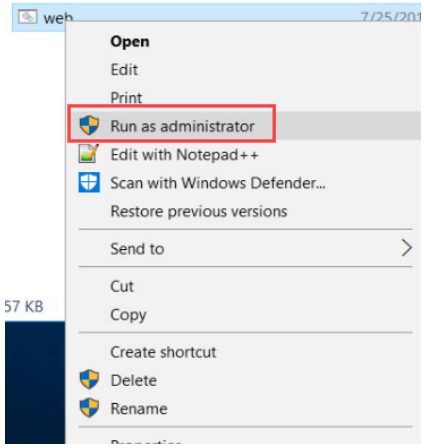
© 2021 MicroStrategy, Inc.

- Remove the current version of Eclipse. This is why you closed Eclipse earlier.
- Install MicroStrategy Web and MicroStrategy Library applications. This is why you shut down Tomcat earlier.
- Configure Tomcat to locate the MicroStrategy applications in c:\sdk_workshop\war.
- Install a recent version of Eclipse with a MicroStrategy plug-in already configured.
- Adjust Eclipse settings for MicroStrategy applications.

Run the installation script

1 In File Explorer, navigate to the **C:\sdk_automation\scripts** folder.

2 Right-click **web** and select **Run As administrator**. In the User Account window, click **Yes**.



Run the script only once, as the script deletes all of your progress if it is executed twice. If you would like to revert the environment to a brand new state, you can run the script again.

Once the script ends, the command line window closes automatically. This might take a few minutes.

Exercise 1.4: Connect Web to the Intelligence Server

To complete the MicroStrategy Web application setup, connect the application to the Intelligence Server located on the Linux server.

© 2021 MicroStrategy, Inc.

Complete the MicroStrategy Web setup

1 From the Windows desktop, right-click **Tomcat_Restart**, and select **Run as Administrator**.

2 In the User Account window, click **Yes**. The Tomcat web server restarts after a minute or so.

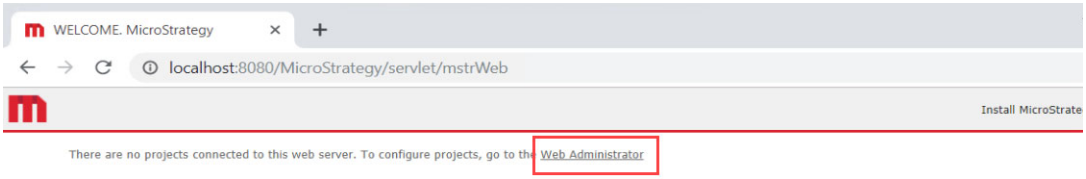
3 From the desktop, double-click **Google Chrome**.

If you want to make Chrome the default browser on the Windows machine, complete the steps displayed in the browser.

4 Navigate to the following URL to validate that Tomcat and MicroStrategy Web are running correctly:

http://localhost:8080/MicroStrategy/servlet/mstrWeb

5 Click the **Web Administrator** link.



6 In the Sign in window, enter the following username and password:

Username = **admin**

Password =**sdkws**

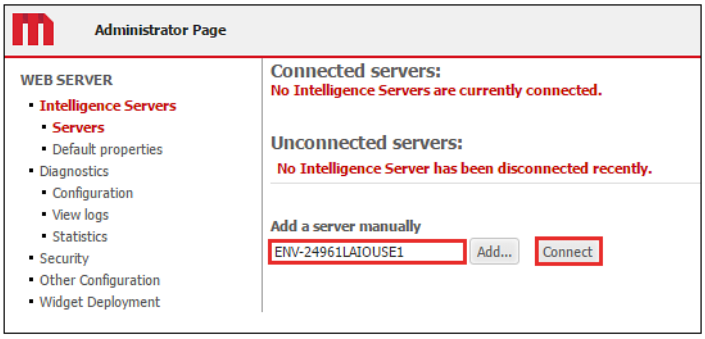
Connect your local instance of MicroStrategy Web to the Intelligence Server

7 From the desktop of the Windows machine, open **hosts.txt** in **Notepad**.

8 At the bottom of the file, copy the Intelligence Server machine name. For example, env-123456laiouse1.

© 2021 MicroStrategy, Inc.

9 In the Administrator Page in Chrome, in the **Add a server manually** box, enter the copied Intelligence Server machine name and click **Connect**.



10 Click the **MicroStrategy Web Home** link.

A list of connected projects is displayed. For all subsequent exercises in this course, access MicroStrategy Web using the Chrome browser on the Windows Developer Machine. This ensures that you are able to access custom files used in this course.

Exercise 1.5: Set the default start page and configure Chrome settings

By default, the MicroStrategy Tutorial project opens on a Welcome/Getting Started page. For convenience throughout this class, change the default start page to the project Home page.

Change the default page for the local web site

1 From the list of projects, click **MicroStrategy Tutorial**.

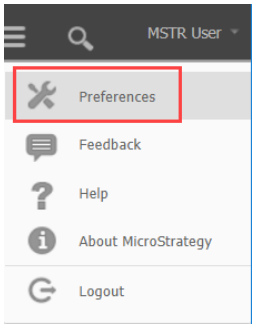
2 Log in using the credentials in the MicroStrategy Cloud email. The MicroStrategy Cloud landing page opens.

3 Scroll over **MicroStrategy Web** and click **Launch**. MicroStrategy Web opens.

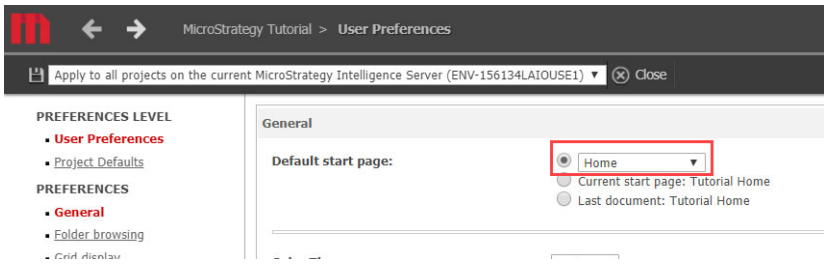
4 Click **Go To MicroStrategy Web**.

© 2021 MicroStrategy, Inc.

5 From the top right corner of the page, click **MSTR User** and select **Preferences**.

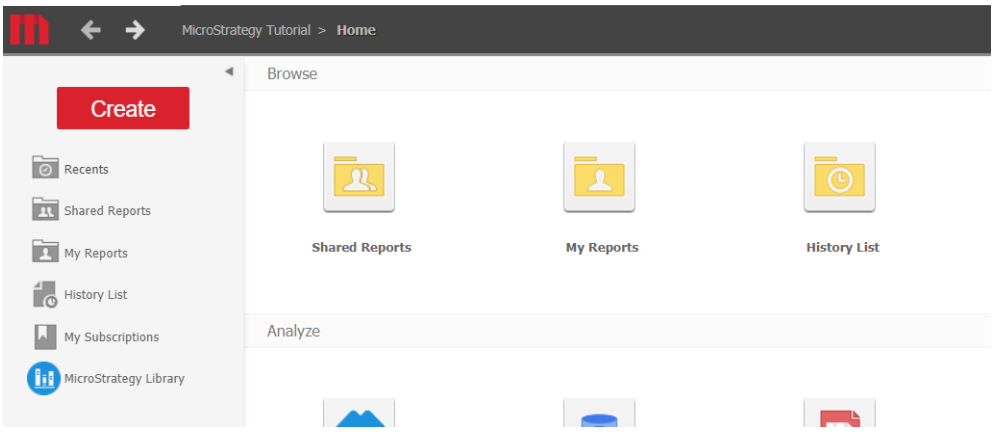


6 In the **Default start page** area, click **Home**, as shown below.



7 Click **Save** on the top left of the screen.

8 Click **Home** . The updated home page is displayed.

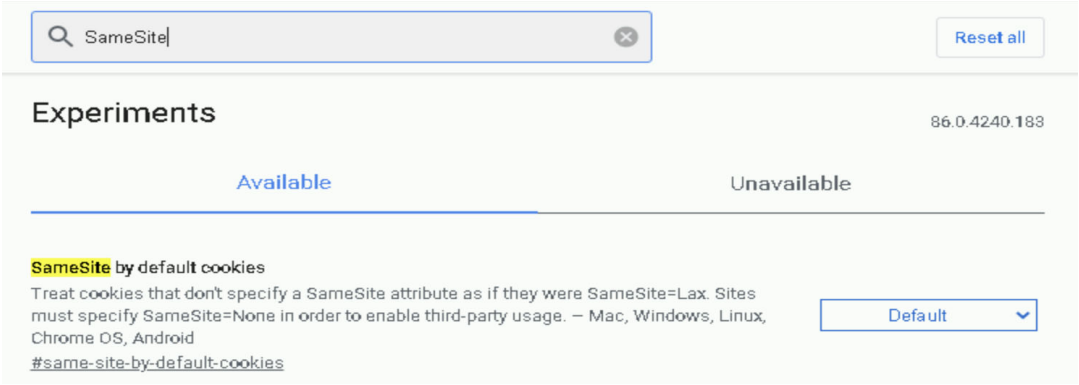


Disable SameSite by default cookies

1 In a Chrome browser, navigate to the following address:

Chrome://flags

2 In the **Search** box at the top, type **SameSite**.



3 From the **SameSite by default cookies** drop-down list, select **Disabled**.

4 Click **Relaunch** at the bottom of the page to load the changes.

5 **Close** the browser.

Exercise 1.6: Configure your local Library

To test your customizations on the local version of MicroStrategy Library, connect MicroStrategy Library to MicroStrategy Web.

Connect Library to MicroStrategy Web

1 On the Windows machine, open a browser and navigate to:

http://localhost:8080/MicroStrategy/servlet/mstrWeb

2 Log in to **MicroStrategy Tutorial**.

3 At the top right, open the **User menu** and click **Preferences**.

4 Under Preferences Level, click **Project Defaults**.

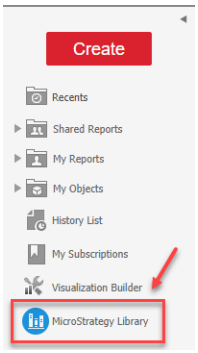
5 In the **MicroStrategy Library Configuration** box, type:

http://localhost:8080/MicroStrategyLibrary

6 Click **Save**.

© 2021 MicroStrategy, Inc.

7 Click **Home** . The MicroStrategy Library link is displayed in the left panel.



Configure the local Library Admin page

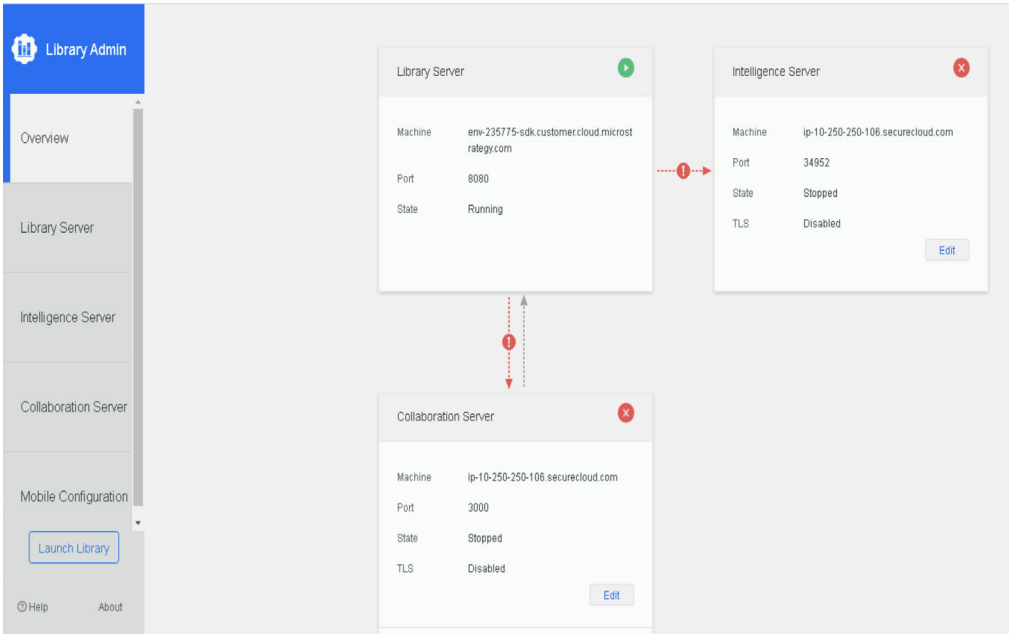
1 On the Windows machine, open a browser and navigate to:

http://localhost:8080/MicroStrategyLibrary/admin

2 Click **Library Admin Page**.

3 If prompted, enter the credentials of **admin** and **sdkws**.

The MicroStrategy Library Admin page opens as shown below. Notice the components that do not contain configuration information.



The configuration needs connectivity details for the following:

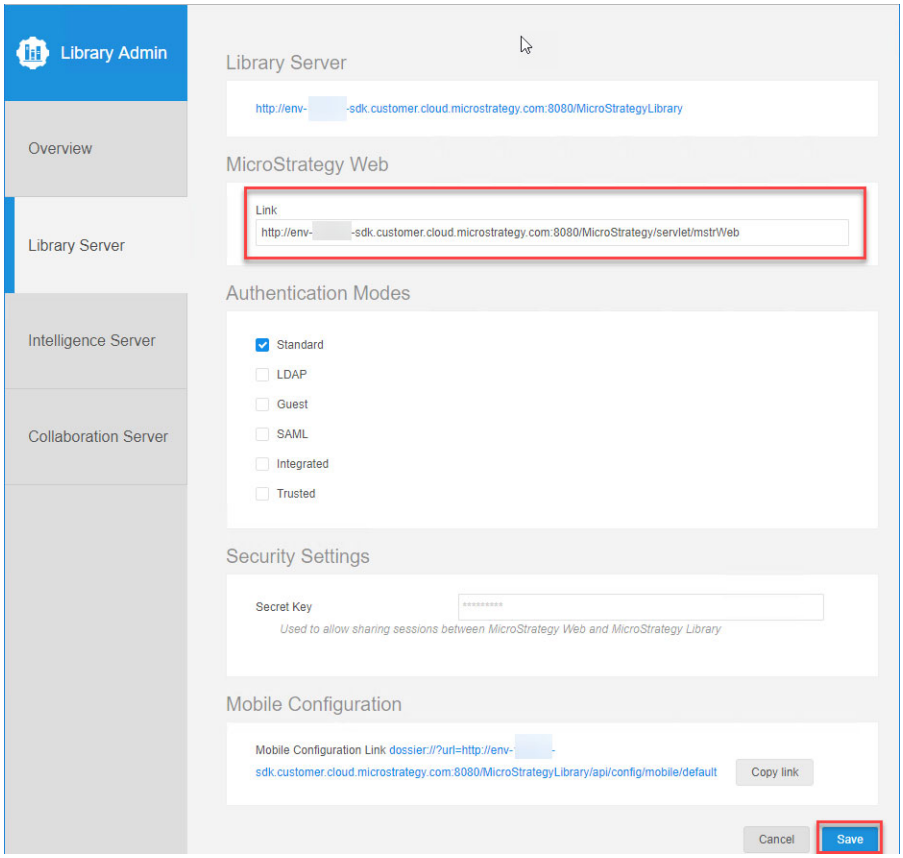
- MicroStrategy Web (within the Library Web Server)
- Intelligence Server
- Collaboration Server

4 Click **Library Server** on the left.

5 In the **Link** box, type **http://env-XXXXXX-sdk.customer.cloud.microstrategy.com:8080/MicroStrategy/servlet/mstrWeb**

where XXXXXX represents your environment number.

The admin page looks like the following image.



6 Click **Save**.

7 In Mobile Configuration, click **Copy Link**. Paste the value in a new Notepad++ document. You use this information later to configure the Library Mobile app to point to your data.

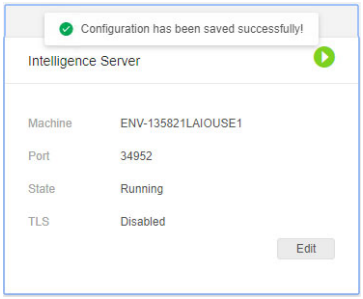
© 2021 MicroStrategy, Inc.

8 Click **Overview** on the left.

9 In the **Intelligence Server** configuration, click **Edit** and modify the following:

a In the **Machine** box, type **ENV-XXXXXLAIOUSE1**, where **XXXXX** represents your environment number.

b In **Port**, enter **34952**.



10 Click **OK**.

The local Windows version of the Library Server is now connected to the Intelligence Server, enabling users to save dossiers to their Library, create filters, bookmark pages, and so on. The Collaboration Server connection is not established on your local installation because it is already connected to the Library Server on your environment's Linux machine.

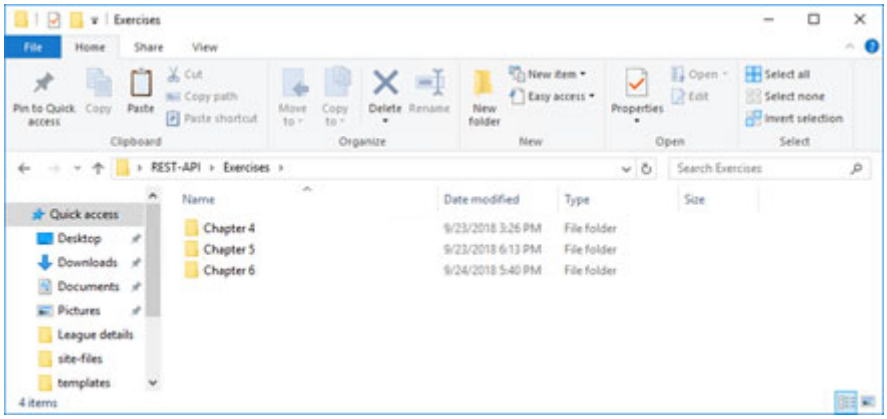
Exercise 1.7: Extract the class files

Your instructor will provide an archive of files needed to complete the exercises in this course. Follow the steps below to copy and extract files to the Windows machine in your Cloud environment.

Extract the class files

1 Copy the **REST-API.zip** file to the desktop of the Windows machine.

2 Right-click the .zip file and select **Extract All**. A REST-API folder is created on the desktop.



Your environment is now ready for the REST API class. Let's begin.

© 2021 MicroStrategy, Inc.

2

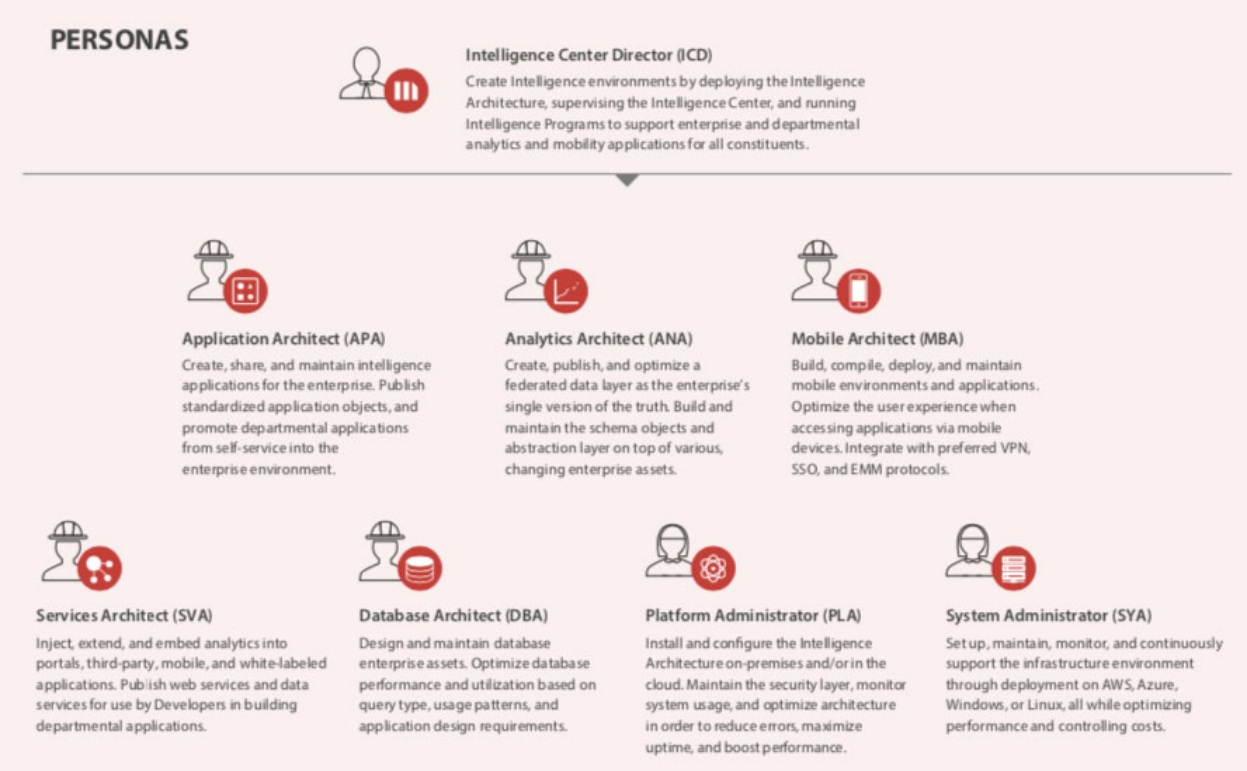
REST API FUNDAMENTALS

Developing in an Intelligent Enterprise

MicroStrategy provides self-service Business Intelligence, and Enterprise BI with data governance, enterprise-level security, user administration, and big data support, all centered around the Intelligence Server. The Intelligence Server delivers world-class monitoring, reporting, and analysis on one integrated platform, offering next-generation Business Intelligence capabilities for the full range of BI applications. MicroStrategy's enterprise platform is the core of the Intelligent Enterprise.

© 2021 MicroStrategy, Inc.

Supporting the Intelligent Enterprise is the Intelligence Center, comprised of a team of expert architects who define, develop, and provide guidance across the enterprise.



The Services Architect is responsible for creating and implementing guidelines and best practices to inject, embed, and extend the MicroStrategy analytics capabilities into custom applications. Development initiatives such as white-labeling and web services must follow the standards established by the Services Architect.

As a developer, you understand that standards play an important role in software. Even a small customization can become a nightmare to maintain and support when standard coding practices are ignored. Intelligent Enterprise best practices can help you efficiently and accurately develop applications in your organization.

The scope of a customization can expand dramatically. A small change requested by a single department can easily evolve into an enterprise standard. By

employing the standards set in place by the Services Architect, development projects can be streamlined to scale with minimal resistance.

Regardless of your organization's size, the Intelligent Enterprise can be leveraged to maximize your investment in the MicroStrategy platform, and to obtain valuable insights that help your organization excel.

API: Communicating with applications

An Application Programming Interface (API) is a collection of communication protocols employed by an application or library to communicate with the external world. For example, you might create your own application that implements an existing map application. You would implement the map application by leveraging its API.

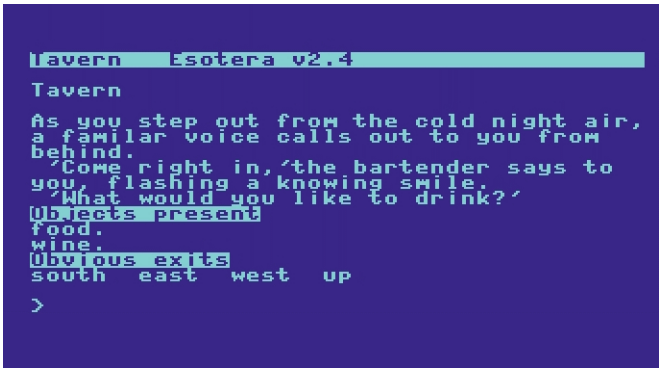
An API provides statements that you can use to communicate with the application. You must apply specific syntax to your statements to ensure that the application can understand your intent.

Think about the digital assistant you use every day: Alexa, Siri, Google, etc. A combination of words like "what is the weather in Los Angeles?" triggers a response.

However, API syntax normally has a strict set of communication rules. For example, if the API expects "what is the weather in <location>?", it cannot understand "What is the *temperature* in <location>?"

In the early days of computing, developers created text adventure games played through the terminal. To progress through the game, you had to conform to a strict set of commands. For example:

Forward	Turn Left	Turn Right
Open door	Examine chest	Open chest
Attack	Parry	Ask information



The interface forms a restricted language between you and the software. An API might include statements to perform the following:

- Retrieve a status
- Set a status
- Perform an action

Let's review each type of interaction available.

Retrieve a status

Make an inquiry like "What is my account balance?" When you retrieve a status, you are reading a property in the application.

Set a status

Modify a specific value like "Set the temperature to 21 degrees Celsius." When you set a status, you set a property. The interface may acknowledge the change by sending a response.

Perform an action

Ask the application to perform an action like "Send an e-mail to a list of contacts." When you perform an action you are executing a method, function, or procedure. The interface may return a response when the task is complete, or a message explaining whether the task is complete.

Specifying context in API calls with parameters

A parameter is a piece of information that provides context in an API call.

When you ask your digital assistant for the weather forecast in Los Angeles, the city or geographical location is a request parameter. When you translate a word in your browser from one language to another, you pass three parameters:

- The word or expression to be translated
- The language the word or expression is from
- The language you want to translate to

Any information that narrows the scope of a request is considered a parameter. Some systems require multiple parameters to complete their task. Parameters can take various forms, such as a string of text, a number, an array of items, or a programming object. The information format is known as a data type.

Each API expects parameters in a specific format or data type. If you specify a parameter with an unexpected data type, the system will not understand the request and respond with an incorrect answer or an error. You can find required and optional parameters and their associated data types in the API Explorer and the documentation.

Best Practice When you implement an external API, become familiar with its documentation to understand its available methods and properties, and the expected parameters.

Required parameters

Required parameters are pieces of information that an API requires to perform its work. For example, if you are using an API to retrieve a dossier, the dossier identifier is a required parameter.

© 2021 MicroStrategy, Inc.

Optional parameter

An API end points can accept more information than is required to accomplish its task. For example, all MicroStrategy API end points have an optional fields parameter that enable you to request specific information in the response message. Let's take a look at another example.

A bank API enables you to create a new account for a client. To process the request, the API requires the client's name and address, as well as the type of account created (checking, savings, loan, or a combination of the three). The interface also has an optional parameter for an initial deposit amount. The following table summarizes the banking example.

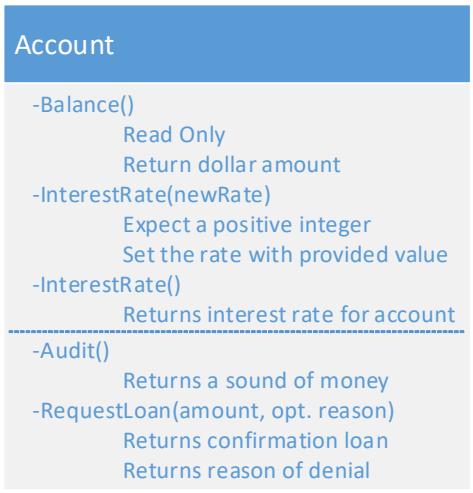
Parameter	Name	Address	Account Type	Initial Deposit	Comment
Necessity	Req.	Req.	Req.	Opt.	
Data Type	String	String	C, S, B, L	Number	
Clients					
Client 1	John Maddox	123 Prairie St, Driken, AL	C	--	Account created
Client 2	--	359 Green Lane, Proxy, CA	B	$500	Error Name missing
Client 3	Stephanie Ryan	1190 Ctr Pkwy, Miami, FL	B	--	Accounts created
Client 4	Thomas Lux	902 Ever St, Chicago, IL	C	$1000	Account created with deposit

© 2021 MicroStrategy, Inc.

