ご利用のコンピュータを設定する方法

事前設定された dCloud ラボを使用してこのラボに取り組む場合は、イベントの事前準備 [英語] とラボの設定 [英語] の両モジュールを確認してください。

自身のコンピュータでこのラボの作業を行うには、Postman という Chrome http クライアントをインストールする必要があります。また Spark アカウントも必要です。

REST API の基本

このラーニング ラボでは、REST API の利用方法の基本と、Postman を使用して REST API のテストを行う方法について学びます。

目標

所要時間:20 分

- REST API の使用方法の基本を理解する
- Postman クライアントを使って API を呼び出す方法について学ぶ
- Spark API を呼び出す方法について学ぶ

前提条件

このラボでは、Cisco Spark API と Postman [英語] を使用して、API を呼び出す方法について学びます。

背景

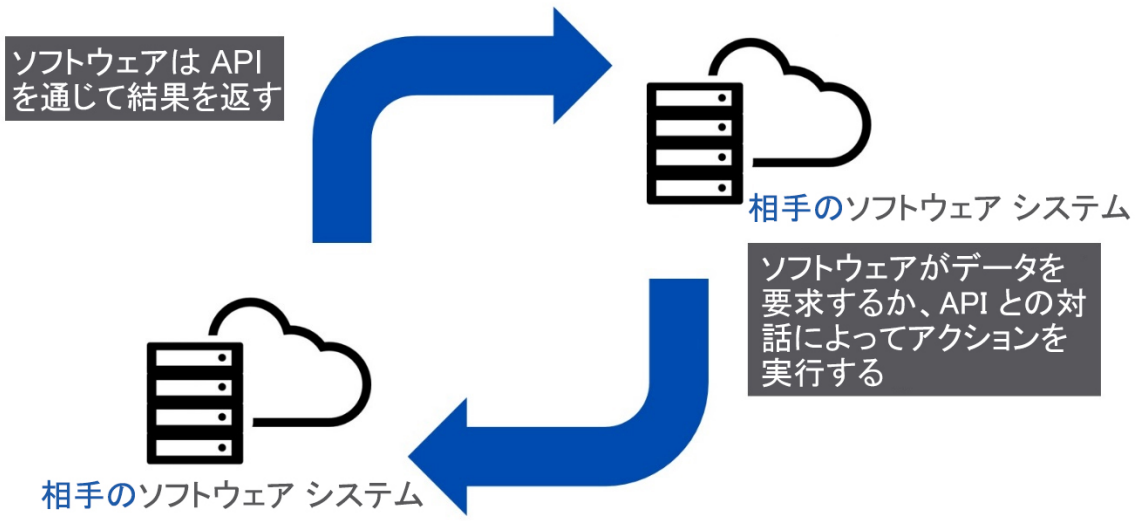
- Python や API に慣れていない場合、またはプログラミングの経験がない場合は、ここからスタートするのが最適です。こうした知識がすでにある場合は、次のモジュールに進んでもかまいません。
- このラボでは、REST API の呼び出しが可能な GUI ベースの Web クライアント ツールである Postman [英語] を使用します。Postman では、エンジニアが REST API の機能と構文を簡単に確認できます。API 構文から実際のコードへの移行が簡単になります。Postman がまだインストールされていない場合は、必ずインストールしてください。

API の概要

ここでは API とは何か、なぜ重要なのかという観点から、ネットワーク プログラマビリティについて考えます。説明の前提として、まずは重要な概念の定義を確認しましょう。

API（アプリケーション プログラミング インターフェイス）は、2 つのソフトウェアが相互に通信する際の方式です。あなたは通常、ソフトウェアとのインターフェイスにどのようなものを使用しているでしょうか。たとえば、Web ブラウザ（Web インターフェイス）を開いて電子メールにアクセスしているかもしれません。メッセージを開いてファイルとして保存するワークフローを決めているかもしれません。こうしたワークフローには、それぞれの「インターフェイス」、つまり、特定のタスクを実行するための方法があります。

API はそれに類似する概念です。ソフトウェアとのインターフェイスを人が使用するのではなく、ソフトウェア同士でそれぞれに向けたインターフェイスを使用します。人が画面上でワークフローを進めるのではなく、API が別のアプリケーションにその機能を公開するわけです。



なぜそのような方法をとるのでしょうか。1 つは、それによって広範な機能を備えたリッチ アプリケーションの開発が可能になるためです。例を見てみましょう。

たとえばレストラン ガイド アプリの開発者が、目的の地域のレストランの一覧を表示する機能と、現在地から指定したレストランまでの経路を表示するマップ アプリケーションの統合を考えたとします。あなたはこのような機能をゼロから作成するでしょうか。おそらくそのようなことはしないでしょう。

ゼロから構築するとなると、専門とする領域とは関係のないことにも取り組まなければなりません。また、リスクや学習時間も多大なものになります。代わりに、サード パーティを活用して既製の機能をアプリケーションに統合するほうが、理にかなっています。

マップ サーバもその例の 1 つです。マップ機能をゼロから構築する代わりに、マップ サーバから提供された API を使用して、アプリケーションにマップ機能をすばやく統合することができます。

API を利用することで、一貫した方法でソフトウェアが提供する機能を呼び出すことができます。次の図を見てください。

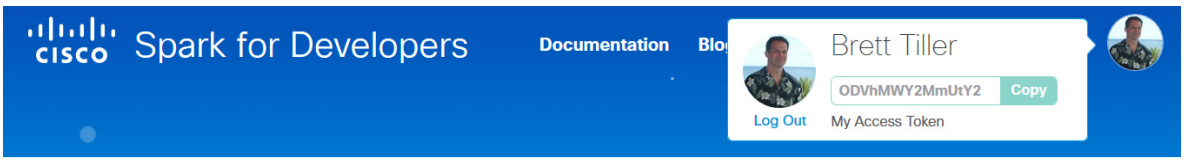
API は壁面のコンセントのように、インターフェイスの仕様を決定します。どのベンダーやデバイスであっても、コンセントを通じてデバイスが適切な電気仕様に合致するのと同様に、API により、ソフトウェアが適切な仕様に準拠することが保証されます。