Exercise 2.1: Build an API - Part I

For this exercise, you play the role of a programmer tasked with defining an API. The goal for this exercise is to understand the process of making a call and receiving a targeted response. Use the following examples as a guide.

Banking



API call	Sample Response	Comments
Balance()	$234.87	Returns balance is in the account.
InterestRate(5.21)	No response	You just set the interest rate to 5.21%.
InterestRate()	5.21%	Returns the current rate.
Audit()	“Cha-Ching”	You hear the sound of money.

RequestLoan(100)	{
	“LoanStatus”: “Granted”
	}
		You get a JSON response and the loan is granted

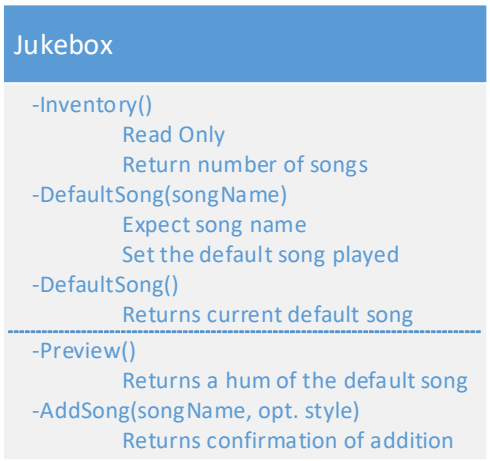
© 2021 MicroStrategy, Inc.

API call	Sample Response	Comments

RequestLoan(1,000,000, Red Ferrari)	{
	“LoanStatus”: “Granted”
	}
		Congratulations, you got a loan for a red Ferrari!

RequestLoan(1,000,000, Blue Ferrari)	{
	“LoanStatus”: “Denied”,
	“Reason”: “What were you thinking? Ferrari should only be red!”
	}
		There is no help for you. Blue?

Music



API call	Sample Response	Comments
Inventory()	5837	Returns number of songs.
DefaultSong(Imperial March)	No response	You just set the default song to the Imperial March (by John Williams)
DefaultSong()	Imperial March	Returns the current default song.
Preview()	“Pam Pam Pam, Pam Po-Pam Pam Po-Pam”	You hear the sound of the default song.

API call	Sample Response	Comments

AddSong(Don't Worry, Be Happy)	{
	"SongStatus": "Added"
	}
		You get a JSON response and the song is added

AddSong(True Colors, Romantic)	{
	"SongStatus": "Added"
	}
		You added True Colors to the Jukebox, in the romantic category.

Establish a business realm

1. Identify a field of interest for which you will design your API.
2. Create a brief list of "calls" in your API.
3. Compare your calls to the examples in the tables below.

Define properties

1. Define one property whose value you want to retrieve. No parameter are required.
2. Define one property whose value you want to set. Define at least one parameter to pass the desired value.
3. Define the expected values in each property.

Define methods

1. Define one method that requires no parameters.
2. Define one method that requires at least one required parameter and one optional parameter.
3. Define the expected values in each parameter you define.

© 2021 MicroStrategy, Inc.

Exchange you API

1. Find a partner for this activity.
2. Exchange your APIs and determine if you can create calls without any further instruction.

Make API calls

In this section, you and your partner take turns making a single API call from each API.

1. Using the API syntax, on a new sheet of paper, Partner A writes down an API call to your partner's system, pass the sheet to your partner, and wait for an answer. Select one of the following calls:
   - Set the value for a property.
   - Get the value of a property.
   - First method without parameter.
   - Second method with required parameter(s).
   - Second method with both required and optional parameter(s).
2. Pass the call from Partner A to Partner B.
3. Partner B reads the call and writes down a response. If a call is not properly formated, respond with a helpful error message.
4. Pass the response from Partner B to Partner A.
5. Partner A reads the response to determine if the call was executed as expected.
6. Repeat these steps with Partner B making a call to Partner A's API.
7. Continue making calls until time is called by your instructor. Keep your calls to use in the next exercise.

© 2021 MicroStrategy, Inc.

Transmitting information in API calls

An API's main purpose is to exchange data. In the API call exercise, you transmitted data when you created your calls, and you received data when your partner sent you a response. When calls are made in an API, the response must be in a format that is easily consumable by an application. Let's explore some of these formats.

Formatting objects with JSON

JSON (**J**ava**S**cript **O**bject **N**otation) is a simple text-based exchange format that represents data structures in web based code. JSON's popularity stems from the ease with which applications can parse its data.

JSON objects format data using key/value pairs. One string of text serves as an identifier (the key), and another string presents the actual value. The JSON object contains the following formatting:

- Keys and values are enclosed in double quotes (" ")
- Keys and values are separated by a colon (:).
- A JSON object is wrapped in curly braces ({ })

The following line of code is an example of a simple JSON object:

```
{ "carModel" : "Porshe 911" }
```

carModel is a key, while Porshe 911 is its value. Multiple key/value pairs in a JSON object are separated by a comma (,), as in the following example:

```
{
    "fname" : "John",
    "lname" : "McLane",
    "Profession" : "Cop",
    "Shoes" : "None"
}
```

Creating values with various data types

While all keys are strings, their values can contain other data types. The following list of data types are derived from JavaScript:

- String (in double quotes “ ”)
- Number
- Object (in curly braces { })
- Array (in square brackets [ ])
- true
- false
- null

Examples

The following JSON objects illustrate some more complex structures.

```
{
    "Vendor": "Sony",
    "Products": ["PS", "PS2", "PSP", "PS3", "PS4",
"Vita","PS4 Pro"],
    "Prices": { "PS": 299,
                "PS2": 299,
                "PSP": 249,
                "PS3": 499,
                "PS4": 399,
                "Vita": 249,
                "PS4 Pro": 399
              }
}
```

```
{   "students": {
      "John": {
       "DOB": "12/21/2001",
       "Topics": ["Math", "Literature", "Computer Science"],
       "Details": {
          "Math": {
              "Teacher": "Paul G.",
              "Room": 1121,
              "Grades": [92, 97, 100, 95, 93],
              "Pass": true
            },
          "Literature": {
```

```
                "Teacher": "Lucy T.",
                "Room": 1315,
                "Grades": [57, 49, 64, 70, 50],
                "Pass": false
              },
            "Computer Science": {
                "Teacher": "Gilles D.",
                "Room": 1099,
                "Grades": [100, 97, 100, 99, 100],
                "Pass": true
              }
              }
         },
       "Alexa": {
         "DOB": "09/17/2000",
         "Topics": ["Math", "Literature", "Computer Science"],
         "Details": {
            "Math": {
                "Teacher": "Paul G.",
                "Room": 1121,
                "Grades": [91, 96, 88, 100, 92],
                "Pass": true
              },
            "Literature": {
                "Teacher": "Erika S.",
                "Room": 1317,
                "Grades": [88, 79, 81, 89, 94],
                "Pass": true
              },
            "Computer Science": {
                "Teacher": "Gilles D.",
                "Room": 1099,
```

```
                "Grades": [91, 94, 98, 86, 100],
                "Pass": true
            }
            }
        }
    },
 "School": "Lady of Grace of Tysons",
 "Year": 2018
}
```

The same example is displayed below in a more compact format. Computers do not require whitespace, but JSON objects are often formatted with whitespace to make them readable by humans.

```
{"students":{"John":{"DOB":"12/21/
2001","Topics":["Math","Literature","Computer
Science"],"Details":{"Math":{"Teacher":"Paul
G.","Room":1121,"Grades":[92,97,100,95,93],"Pass":true}
,"Literature":{"Teacher":"Lucy
T.","Room":1315,"Grades":[57,49,64,70,50],"Pass":false}
,"Computer Science":{"Teacher":"Gilles
D.","Room":1099,"Grades":[100,97,100,99,100],"Pass":
true}}},"Alexa":{"DOB":"09/17/

2000","Topics":["Math","Literature","Computer
Science"],"Details":{"Math":{"Teacher":"Paul
G.","Room":1121,"Grades":[91,96,88,100,92],"Pass":true}
,"Literature":{"Teacher":"Erika
S.","Room":1317,"Grades":[88,79,81,89,94],"Pass":true},
"Computer Science":{"Teacher":"Gilles
D.","Room":1099,"Grades":[91,94,98,86,100],"Pass":true}
}}},"School":"Lady of Grace of Tysons","Year":2018}
```

Encoding data structures in XML

XML (E**x**tensible **M**arkup **L**anguage) is a markup language that defines rules to encode documents. This format uses tags to delimit elements. You can define your own schema (data structure) for a document type and use it to store any kind

of data. For example, HTML is a specialized version of XML used for web pages. The following example data structure is formatted in XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <isAlive>true</isAlive>
  <age>27</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021-3100</postalCode>
  </address>
  <phoneNumbers>
    <type>home</type>
    <number>212 555-1234</number>
  </phoneNumbers>
  <phoneNumbers>
    <type>office</type>
    <number>646 555-4567</number>
  </phoneNumbers>
  <phoneNumbers>
    <type>mobile</type>
    <number>123 456-7890</number>
  </phoneNumbers>
  <children/>
  <spouse/>
</root>
```

© 2021 MicroStrategy, Inc.

JSON vs XML

Take a look at the following side-by-side comparison of a data structure in two different formats.

XML	JSON

<?xml version="1.0" encoding= "UTF-8" ?>
<Vendor>Sony</Vendor>
<Products>PS</Products>
<Products>PS2</Products>
<Products>PSP</Products>
<Products>PS3</Products>
<Products>PS4</Products>
<Products>Vi-ta</Products>
<Products>PS4 Pro</Products>
<Prices>
<PS>299</PS>
<PS2>299</PS2>
<PSP>249</PSP>
<PS3>499</PS3>
<PS4>399</PS4>
<Vita>249</Vita>
<PS4 Pro>399</PS4 Pro>
</Prices>
	{
	"Vendor": "Sony",
	"Products": ["PS", "PS2", "PSP", "PS3", "PS4", "Vita","PS4 Pro"],
	"Prices": { "PS": 299,
	"PS2": 299,
	"PSP": 249,
	"PS3": 499,
	"PS4": 399,
	"Vita": 249,
	"PS4 Pro": 399
	}
	}

Human readable

The first thing to notice is that both structures are readable. You can easily identify the desired information in both structures.

Hierarchical

In both structures, information is stored in a hierarchical format, which makes the data simple to navigate and parse.

Can be parsed

Both formats can be parsed and read or even written via a programming language. For most languages, such as JavaScript, you can find native capabilities to consume the JSON format. If the capabilities are not native to a certain programming language, there is a high probability that a parsing library is available.

© 2021 MicroStrategy, Inc.

For XML, support is provided by an XML parser (an external library you need to add to your application).

Can be read by XMLHttpRequest

In a web page, usually through Javascript, you can make a request to fetch information using the XMLHttpRequest method. The method can handle a response with either XML or JSON.

JSON uses less wrapping

Although both formats are structured, you can see that XML uses opening and closing tags for everything. XML uses much more disk space than JSON, which has a minimal shell and mostly surrounds its data with punctuation.

In the AJAX context

When working with AJAX (Asynchronous Javascript and XML), using JSON is preferable due to its native support structure in Javascript.

Let's revisit the API you created in the previous exercise and add a dose of JSON in it.

© 2021 MicroStrategy, Inc.

Exercise 2.2: Build an API - Part II

In this exercise, practice creating JSON objects.

1 Write down your identity using JSON. Include the following:

- First Name
- Last Name
- Occupation
- State
- Phone Number
- Age

{

_____

_____

_____

_____

_____

_____

}

2 Using the API you developed in the previous exercise, write your call responses in JSON format.

_____

_____

_____

_____

_____

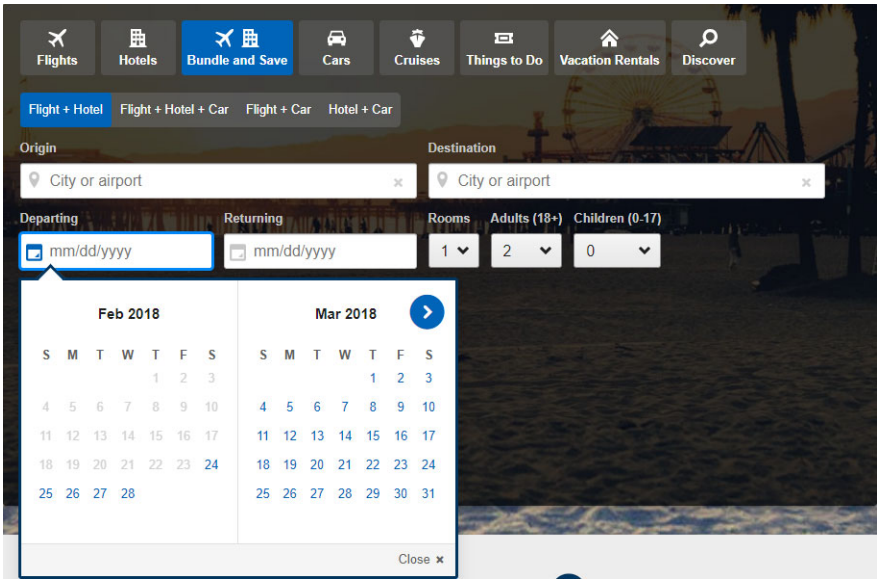
Exposing APIs through web services

An API's value is its ability exposed an application's internal functionality to the external world. This exposure enables developers to implement an application's functionality in their own software. In this section, we explore a few formats for exposing an API.

Most web sites are designed for human-to-machine interaction and communication through HTML, CSS, and Javascript. Sound design and development are essential to a successful interaction with any web site.

© 2021 MicroStrategy, Inc.

For example, to select a date-range for a trip on a travel site, visually pleasing and easily navigable calendars are displayed to help you communicate the required information.



In comparison, recall the visual display of a JSON object with no whitespace:

```
{"students":{"John":{"DOB":"12/21/
2001","Topics":["Math","Literature","Computer
Science"],"Details":{"Math":{"Teacher":"Paul
G.","Room":1121,"Grades":[92,97,100,95,93],"Pass":true}
,"Literature":{"Teacher":"Lucy
T.","Room":1315,"Grades":[57,49,64,70,50],"Pass":false}
,"Computer Science":{"Teacher":"Gilles
D.","Room":1099,"Grades":[100,97,100,99,100],"Pass":
true}}},"Alexa":{"DOB":"09/17/
2000","Topics":["Math","Literature","Computer
Science"],"Details":{"Math":{"Teacher":"Paul
G.","Room":1121,"Grades":[91,96,88,100,92],"Pass":true}
,"Literature":{"Teacher":"Erika
S.","Room":1317,"Grades":[88,79,81,89,94],"Pass":true},
"Computer Science":{"Teacher":"Gilles
D.","Room":1099,"Grades":[91,94,98,86,100],"Pass":true}
}}},"School":"Lady of Grace of Tysons","Year":2018}
```

This is far from user-friendly, but a computer is able to easily process this data without any visual design considerations. Machine-to-machine communication does not require the same design elements that are used in human-to-machine interactions. Instead, computers use web services to communicate.

© 2021 MicroStrategy, Inc.

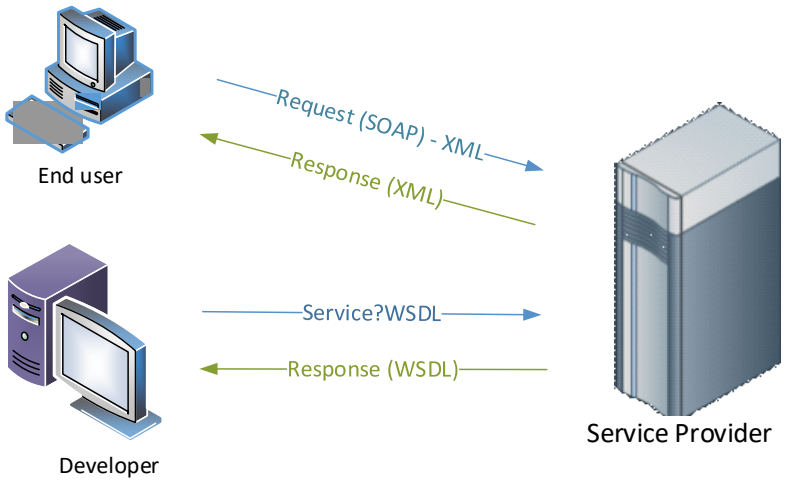
A web service establishes data communication between two electronic devices through a machine-readable format like XML or JSON. Computer-to-computer information transfer over HTTP usually occurs using either the SOAP protocol or the REST architectural pattern.

Communicating with web services through SOAP

SOAP (**Si**mple **O**bject **A**ccess **P**rotocol) uses a strict XML structure to describe the transfer of information between the provider of a service and its consumer. SOAP was developed by Microsoft and is effective when using any of the .NET languages.

SOAP uses a service definition file called WSDL, or Web Service Description Language, to describe the capabilities of the web service, its available calls, and their respective parameters.

Although SOAP is able to maintain a state between calls it requires a great deal of bandwidth overhead. SOAP is also difficult to parse, especially with Javascript, which requires long lines of code to accomplish simple tasks. This is partly due to the XML structure that SOAP requires.



Stateless communication with REST

Representational **s**tate **t**ransfer, or REST is a stateless approach to communication for web services. Whereas SOAP is a protocol, REST is an architectural style (or design pattern) for building services. A service built using this pattern exposes resources (objects) that can be used by an application. By definition, a REST service usually offers only specific methods (verbs), such as:

- GET

- POST
- PUT
- DELETE

Each exposed URI (Universal Resource Identifier) triggers a response that is considered a representation of the resource invoked. That representation can use comma separated values (CSV), JSON, Really Simple Syndication (RSS), or other formats. The response includes a confirmation and any requested data, or simply a status of success.

Stateless

Because RESTful APIs are stateless, no client context is stored on the server between requests. Each request from a client contains all the information necessary to service the request, and the session state is held in the client. The session state can be transferred by the server to another service such as a database to maintain a persistent state for a period and allow authentication. The client begins sending requests when it is ready to make the transition to a new state.

Methods vs resources

A resource is an object with a type, and associated data. Optionally, a resource may have relationships to other resources. A resource has a name, a URI (location in the server) and a representation (how the resource data is returned).You can invoke the methods (or verbs) to a resource. For example, you can use the user PUT method to edit an existing user.

Some resources can return data in various formats. For example, you might specify a parameter to request the response in JSON or in XML.

Accessing a web service through end points

The URI (Uniform Resource Identifier), or end point, is the web address by which you access a web service resource. This is an address to interact with the exposed resources of the web service. You may also use the term URL (Uniform Resource Locater, a subset of URI) to designate end points.

© 2021 MicroStrategy, Inc.

Exercise 2.3: Make a call

In this exercise, experiment with a RESTful interface.

1 Using your phone, make a call to a talking time service. In the United States, try the following:

- 1 (202) 762-1401
- 1 (719) 567-6742

For other countries, browse to **https://en.wikipedia.org/wiki/Speaking_clock** and locate the appropriate number. Not all services return the exact same info in the same format.

You receive the same information (data from the resource) in two representations: Eastern or Mountain time, and Universal time.

In this example, the resource call is:

```
1(202)762-1401/
```

You can consider the above line the URI for the resource. This is the same principle as a REST web call, but you are using a phone as the medium.

2 In the United States, dial 1 (866) 895-3124.

Consider the following phone number the base URL:

```
1(866)895-3124/
```

A message lists the available services.

3 Say **Time**.

Listen to the time information.

4 Select a time zone by saying its name.

Listen to the response.

5 Say **Main Menu**.

6 Say **Weather**.

Listen to the prompt.

© 2021 MicroStrategy, Inc.

7 Say **Los Angeles, California**.

Listen to the forecast.

8 Say **Stop**.

9 Say **90210**. Make sure to say “zero”.

Listen to the forecast.

10 Hang up.

You just created a series of calls by submitting a command, passing parameters and getting a response. In the last exchange, you sent two types of parameters to obtain the weather forecast: a city name and a ZIP code. This illustrates that some API calls can be made with different types of parameters to obtain the same result.

MicroStrategy RESTful APIs

To embed MicroStrategy functionality in your own custom web applications, you can use a series of RESTful APIs that leverage many features of the MicroStrategy platform. The following table describes the APIs:

Category	Description
Authentication	The Authentication API family enables developers to authenticate users against the Intelligence Server or third-party servers. It includes: • Login API (Create and terminate sessions) • Session API (Session info) • Identity Token (manage tokens for third party applications
Browsing	The Browsing API family enables developers to leverage traditional metadata folder browsing through the Folders endpoints. It also exposes the ability to perform a search with a wide variety of search criteria.
Cubes	The Cubes API family enables developers to create a report instance based on a cube and incrementally fetch slices of the report instance. The Cube API lets developers leverage the performance advantage of in-memory cubes. The cube can be either an Intelligent Cube or a DDA (Direct Data Access)/ MDX-based Live Connect cube.

© 2021 MicroStrategy, Inc.

Category	Description
Datasets	The Datasets API family enables developers to push external data directly to the MicroStrategy Intelligence Server. This data can be used to create a dataset or to modify an existing dataset in a Data Import cube. Once the data has been pushed to the Intelligence Server, developers can use the Cube API or the Report API to manipulate that data.
Datasource Management	The Datasource Management API family enables administrators to create, update, and delete database connections in the MicroStrategy metadata. This functionality can be used to automate the update of data source credentials.
Dossier and Documents	The Dossiers and Document API family exposes functionality that enables the execution and exporting of dossiers and documents to a variety of export formats.
Emails	The Emails API family enables administrators and developers to send emails using MicroStrategy Distribution Services email transmitters.
Hierarchies	The Hierarchies API family retrieves attributes in a hierarchy.
Library	The Library API family exposes the publishing workflow, allowing requests to publish or unpublish to a set of users/user groups, and obtain the library for the authenticated user.
Misc	The Misc API retrieves the status of the REST API server.
Monitors	The Monitors API family enable you to manipulate node, project, and database connections.
Object Management	The Object Management API family exposes the ability to programmatically manage objects in the MicroStrategy metadata. Endpoints can be used to certify or decertify a report or dossier, delete objects, and update object info for any object type.
Projects	The Projects API family enables an authenticated user to obtain the list of projects which they have access to. It also exposes functionality for an administrator to administer the quota settings for users and projects.
Reports	The Reports API family enables developers to retrieve the data from a specific report. The Report API currently supports reports with attributes on rows and metrics on columns.
Scripts	The Scripts API enables you to create and manipulate scripts in MicroStrategy projects.
Security Roles	The Security Roles API family enables administrators and developers to obtain a list of roles, its members and current privileges, and to manage security role membership.

© 2021 MicroStrategy, Inc.

Category	Description
System Administration	The System Administration API family enables administrators and developers to programmatically configure the settings related to the MicroStrategy Library web application. For example, this functionality can be used to configure default values related to the authentication process, set up trust relationships to the Intelligence Server, and more.
User Management	The User Management API family enables developers to create, update, and delete users and user groups; perform user group membership management; and manage membership within Security Roles. This functionality can be used to automate user creation and management in conjunction with the on boarding process.

Summary

After an overview of the Intelligence Center, we explored the fundamentals of API and parameters that enable external developers to leverage an application's features in their own software. You built your own API framework and tried to access and respond to a partner's API. We then explored the JSON format's role in transmitting data structures between systems.

You also learned about web services and their underlying communication protocols and frameworks. You simulated REST services using your phone. Finally, you reviewed the MicroStrategy family of REST services.

In the next chapter we discuss the array of end points MicroStrategy's REST API has to offer and the tools needed to explore this API.

© 2021 MicroStrategy, Inc.

Activity 2.4: Quiz

Answer the following questions. You will find the answers in *Quiz Answers, page 153*.

1 API stands for

a Array of Programming Indexes

b Application Programming Interface

c Application Programming Index

d Application Provider Interlace

---

2 True or False: All APIs use a text interface.

---

3 True or False: A parameter can be either required or optional.

---

4 True or False: JSON is considered difficult to parse.

---

5 What other format is JSON often compared with?

a HMTL

b RAR

c XML

d MKV

6 True or False: Spacing is not important in JSON.

---

7 True or False: SOAP stands for Simple Object Access Protocol

---

© 2021 MicroStrategy, Inc.

8 True or False: SOAP requests and responses use the JSON format.

---

9 True or False: A REST API possesses a base URL and end points for each available function.

---

10 Which of the following is not part of MicroStrategy REST API?

a Authentication

b Cubes

c Projects

d System Administration

e User Management

f Warehouse

© 2021 MicroStrategy, Inc.

3

Tools of the API Trade

REST APIs enable developers to leverage an application's functionality in their own software through a two-way communication interface. The MicroStrategy REST API enables you to leverage analytics features in your own systems.

In this chapter, you will gain hands-on experience with REST API tools that you can use to test MicroStrategy end points.

MicroStrategy REST API Explorer

The MicroStrategy REST API Explorer enables you to test all MicroStrategy end points, understand optional and required parameters, and examine responses to your calls. The API Explorer links to the API documentation, where you can find detailed information on methods and parameters in the API. You can also navigate a live demo that provides a hands-on experience with some commonly used end points.

Walk-through: MicroStrategy REST API Explorer

Follow these steps to launch the tool and discover its main components.

© 2021 MicroStrategy, Inc.

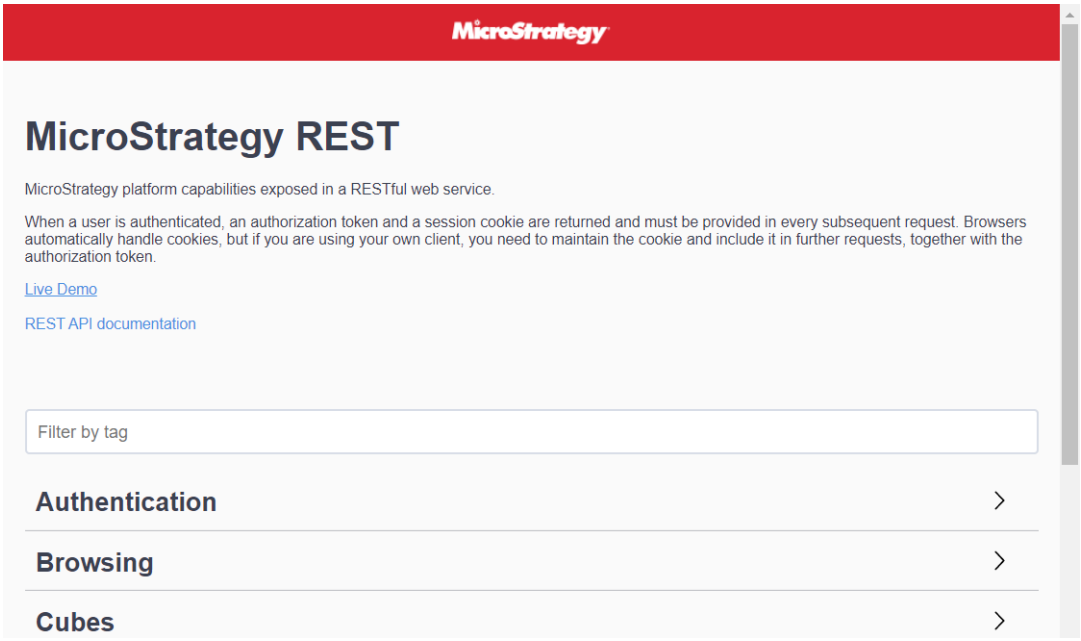
Launch the MicroStrategy REST API Explorer

1 Open a browser session and navigate to:

https://env-XXXXXX.customer.cloud.microstrategy.com/MicroStrategyLibrary/api-docs

where XXXXX represents your environment number.

The following page is displayed.

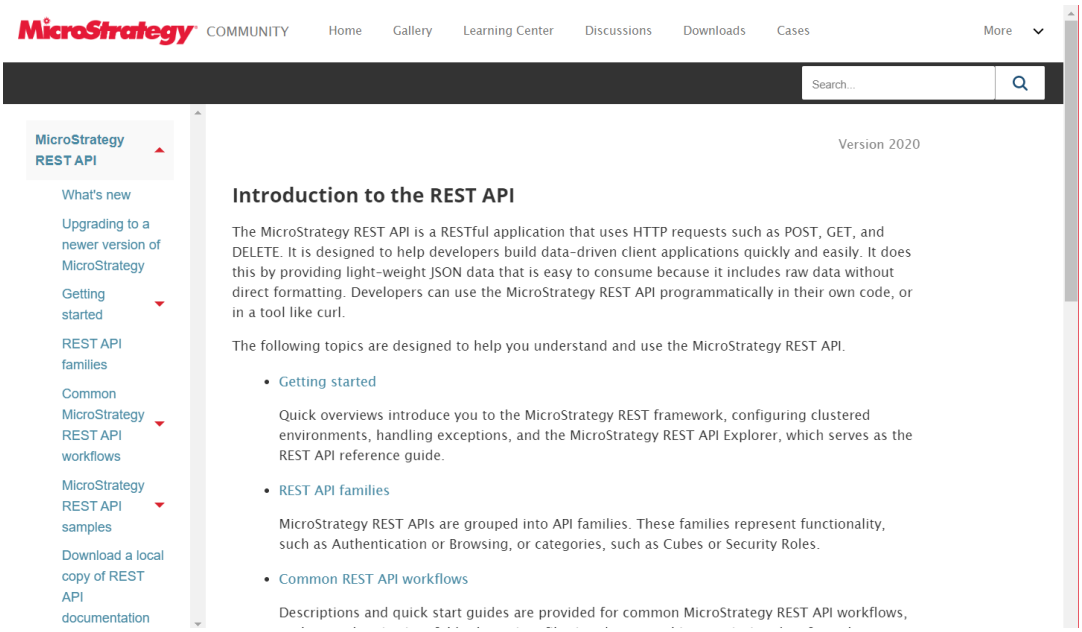


MicroStrategy Library hosts the REST API server, which is a part of your MicroStrategy Library installation. When you access the server, a list of available APIs is displayed in distinct categories. You can drill down into each category to see end point details, HTTP headers, input and output information for each request, placeholders to enter values, and test the APIs for yourself.

2 Click **REST API Documentation** to open the documentation.

© 2021 MicroStrategy, Inc.

A new tab opens as shown below.



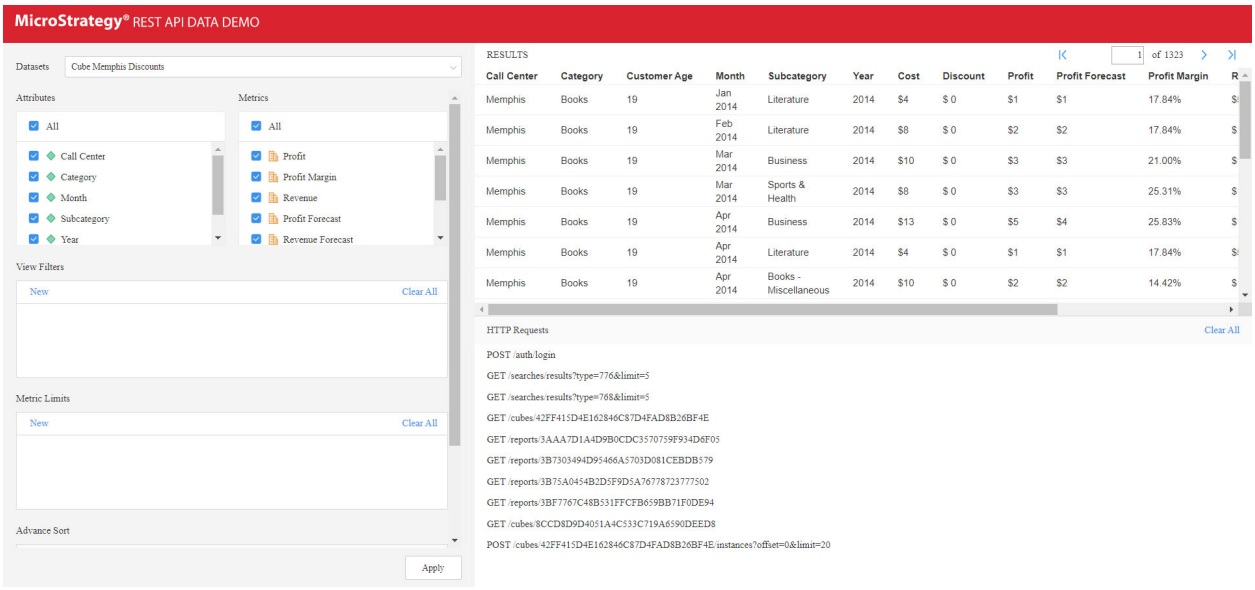
The REST API Help site discusses in detail various workflows and samples for common scenarios.

3 Close the documentation tab.

4 On the MicroStrategy REST page, click **Live Demo**.

The live site opens and displays a series of cubes and reports you can explore. With your selections, you can generate on-the-fly reports displayed on the right side of the page.

This live site is one of the available samples in the documentation. We will take some time later to install and configure it to our needs.



5 Take a few moments to explore the available datasets.

6 In the HTTP Requests area, click one of the **GET/v2/reports/** calls.

7 Click the **Response** tab. The attributes and metrics in the report are displayed, as in the following example.

8 Scroll through the response JSON object and explore the report components.

9 Close the tab.

 © 2021 MicroStrategy, Inc.

Exercise 3.1: Make your first call

Before you integrate a MicroStrategy REST end point with your own application or web pate, use the API Explorer to test the end point and verify its results.

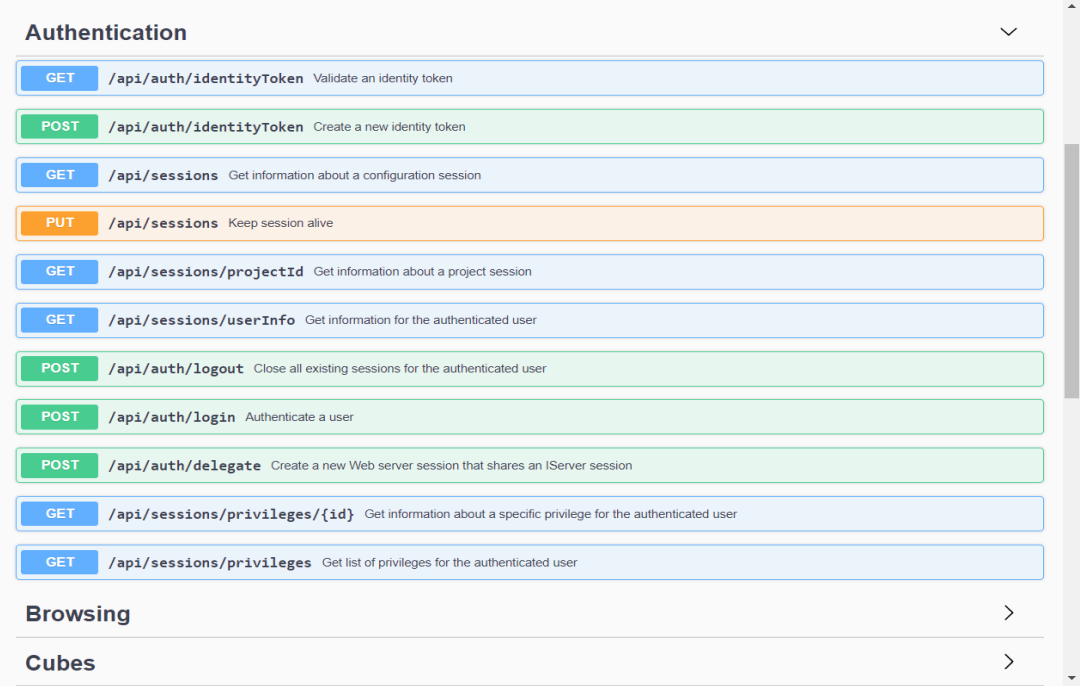
Visiting the REST API tool

1 Open a browser session and navigate to:

https://env-XXXXXX.customer.cloud.microstrategy.com/MicroStrategyLibrary/api-docs

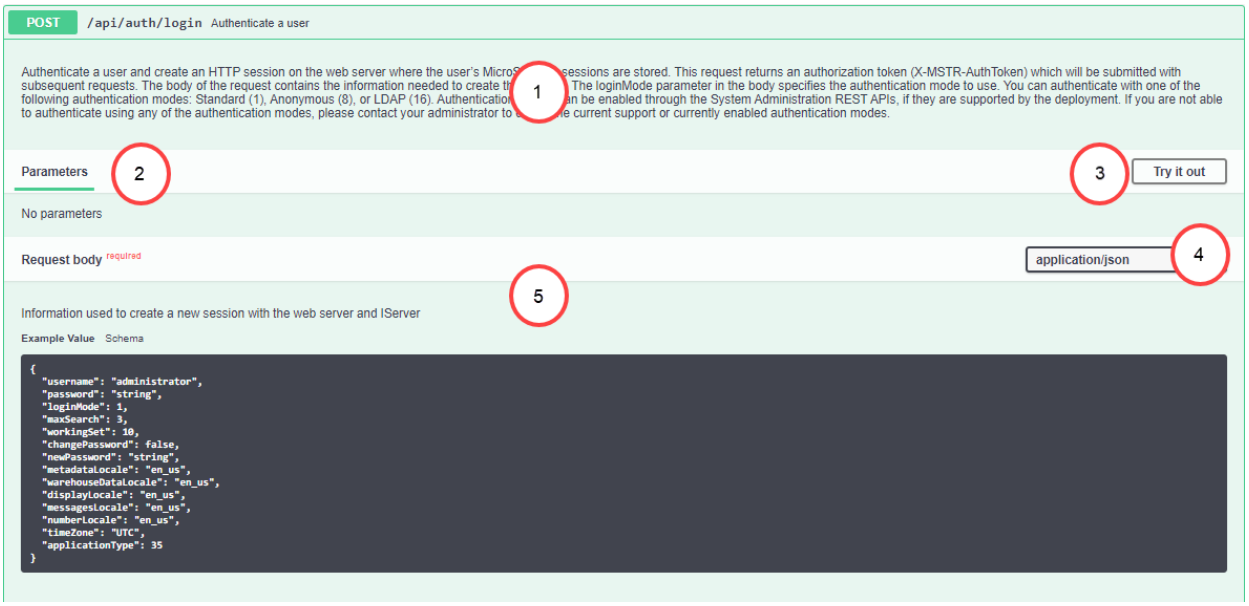
where XXXXXX is the environment number from your MicroStrategy Cloud email.

2 Click the **Authentication** category to view its associated GET, PUT, and POST statements.



3 Click **POST api/auth/login** to view its details.

Review the following API Explorer component descriptions, indicated by the red circles in the image.



1 The Implementation Notes provide context and specific details on using the end point. The text may include constant values explanations or other relevant details.

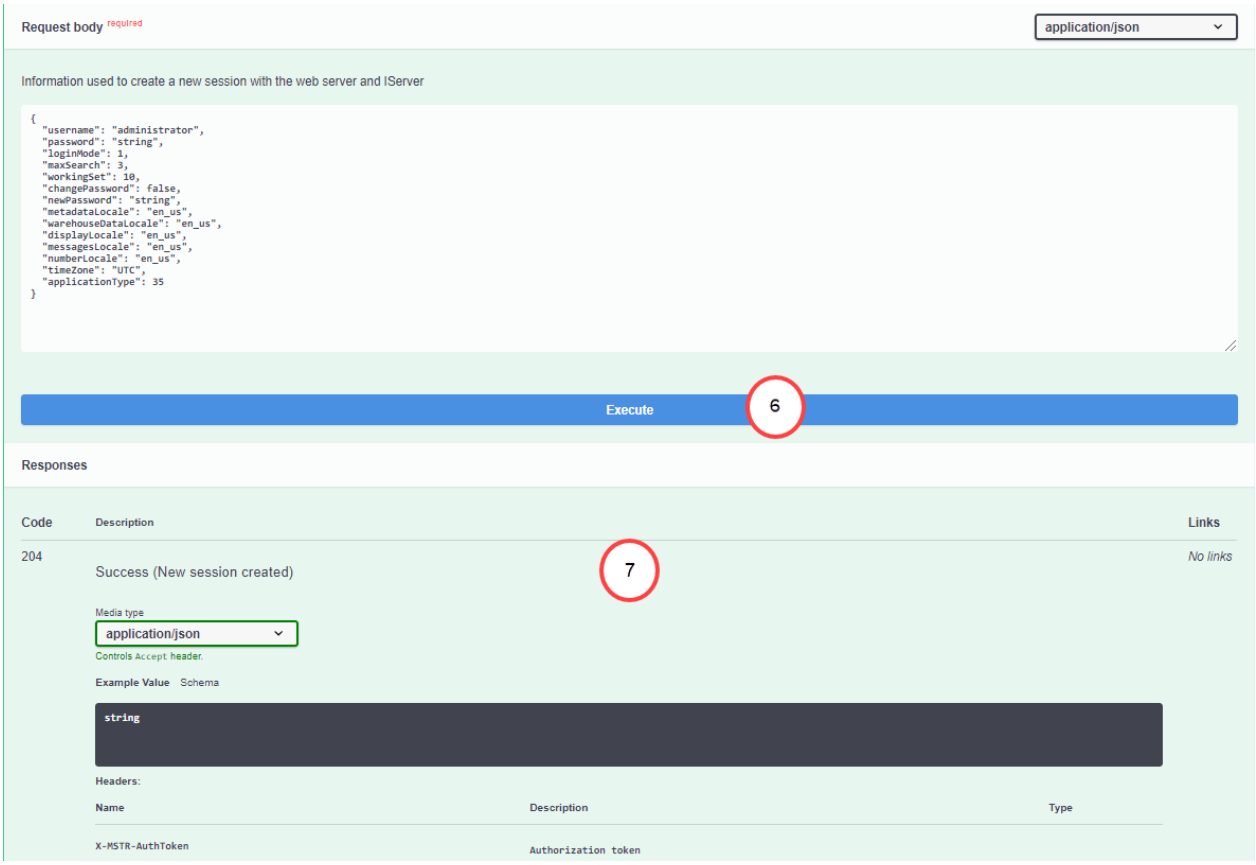
2 The Parameters section contains the available parameters (required or optional).

3 The Request Body section contains an example message body. You can click Schema to view descriptions, data types, and sample values for each component of the message body.

4 The message content can accept one or more formats for its content. JSON is the most common values, but some endpoints can use other formats such as text.

© 2021 MicroStrategy, Inc.

5 Response are possible answers returned following the call to this end point. Use this information with the browser's developer tools to debug a page.



- Code 204 indicates that the call was a success.
- Code 401 indicates that a problem was encountered with the call. An example of a returned value is present to help you parse an error and grab the proper error meaning.

6 You can click Try It Out to convert the example to an editable message. Modify the message with the desired values.

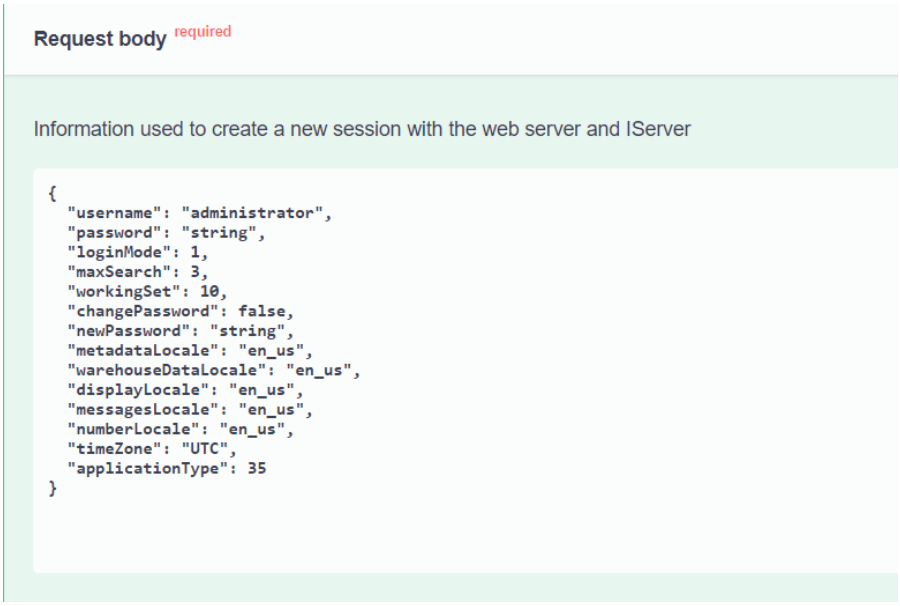
7 You can click Execute to send your call to the REST API server connected to the Intelligence Server. A response is displayed below the Execute button.

© 2021 MicroStrategy, Inc.

Authenticate with the tool

Use the login end point to retrieve the authorization token that is required to use all other end points.

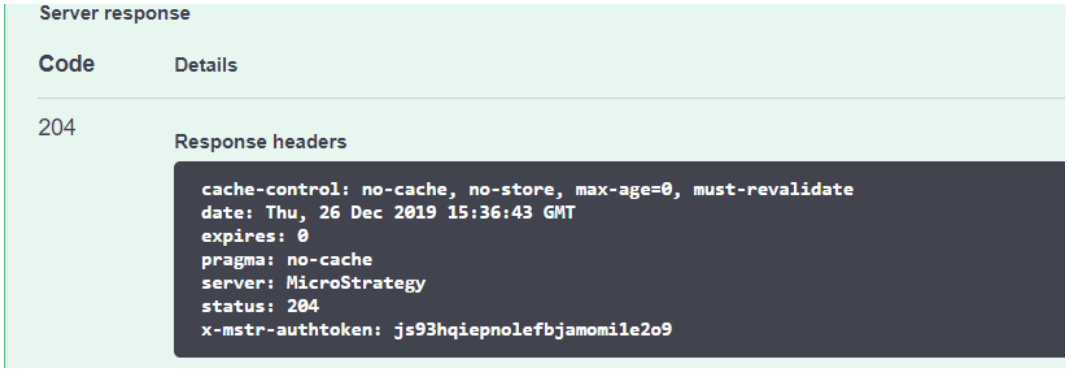
1. From the MicroStrategy REST API Explorer in your browser, expand **Authentication**.
2. Click **POST /api/auth/login**.
3. Click **Try It Out**. The request body is now editable and displayed with a white background.



4. Modify the **username** value to **mstr**. Do not erase the quotes surrounding the value.
5. Modify the **password** value to the password in the MicroStrategy Cloud email.

© 2021 MicroStrategy, Inc.

6 Click **Execute**. What response code do you see?



The server response contains the following components:

- The 204 Response Code block indicates that the request was processed and a valid response was received.
- The Response Headers area contains the x-mstr-authtoken, which is an authorization token that validates the username and password on the Intelligence Server. The token is required to create other REST API calls.

7 In the Response Headers section, copy the **x-mstr-authtoken** value and paste it in an empty Notepad++ file.

Review a user's privileges

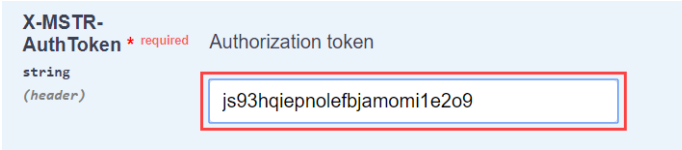
Use the authorization token to create a new call that retrieves user privileges.

1 In the Authentication area, click **GET api/sessions/privileges**.

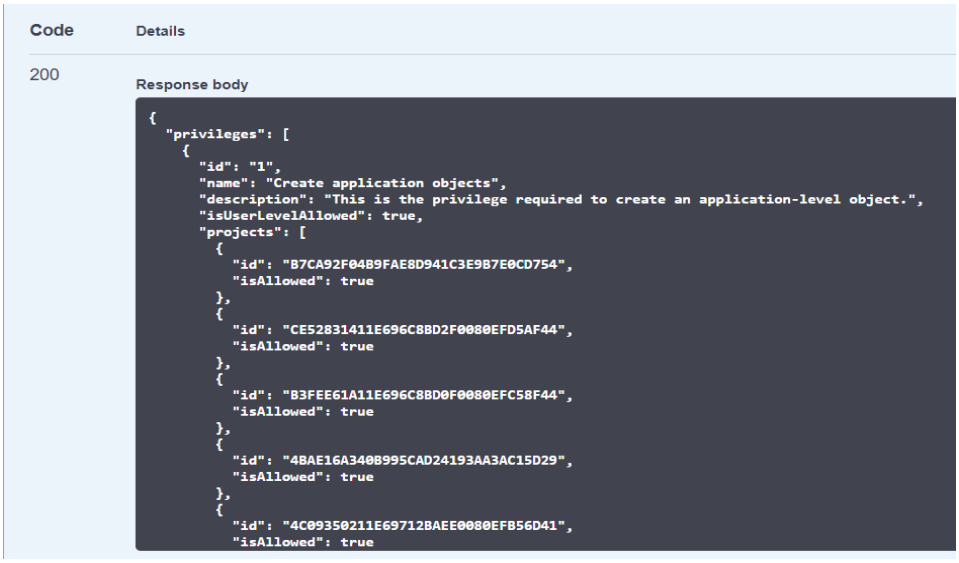
2 Click **Try It Out**.

3 From Notepad++, copy the authorization token.

4 In the **X-MSTR-AuthToken - Authorization Token** box, paste the authorization token.



5 Click **Execute**. The response is displayed in the Responses area.



When you execute the call a 200 Response code is sent back. The response body contains the mstr user privileges.

If you receive an authentication error, your token has expired. The token expired between the login call and the privileges call. Return to the previous section to create the login call and get a new token.

Retrieve a Project ID

Get the Project ID for MicroStrategy Tutorial. You will use this ID later in the class.

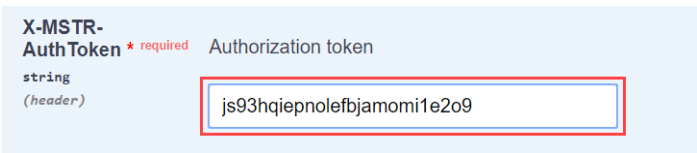
1 Click **Authentication** to collapse the Authentication area.

2 Scroll to the middle of the page and click **Projects**.

3 Click **GET api/projects**.

4 Click **Try It Out**.

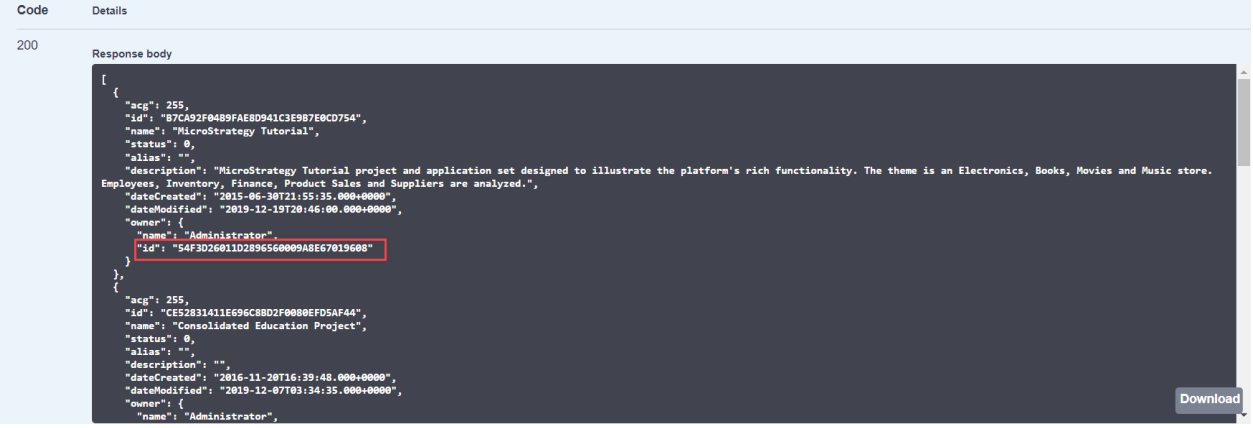
5 In the **X-MSTR-AuthToken - Authorization** token field, paste your authorization token.



 © 2021 MicroStrategy, Inc.

6 Click **Execute**. The response is displayed in the Responses area.

7 In the Response Body Section look for the **MicroStrategy Tutorial** project and copy the **id** for the project.



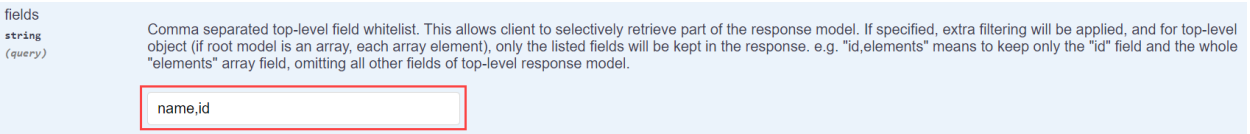
8 Paste the **32-character Project ID** into a text editor and save it.

Retrieve only project names and IDs

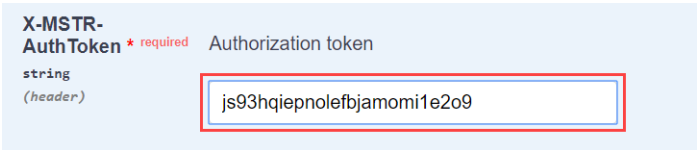
When you invoked the projects API, you received several pieces of information about each project including alias, description, status, date created, and so on. Retrieve the project information again, but this time only retrieve the project names and IDs. In the Projects area, click **GET api/projects**.

9 Click **Try It Out**.

10 In the **Fields** box, type **name,id**. Do not type any spaces.

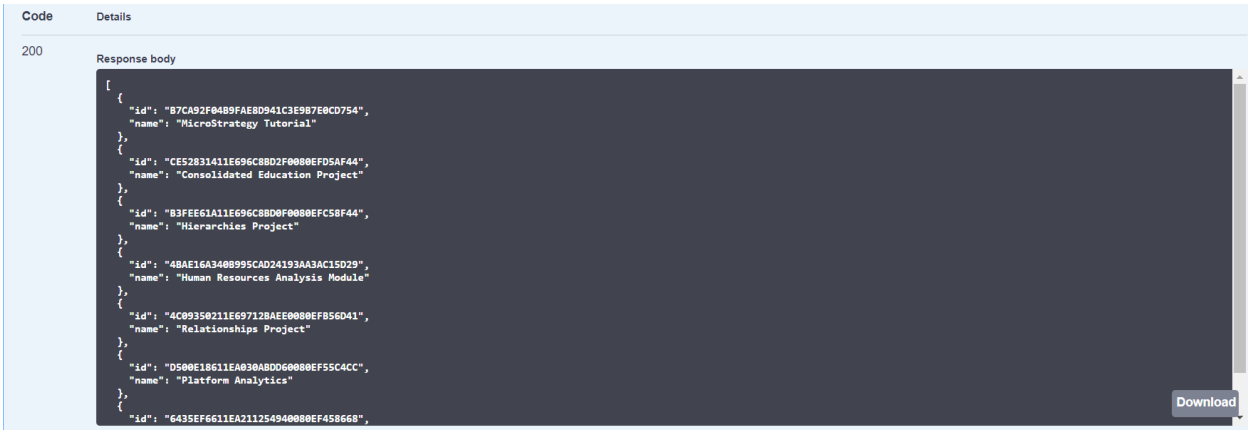


11 In the **X-MSTR-AuthToken - Authorization** token field, paste your authorization token.



12 Click **Execute**. The response is displayed in the Responses area.

In the Response Body Section, notice that only the project names and IDs are displayed.



You used the MicroStrategy REST API Explorer to create an authorization token, retrieve user privileges, and identify a project ID. Now learn to use cURL to test MicroStrategy API end points.

Testing API end points with cURL

cURL is a library and command line tool that can send and receive information using URL syntax.

With cURL, you can test web services endpoints using a variety of protocols including http and https. cURL processes web request in the same way a browser does. You can transmit API calls at the command line and receive summary information in a response header, along with response details in a body in JSON, XML, or some other format.

Why use cURL?

cURL's advantage over other API testing tools like Postman lies in its transparency. cURL does not have a hidden back-end process that obscures data processing.

© 2021 MicroStrategy, Inc.

Installing cURL

cURL is installed by default on Linux and Mac systems. You can install cURL on Windows systems by downloading it from https://curl.haxx.se/.

Using proper syntax

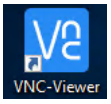
Syntax errors like missing spaces, incorrect quoting, or improper casing can produce error messages that are difficult to parse. To avoid these errors, cURL requires strict adherence to syntax rules.

Exercise 3.2: Create a VNC session

cURL is already installed on the Linux server in your Cloud environment. In this exercise, set up a remote desktop connection to the Linux server to use in future exercises.

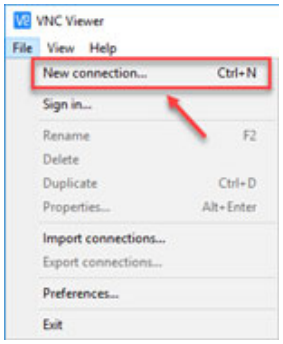
Creating the connection

1 On the Windows desktop, double-click **VNC-Viewer**.



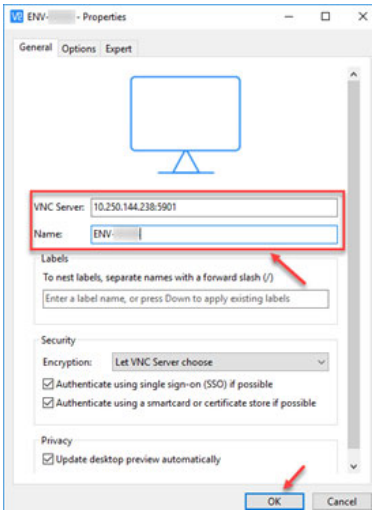
2 In the Get Started window, click **Got it**.

3 From the **File** menu, click **New connection**.The Properties window opens.



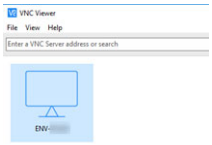
4 In the **VNC Server** box, type the IP address of the Intelligence Server machine followed by **:5901**. For example, 10.250.147.72:5901.

The Intelligence Server IP address is at the bottom of the Desktop\hosts.txt file.



5 In the **Name** box, type **ENV-XXXXXX**, where **XXXXXX** is your environment number.

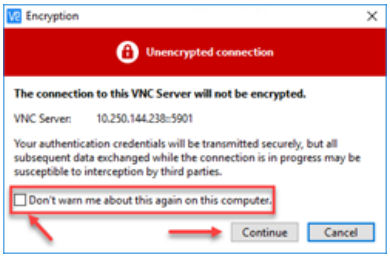
6 Click **OK**. A new connection is created.



Launching the remote session

1 In the VNC Viewer window, double-click the new connection icon.

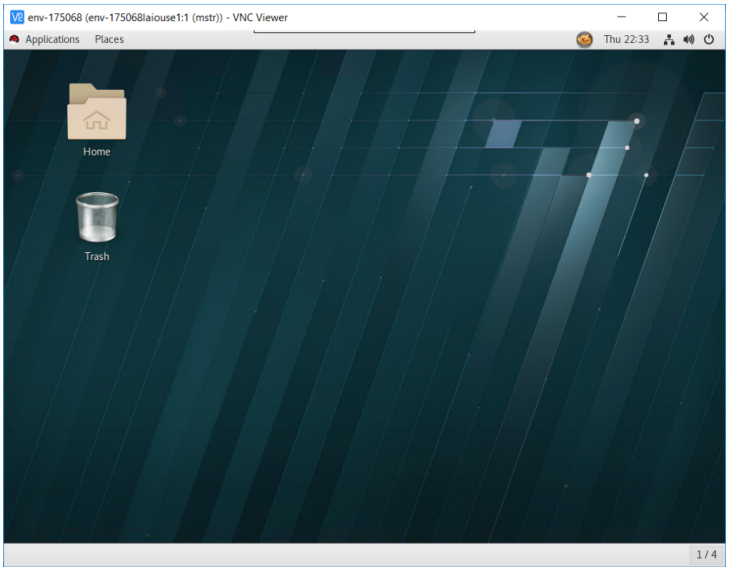
2 In the warning window, check **Don't warn me about this again on this computer**, and click **Continue**.



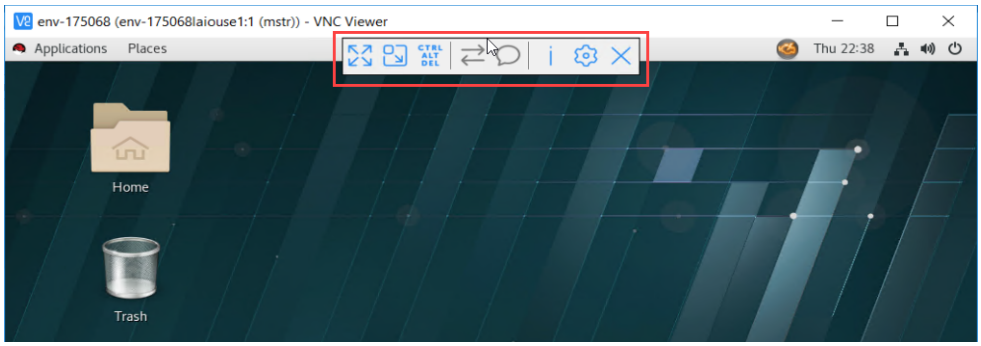
© 2021 MicroStrategy, Inc.

3 In the Authentication window, type the password from your Cloud email and click **OK**.

The remote session opens as shown below.



4 Hover your cursor over the middle of the toolbar to reveal the following menu.



The menu contains options including Full Screen, the CTRL+ALT+DEL command, and advanced properties.

Your VNC session is up and running. Now let's use it to explore the cURL tool.

© 2021 MicroStrategy, Inc.

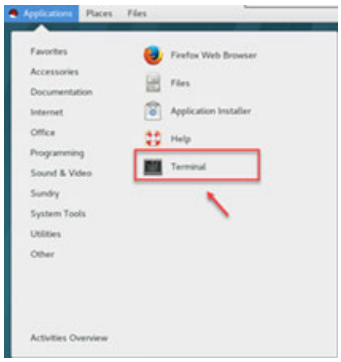
Exercise 3.3: Test API end points with cURL

Use the Terminal application to perform a few cURL calls.

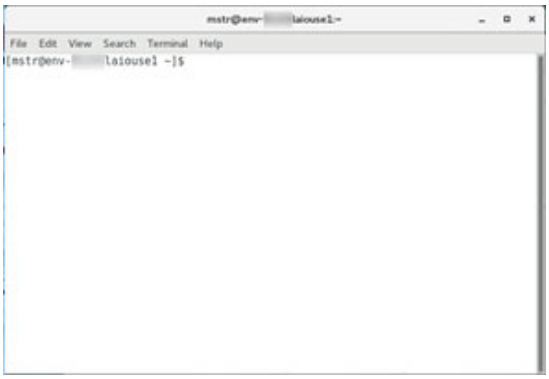
Retrieve the date and time with a cURL command

1 Make the VNC window larger. If desired, make it full screen.

2 From the **Applications** menu, point to **System Tools**, then click **Terminal**.



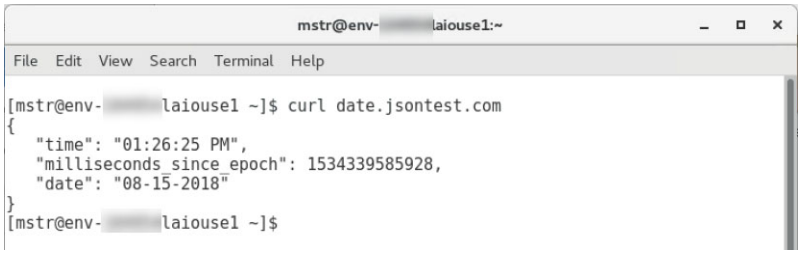
The Terminal window opens as shown below.



© 2021 MicroStrategy, Inc.

3 Enter the following command and press **Enter**.