ステップ 1：Spark REST API にアクセスする

ここでは Spark REST API を使用して、プログラマビリティについての学習を始めます。Spark は、チャットやファイル共有などのさまざまな機能を備えたコラボレーション ツールです。

Spark REST API を使用するには、Spark 開発者アカウントをセットアップする必要があります。

1. Spark Developer Web サイト [英語] にアクセスします。
2. [サイン アップ（Sign Up）] ボタンをクリックして、アカウントを作成します。
3. 新しいクレデンシャルでログインし、ページの右上隅にあるメンバー アイコンをクリックし、[コピー（Copy）] ボタンをクリックして、**アクセス トークン**をコピーします。



このトークンを使用して、Spark API の呼び出しを行うことができます。

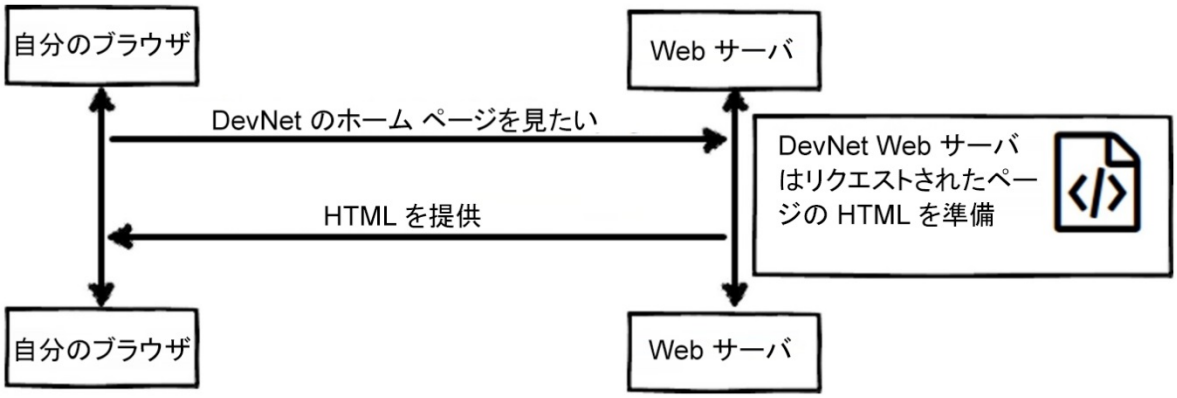
次に進みます。API の基礎を確認できたので、REST API の機能を詳しく見てみましょう。

ステップ 2：REST Web サービスとは

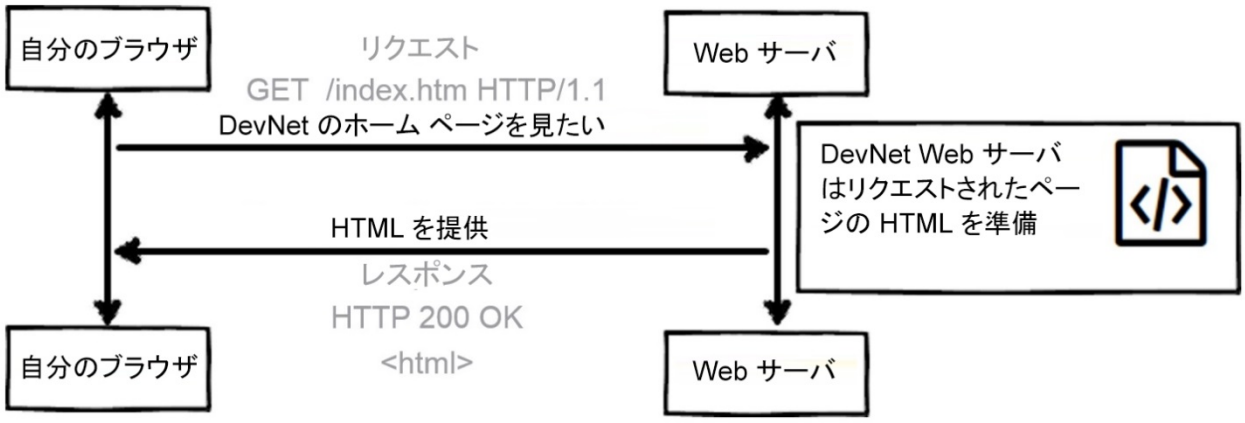
一般的に Web サービスとは、定義されたインターフェイスを介して 2 つのシステム間で情報をやり取りする際の方式です。過去 20 年間、Web サービスには大きく分けて **REST** と **SOAP** の 2 つのタイプがありました。ここ 10 年は、REST によるアプローチが普及しています。

REST とは何でしょうか。REST（Representational State Transfer）は、ネットワーク アプリケーションを設計するためのアーキテクチャ スタイルです。REST Web サービスは、HTTP リクエストのように簡単にコールできる Web サービスです。