```
curl date.jsontest.com
```



The following response is received:

```
{
 "date": "03-15-2020"
 "milliseconds_since_epoch": 1534338569412,
 "time": "01:09:29 PM",
}
```

Your time and date will vary, but the information is delivered in the same format. You created the simplest cURL command in the following format:

```
curl url
```

The response is a JSON object with three key-value pairs.

Now add a parameter to include the http response header in the response.

4 Enter the following command and press **Enter**:

```
curl -i date.jsontest.com
```

The response body is the same, but the http header is included.

```
[mstr@env-        laiouse1 ~]$ curl -i date.jsontest.com
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=ISO-8859-1
X-Cloud-Trace-Context: 7ab86b0056e25d99444c05035c4020c2
Date: Wed, 15 Aug 2018 13:30:56 GMT
Server: Google Frontend
Content-Length: 100

{
   "time": "01:30:56 PM",
   "milliseconds_since_epoch": 1534339856129,
   "date": "08-15-2018"
}
```

Notice the line containing 200 OK. A response in the 2xx number range indicates a successful call. The following table outlines the response number ranges and their significance.

Code range	Description
1xx	Transient response; more data is coming
2xx	Success
3xx	A redirect
4xx	The client asked for something the server could not deliver. For example, the classic 404 page not found.
5xx	A problem was encountered on the server

Authenticate the mstr user

Now that you have created a basic cURL command, invoke the MicroStrategy login API.

1 From the **Exercise Files\Chapter 3** folder, open the **Exercise 3_3 cURL commands.txt** document with Notepad++.

© 2021 MicroStrategy, Inc.

The following cURL command is displayed in a single line under Authenticate the mstr User:

```
curl -X POST -i -c ~/cookie-jar.txt --header
'Content-Type: application/json' --header 'Accept:
application/json' -d '{ "username": "mstr",
"password": "PPPPP", "loginMode": 1, "maxSearch":
3, "workingSet": 0, "changePassword": false,
"newPassword": "string", "metadataLocale":
"en_us", "warehouseDataLocale": "en_us",
"displayLocale": "en_us", "messagesLocale":
"en_us", "numberLocale": "en_us", "timeZone":
"UTC", "applicationType": 35}' https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary/api/auth/login
```

2 Replace **PPPPP** with the password from the MicroStrategy Cloud email.

3 Replace **XXXXX** with your environment number.

4 Copy the command and paste it in the Terminal window in the VNC session.

5 Press **Enter**. The following response is received:

```
[mstr@env-[redacted]laiouse1 ~]$ curl -X POST -i -c ~/cookie-jar.txt --header 'Conte
nt-Type: application/json' --header 'Accept: application/json' -d '{ "username":
 "mstr", "password": "[redacted]", "loginMode": 1, "maxSearch": 3, "workingSet
": 0, "changePassword": false, "newPassword": "string", "metadataLocale": "en_us
", "warehouseDataLocale": "en_us", "displayLocale": "en_us", "messagesLocale": "
en_us", "numberLocale": "en_us", "timeZone": "UTC", "applicationType": 35}' http
s://env-[redacted].customer.cloud.microstrategy.com/MicroStrategyLibrary/api/auth/lo
gin
HTTP/1.1 204
Date: Thu, 23 Aug 2018 12:46:41 GMT
Connection: keep-alive
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Set-Cookie: JSESSIONID=776E370EA9E9AACE81858AAC52401E5C; Path=/MicroStrategyLibr
ary; Secure; HttpOnly
X-MSTR-AuthToken: gek6rfkbo492ak595b6k134g7f
Server: MicroStrategy

[mstr@env-[redacted]laiouse1 ~]$
```

6 Copy the **X-MSTR-AuthToken** value and paste it on a new line in Notepad++.

As with the login call you previously made in the MicroStrategy REST API Explorer, the response contains an authorization token. However, the cURL command contains some additional information:

```
-i -c ~/cookie-jar.txt
```

We already know that the -i option requests the response header, but what does the -c option do?

The MicroStrategy REST API uses an authorization token and a session cookie to establish a connection. When you use the MicroStrategy REST API tool, the browser handles the session cookie in the background.

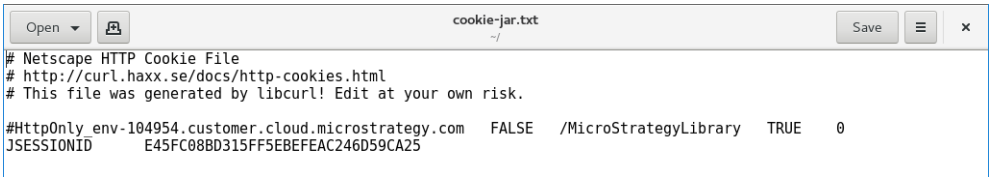
When you use cURL in the Terminal, you must manually handle the session cookie. To do this, use the -c and -b options outlined in the following table:

command	Description
-c path	Write a cookie file using the info provided by the content of the value path. This is usually a path and the filename of the cookie container file. The filename cookiejar.txt is often used.
-b path	Read a cookie file using the info provided by the content of the value path.

The -c option in the command you executed writes the session to the file in the Linux environment using the path provided (~/cookie-jar.txt). Let's open the file and examine its contents.

7 From the desktop of the Linux server (through the VNC session), double-click **Home**.

8 Right-click **cookie-jar.txt** and select **Open with Text Editor**. The session cookie information is displayed.



 © 2021 MicroStrategy, Inc.

List of projects

Retrieve a list of projects from the MicroStrategy Intelligence Server, this time using cURL.

By the time you begin this exercise, your authorization token and session cookie will have expired. To retrieve a new authorization token and session cookie, re-execute the command you previously pasted in Notepad++.

1 In Notepad++, view the following cURL command under Retrieve a List of Projects:

```
curl -X GET -b ~/cookie-jar.txt --header 'Accept:
application/json' --header 'X-MSTR-AuthToken:
32kjeke15h3hhi0fa6fo1qda26' --output projects.jsn
https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary/api/projects
```

2 Replace **32kjeke15h3hhi0fa6fo1qda26** with your authorization token.

3 Replace **XXXXX** with your environment number.

4 Copy the completed command and paste it in Terminal.

5 Press **Enter**.

The response is received.

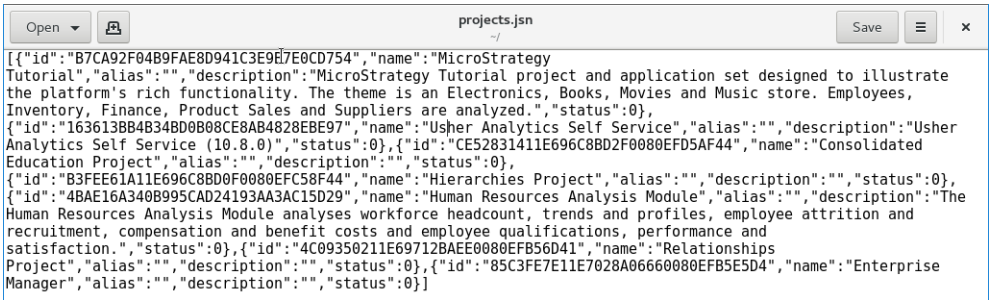
```
[mstr@env-104954laiouse1 ~]$ curl -X GET -b ~/cookie-jar.txt --header 'Accept: a
pplication/json' --header 'X-MSTR-AuthToken: 32kjeke15h3hhi0fa6fo1qda26' --outpu
t projects.jsn https://env-104954.customer.cloud.microstrategy.com/MicroStrategy
Library/api/projects
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1297  100  1297    0     0   6414      0 --:--:-- --:--:-- --:--:--  6420
[mstr@env-104954laiouse1 ~]$
```

The --output option captures the response body in a JSON file (~/projects.jsn) instead of the terminal window. The call also does not contain the -i option, so the header information is not included in the file.

6 From the Linux server desktop, double-click **Home**,

7 Double-click **projects.jsn** to open it with Text Editor.

The JSON data is displayed.



This file is difficult to read, so let's reformat its contents.

8 Copy the content of the file.

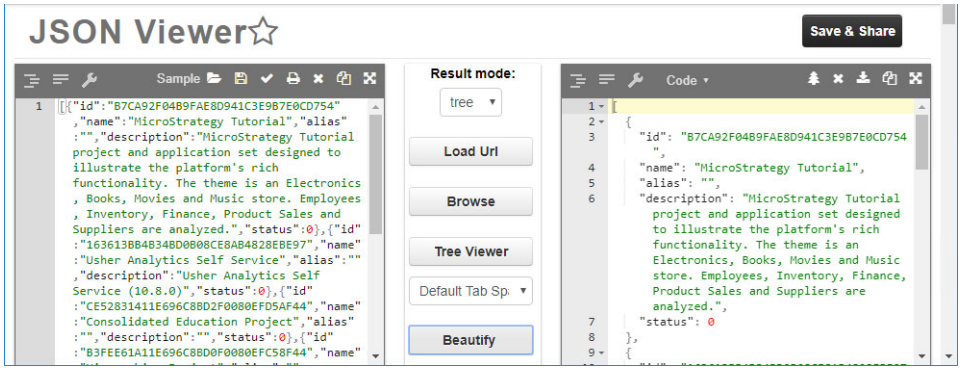
9 On the Windows machine, open **Chrome** and navigate to:

https://codebeautify.org/.

10 Click **JSON Viewer**.

11 In the left pane, paste the content of projects.jsn.

12 Click **Beautify**. The information is displayed in a readable format.



Although the right side is easier to read by humans, applications do not need the extra white space to effectively parse this file. The output produced by the cURL command can be used by an application without any reformatting.

© 2021 MicroStrategy, Inc.

Summary

Though the MicroStrategy REST API Explorer, you reviewed and tested various end points, and utilized a live demo to quickly create calls. You then learned about the authorization token required to create calls.

You also tested REST API calls through the cURL tool, which required you to handle session cookies. You used cURL to retrieve a list of projects, invoked various options to retrieve specific information, and wrote the response body in an external file that can be consumed by another application.

In the next chapter we return to the MicroStrategy REST API Explorer and experiment with the certify end point.

© 2021 MicroStrategy, Inc.

Activity 3.4: Quiz

Answer the following questions. You will find the answers in *Quiz Answers, page 153*.

1 True or False: The REST API Explorer is provided within MicroStrategy Library.

_____

2 The api/auth/login end point provides what type of information in its response?

_____

3 A response code in the 2XX range signifies:

a Success

b A redirect

c Something the server could not deliver like a page not found.

d A problem in the server

_____

 © 2021 MicroStrategy, Inc.

4

Certify a Dossier Using APIs

In MicroStrategy, a certification indicates that an object has been reviewed, verified for data accuracy and integrity, and is promoted for consumption by end users. The certification process ensures that report developers derive data from vetted sources and analysts consume information from a set of curated objects.

You can use the certify API to certify (or remove certification from) data sources, dossiers, and documents. This enables you to create applications that implement workflows to certify MicroStrategy objects stored in the MicroStrategy metadata.

© 2021 MicroStrategy, Inc.

Exercise 4.1 Certify a dossier through the API

In this exercise, use the API Explorer to obtain an authorization token and use it with other details to certify a dossier.

Prepare the connection file

1 Open **File Explorer**, and navigate to **Desktop\REST-API\Exercises\Chapter 4**.

2 Right-click **ENV-XXXXX.mstrc,** and select **Edit with Notepad++**.

```
{
  "environmentName": "ENV-XXXXX",
  "dossierServerURL": "https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary/",
  "authenticationMode": 1,
  "dateCreated": "2018-04-16T10:55:58.6561868-04:00",
  "dateModified": "2018-04-16T10:55:58.6561868-04:00"
}
```

3 Replace the two instances of **XXXXX** with your environment number.

4 Save the file and keep File Explorer open.

Connect Workstation to your environment

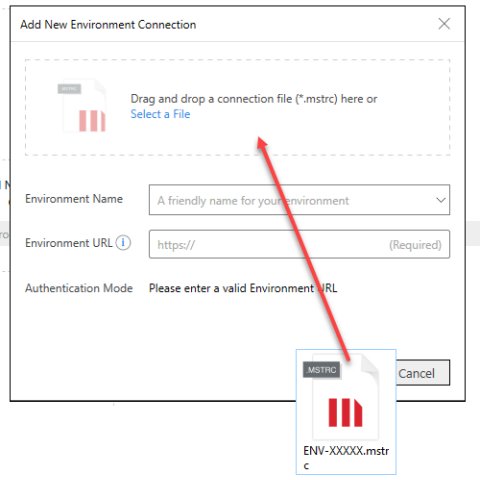
1 From the Windows desktop, double-click **MicroStrategy Workstation**.

2 In the left pane, click **Environments**.

3 Click **Add New Environment Connection**.

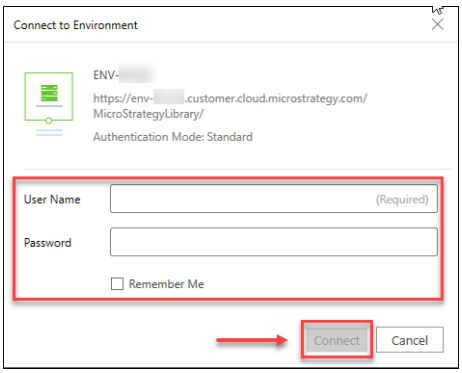
© 2021 MicroStrategy, Inc.

4 Drag and drop **ENV-XXXXX.mstrc** in the proper zone, as shown below.



The details of the connection will automatically be filled using the content of the file you just dragged.

5 Click **Continue**.



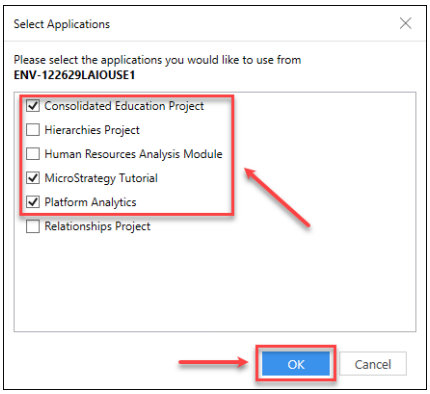
6 Enter the credentials from the Welcome to MicroStrategy Cloud email.

7 Click **Remember Me**.

8 Click **Connect**.

© 2021 MicroStrategy, Inc.

The Select Applications window opens.



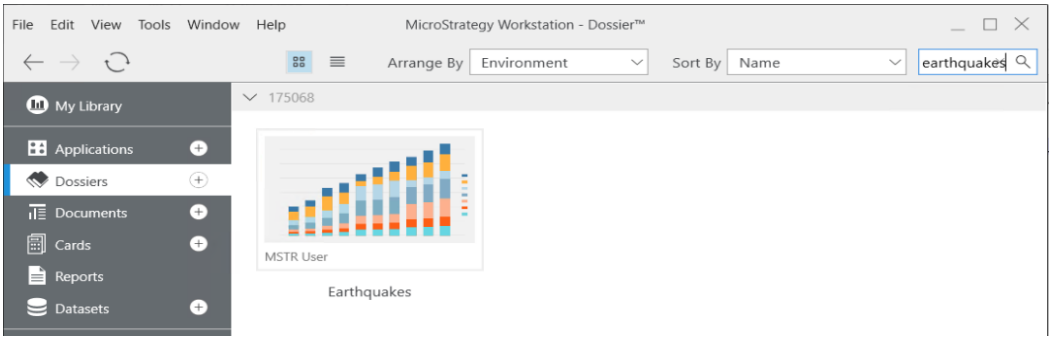
9 Select **MicroStrategy Tutorial**, **Platform Analytics**, and **Consolidated Education Project**.

10 Click **OK**.

The window closes and Workstation now displays the connected environment as shown below.

11 In the left pane, click **Dossiers**.

12 In the **Search** box, type **Earthquakes** and press **Enter**.



© 2021 MicroStrategy, Inc.

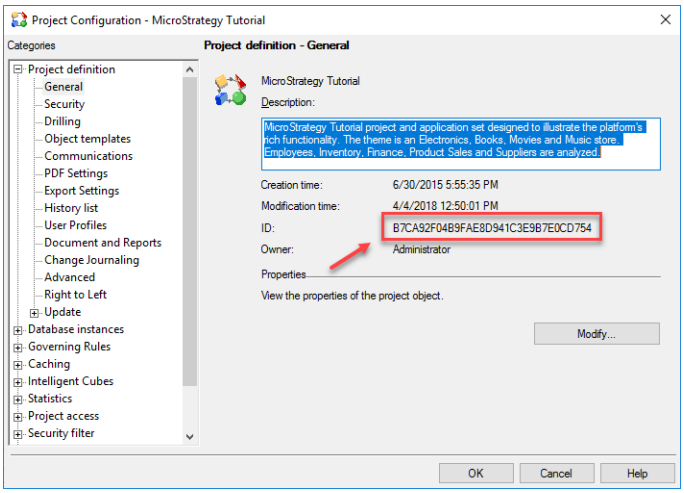
Copy the project and dossier IDs

1 From the Windows desktop, double-click **Developer**.

2 Log in using your MicroStrategy Cloud credentials.

3 Right-click **MicroStrategy Tutorial** and select **Project Configuration**.

4 Copy the **ID** as shown below.



5 Paste the value in a Notepad++ document.

6 Close the window.

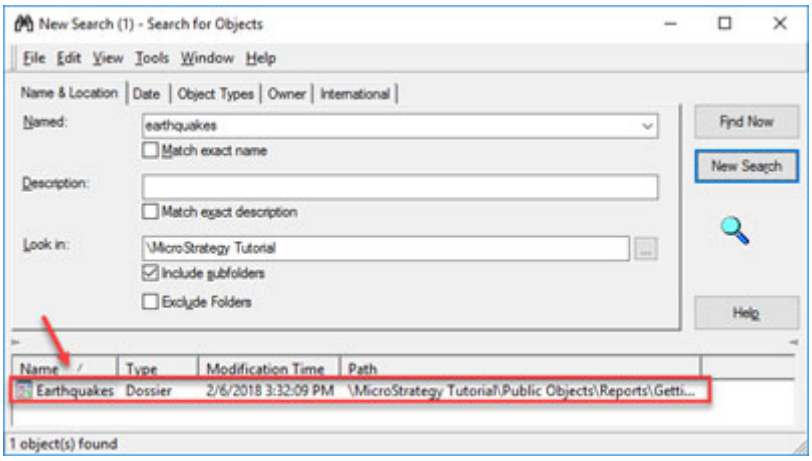
7 Expand **MicroStrategy Tutorial**.

8 On the toolbar, click **Search for objects in the project** .

9 In the **Named** box, type **earthquakes** and click **Find Now**.

© 2021 MicroStrategy, Inc.

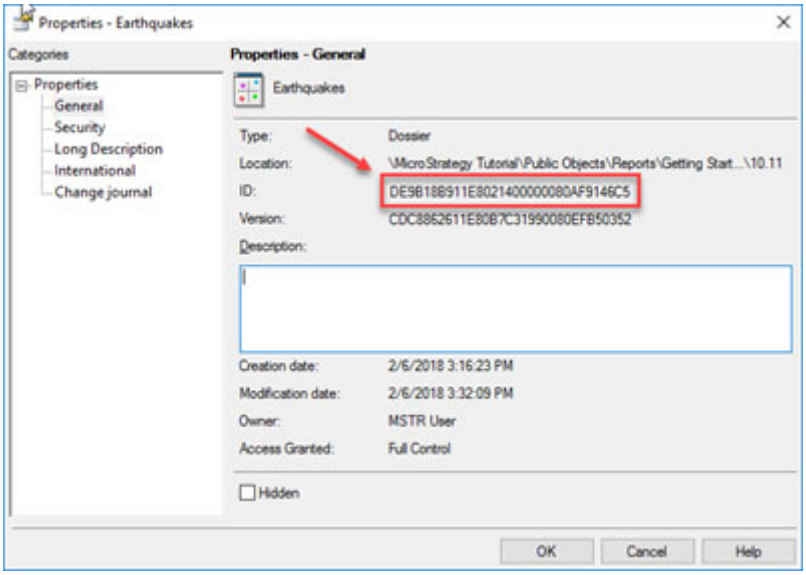
The Earthquakes dossier is displayed in the results area, as shown below.



10 Right-click **Earthquakes** and select **Properties**.

The Properties window opens.

11 Copy the **ID** as shown below.



12 Paste the ID in your Notepad++ file.

13 Click **OK** to close the Properties window.

14 Close the Search window, and click **No** to save the changes.

15 Close **Developer**.

© 2021 MicroStrategy, Inc.

Launch the API Explorer

1 Open a browser and navigate to the following address:

https://env-**XXXXXX**.customer.cloud.microstrategy.com/MicroStrategyLibrary/api-docs

where **XXXXXX** is your environment number.

2 Expand **Authentication**.

3 Use the POST /api/auth/login end point to obtain an authorization token. If you need a reminder, do the following:

a Click **POST /api/auth/login**.

b Click **Try It Out**.

c Modify the **username** value to **mstr**.

d Modify the **password** value to the password in the MicroStrategy Cloud email.

e Click **Execute**.

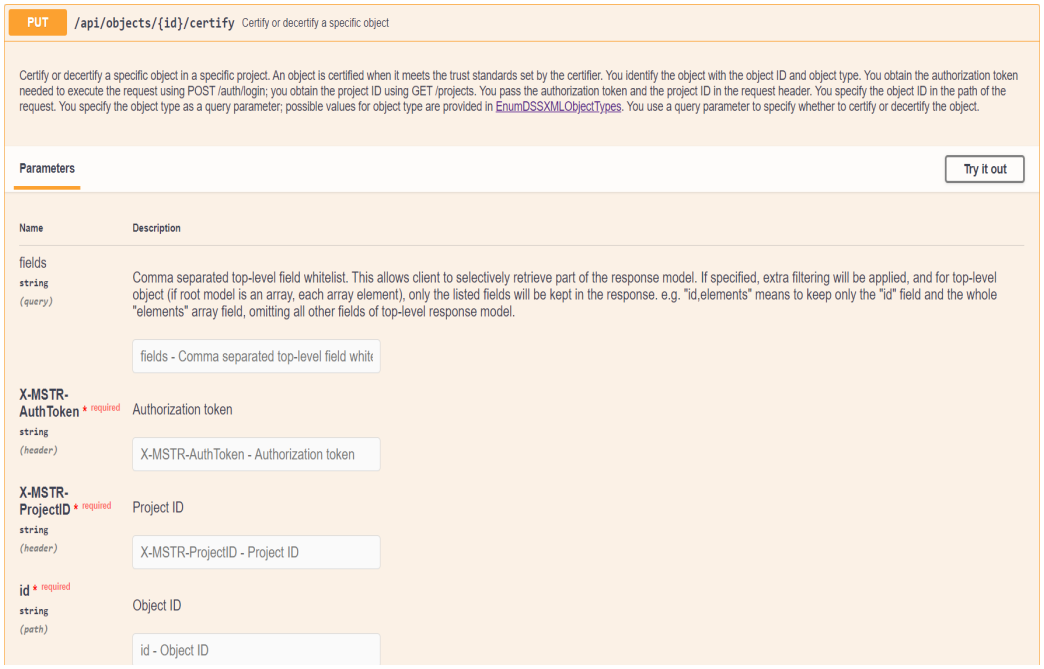
4 In the Response Headers section, copy the **x-mstr-authtoken** value and paste it in the Notepad++ document.

The authorization token is only valid until the login times out. This is why you retrieved the required IDs and token and pasted them in a convenient location.

5 Click **Object Management**.

© 2021 MicroStrategy, Inc.

6 Click **PUT /api/objects/{id}/certify**, as shown below.

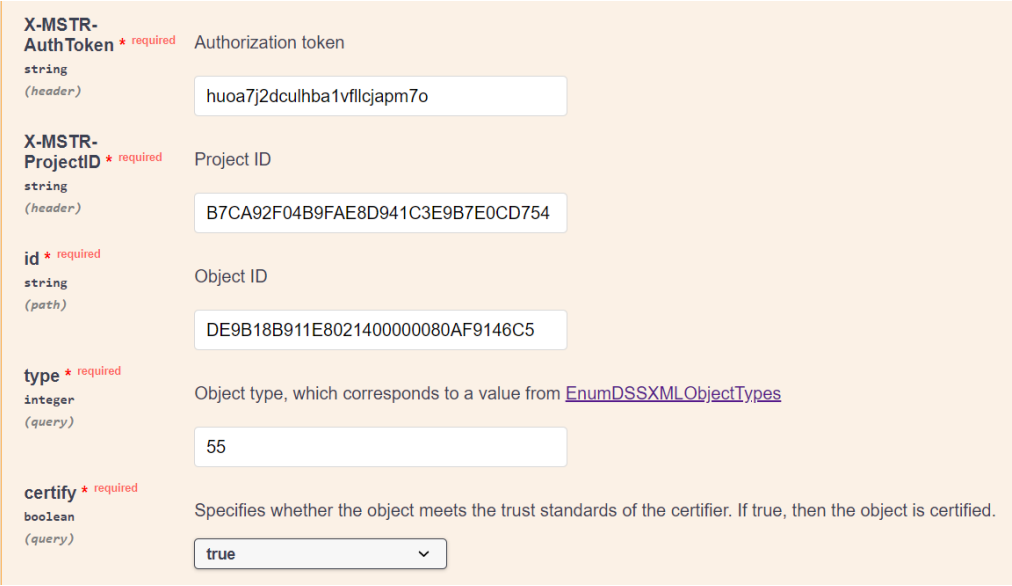


7 Click **Try It Out**.

© 2021 MicroStrategy, Inc.

8 In the Parameters area, paste the values from your Notepad++ document and the information in the table below. The following parameters are required.

Parameter	Value
X-MSTR-AuthToken	**Paste the value from your Notepad++ file**. The authorization token from the mstr user session.
X-MSTR-ProjectID	**Paste the value from your Notepad++ file**. The identifier for MicroStrategy Tutorial.
id	**Paste the value from your Notepad++ file**. The identifier for the Earthquakes dossier.
type	**55**. From the EnumDSSXMLObjectTypes enumeration's document identifier.
certify	**true**. Binary choice to certify the object.

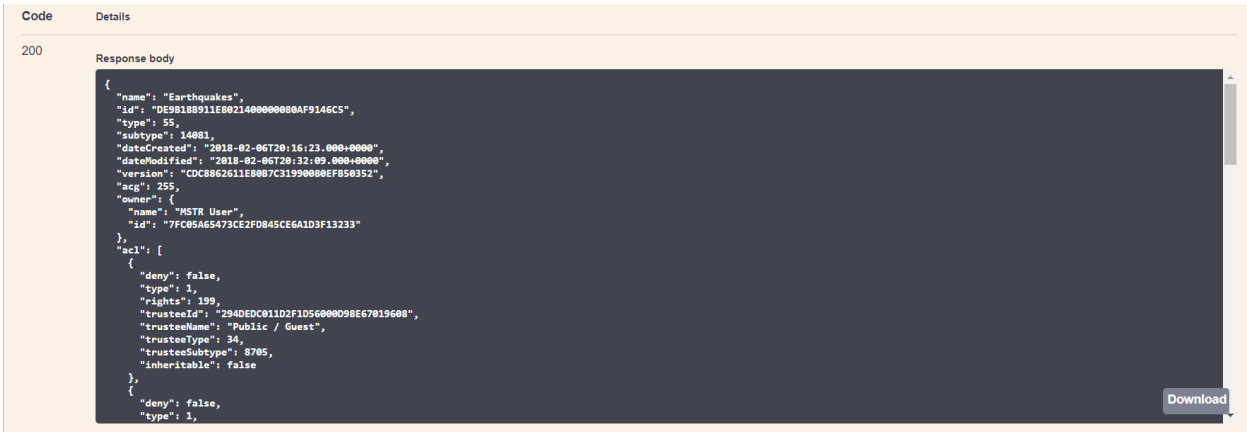


9 Click **Execute**.

In the Responses area, a successful code 200 is displayed.

© 2021 MicroStrategy, Inc.

The Response Body contains the object name, identifier, type, icon path, owner, ancestors (hierarchy of the path of the object in the project), permission and more.



The end of the response body contains relevant information about certification.

```
"certifiedInfo": {
  "certified": true,
  "dateCertified": "2018-04-16T17:31:21.665+0000",
  "certifier": {
    "id": "7FC05A65473CE2FD845CE6A1D3F13233",
    "username": "mstr",
    "fullName": "MSTR User"
  }
}
```

The date and time the dossier was certified and specific information about the user who performed the certification are displayed.

© 2021 MicroStrategy, Inc.

Verify the certification

Now that you have used the API to certify the Earthquakes dossier, verify that the certification was completed.

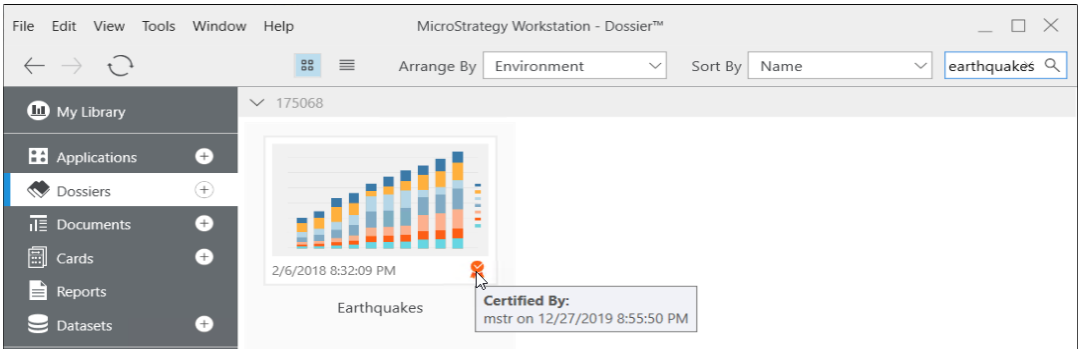
1 Open **Workstation** and open the environment connection you created.

2 If the Dossiers screen is already open, click **Applications**.

3 In the left pane, click **Dossiers**.

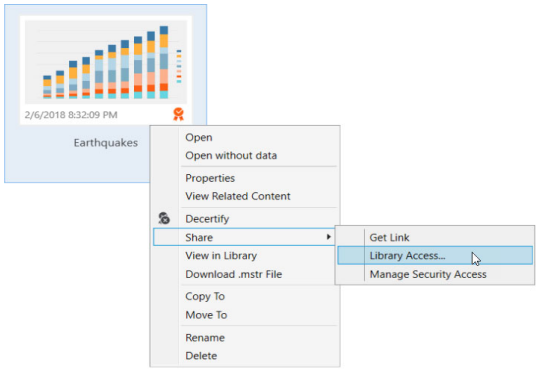
4 In the **Search** box, type **Earthquakes**.

The dossier icon contains the certification logo. Hover over the certification logo to see certification details.



Add the dossier to your Library

5 Right-click **Earthquakes**, click **Share**, and click **Library Access**, as shown below.

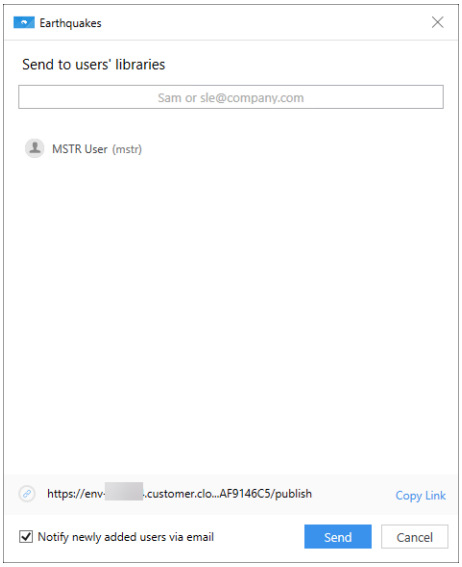


6 In the **Manage Users' Library Access** box, type **mstr**.

© 2021 MicroStrategy, Inc.

7 Click **MSTR User (mstr)**.

The mstr user is added to the list of recipients.



8 Click **Done**. The dossier is added to the mstr user's Library.

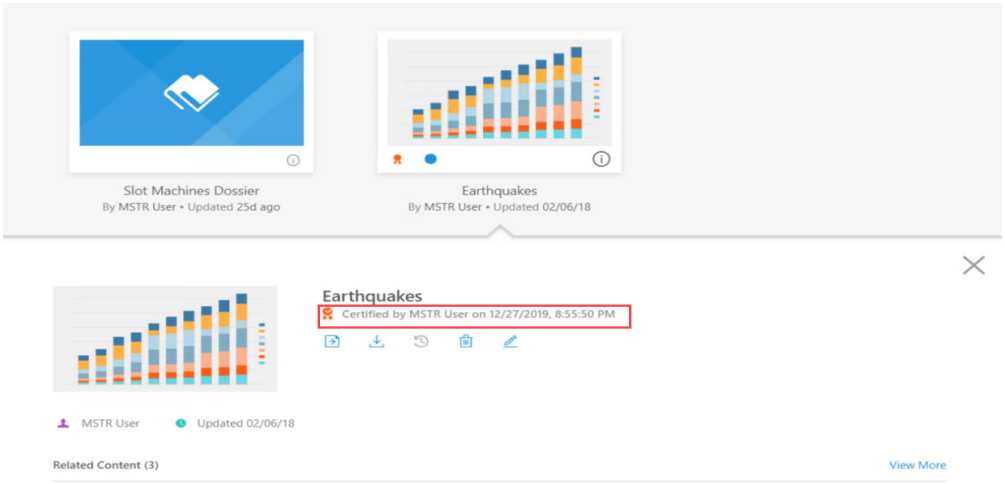
9 Open a browser tab and navigate to:

http://localhost:8080/MicroStrategyLibrary

10 Log in using the credentials from the MicroStrategy Cloud email.

11 Find the **Earthquakes** dossier.

12 Expand the info box to see the certification details as shown below.



© 2021 MicroStrategy, Inc.

Remove the certification

1. Go back to the API Explorer.
2. Use the POST /api/auth/login end point to obtain an authorization token.
3. Copy the authorization token and paste it in the Notepad++ document.
4. Click **Object Management**.
5. Click **PUT /api/objects/{id}/certify**, as shown below.
6. In the **X-MSTR-AuthToken** box, paste the authorization token.
7. Paste the project ID and dossier ID from your Notepad++ file.
8. From the **Certify** drop-down list, select **False**.
9. Click **Execute.**
10. When you receive the 200 response code, scroll down to the Response Body and identify the following lines:

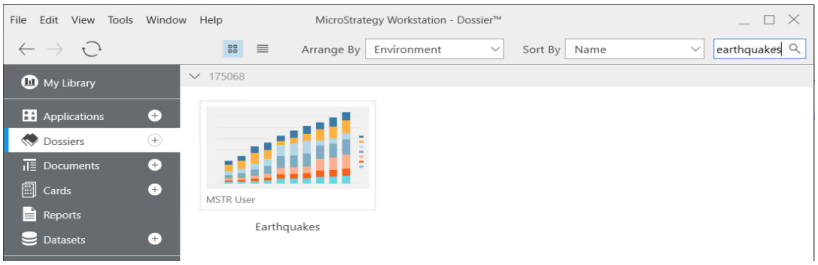
```
"certifiedInfo": {
    "certified": false
  }
```

    The certification was removed.

11. Return to Workstation.
12. In the left pane, click **Applications**, and then click **Dossiers**.
13. In the **Search** box, type **Earthquakes**.

© 2021 MicroStrategy, Inc.

Your dossier is no longer certified, as shown below.



Summary

In this chapter, you used a REST API end point to manipulate the certification status of a dossier. Now you can implement the certify end point into your own application to certify reporting objects.

Activity 4.2: Quiz

Answer the following questions. You will find the answers in *Quiz Answers, page 153*.

1 What is the MicroStrategy environment connection file extension?

a .mstrc

b .mstr

c .mstrx

d .mstrcc

---

2 Which Response Code is received when the certify end point call is successful?

a 200

b 204

c 300

d 304

e 401

 © 2021 MicroStrategy, Inc.

3 True or False: In the Response Body, the certifiedInfo area contains details like the certifying user and certification date.

_____

4 True or False: In the Response Body, the certifiedInfo section is removed when a dossier is no longer certified.

_____

© 2021 MicroStrategy, Inc.

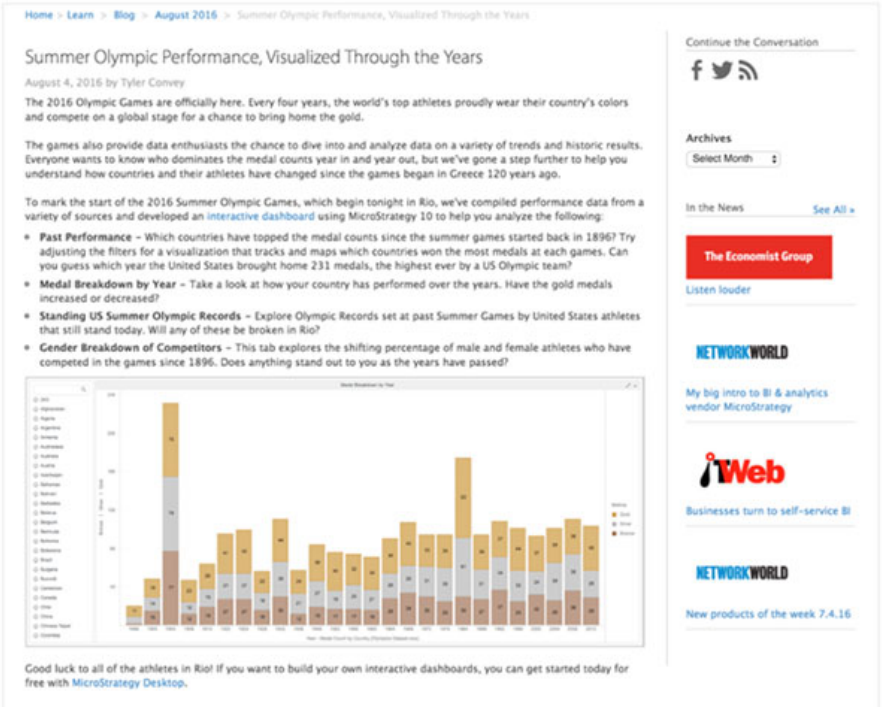
5

Embed a Dossier in an HTML Page

Dossiers are reporting objects that enable you to bring data to life with compelling visualizations and layouts that can be easily consumed by data analysts. Although you can create, distribute, and consume dossiers in

© 2021 MicroStrategy, Inc.

MicroStrategy, your organization might also want to display some of your dossiers on your web pages.



The MicroStrategy REST API contains several end points that help you consume and manipulate data and MicroStrategy objects. The Embedding Software Developer Kit (SDK) employs some of these endpoints in a JavaScript library to embed dossiers on web pages.

Creating custom applications with the Embedding SDK

The Embedding SDK contains a JavaScript library that enables you to display dossiers in any HTML page, providing access to content outside of MicroStrategy. Users who view your web page do not require any MicroStrategy experience to consume your data.

To embed a dossier built with MicroStrategy in an HTML page you need at minimum:

- A link to the MicroStrategy Library JavaScript library
- A destination for the dossier
- JavaScript code to retrieve the dossier

 © 2021 MicroStrategy, Inc.

Linking to the JavaScript library

Your installation of MicroStrategy Library contains the JavaScript embedding library that contains the code required to display dossiers on web pages. Link to the embeddinglib.js library in the MicroStrategy Library installation location to ensure you are using the latest installed version. Alternatively, you can install the library directly on your website.

To associate your web page with the embeddinglib.js library, create a reference to the library, as in the following example:

```
<script type="text/javascript" src="http://
localhost:8080/MicroStrategyLibrary/javascript/
embeddinglib.js"></script>
```

In the example link, the **src** value is the URL (or path) to your MicroStrategy Library application. In the code excerpt above, the JavaScript library is located on your local web site. This application only functions for a user who reaches the web page from the computer that hosts the web site.

The link you create in your organization must accommodate access from outside the local host. For your Cloud environment, the following link format can be accessed externally:

```
<script type="text/javascript" src="http://
env-XXXXX-sdk.customer.cloud.microstrategy.com:8080
/MicroStrategyLibrary/javascript/
embeddinglib.js"></script>
```

where **XXXXX** represents your environment number.

To access the MicroStrategy Library located in the Linux component of your MicroStrategy Cloud environment, the link is in the following format:

```
<script type="text/javascript" src="https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary/javascript/embeddinglib.js"></
script>
```

where **XXXXX** represents the environment number.

Mixing http and https content may lead to browser loading errors.

To ensure that MicroStrategy Library is able to embed information on external sites, you must also configure CORS (Cross Origin Resource Sharing).

Specifying a location for the dossier

When you design your web page, create a specific location to hold your dossier. To do this, insert a DIV section in the HTML code, as in the following example:

```
<TR>
    <TD>
        <DIV id="leftMenu">
            <H2>Table of Content</H2>
        </DIV>
    </TD>
    <TD>
        <DIV id="dossierContent">
        <!-- Dossier will appear here -->
        </DIV>
    </TD>
</TR>
```

The code above depicts a single row in a two column table. The left cell displays a Table of Contents (not related to MicroStrategy content), and the cell on the right contains a DIV structure named dossiercontent, which serves the following purposes:

- An identifier for the div to which a visual style can be applied.
- An identifier for the div to which the JavaScript code anchors the dossier.

Retrieving the dossier with JavaScript code

Once the embeddinglib.js library is linked to your web page, you can implement its methods to request and manage dossier content, as in the following simple example:

```
<script type="text/javascript">
    //Get the div from the web page that will hold
the MicroStrategy dossier
```


© 2021 MicroStrategy, Inc.