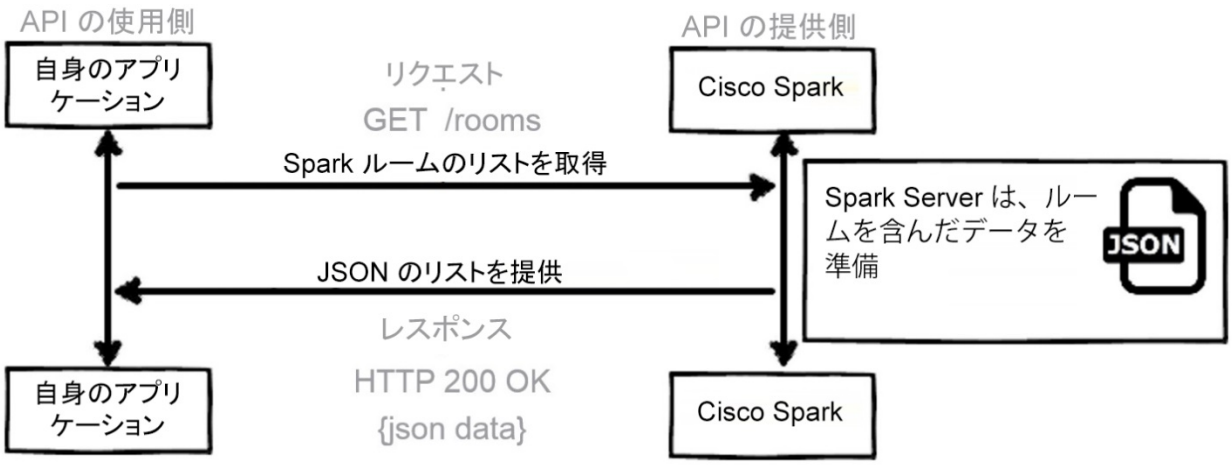
次の図を見てください。これはブラウザがどのように Web ページを取得するかを示しています。通常は、ユーザがブラウザ内で特定のリソースを要求すると、該当する Web サーバが適切な HTML で応答し、クライアント ブラウザにページが表示されます。



バックグラウンドでは、HTTP（HTTPS）がデータ要求の通信を通して CRUD（Create/作成、Read/読み取り、Update/更新、Delete/削除）の操作を行います。次の例では、ブラウザが GET 操作を実行し、関連する Web ページを読み取っています。Web サーバは関連するデータと HTTP 応答をクライアント ブラウザに返します。



RESTful インターフェイスでは、この CRUD（作成、読み取り、更新、削除）の操作を HTTP（HTTPS）によって可能にします。ブラウザは、RESTful サービスのインターフェイスを使用するソフトウェアに置き換えられます。次の図も概念は同じですが、ブラウザが、REST API を使用するソフトウェアに置き換わっています。

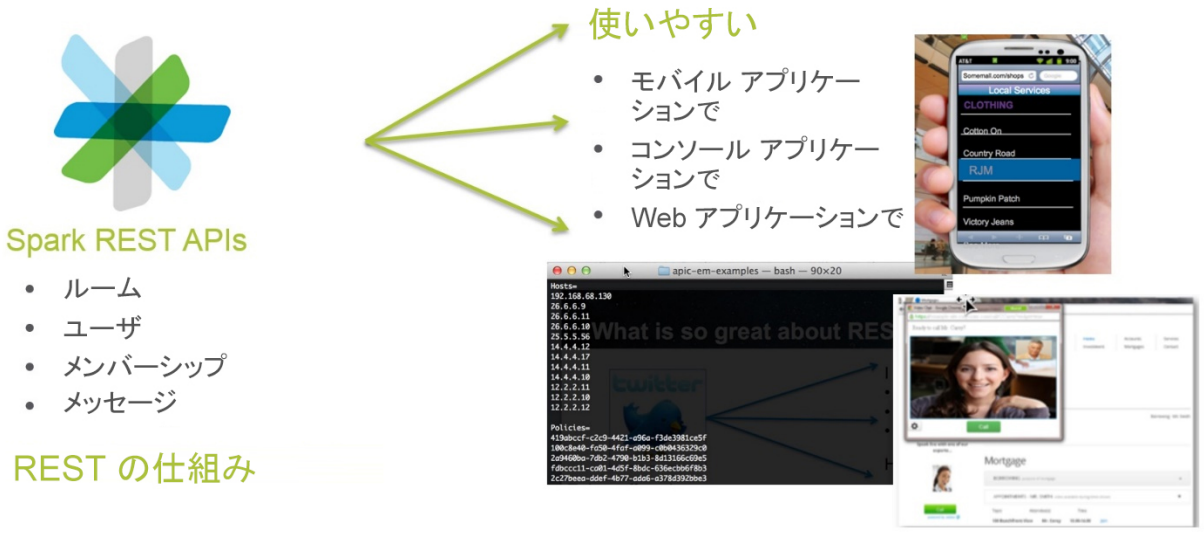


さらに詳細を知りたい場合は、REST チュートリアル [英語] を参照してください。

REST のメリット

REST はプラットフォームを問わず簡単に使用できます。

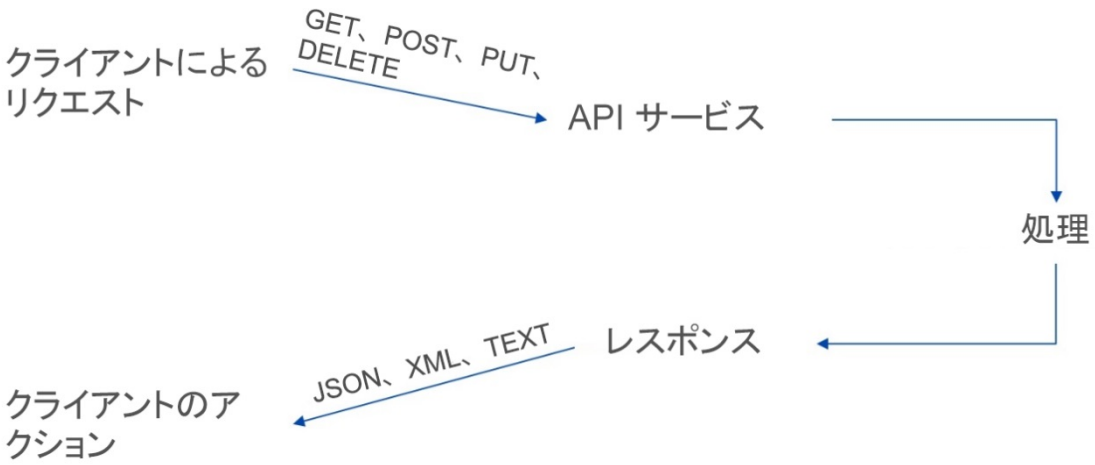
REST API のメリットを確認してみます。まずは、さまざまなプラットフォームに展開可能であるというコンセプトが挙げられます。このラボでは、REST API について学習するためのツールとして、Spark REST API を使用します。Spark は、コラボレーション用のプラットフォームです。