```
    var dossierContainer =
document.getElementById("dossierContent");

    // Define the Dossier URL (can be found with
Workstation)

        var secureCloudBaseURL = "https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary/";

        var dossierURL = secureCloudBaseURL + "app/
B7CA92F04B9FAE8D941C3E9B7E0CD754/
81B1C74711E791A31E480080EF75C3E0";

        // Show Dossier

        microstrategy.dossier.create({

            url: dossierURL,

            placeholder: dossierContainer

        });

</script>
```

The code above defines an anchor (the dossierContent div) for the dossier content. The secureCloudBaseURL variable is defined to hold the location of the MicroStrategy Library installation that contains the dossier. This value is appended to the project and dossier identifiers to complete the dossier address.

In the long URL in the snippet above, the string B7CA92F04B9FAE8D941C3E9B7E0CD754 is the project ID while 81B1C74711E791A31E480080EF75C3E0 is the dossier ID.

Both the project and dossier IDs can be found using MicroStrategy Web or Workstation.

The .create method is the actual hero, passing as arguments the source of information and the destination on the page.

Many more parameters are available. We will discover some of them as we do some exercises later in the chapter.

First, we need to prepare the dossier we will use for this chapter.

© 2021 MicroStrategy, Inc.

Exercise 5.1: Import a dossier into your environment

You are working as a developer at a video gaming console company and you want to showcase research data on the corporate web site.

In this exercise, import a dossier into your environment, make a few enhancements, and place the dossier in Library.

Import a .mstr file

1 Open Chrome and navigate to:

https://env-**XXXXXX**.customer.cloud.microstrategy.com/MicroStrategy/servlet/mstrWeb

Where **XXXXXX** represents the environment number.

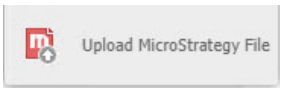
2 Log in using the credentials from your MicroStrategy Cloud environment.

3 Click **MicroStrategy Tutorial**.

4 On the left, click **My Reports**.

5 Click **Create**.

6 Click **Upload MicroStrategy File**, as illustrated below.



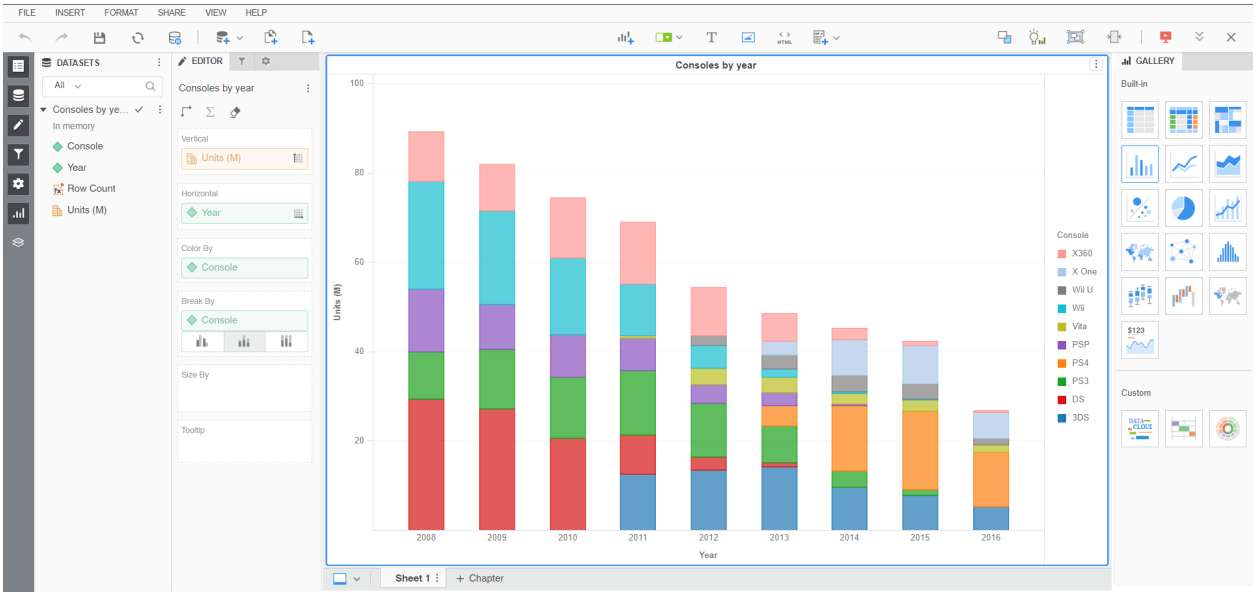
7 Navigate to **Desktop\REST-API\Exercises\Chapter 5**.

8 Select **Consoles by year.mstr.** Click **Open**.

9 In the confirmation dialog, click **View Dossier**.

© 2021 MicroStrategy, Inc.

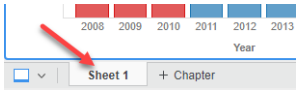
The dossier opens as illustrated below.



Enhance the dossier

Add a new chapter to the dossier to present the data in a different way.

1 Double-click **Sheet 1**and type **By console**.

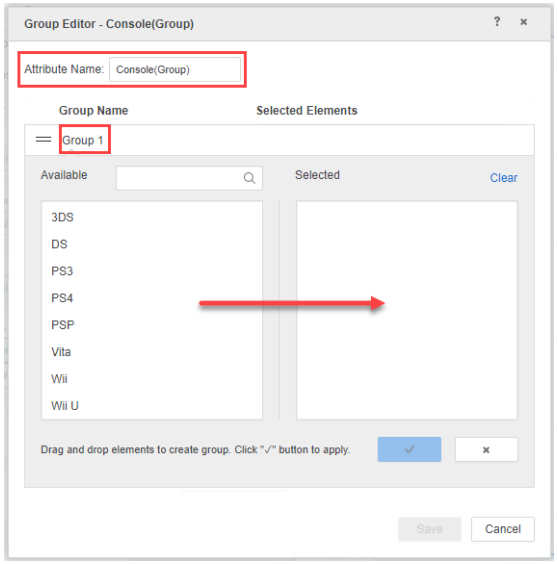


2 Right-click **By console** and select **Duplicate Chapter**.

3 Double-click the new tab name and replace it with **By MFG**.

4 In the Editor pane, in the Break By area, right-click **Console** and select **Create Groups**.

The Group Editor opens, as illustrated below.

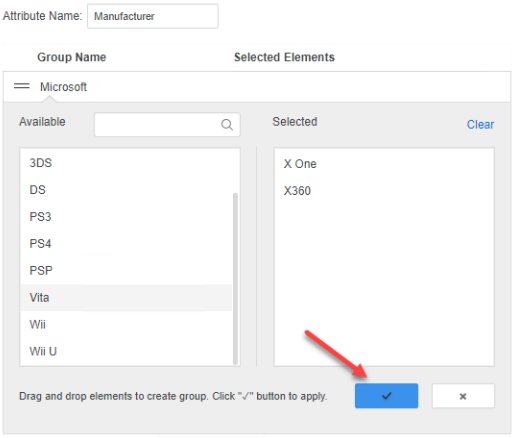


To create a group, you can combine attribute elements into a new attribute. You define the name of the new attribute, provide a group name, and then select the attributes elements.Once the attribute is created, you can use it like any other attribute in any visualization.

5 Change the **Attribute Name** to **Manufacturer**.

6 Change **Group 1** to **Microsoft**.

7 Drag the elements **X One** and **X360** to the Selected box.



8 Click the blue Check mark to apply your selections.

© 2021 MicroStrategy, Inc.

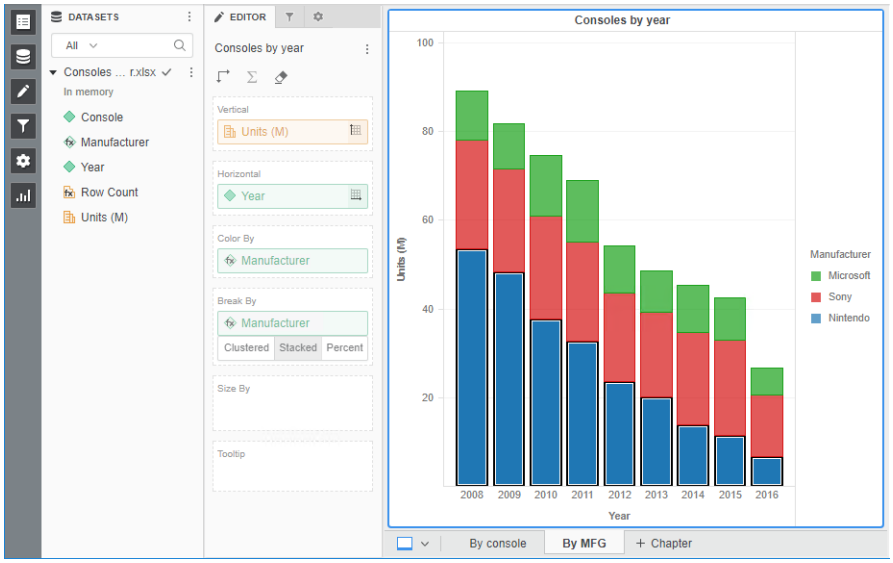
9 Using the previous steps as a guide, and the **Add a Group** button, create the two groups in the following table.

Manufacturer	Consoles
Sony	PS3, PS4, PSP, Vita
Nintendo	3DS, DS, Wii, Wii U

The groups are listed in the Group Editor.

10 Click **Save**.

The second chapter displays data by the groups you just created.



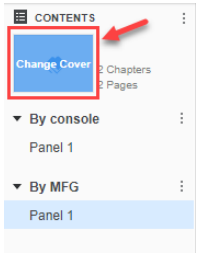
11 **Save** the dossier.

Change the dossier thumbnail

To distinguish you dossier, change its thumbnail image.

1 Below the Consoles by Year visualization, open the navigation menu and select **List View**.

2 In the Table of Contents, click the dossier cover icon.



The Change Cover window opens and displays cover images. You can also provide your own image URL.

3 Select one of the images and click **Save**.

Your new thumbnail image is displayed in the Contents pane.

4 In the Contents pane, click the menu (three vertical dots), and select **Tab View -Bottom**.

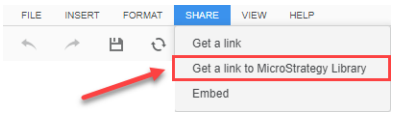
5 **Save** the dossier.

© 2021 MicroStrategy, Inc.

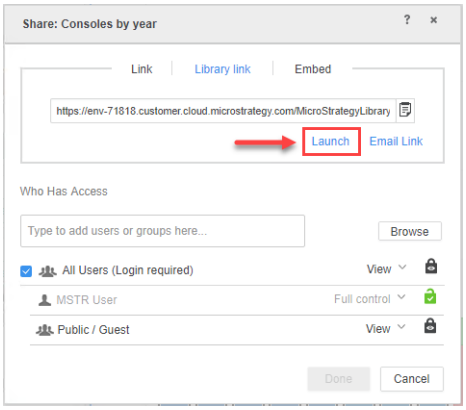
Share the dossier to Library

Now that your dossier has been enhanced with a second chapter, share the dossier with MicroStrategy Library.

1 From the **Share** menu, click **Get a link to MicroStrategy Library**.



The Share window opens.



2 Click **Launch**.

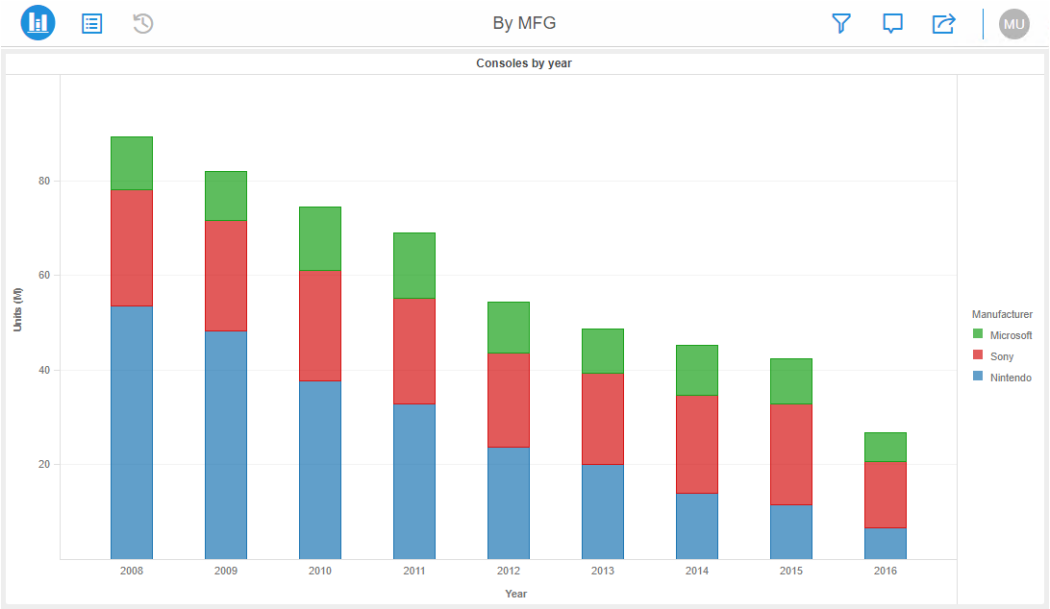
If no new tab opens, the Chrome pop up blocker is in effect for this site. Click the icon with the red x in the address bar and authorize pop ups from this site and click Launch again.

3 Log in with the credentials from your MicroStrategy Cloud email. A new tab opens in your browser and displays the dossier.

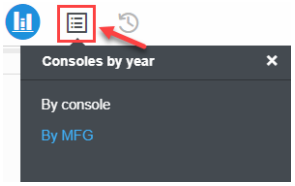
4 Click **Add to Library**.

© 2021 MicroStrategy, Inc.

The dossier is displayed as illustrated below.

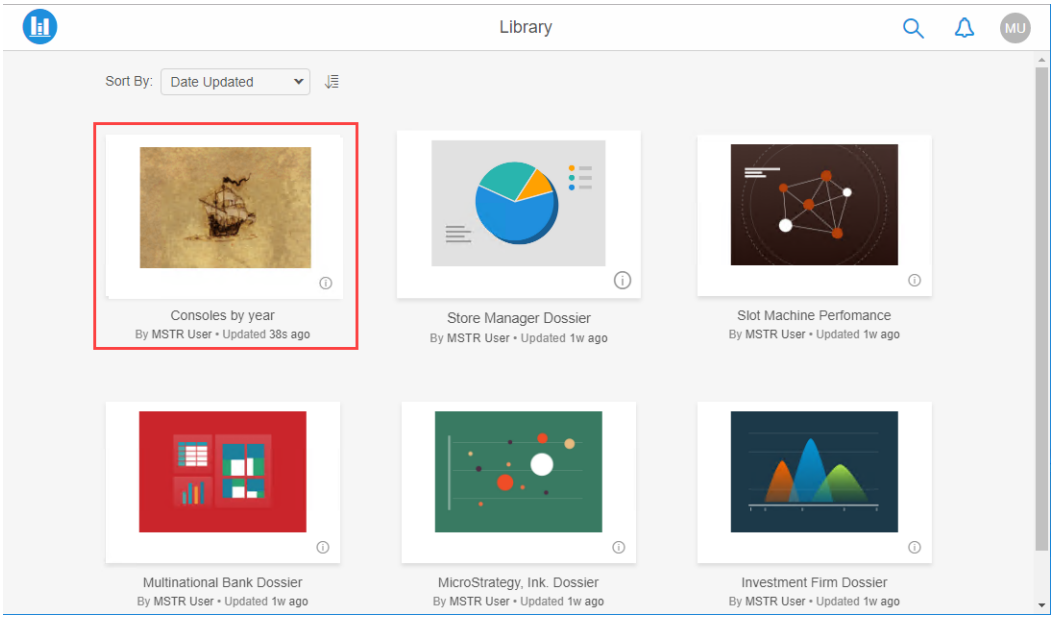


5 Open the Table of Contents, and select **By console**.



© 2021 MicroStrategy, Inc.

6 Click **Library** (the blue icon on the left of the menu area).



Now that the dossier is in the Library, we are ready to use it in our own HTML pages.

Exercise 5.2: Configure the development environment

The Embedding SDK enables you to create an application that displays MicroStrategy content. In this exercise, prepare the environment to deploy your web application and collect the information required to retrieve the dossier.

Set up the development applications in your environment

1. From the Windows desktop, right-click **Tomcat_Stop** and select **Run as Administrator** to stop the Tomcat server.
2. In the User Account window, click **Yes**.
3. Double-click **Google Chrome** and do the following:
   a. In the top right corner, click **Customize and Control Google Chrome**.
   b. Point to **More Tools** and select **Clear browsing data**.
   c. Click **Clear Data**.
4. From File Explorer, navigate to **Desktop\REST-API\Exercises\Chapter 5** and copy the following files:
   - **CreateAuthToken.html**
   - **Embedding.html**
   - **Embedding2.html**
5. Navigate to **C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT**.
6. Create a folder named **Embed** to host your web pages.
7. Paste the three files inside the **Embed** folder. Click **Continue** at each request.
8. From the desktop, right-click **Tomcat_Restart** and select **Run as Administrator** to restart the Tomcat server.

© 2021 MicroStrategy, Inc.

Retrieve dossier identifiers

To retrieve a dossier and embed it on a web page, collect the project and dossier identifiers.

1. Log in to your web environment using the following link:

   https://env-**XXXXXX**.customer.cloud.microstrategy.com/MicroStrategy/servlet/mstrWeb

   where **XXXXXX** is your environment number.

2. Log in using the credentials from the MicroStrategy Cloud email.
3. Click **MicroStrategy Tutorial**.
4. Click **Go to MicroStrategy Web**.
5. In the left panel, click **My Reports**.
6. Right-click **Consoles by Year**, and select **Properties**.
7. In the Properties window, copy the **ID** value.
8. Paste this value in a blank document in **Notepad++**.
9. Use Workstation or Developer to retrieve the **Project ID** for **MicroStrategy Tutorial**. Refer to previous exercises if you need a refresher.
10. Double-click **Notepad++**, and paste this value.

You now have the identifiers required to complete the embedding process.

Exercise 5.3: Create an authorization token

Each time you connect to the MicroStrategy REST API server by making end point calls or using the embedding library, you must use an authorization token that authenticates your MicroStrategy user.

In this exercise, use the CreateAuthToken.html page to learn about authorization tokens.

Edit the CreateAuthToken.html page

1. In File Explorer, navigate to **C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\Embed**.
2. Right-click **CreateAuthToken.html**, and select **Edit with Notepad++**.
3. Locate line 22:

```
var baseRestURL = "https://
env-xxxxxx.customer.cloud.microstrategy.com/
MicroStrategyLibrary";
```

4. Replace **XXXXXX** with your environment number.
5. Locate line 23:

```
var username = "user";
```

6. Change **user** to **mstr**.
7. Locate line 24:

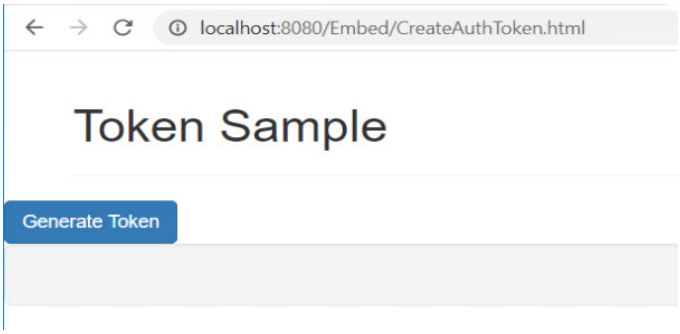
```
var password = "passwd";
```

8. Change **passwd** to the password in the Welcome to MicroStrategy Cloud email.
9. **Save** your changes. If required to relaunch in Administrator mode, click **Yes** and save again.
10. In Chrome, navigate to

© 2021 MicroStrategy, Inc.

http://localhost:8080/Embed/CreateAuthToken.html

The following page is displayed:



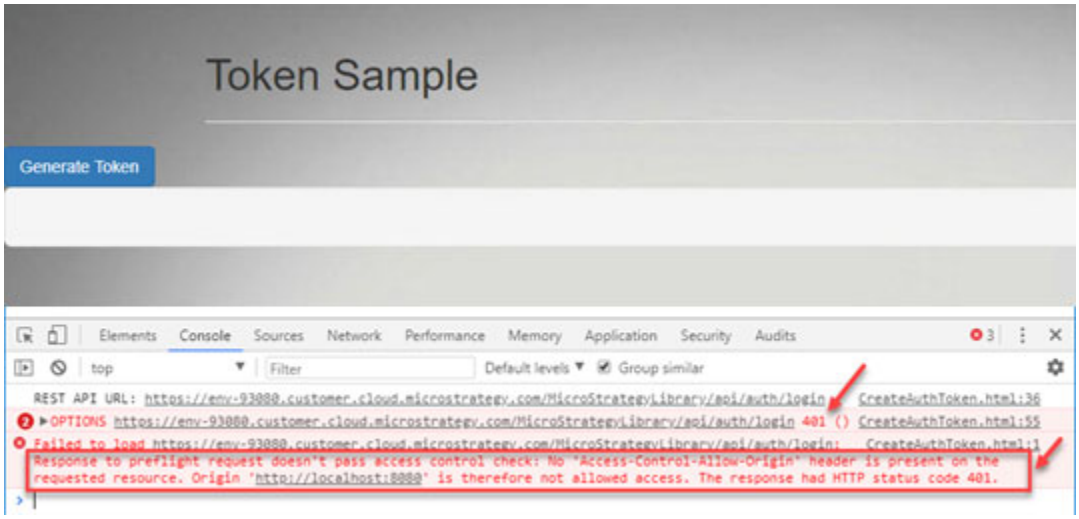
11 Click **Generate Token**. An authorization token value is displayed under the button. You can use this token in your web application to authenticate to MicroStrategy.

Configuring Library to distribute content across domains

By default, your MicroStrategy Cloud environment is configured to distribute content from Library to a web page on another domain. This configuration enables you to install Library on one server, and request content from a web application installed on another server.

If MicroStrategy Library is configured to not allow content distribution across domains, you would see the following error when you request a token from a web

page installed on localhost that access Library on your Cloud environment's Linux server:



Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to access selected resources from a server on a different domain. A user agent makes a cross-origin HTTP request when it requests a resource from a different domain, protocol, or port than the one from which the current document originated.

In our example, a page located on localhost requests content from https://env-XXXXXX.customer.cloud.microstrategy.com/MicroStrategyLibrary/api/auth/login. Those two domains are different, hence the CORS response.

To implement CORS properly on an on-premises installation and allow cross domain requests, you must adjust the settings in your Web server. In this example, we use the Tomcat instance located on the Linux server.

If the embedding web site files were installed on the Tomcat server located on the Linux server, there would be no error because the pages and calls are served from the same domain. Similarly, on the Cloud environment, CORS does not need to be configured because the Library administrative options have been configured to allow embedding on all external pages.

Adapt the steps in this section to configure CORS on a MicroStrategy Library installation in a Linux environment. Alternatively, you can configure Library to allow embedding on external pages through the Library admin page.

The steps in this section are not to be performed in-class; they are for informational purposes only.

© 2021 MicroStrategy, Inc.

Adjusting the Tomcat server on the Linux server

1 Launch **VNC** from the desktop.

2 Open the session you created earlier.

3 Double-click **Home**.

4 Click **Other Locations**.

5 Double-click **Computer**.

6 Navigate to **/opt/apache/tomcat/latest/webapps/MicroStrategyLibrary/WEB-INF/classes/config**.

7 Right-click **configOverride.properties**, and select **Open with Application**.

8 Click **Text Editor**, and click **Select**.

9 Enter the following lines at the end of the file.

```
X-Frame-Options=SAMEORIGIN

auth.cors.origins=*
```

10 Click **Save**. Close the text editor.

11 Right-click **security_headers-index.properties**, and select **Open with Application**.

If the file does not exist, you do not need to create it. Skip to Restarting the Web server.

12 Click **Text Editor**, and click **Select**.

13 Replace the word **SAMEORIGIN** with **ALLOW-FROM *** to allow requests from all domains.

To create a more restrictive security policy, instead of *, enter the specific domain to authorize, for example, https://www.mysite.com.

14 Click **Save**. Close the text editor.

© 2021 MicroStrategy, Inc.

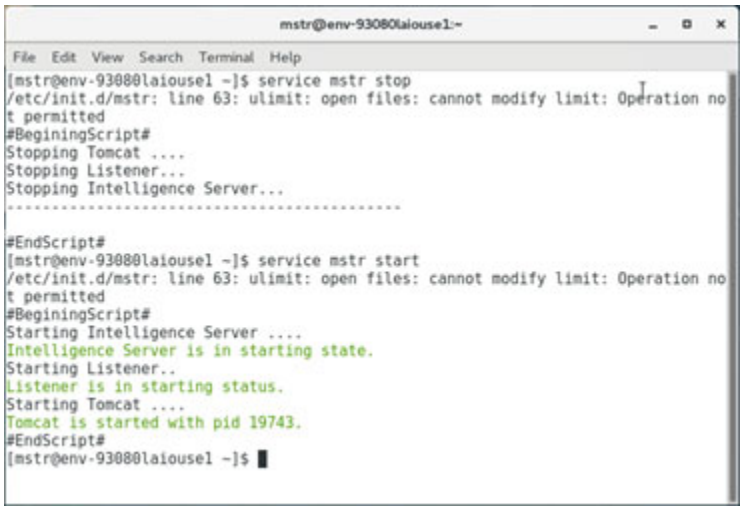
Restarting the Web server

To apply your changes to MicroStrategy Library, restart the Tomcat server.

1. From the **Applications** menu, point to **System Tools**, and click **Terminal**.
2. Enter the following command: **service mstr restart**.

   Wait a few minutes to allow the Intelligence Server to completely restart.
3. Click **Reconnect**.
4. Enter the following command: **service mstr status**.

   Three messages in green indicate that all components restarted properly, including Tomcat.
5. Close the VNC session.



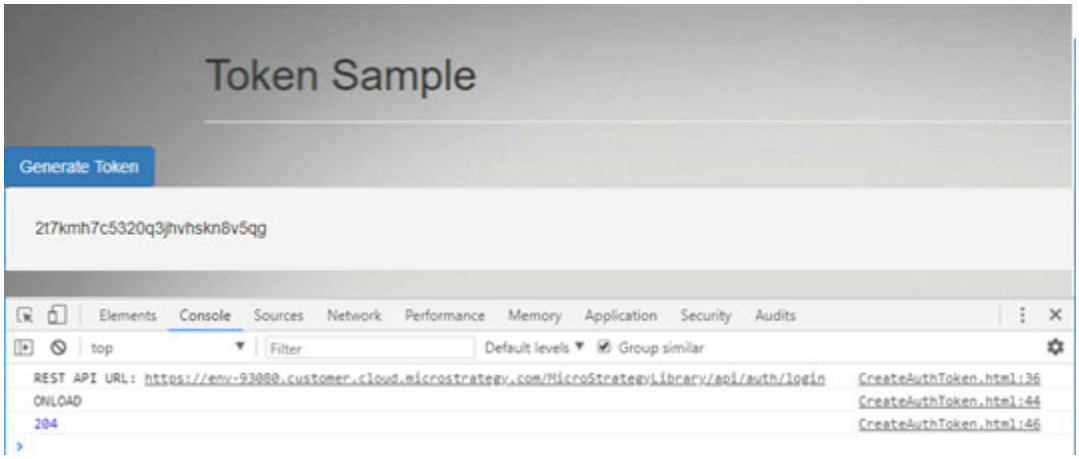
Testing the CreateAuthToken.html again

Now that you have configured Library to process data requests from external domains, test the Token Sample page to verify your updates.

1. Return to the browser and refresh the Token Sample page.
2. Click **Generate Token**.

 © 2021 MicroStrategy, Inc.

This time the request is processed and a token is returned, as in the following example.



3 Close the Developer tools. MicroStrategy Library is now configured to allow data retrieval from outside domains.

Exercise 5.4: Embed a dossier on a page

The embedding web site files contain generic code that must be modified to connect to your environment. In this exercise, modify Embedding.html to do the following:

- Connect to the embedding JavaScript library located on your local installation of MicroStrategy Library
- Retrieve the dossier from the local installation of MicroStrategy Library
- Use the dossier and project identifiers you retrieved
- Authenticate with the mstr user

Modify the Embedding.html page

1. Using File Explorer, navigate to **C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\Embed**.
2. Right-click **Embedding.html** file and select **Edit with Notepad++**.
3. Locate line 21:

```
<script src="https://
env-xxxxx.customer.cloud.microstrategy.com/
MicroStrategyLibrary/javascript/
embeddinglib.js">
```

4. Modify the line to point to localhost:

```
<script src="http://localhost:8080/
MicroStrategyLibrary/javascript/
embeddinglib.js"></script>
```

5. Locate the line that starts with // Begin Config Parameters.
6. Find the following line:

```
baseRestURL="https://
env-xxxxx.customer.cloud.microstrategy.com/
MicroStrategyLibrary";
```


© 2021 MicroStrategy, Inc.

7 Modify the line to:

```
baseRestURL = "http://localhost:8080/
MicroStrategyLibrary";
```

These changes point to the local instance of the MicroStrategy Library on your Windows Server.

8 Find the following line:

```
username = "user";
```

9 Change user to **mstr**.

10 Find the line:

```
password = "passwd";
```

11 Change **passwd** to the password from the MicroStrategy Cloud email.

The embedding API also supports guest authentication, which enables you to embed data that does nor require authentication to MicroStrategy Library. No special coding is necessary for guest authentication other than the embedding JavaScript to display a dossier.

12 Find the line:

```
Project ID = "yyyy";
```

13 Change **yyyy** to the **project ID** saved in your Notepad++ text file.

14 Find the line:

```
dossier ID = "zzzz";
```

15 Change **zzzz** to the **dossier ID** saved in your Notepad++ text file.

16 **Save** Embedding.html. If Notepad++ request Administrator mode, click **Yes** twice, and then save your file.

© 2021 MicroStrategy, Inc.

Locate the embedding code

Before you test the web page, identify the code that embeds the dossier.

1 Locate the line:

```
<div id="dossierContainer1" style="width: 80%;
height: 100%;"></div>
```

The dossierContainer1 div is the web page position where the dossier is going to be embedded.

2 Locate the lines:

```
microstrategy.dossier.create({

        placeholder:
document.getElementById("dossierContainer1"),
```

The .create method points to the dossierContainer div as the destination and renders the dossier.

Test the page

1 Right-click **Tomcat_Restart** and select **Run as Administrator**.

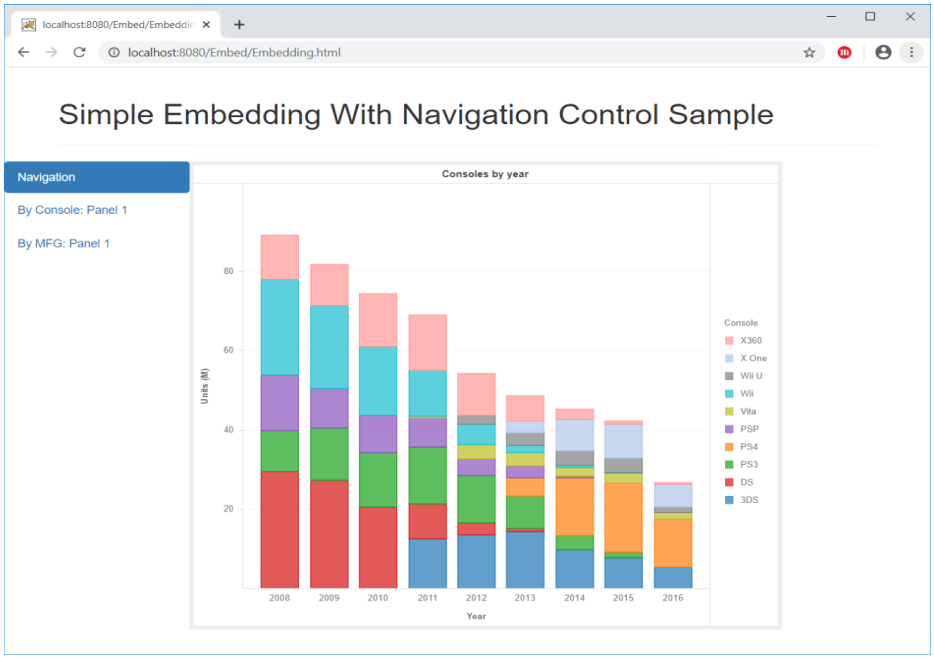
2 In the User Account window, click **Yes**.

3 In Chrome, navigate to:

http:/localhost:8080/Embed/Embedding.html

© 2021 MicroStrategy, Inc.

The dossier is embedded on your web page. Change the size of your browser window, and notice that the dossier automatically resizes to fit the available screen real estate.



Exercise 5.5: Create an interactive web page

You have been tasked with creating a web page that displays data about famous pirates. You want to create an interactive experience that enables data analysts to make selections in one dossier to filter data in a second dossier.

In this exercise, leverage the JavaScript Embedding API to embed a dossier and capture selection events from one dossier and apply them as a filter to a second dossier.

Import the data

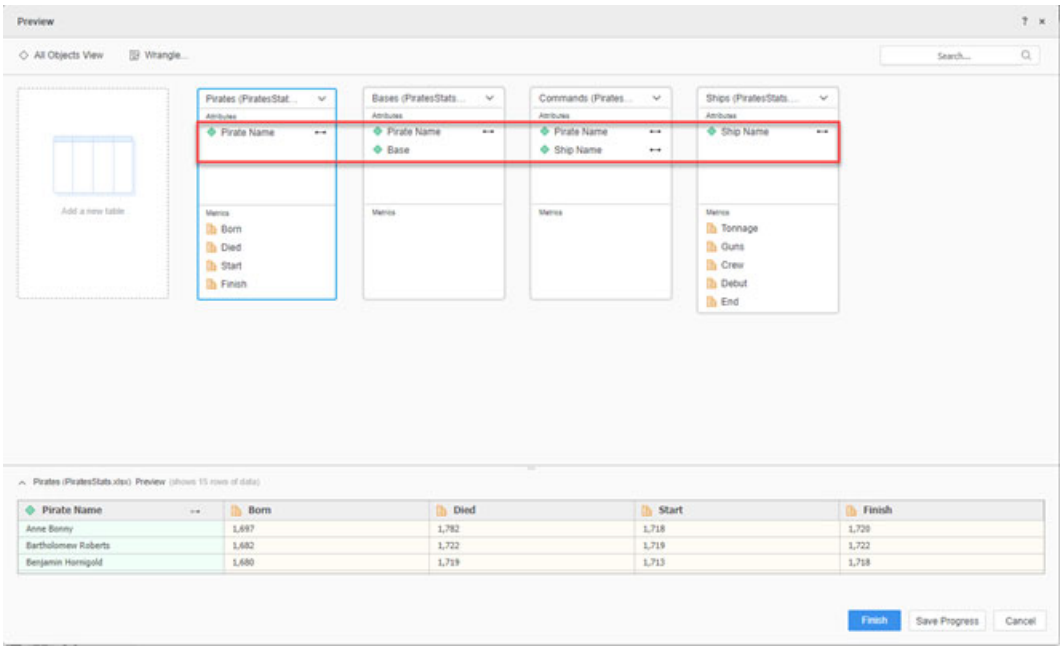
1. In Chrome, log in to the local instance of MicroStrategy Web:

   http://localhost:8080/MicroStrategy/servlet/mstrWeb

2. Navigate to **MicroStrategy Tutorial**.
3. Log in with the credentials from your MicroStrategy Cloud email.
4. Click **Create** and select **Add External Data**.
5. In the Connect to Your Data window, click **File from Disk**.
6. Click **Choose Files**.
7. Navigate to **Desktop\REST-API\Exercises\Chapter 5**. and double-click **PirateStats.xlsx**.
8. Click **Prepare Data**.
9. Select **Pirates**, **Bases**, **Commands**, and **Ships**, and click **Select**.

© 2021 MicroStrategy, Inc.

The Preview window displays the table structure. Notice that the tables are linked by specific attributes.



10 Click **Finish**.

11 In the **Name** box, type **PiratesImport** and click **Save**.

12 In the Start Your Analysis window, click **Create Dossier**.

Create the first dossier

Now that your pirate data is imported into MicroStrategy, create your first dossier.

1 From the Datasets pane, drag **Pirate Name** into the **Rows** drop-zone in the Editor pane.

2 Drag the following metrics to the **Metrics** drop-zone:

- **Born**
- **Start**
- **Finish**
- **Died**

© 2021 MicroStrategy, Inc.

3 Hide the Visualization1 title bar. To do this, in the visualization header, click the menu (three vertical dots) and select **Hide Title Bar**.

4 Click **Save**.

5 In the Name box, type **Pirate1** and click **OK**.

6 Click **Run newly saved dossier**.

7 Close the dossier.

Create the second dossier

1 Click **Create** and select **New Dossier**.

2 In the Datasets pane, click **Existing Dataset**.

3 From **My Reports**, click **PiratesImport** and click **Select**.

4 Open the menu for Visualization 1, and click **Change Visualization**.

5 From the list of visualization types, select **More**, and click **Heat Map**.

6 Close the Change Visualization window.

7 Drag the following attributes to the **Grouping** drop-zone:

- **Pirate Name**
- **Base**
- **Ship Name**

8 Drag **Crew** to the **Size By** drop-zone.

9 Drag the following metrics to the **Tooltip** drop-zone:

- **Born**
- **Died**
- **Guns**
- **Tonnage**
- **Debut**

© 2021 MicroStrategy, Inc.

- **End**

10 Hide the title bar.

11 Click **Save**.

12 Save the dossier in **My Reports** and name it **Pirate2**.

13 Click **Run newly saved dossier**.

14 Close the dossier.

Get the dossier identifiers

To display the two dossiers on your web page, retrieve the dossier and project identifiers.

1 From the MicroStrategy Tutorial Shared Reports screen, in the left pane, click **My Reports**.

2 Right-click **Pirate1** and select **Properties**.

3 Copy the ID value and paste it in a Notepad++ document.

4 Copy and paste the ID for **Pirate2**.

5 Using Developer, retrieve the ID for MicroStrategy Tutorial.

6 Paste the Project ID in your Notepad++ document.

Connect the web page to your environment

Connect to the embedding library and specify the environment variables.

1 Navigate to **C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\Embed**.

2 Right-click **Embedding2.html** and select **Edit with Notepad++**.

© 2021 MicroStrategy, Inc.

3 Locate line 32:

```
<script src="https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary/javascript/
embeddinglib.js"></script>
```

4 Replace **XXXXX** with your environment number.

5 Locate line 35:

```
baseRestURL = "https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary";
```

6 Replace **XXXXX** with your environment number.

7 Locate the following line:

```
username = "user";
```

8 Replace **user** with **mstr**.

9 In the **password** line, replace **passwd** with the password from the Welcome to MicroStrategy Cloud email.

10 In the **projectID** line, replace **PPP** with the MicroStrategy Tutorial Project ID stored in your Notepad++ document.

11 In the **dossier1ID** line, replace DS1 with the ID for Pirate1 stored in your Notepad++ document.

12 In the **dossier2ID** line, replace DS2 with the ID for Pirate 2 stored in your Notepad++ document.

13 **Save** your file. If you see a message about Administrator mode, click **Yes** twice and save again.

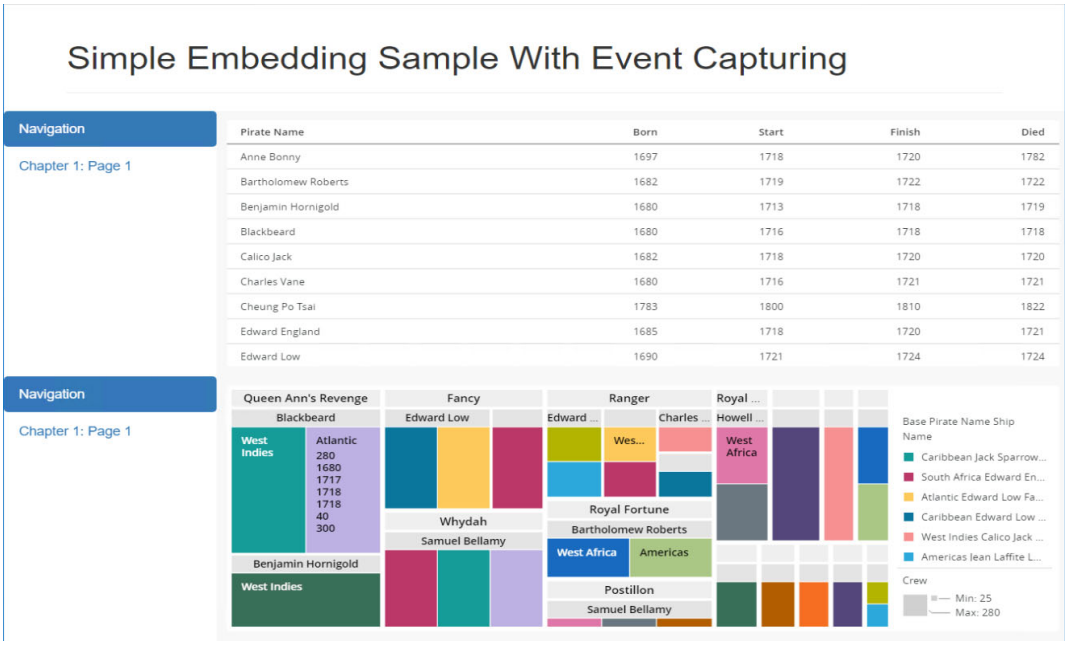
Test the Embedding2 web page

1 Open a browser and navigate to:

http:/localhost:8080/Embed/Embedding2.html

© 2021 MicroStrategy, Inc.

A web page with two embedded dossiers is displayed.



2 Click any Pirate Name in the top dossier. Nothing happens because the Pirate 2 dossier does not contain a filter.

Add a filter to the Pirate 2 dossier

To display data in the Pirate 2 dossier based on user selections in the Pirate 1 dossier, create a filter in the Pirate 2 dossier.

1 In Web, open the **Pirate2** dossier.

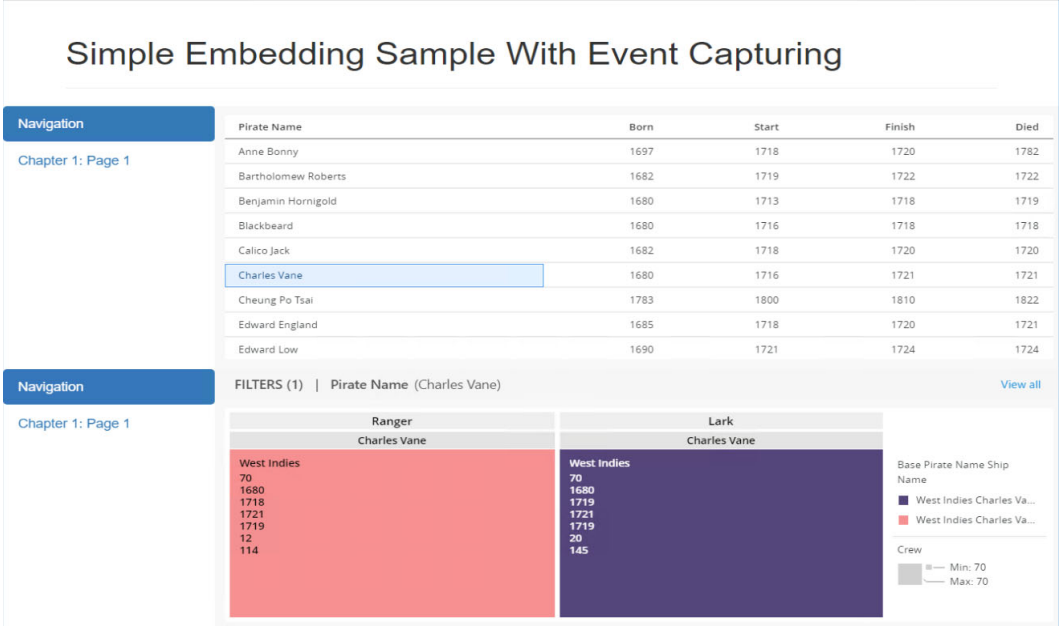
2 At the top of the Editor pane, click **Filter**.

3 Drag **Pirate Name** to the Filter panel.

4 **Save** your dossier.

5 In Chrome, **refresh** the Embedding2.html page.

6 In the Pirate 1dossier, click any **Pirate Name**. The Pirate2 dossier is filtered based on your selection, as in the following example.



Let's examine some of the code in the Embedding2.html page. The page contains the following methods from the dossier class in the Embedding JavaScript library:

- The dossier.create() method uses the configuration parameters you modified to retrieve the desired dossiers and embed them in specific HTML containers.
- The getFilterList() method returns a list of filters defined on the current chapter (Chapter 1) of Pirate 2 (the target dossier) as an array of JSON objects that describe a specific filter.

Once the dossiers are embedded in the web page, users are able to click the dossier components. A for loop compares the selection made in Pirate 1 to the available filters in Pirate 2. If a matching filter is found, the selection is applied in Pirate 2.

Congratulations, you have successfully embedded two dossiers with interactive filtering.

Summary

In this chapter, you built web applications that leverage MicroStrategy reporting objects. Your applications employed the Embedding SDK to create REST API calls to retrieve the desired data. You learned about the authentication requirements and CORS settings required to interact with the MicroStrategy Library server.

© 2021 MicroStrategy, Inc.

Activity 5.6: Quiz

Answer the following questions. You can find the answers in *Quiz Answers, page 153*.

1 To embed a dossier, you need a link to the embedding library, a destination for the dossier, and which of the following?

a JavaScript code to retrieve the dossier

b HTML code to retrieve the dossier

c CSS code to retrieve the dossier

d A plug-in to expose the dossier

---

2 Which HTML tags are used to embed a dossier?

a
</BR>

b <Dossier></Dossier>

c <DIV></DIV>

d <TD></TD>

---

3 True or False: To embed a dossier, the <DIV> in your web page requires an identifier.

---

4 True or False: The <DIV> identifier is used only in the JavaScript code.

---

5 Which Embedding API JavaScript method is responsible for rendering a dossier?

a microstrategy.dossier.generate

b microstrategy.dossier.create

c microstrategy.dossier.appear

d microstrategy.dossier.show

© 2021 MicroStrategy, Inc.

6 True or False: A project ID and dossier ID can only be retrieved through MicroStrategy Workstation.

7 True or False: Cross-Origin Resource Sharing enables an application to process requests and deliver information to servers outside of its own domain.

© 2021 MicroStrategy, Inc.

6

Import Data using the Push API

Up to this point, you have used the MicroStrategy REST APIs to retrieve and manipulate existing MicroStrategy data. You can also use the REST API to insert new data and create new objects in MicroStrategy. For example, you can use API end points in a custom application that inserts new data into Intelligent Cubes.

In this exercise, you will develop a supporting application for Ships Ahoy!, an online pirate game based on loot boxes. Players can buy loot boxes with a random allotment of in-game goodies. You will create an application that collects external sales data related to loot boxes. The data collected by your application will help your organization identify the loot box items that appeal to users, and adjust allocation to maximize sales.

Publishing datasets with the Push API

To streamline the data loading process and ensure that the latest data is analyzed by MicroStrategy users, you can publish and update datasets. For example, you might create an application that implements APIS to push new data to an Intelligent Cube and republish the cube.

You can use the APIs to create new datasets or update existing datasets from CSV or JSON files without interacting with the data warehouse. Refresh your data based on the following policies:

© 2021 MicroStrategy, Inc.

- **Replace**: Replace all data with new data.
- **Insert**: New data is added, but existing data is not updated.
- **Update**: No new data is added. Existing data is updated.
- **Upsert**: Existing data is updated and new data is added. This is essentially a combination of Update and Insert.

Exercise 6.1 Create a cube using the REST API

You are working on a custom application that leverages the MicroStrategy REST API to extract data from a file and create an Intelligent Cube in a MicroStrategy project. The majority of the JavaScript code is already written, but you must specify the configuration parameters to connect to your environment and project.

To ensure that the application runs on all browser clients without errors, you will also modify a library protocol from http to https. This process conforms with the web development best practice to employ a single communication protocol in your application.

Configure the application

1. In File Explorer, navigate to **Desktop\REST-API\Exercises\Chapter 6** and do the following:
   a. Right-click **JSPushDataDemo.zip** and click **Extract All**.
   b. In the new window, click **Extract**. The JSPushDataDemo folder is created.
   c. Change the JSPushDataDemo folder name to **REST**.
2. Inside the **REST** folder, right-click **index.html** and select **Edit with Notepad++**.
3. Locate the following line:

```
<script src="http://maxcdn.bootstrapcdn.com/
bootstrap/3.3.7/js/bootstrap.min.js"></script>
```


© 2021 MicroStrategy, Inc.

4 Replace **http** with **https** as below:

```
<script src="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

5 **Save** the file.

6 Inside the **js** folder, right-click **demo.js** and select **Edit with Notepad++**.

7 Locate the following section:

```
// ******** Configuration

    var restServerUrl = "Rest Server URL";
// URL of rest server, for example, http://
localhost:8080/web-dossier/

    var projectId = "Project ID";
// ID of the MicroStrategy Project, For
example, "B19DEDCC11D4E0EFC000EB9495D0F44F"

    var username = "administrator";
// Credentials to connect to the MicroStrategy
Project

    var password = "";
```

8 Modify the configuration parameters based on the information in the following table:

Parameter	Value
restServerUrl	https:// env-**XXXXX**.customer.cloud.microstrategy.com/ MicroStrategyLibrary/ where **XXXXX** is your environment number.
projectId	The MicroStrategy Tutorial identifier from your Notepad++ document
username, password	Credentials from the Welcome to MicroStrategy Cloud email.

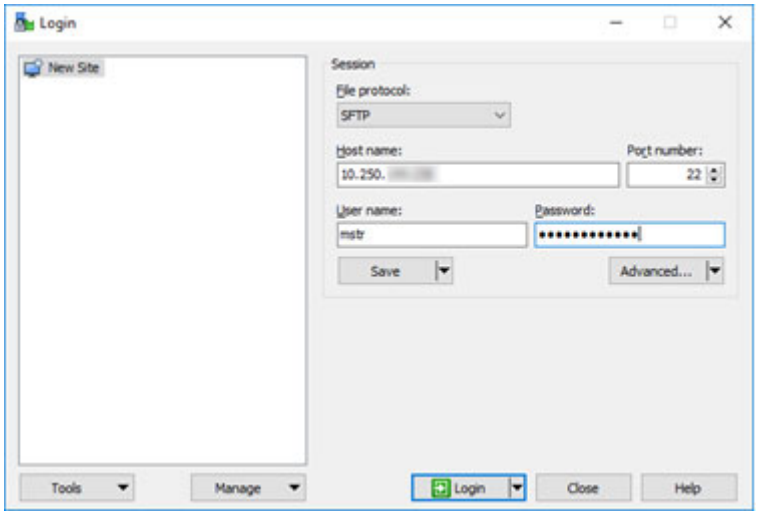
9 **Save** the file.

Deploy the application on the web server

Move the application to the Linux server in the MicroStrategy Cloud environment.

1 From the Windows desktop, double-click **WinSCP**.

2 In the **Host Name** box, type the IP address for the Intelligence Server found in the hosts.txt file.



3 Type the credentials from the MicroStrategy Cloud email.

4 Click **Save** and then click **OK**.

5 Click **Login**.

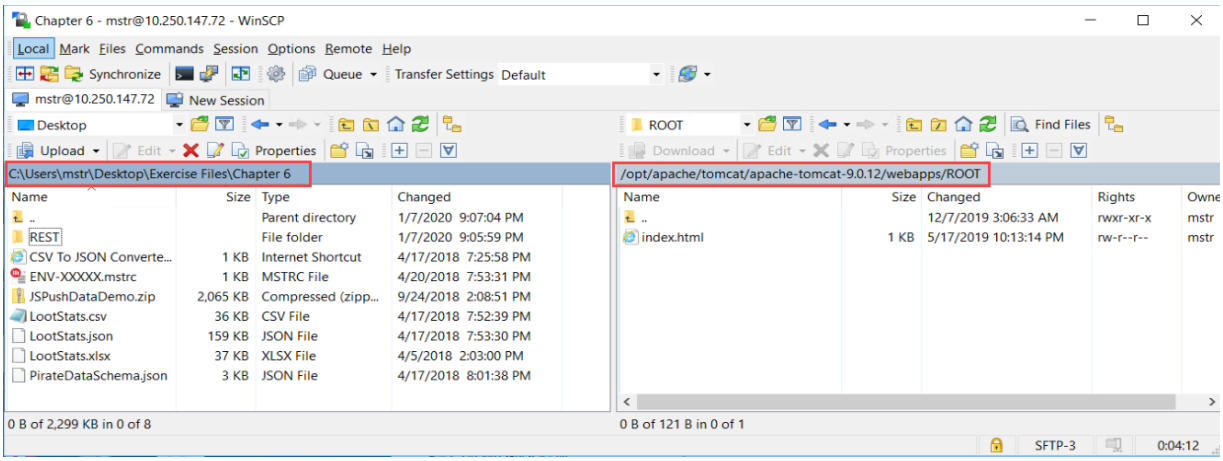
6 In the Warning window, click **Yes** and then click **Continue.**

7 Type the password from the MicroStrategy Cloud email and click **OK**.

8 On the left side of the WinSCP window (the Windows server), navigate to **Desktop\REST-API\Exercises\Chapter 6**.

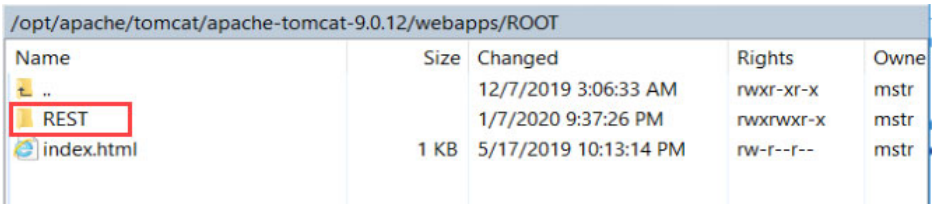
© 2021 MicroStrategy, Inc.

9 On the right side of the WinSCP window (the Linux server), navigate to **/opt/apache/tomcat/apache-tomcat-9.0.39/webapps/ROOT**, as show below.



10 In the left pane, click the **REST** folder and click **Upload**.

11 In the Upload window, click **OK**. The REST folder is copied to the Linux server.



12 Close the WinSCP session.

Starting the application

You placed the application folder on the web server. Restart the Tomcat server to deploy the application.

1 From the Windows desktop, double-click **VNC - Viewer**.

2 Double-click the session you created earlier.

3 From the **Applications** menu, point to **System Tools**, and select **Terminal**.

4 Enter the following command: **service mstr restart**.

Wait a few minutes to allow the Intelligence Server to completely restart.

5 Click **Reconnect**.

© 2021 MicroStrategy, Inc.

6 In the **Terminal**, enter the command: **service mstr status**.

Three green lines indicate that all servers, including Tomcat, restarted.

7 Close the VNC session.

Reviewing the dataset

Now that the application is deployed, you are ready to review the data that will eventually be loaded into your MicroStrategy project.