また REST は、シスコの APIC-EM コントローラなど、各種のネットワーク プラットフォームでも使用されています。API と APIC-EM のデータ モデルは異なりますが、基盤にあるツールは共通しています。

REST の仕組み

次に REST API の仕組みを確認します。REST は、HTTP のリクエスト/レスポンス モデルの中核となるものです。API の実行は、HTTP リクエストのようにシンプルです。



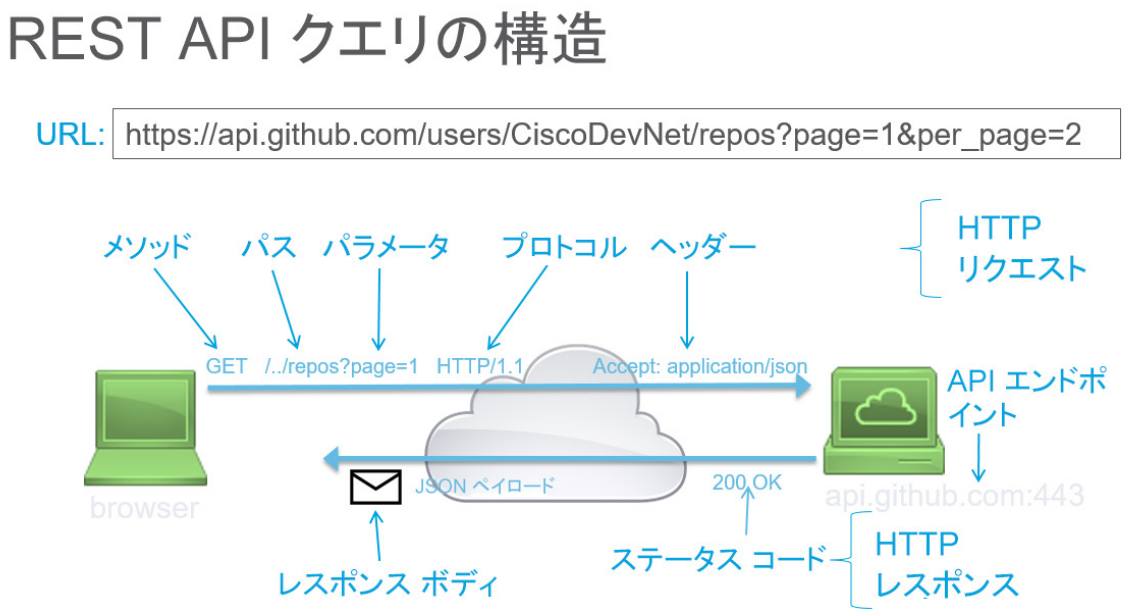
たとえば、API サービスにリクエストを送信すると、結果がレスポンスとして返されます。レスポンスで返されるデータは、通常は JSON か XML です。

（JSON：JavaScript Object Notation は、人が判読可能な形式でデータ交換できるように設計された、テキスト ベースの軽量でオープン スタンダードなデータフォーマットです。）

ステップ 3：REST API クエリの構造を確認する

リクエストを作成するには、呼び出す API に関して、次の情報が必要です。この情報は、API リファレンスドキュメントで確認できます。

- **メソッド**
  - GET：データを取得します。
  - POST：新しいデータを作成します。
  - PUT：データを更新します。
  - DELETE：データを削除します。
- **URL**
  - 呼び出し先のエンドポイントの URL
  - 例：http://api.ciscospark.com/v1/rooms
- **URL パラメータ**
  - URL の一部として渡すパラメータのことです。
- **認証**
  - 使用する認証の種類を知る必要があります。HTTP の Basic 認証、トークン ベース、OAuth が一般的です。
  - 認証のクレデンシャル
- **カスタム ヘッダー**
  - API が HTTP ヘッダーを送信する必要があるかを確認します。
  - 例：Content-Type：application/json
- **リクエスト ボディ**
  - リクエストの処理に必要なデータを含んだ JSON や XML は、リクエストのボディで送信することができます。



認証について

REST API では各種の認証があります。認証は、REST API へのアクセス制御とアクセス権限に使用されます。たとえば、読み取り専用のアクセス権限を割り当てられているユーザの場合、データを読み取る API の一部のみを使用できます。他のユーザが読み取りと書き込み両方のアクセス権限を持つこともできます。つまり API を使用して、データを読み取るだけでなく、データの追加、編集、削除を行うこともできるのです。通常、これらのアクセス権限は、データ変更に関するすべての権限を持つ**管理者**などのような、ユーザに割り当てられたロールに基づいています。たとえば単純な**ユーザ** ロールには、一般的に読み取り専用のアクセス権限が割り当てられます。