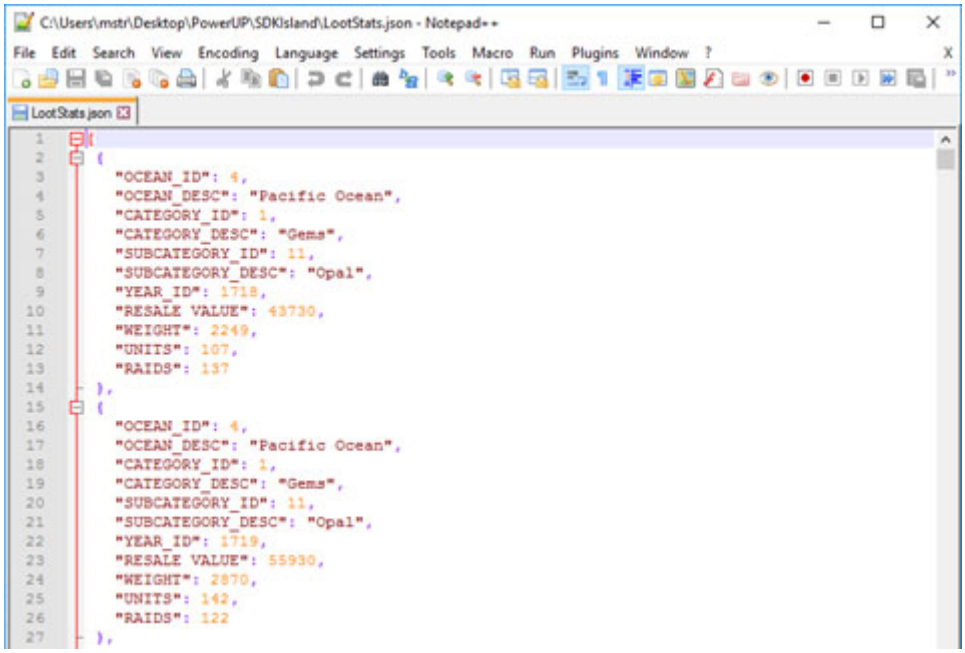
The sales system generates data in the form of a csv file. Because your application expects data in JSON format, the file has been converted to JSON.

To convert a .csv to JSON, you can use an online converter such as the following application: http://www.convertcsv.com/csv-to-json.htm

1 From File Explorer, navigate to **Desktop\REST-API\Exercises\Chapter 6**.

2 Right-click **LootStats.json** and select **Edit with Notepad++**.

Each element represents a row in the .csv file.



3 Close the file.

 © 2021 MicroStrategy, Inc.

Review the schema

Your application uses a schema file to define the cube, tables, metrics, attributes, and attributes forms that will be created in your MicroStrategy project. Examine the schema to understand how the schema is created.

1 In file Explorer, navigate to **Desktop\REST-API\Exercises\Chapter 6**.

2 Right-click **PirateDataSchema.json** and select **Edit with Notepad++**.

Let's review some key components:

```
{
    "name": "PiratesData",
    "tables": [
        {
            "data": "Select file to insert data here",
            "name": "LootStats",
            "columnHeaders": [
                {
                    "name": "OCEAN_ID",
                    "dataType": "INTEGER"
                },
                {
                    "name": "OCEAN_DESC",
                    "dataType": "STRING"
                },
```

The first element is the name of the Intelligent Cube that will be created. The next element is an array of tables. This array includes a single table which contains a table name and an array of column headers.

Each column is defined with a name and a type.

```
"metrics": [
        {
            "name": "ResaleValue",
            "dataType": "number",
            "expressions": [
                {
                    "formula": "LootStats.RESALE VALUE"
                }
            ]
        },
        {
```

```
            "name": "Weight",
            "dataType": "number",
            "expressions": [
                {
                    "formula": "LootStats.WEIGHT"
                }
            ]
        }
```

This next element is an array that defines the metrics. Each metric element has a name, a data type, and expression that reference specific columns.

In the sample above, two metrics are mapped to two columns in the .csv file.

```
"attributes": [
        {
            "name": "Ocean",
            "attributeForms": [
                {
                    "category": "ID",
                    "expressions": [
                        {
                            "formula": "LootStats.OCEAN_ID"
                        }
                    ],
                    "dataType": "number"
                },
{
                    "category": "DESC",
                    "expressions": [
                        {
                            "formula": "LootStats.OCEAN_DESC"
                        }
                    ],
                    "dataType": "string"
                }
            ]
        }
```

The next element is an array that defines the attributes in the dataset. Each attribute is an element defined with a name and an array of attribute forms. An attribute form element is composed of a category (the attribute element name), and an expression that references a column in the .csv file. The preceding sample defines the Ocean attribute with ID and Description attribute forms.

3 At the top of the PirateDataSchema.json file, change the name of the cube to **MyPirateDataXX**, where **XX** are your initials.

 © 2021 MicroStrategy, Inc.

4 **Save** the file.

Create the cube

Now that you understand the data schema, use the deployed application to create the Intelligent Cube.

1 Open a browser and navigate to:

https://env-**XXXXXX**.customer.cloud.microstrategy.com/REST/index.html

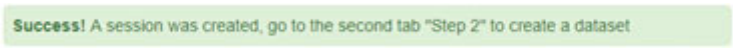
where **XXXXXX** is your environment number.

The application page opens, as shown below.

Notice that the configuration parameters you specified in the demo.js file are automatically populated.

2 Click **Login**.

If successful, a confirmation box displays as shown below.



3 Click **Step 2: Create dataset**.

4 Return to Notepad++ to the **PirateDataSchema.json** file.

If you closed the document, it can be found in **Desktop\REST-API\Exercises\Chapter 6\REST**.

5 Copy the content of the file.

6 Back in the Push Data API Demo application, delete the content of **A. Configure dataset creation**.

The populated schema is part of the demo.

7 Paste the content of **PirateDataSchema.json**.

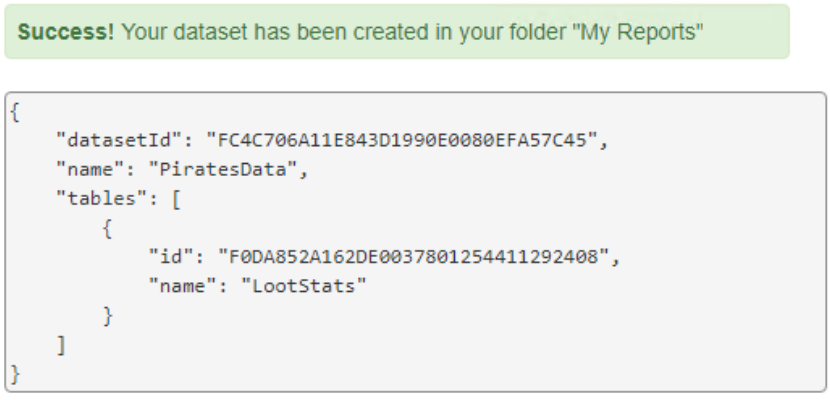
8 Click **Choose File**.

If you want to run the demo, the REST\assets folder contains a data.json file that matches the default schema.

9 Navigate to **Desktop\REST-API\Exercises\Chapter 6**, select **LootStats.json**, and click **Open**.

10 Click **Create Dataset**.

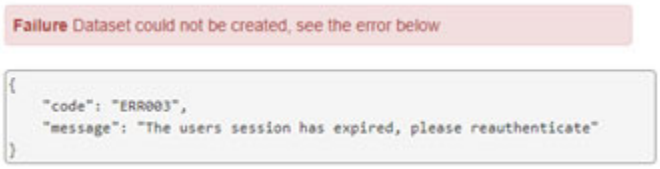
If successful, the response is similar to the following image.



The message contains the dataset ID you just created, its name, and information about the table in the cube.

 © 2021 MicroStrategy, Inc.

If you get the following message in the response, return to the Login tab, click **Login**, and return to the second tab to try to push the data again.



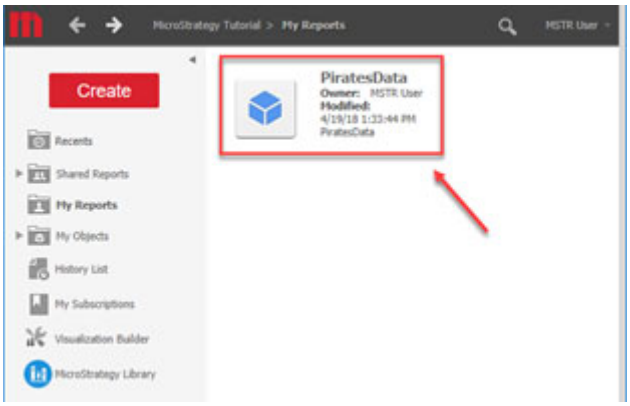
11 Open a browser and navigate to:

https://env-**XXXXX**.customer.cloud.microstrategy.com/MicroStrategy/servlet/mstrWeb

where **XXXXX** is your environment number.

12 Log in and navigate to the MicroStrategy Tutorial project **My Reports**.

Your newly created cube is displayed.



Test the new cube

Test your new cube by creating a simple dossier.

1 From the My Reports folder, click MyPirateDataXX.

A new dossier opens.

2 Drag the following attributes to the **Rows** drop-zone:

- **Ocean**

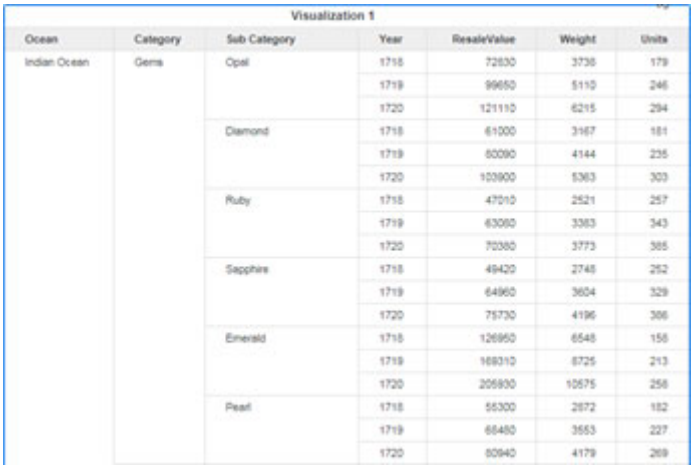
- **Category**
- **Sub Category**
- **Year**

3 Hide the ID attribute forms displayed in the grid. To do this, right-click the attribute, point to **Display Attribute Forms**, clear the **ID** check box.

4 Drag the following to the **Metrics** drop-zone:

- **Resale Value**
- **Weight**
- **Units**

5 Compare the values with the .csv file to verify that the import was successful.



Ocean	Category	Sub Category	Year	ResaleValue	Weight	Units
Indian Ocean	Gems	Opal	1718	72830	3736	179
			1719	99650	5110	246
			1720	121110	6215	294
		Diamond	1718	61000	3167	181
			1719	80090	4144	235
			1720	103900	5363	303
		Ruby	1718	47010	2521	257
			1719	63080	3383	343
			1720	70380	3773	385
		Sapphire	1718	49420	2748	252
			1719	64960	3604	329
			1720	75730	4196	386
		Emerald	1718	126950	6548	158
			1719	168310	8725	213
			1720	206930	10675	258
		Pearl	1718	55300	2872	182
			1719	68480	3553	227
			1720	80940	4179	269

6 Save the dossier as **Pirate Loot Report**.

Congratulations, you just created a cube programmatically and used it in a basic dossier!

Code walk-through

Now that you have created a cube using the web application, examine some of the code in demo.js.

The script defines a Demo object that contains three main methods accessible from the outside (defining its own API):

© 2021 MicroStrategy, Inc.

- init() - called immediately when index.html loads
- createDataset(); - called when the user clicks Create Dataset on the second tab
- createSession(); - triggered when the user clicks Login on the first tab

All other methods in the file are support methods that are not exposed externally but accessible within the script when executed.

Let's get started.

```
function Demo (){}

new Demo();

Demo.prototype = function () {
```

The first line calls the Demo object constructor. The second line creates the Demo object. The third line uses the prototype method to add methods to the object constructor. The remainder of the script file is part of this prototype structure.

With that in mind, let's explore the script structure.

```
// ******** Configuration



    var restServerUrl = "https://
env-XXXXX.customer.cloud.microstrategy.com/
MicroStrategyLibrary/";         // URL of rest server, for
example, http://localhost:8080/web-dossier/

  var projectId = "B7CA92F04B9FAE8D941C3E9B7E0CD754";
// ID of the MicroStrategy Project, For example,
"B19DEDCC11D4E0EFC000EB9495D0F44F"

    var username = "mstr";               // Credentials to
connect to the MicroStrategy Project

    var password = "password";



    // ******** Global variables



    var json;                                       // Contains
the retrieved json output

    var authToken;                                  // Contains
the session id which will be used in every api call
```

```
    var editor;                                          // Contains
the ACE editor of the body request in json
```

This section of code begins with the connectivity details you modified to connect to your environment.

The global variables section defines the following variables that are used throughout the script:

- json is a container for the return value of a REST call.
- authToken holds the authorization token retrieved as the first call to the API, allowing us to authenticate against the REST API server for each subsequent call.
- editor is a variable used to manage the JSON editor window in the second tab of the application. This is an implementation of an embedded code editor (more details at https://ace.c9.io/). In our application, we use it to manage the JSON schema.

```
var  init = function () {
    initConfigurations();
    initDatasetEditor();
    initTabNav();
},
```

The init function relies on three internal support functions. These are:

- initConfigurations() - Reads the configuration parameters like the server address, the project ID, the user name, and password.
- initDatasetEditor() - Initializes the ACE code editor that contains the JSON schema on the second tab. This editor relies on an external JavaScript library declared in the HTML page.
- initTabNav() - Initializes the two tabs on the HTML page of the application.

.

```
/**
 * Authenticate the user
 */
createSession = function () {
```


© 2021 MicroStrategy, Inc.

```
        loadConfigurations();
        var authOptions = {
            loginMode: 1,
            username: username,
            password: password
        };
        authenticate(restServerUrl, authOptions);
    },
```

The createSession function logs in to the MicroStrategy Library API server and obtains an authorization token for subsequent REST calls. This block of code begins by calling a support method that reads the parameter values (server, project ID, user name, and password) entered on the HTML page.

The authentication options are defined, based on the values read from the UI. Finally, the authenticate function is called to retrieve the authorization token. More on that function when we get there in the script. For now, let's continue with the third exposed method.

.

```
    /**
     * Create a dataset using the request body and the chosen
    file
     */
    createDataset = function (){
        var datasetCreatorString = editor.getValue();
        var file = $('#upload-file').prop("files")[0];
        if(!file){
            $('#file-not-selected-label').show()
        }else {
            $('#file-not-selected-label').hide()
        }
        if (file && datasetCreatorString) {
            prepareDataAndCreateDataset (file,
    datasetCreatorString);
        }
    };
```

createDataset, grabs the content of the ACE code editor on the second tab of the HTML page, retrieves the content of the data files, and invokes a support method that makes the REST call to create the data cube.

```
/**
* Set default values for the REST server and auth information
*/
function initConfigurations () {
    $('#project-id')[0].value=projectId;
    $('#username')[0].value=username;
    $('#password')[0].value=password;
    $('#rest-server-url')[0].value = restServerUrl;
}
```

initConfigurations is the first of many support functions we will explore. Some are simpler than others. This method uses the configuration parameters to populate the text boxes on the first tab of the HTML page.

```
/**
 * Initialize the Ace Editor used to create the body of the
request in json
 */
function initDatasetEditor () {
    editor = ace.edit("editor");
    editor.setTheme("ace/theme/github");
    editor.getSession().setMode("ace/mode/json");
    editor.setValue(JSON.stringify(datasetCreator, null, '\
t'), 1);
    $("#upload-file").change(function() {
        $('#file-not-selected-label').hide();
    });
}
```


© 2021 MicroStrategy, Inc.

initDatasetEditor is responsible for initializing the ACE code editor on the second tab.

```
/**
 * Initialize the tabs used for navigation in the demo menu
 */
function initTabNav (){
    $('#navTabs').find('a').click(function (e) {
        e.preventDefault();
        $(this).tab('show');
    });
}
```

initTabNav initializes the tabs displayed on the HTML page and makes the first tab visible. For more details, consult the JQuery documentation.

```
/**
 * Load the rest server and auth information from the user
inputs
 */
function loadConfigurations() {
    projectId = $('#project-id')[0].value;
    username = $('#username')[0].value;
    password = $('#password')[0].value;
    restServerUrl = $('#rest-server-url')[0].value;
}
```

loadConfigurations retrieves the values from the text boxes on the first tab of the HTML page and sets the variables used in the authentication and other REST calls.

```
/**
 * From the chosen file and editor value, prepare and send
the request for a dataset creation
 * @param filePath
```

```
 * @param datasetCreatorString

 */

function prepareDataAndCreateDataset (filePath,
datasetCreatorString) {

    var reader = new FileReader();

    reader.onload = function () {

        var binaryFile = reader.result.split("base64,")[1];

        var datasetCreator =
JSON.parse(datasetCreatorString); // Parse the request input
into json

        datasetCreator.tables[0].data = binaryFile; // Add
the data from the chosen file into the request

        postDataset(restServerUrl, datasetCreator);

    };

    reader.onerror = function (error) {

        console.log('Error: ', error);

    };

    reader.readAsDataURL(filePath);

}
```

prepareDataAndCreateDataset begins the process of creating the cube. It requires two parameters: a filepath (selected by the user on the second tab of the HTML page), and a schema (content of the ACE editor object passed by the datasetCreatorAString variable).

A FileReader object is instantiated to take care of the data file. If read properly, the content of the ACE editor object (the actual schema in JSON format) is parsed into a JSON structure, called datasetCreator. The content of the data file is added as part of that object. Then the postDataset function is invoked to make the REST call using both the server address and the datasetCreator object (which now contains both the data and the schema).

.

```
function submitRequest(options) {

return new Promise(function(resolve, reject) {

    var xhr = new XMLHttpRequest();

    xhr.open(options.method, options.url);

    // Default value of the request headers
```


© 2021 MicroStrategy, Inc.

```
        xhr.setRequestHeader('Content-Type', 'application/json');
        xhr.setRequestHeader("Accept", "application/json");
        xhr.setRequestHeader("X-Requested-With",
    "XMLHttpRequest");
        if (options.headers) {
            Object.keys(options.headers).forEach(function (key) {
                xhr.setRequestHeader(key, options.headers[key]);
            });
        }

        xhr.onload = function () {
            if (this.status >= 200 && this.status < 300) {
                resolve(xhr);
            } else {
                reject(xhr.response);
            }
        };
        xhr.onerror = function () {
            reject(xhr.response)
        };
        xhr.send(JSON.stringify(options.body));
    });
    }
```

submitRequest posts an HTTP request for the REST API calls and returns a Promise object. For more details on the Promise object, consult the following page:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

The promise is resolved positively if the status returned is in the 200s.

```
    /**
    * Create a session using the User Input set in the Servers
    Configuration tab
```

```
* @param baseUrl
* @param requestOptions
*/
function authenticate(baseUrl, requestOptions) {
var options = {
    method: 'POST',
    url: baseUrl + 'api/auth/login',
    body: requestOptions
};

$("#create-session-button").button('loading'); // Start the
loader
$("#login-success-alert").hide();
$("#login-failure-alert").hide();

var success = function (xhr) {
        $("#create-session-button").button('reset');// Stop
the loader
        $("#login-success-alert").show();
        authToken =
xhr.getResponseHeader('x-mstr-authToken');  //Cache the
authToken
    },
    error = function () {
        $("#create-session-button").button('reset');// Stop
the loader
        $("#login-failure-alert").show();
        console.log(error);
    };

submitRequest(options).then(success, error);
}
```

authenticate is at the heart of the createSession() method. It defines the options object mentioned earlier. The url variable points to the target REST API end point., auth/login in this case.

© 2021 MicroStrategy, Inc.

The submitRequest function is called and a Promise object is returned. The .then() structure determines if the function succeeded or if an error will be executed when the promise is resolved. If the success function method is triggered, the authToken variable is populated with the authorization token needed for all subsequent REST calls.

```
/**
* send request to create dataset with uploaded files and
defined format
* @param baseUrl
* @param requestOptions
*/
function postDataset(baseUrl, requestOptions) {
var options = {
    method: 'POST',
    url: baseUrl + 'api/datasets',
    body: requestOptions,
    headers: {
        'X-MSTR-AuthToken': authToken,
        'X-MSTR-ProjectID': projectId
    }
};

$("#create-dataset-button").button('loading'); // Start the
loader
$("#dataset-creation-failure-alert").hide();
$("#dataset-creation-success-alert").hide();
$("#json-response-container").css("visibility", "hidden");
$("#json-response").empty();

// Handle what should be done once the dataset creation
succeeded
var success = function (xhr) {
        $("#create-dataset-button").button('reset');// Stop
the loader
```

```
            $("#dataset-creation-success-alert").show();

            $("#json-response-container").css("visibility",
"visible")


$("#label-dataset-creation-successful").css("visibility",
"visible");


$("#json-response").html(JSON.stringify(JSON.parse(xhr.respo
nse), null, 4));

        },

        // Handle what should be done once the dataset creation
failed

        error = function (response) {

            $("#create-dataset-button").button('reset'); // Stop
the loader

            $("#dataset-creation-failure-alert").show();

            $("#json-response-container").css("visibility",
"visible");


$("#label-dataset-creation-successful").css("visibility",
"hidden");


$("#json-response").html(JSON.stringify(JSON.parse(response)
, null, 4));

            console.log(response);



        };



submitRequest(options).then(success, error);

}
```

postDataset defines the options object. The url variable points to the REST API end point we want to target, datasets in this case.

The submitRequest function is called and a Promise object is returned. The .then() structure decides if the function succeeds or if an error will be executed when the promise is resolved. If the success method is triggered, the HTML page is modified

© 2021 MicroStrategy, Inc.

to display successful cube creation response along with its details. If the error method is executed, an error message is displayed instead.

```
return {
    init: init,
    createDataset: createDataset,
    createSession: createSession
};
```

The return block is the completion of the Demo object. It establishes that the Demo object returns an object with three methods, as presented earlier.

That's it. As you can see some well-placed code is all that's necessary to make REST API calls in your applications.

Summary

In this chapter, you explored another practical application of the MicroStrategy REST API. You used a custom application to load the data from a third-party system and push the data into a new Intelligent Cube. We took some time to review the code and understand the authentication and cube creation processes. Armed with this knowledge, you can build your own custom applications and use the snippets presented here to inject the proper REST API calls.

Activity 6.2: Quiz

Answer the following questions. You will find the answers in *Quiz Answers, page 153*.

1 Which refresh policy updates existing data and adds new data?

a Replace

b Insert

c Update

d Upsert

2 Which files were required to create the data cube in the custom application?

a A data file and a configuration file

b A data file and a schema file

c A schema file and a configuration file

d A schema file and an event file

---

3 True or False: All REST API calls are built on a Promise object.

© 2021 MicroStrategy, Inc.

AFTERWORD

Where to go from here

With the knowledge acquired in this course, you are in a key position to leverage the expansive MicroStrategy REST API library to build your own customized applications that leverage MicroStrategy functionality. Make sure you adhere to security protocols and policies set in place in your enterprise while developing these new applications.

As you continue your journey, follow the standards and guidelines established by your Services Architect. In the same vein, when you add data to MicroStrategy, follow the best practices put in place by the Platform Administrator and Application Architect. The best practices established by your organization's Intelligence Center architects will help you maintain efficiency and integrity.

Best Practice

As you create your applications, always perform extensive testing in an appropriate test environment and make sure you are confident in the behavior of your application before deploying to a production server.

Revisit the exercises whenever you need a refresher on specific workflows. When you are ready to further explore the MicroStrategy REST API, dig deeper in the REST API documentation and experiment with the REST API Explorer to supplement this text.

© 2021 MicroStrategy, Inc.

Additional classes

Expand your knowledge of the various MicroStrategy SDKs, including:

- **SDK for Customizing Analytical Applications**. Learn to expand MicroStrategy Web capabilities. Create plug-ins to modify the interface and customize application functionality.
- **Advanced SDK for Customizing Branding**. Learn to customize MicroStrategy components to match your organization's branding. Topics include using the URL API to trigger another app, or create an email on the fly.
- **Advanced SDK for Customizing Visualizations**. Build custom visualizations from scratch or modify existing customizations to display data in a unique way and trigger new analyses by end users.

© 2021 MicroStrategy, Inc.

A

Quiz Answers

Chapter 2

Activity 2.4: Quiz

1. Application Programming Interface
2. False. Think about the voice assistants, or the Shazaam application.
3. True
4. False. Quite the opposite, JSON is known for its simplicity to parse.
5. XML
6. True
7. True
8. False. XML
9. True

© 2021 MicroStrategy, Inc.

10 Warehouse

Chapter 3

Activity 3.4: Quiz

1 True

2 Authorization token

3 Success

Chapter 4

Activity 4.2: Quiz

1 .mstrc

2 200

3 True

4 False. An single element called certified is set to false in the section.

Chapter 5

Activity 5.6: Quiz

1 a. JavaScript code.

2 c. <DIV></DIV> tags.

3 True.

4 False. The identifier can be used in CSS as well.

5 b. create.

6 False. You can also open the dossier and find the information in the URL displayed. You can also use MicroStrategy Developer.

© 2021 MicroStrategy, Inc.

7 True. This is the definition of CORS.

Chapter 6

Activity 6.2: Quiz

1 Upsert.

2 A data file and a schema file.

3 True.

© 2021 MicroStrategy, Inc.

© 2021 MicroStrategy, Inc.

Copyright Information

All Contents Copyright © 2021 MicroStrategy Incorporated. All Rights Reserved.

Trademark Information

The following are either trademarks or registered trademarks of MicroStrategy Incorporated or its affiliates in the United States and certain other countries:

Dossier, Enterprise Semantic Graph, Expert.Now, HyperIntelligence, HyperMobile, HyperScreen, HyperVision, HyperVoice, HyperWeb, Information Like Water, Intelligent Enterprise, MicroStrategy, MicroStrategy 2019, MicroStrategy 2020, MicroStrategy 2021, MicroStrategy Analyst Pass, MicroStrategy Architect, MicroStrategy Architect Pass, MicroStrategy Badge, MicroStrategy Cloud, MicroStrategy Cloud Intelligence, MicroStrategy Command Manager, MicroStrategy Communicator, MicroStrategy Consulting, MicroStrategy Desktop, MicroStrategy Developer, MicroStrategy Distribution Services, MicroStrategy Education, MicroStrategy Embedded Intelligence, MicroStrategy Enterprise Manager, MicroStrategy Federated Analytics, MicroStrategy Geospatial Services, MicroStrategy Identity, MicroStrategy Identity Manager, MicroStrategy Identity Server, MicroStrategy Integrity Manager, MicroStrategy Intelligence Server, MicroStrategy Library, MicroStrategy Mobile, MicroStrategy Narrowcast Server, MicroStrategy Object Manager, MicroStrategy Office, MicroStrategy OLAP Services, MicroStrategy Parallel Relational In-Memory Engine (MicroStrategy PRIME), MicroStrategy R Integration, MicroStrategy Report Services, MicroStrategy SDK, MicroStrategy System Manager, MicroStrategy Transaction Services, MicroStrategy Usher, MicroStrategy Web, MicroStrategy Workstation, MicroStrategy World, Usher, and Zero-Click Intelligence.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Specifications subject to change without notice. MicroStrategy is not responsible for errors or omissions. MicroStrategy makes no warranties or commitments concerning the availability of future products or versions that may be planned or under development.

The Course and the Software are copyrighted and all rights are reserved by MicroStrategy. MicroStrategy reserves the right to make periodic modifications to the Course or the Software without obligation to notify any person or entity of such revision. Copying, duplicating, selling, or otherwise distributing any part of the Course or Software without prior written consent of an authorized representative of MicroStrategy are prohibited.