認証制御のタイプ

- **なし**: Web API リソースは公開状態であり、誰でも API の呼び出しを行うことができます。通常は GET メソッドの場合に使用され、POST、PUT、DELETE で使用されることは稀です。
- **Basic 認証**: ユーザ名とパスワードがエンコードされた文字列でサーバに渡されます。
    - Authorization: Basic ENCODEDSTRING
    - 詳細については、Basic 認証 [英語] を参照してください。
- **トークン**: 一般的に Web API の開発者ポータルから取得される秘密鍵です。
    - ベアラー、トークンなど、キーワードは Web API によって異なります。
    - 各 API の呼び出しと共に渡されます。
    - 詳細については、トークン ベースの認証 [英語] を参照してください。
- **OAuth**: アイデンティティ プロバイダーからアクセス トークンを取得する際にシーケンス フローが開始されます。トークンは各 API コールと合わせて渡されます。
    - オープン スタンダード。ユーザ権限はトークン（OAuth スコープ）と関連付けられています。
    - トークンには有効期限があります。失効させることができます。トークンの更新によって再発行することも可能です。
    - 詳細については、OAuth [英語] を参照してください。

API リファレンス ドキュメント

API リファレンス ドキュメントには、公開されているすべての API メソッドと、リクエスト方法の詳細が記載されています。新しい API を使用する際には、API リファレンスが最も重要な情報源の 1 つとなります。詳細については、Spark 向け API リファレンス ガイド [英語] を参照してください。

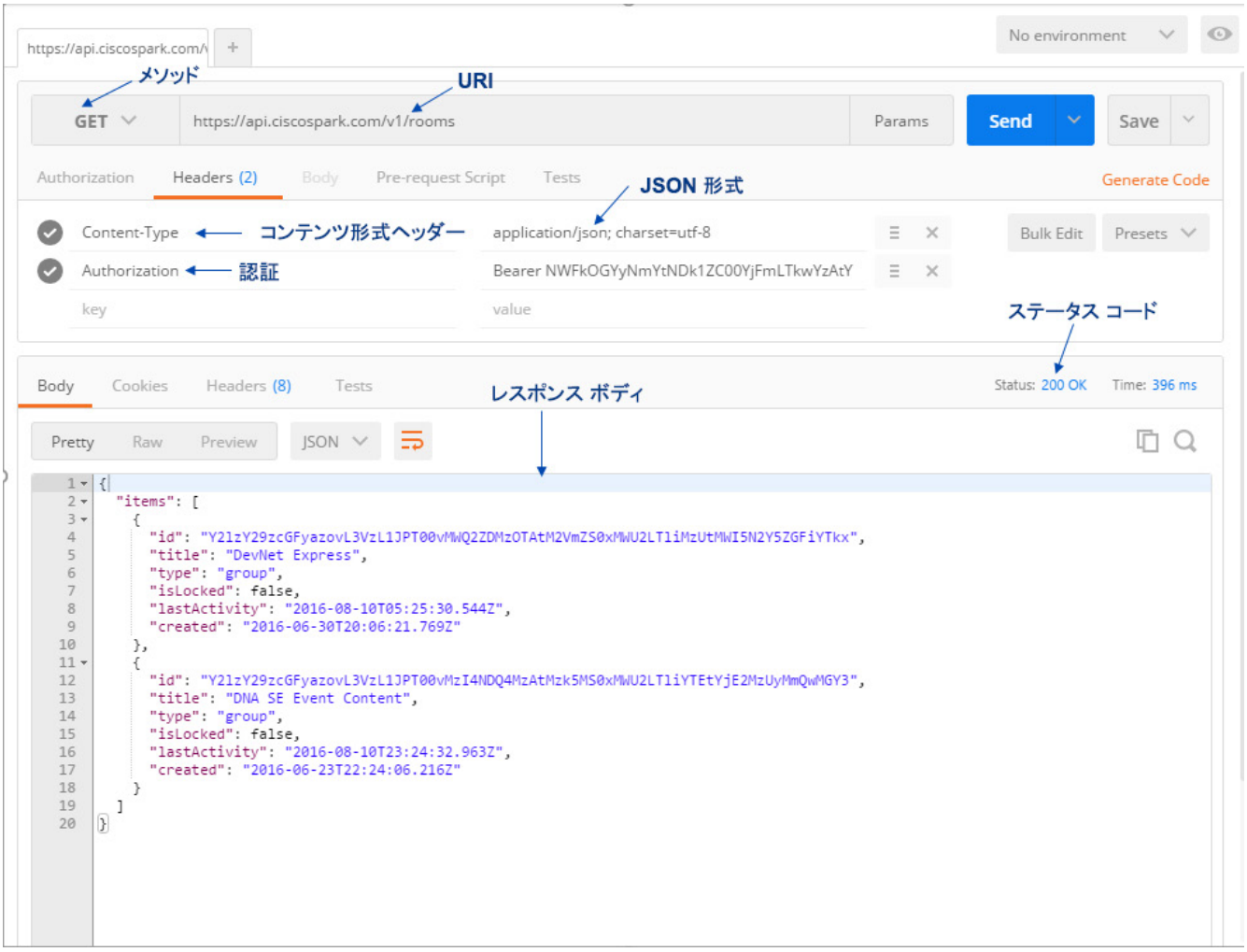
レスポンスの内容

API リファレンス ガイドには、送受信される属性に関する情報が記載されています。受信するデータはレスポンス部分で定義されており、データ形式や属性とともに HTTP ステータス コードが含まれています。

- HTTP ステータス コード
  - HTTP ステータス コードは、成功、失敗、または他のステータスを返すために使用されます。http://www.w3.org/Protocols/HTTP/HTRESP.html
  - 次のような例が一般的です。
    - 200 OK
    - 202 Accepted/Processing
    - 401 Not Authorized
- コンテンツ
  - リクエストに基づいてさまざまな形式で返されます。一般的な形式は JSON、XML、テキスト形式です。
    - JSON（最も一般的に使用される）

```
{
 "data": [{
       "company": "Cisco Systems",
       "event": "DevNet Express",
       "location": "Las Vegas, NV, USA"
}, {
      "company": "Cisco Systems",
      "event": "Cisco Live America",
      "location": "San Jose, CA, USA"
}]
}
```

REST API のリクエストとレスポンスが同じウィンドウに表示されている例を示します。



ステップ 4：Postman を使用して Spark API を呼び出す

先に簡単に説明したように、Postman は、HTTP の呼び出しを行うことができる HTTP Web ユーザ インターフェイス（Web UI）クライアントです。同様の機能を持つ Web UI クライアントは多数ありますが、ここでは Postman に焦点を当てます。

Postman とは

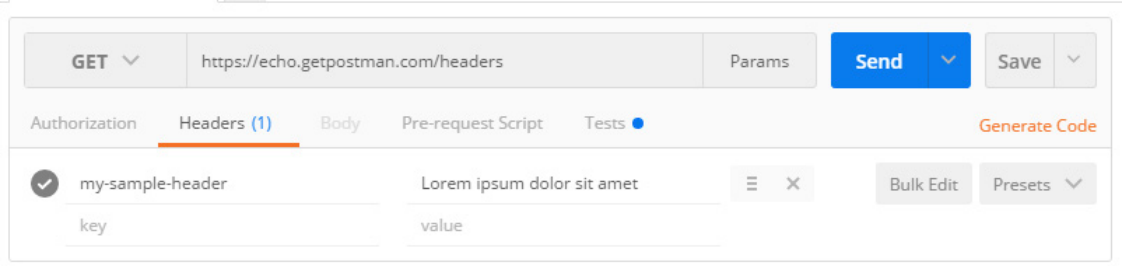
Postman は Google Chrome のアプリケーションです。REST API について学習し、また、試しに実行してみるためには使いやすいインターフェイスになっています。

ユーザが API 呼び出しを実行すると、同じウィンドウにレスポンスが表示されます。Postman は、python などの言語のコードを生成する場合にも使用できます。Postman はエントリ レベルのユーザにとって有益なツールです。Postman をダウンロードして実行するには、こちらをクリックしてください。これ以降のモジュールでは Postman を使用します。ワークステーションに Postman がインストールされていることを確認してください。

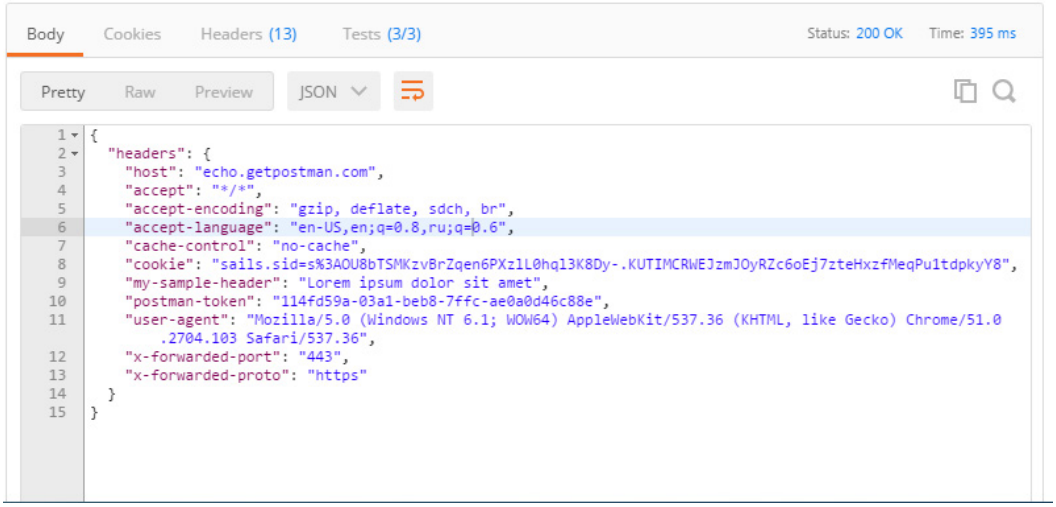
Postman のウィンドウ

Postman にはメインの作業領域が 3 つあります。ウィンドウの左側には、[履歴（History）] タブと [コレクション（Collections）] タブがあります。[履歴（History）] タブには、以前に呼び出したリクエストのリストが表示されます。[履歴（History）] タブから特定の API 呼び出しを保存するには、目的の呼び出し履歴をクリックして強調表示させ、[コレクションに保存（Save to collection）] リンクをクリックして、保存先を指定します。

画面の中央が最も中心的な領域になります。ここでは、HTTP のリクエストメソッド（GET、POST、PUT、DELETE など）を指定して URL 情報を入力し、API の呼び出しを行います。[ヘッダー（Headers）] タブでは、呼び出し時に必要なすべてのヘッダーをキーと値のペア（「ヘッダー名」と「ヘッダーの値」のペア）で指定します。すべてのパラメータを設定したら、[送信（Send）] ボタンをクリックします。



リクエストの形式が正しければ、[送信（Send）] をクリックすると、ウィンドウの下部にレスポンスが表示されます。[レスポンス（Response）] 表示エリアには、レスポンスの**ステータス** コード、形式（JSON、XML など）、レスポンス メッセージのボディなどの情報が表示されます。



Postman のテスト：Spark API に対する呼び出し

Spark API に対する呼び出しを行います。下にあるスクリーンショットは、Spark サーバに対する API の呼び出しを示しています。アルファベットの記号は各ステップを示すとともに、Postman ウィンドウの各領域に対応しています。

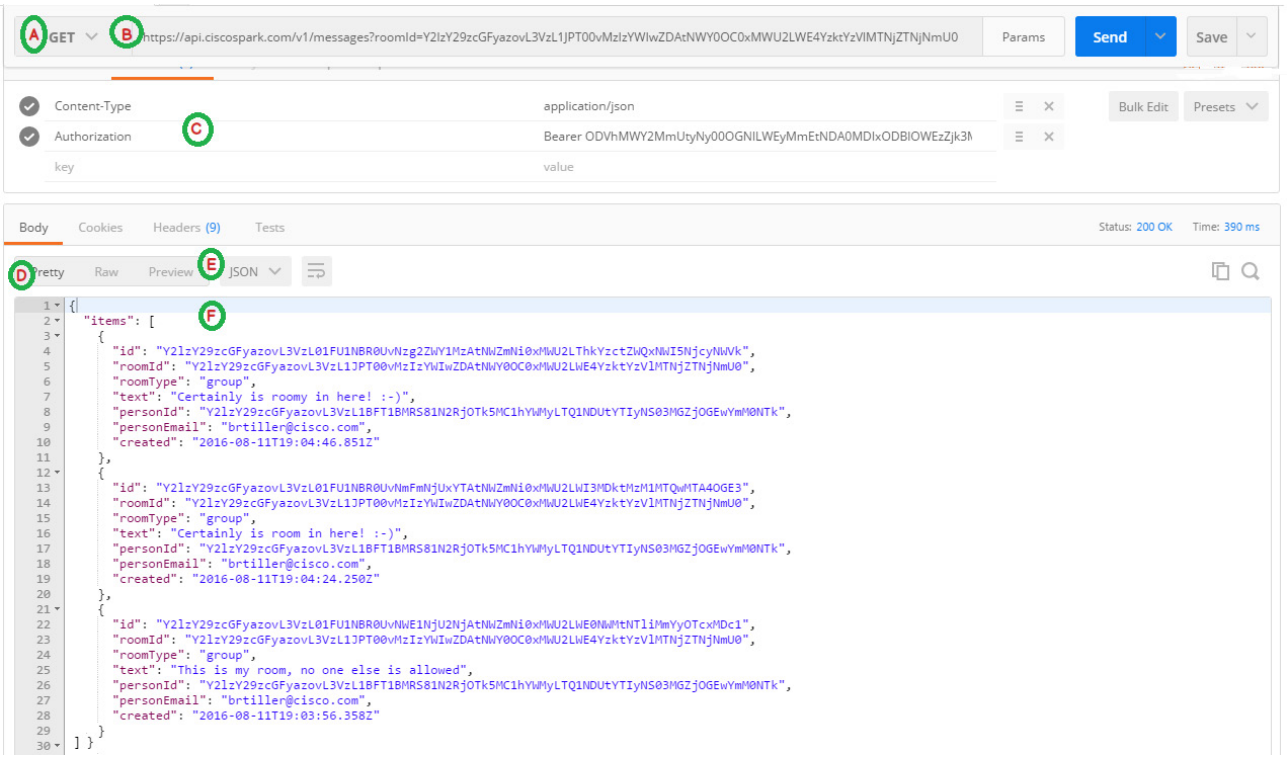
A：リクエストの送信に使用する HTTP メソッドを示します。メソッドはドロップダウン リストから選択できます。すでに説明したように、一般的なメソッドは GET、POST、PUT、DELETE です。ここでは、特定のユーザから特定のルームに投稿されたメッセージのリストを読み込みます。この場合、**GET** メソッドを使用します。

B：ここでコール先の URL を指定します。Spark API の URLは、一般的に、**https://api.ciscospark.com/v1/**から始まります。これに続けて、コール対象の機能として **messages** を指定します。さらに roomid が続き、ID 自体を **?roomid=<the room ID>** の形式で指定します。

C：Spark で機能を呼び出すには、アクセストークンが必要になります。そのため、ヘッダーに **Authorization** というキーが追加されています。また、**Bearer <the access token>** という値が入力されています。**Content-Type** キーは、どのようなフォーマット タイプのコンテンツが 先ほどの URLの中で指定した **api.ciscospark.com** という HTTP サーバに送信されるかを指定するものです。この時点でリクエストが送信可能になります。[送信（Send）] ボタンをクリックすると送信されます。

D および E：[ボディ（Body）] には、リクエストの結果として返されたデータ（レスポンス）が含まれています。レスポンスの形式として **JSON** を選択しているため、さまざまな出力方法を指定して、レスポンス データを見ることができます。[Raw] 出力の場合は、受信した形式のままデータが表示されます。[Pretty] 出力を選択した場合は、読みやすい形式に変換されたデータを表示することができます。

F：返信された JSON コンテンツが Pretty 形式で表示されます。



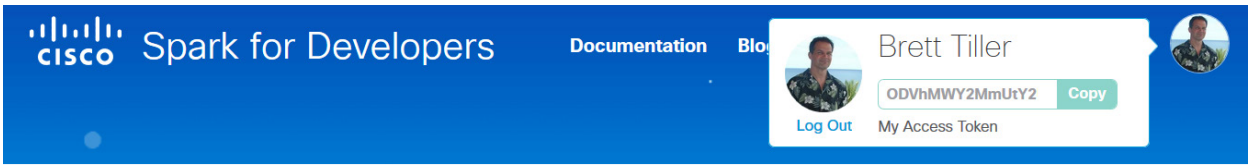
実際に確認する

次の手順に従い、Postman を使用して Spark API の呼び出しを行います。

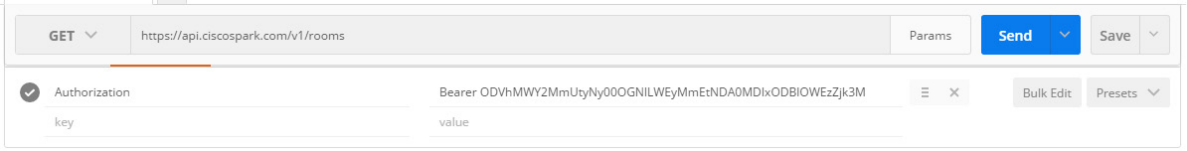
1. ここでは、自身が属する Spark ルームの一覧を取得する機能を呼び出します。Spark デスクトップ アプリケーションまたは Web クライアント（http://web.ciscospark.com）を使用して Spark にアクセスし、ログインします。属しているルームがない場合は、新たに作成す

ると、自動的にそのルームのメンバーになります。もしくは、他のユーザが属するルームに招待してもらいます。

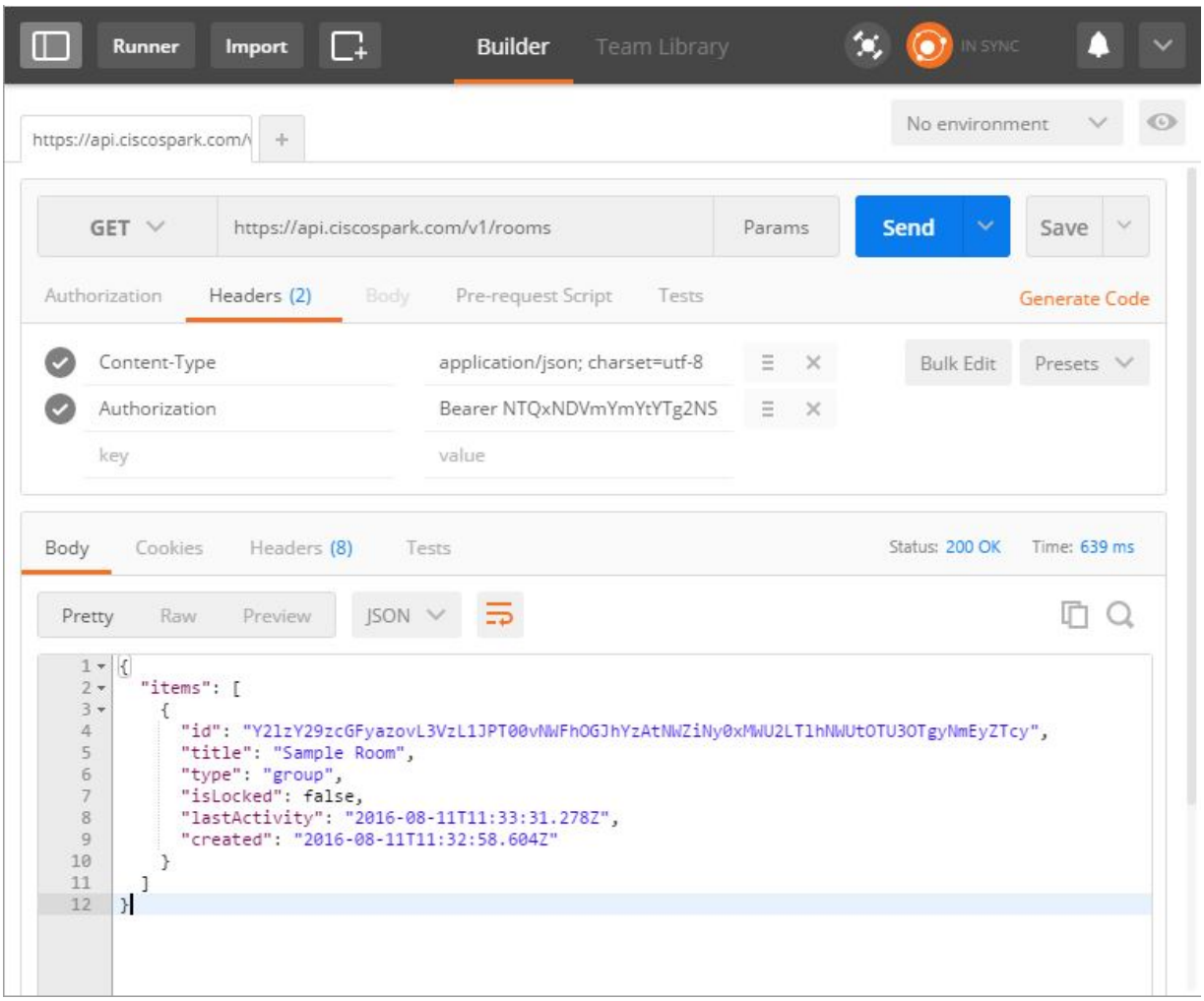
2. 機能を呼び出しするには、アクセストークンも必要です。
   https://developer.ciscospark.com に移動し、サインアップしていない場合は、サインアップしてログインします。ログインしたら、ページの右上隅にあるメンバー アイコンをクリックし、[コピー（Copy）] ボタンをクリックします。



3. Postman を開きます。
4. ドロップダウン メソッドから [GET] を選択します。
5. URL フィールドに「https://api.ciscospark.com/v1/rooms」と入力します。
6. ヘッダー フィールドに「Authorization」キーを追加します。
7. 値フィールドに「**Bearer**」とスペースを入力し、アクセストークンを貼り付けます。
   ※ アクセストークンは誰にも公開しないようにしてください。



8. [送信（Send）] ボタンをクリックします。
9. 次のスクリーンショットのようになります。



自身が属するルームの一覧が表示されます。データの形式は JSON です。エラーが表示された場合は、各手順を再確認して、メソッド、URL、ヘッダー情報が正しいことを確かめます。

これで最初の API の呼び出しが完了しました。