PART 1: Designing APIs

At the end of this part, you should be able to

- Describe REST API architecture
- Describe API development lifecycle
- Translate functional requirements for APIs into resources and HTTP methods
- Navigate Anypoint Platform

MuleSoft

Module 1: Introducing RESTful API Design

Goal

MuleSoft

SOAP

```
▼<wsdl:definitions xmlns:xsd="http://www.v
 xmlns:soap="http://schemas.xmlsoap.org/ws
  ▼<wsdl:types>
   ▼<xs:schema xmlns:xs="http://www.w3.or
    targetNamespace="http://soap.training
     <xs:element name="findFlight" type="
     <xs:element name="findFlightResponse
     <xs:element name="listAllFlights" ty
     <xs:element name="listAllFlightsResp
    ▼<xs:complexType name="listAllFlights
      <xs:sequence/>
     </xs:complexType>
    ▼<xs:complexType name="listAllFlights
      ▼<xs:sequence>
```

RPC

```
POST /SendUserMessage HTTP/1.1
Host: api.example.com
Content-type: application/json

{"userId": 501, "message": "Hello!"}
```

REST

```
#%RAML 1.0
title: ACME Banking API
version: 1.0

/customers:
  get:
  post:
  /{customer_id}:
    get:
    patch:
```

At the end of this module, you should be able to

MuleSoft

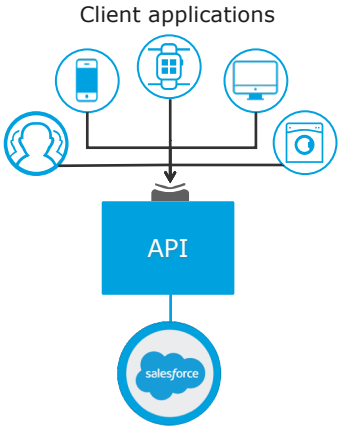
- Describe the common web API formats including SOAP, RPC, and REST
- Describe REST API architecture
- List the rules for retaining REST principles in APIs
- Describe design-first approach for REST APIs

Introducing APIs

Application Programming Interfaces (APIs)

MuleSoft

- Interface to allow applications to talk to each other
- It provides information for how to communicate with a software component, defining the
  - Operations (what to call)
  - Inputs (what to send with a call)
  - Outputs (what you get back from a call)
  - Underlying data types



Benefits of APIs

MuleSoft

- Standard interface for communication
- Easy to use
- Layer of abstraction
  - Acts as a security layer
  - Hides the specifics of the systems talking to each other

Terminology used with APIs

MuleSoft

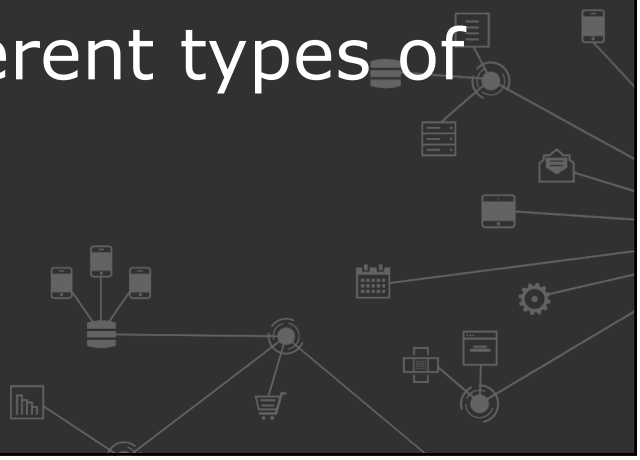
- Client
  - Initiating party that sends requests to APIs (many clients can consume an API)
- Server
  - Provides a service by sending a response to the API requests
- Stateless
  - Server does not store client information between multiple requests
- Resources
  - Entities or categories that has a URI to which requests can be sent
- Methods
  - Part of a request that tells the server the action the client wants to perform

Types of APIs

MuleSoft

- APIs belong to two categories
  - Traditional server to server, libraries in source code
  - Web APIs (focus will be on this throughout the class)
- Over tens of thousands of public web APIs that help connect applications via the internet exists
- Web APIs are created in a wide range of web formats and standards
  - SOAP (Simple Object Access Protocol) APIs
  - RPC (Remote Procedure Call) APIs
    - JSON-RPC APIs
    - XML-RPC APIs
  - REST (Representational State Transfer) APIs

Introducing different types of web APIs



SOAP APIs

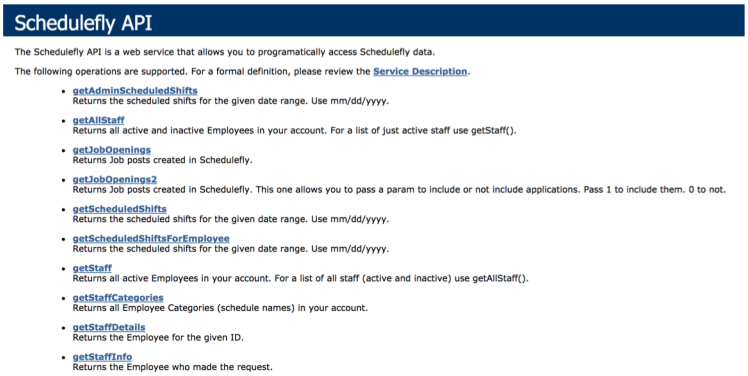
MuleSoft

- Can be used across different communication protocols (HTTP, SMTP, UDP)
- Require SOAP library in the client to build and receive requests
- Can be stateful or stateless
- Requires extensive programming knowledge
- Rely heavily on XML format for defining a web service (WSDL)
  - XML format results in more lines of code which results in bigger message sizes making them unwieldy

Walkthrough 1-1: Review and make a call to a SOAP API

MuleSoft

- Examine an example SOAP API
- Make a call to a SOAP API endpoint to retrieve information



Schedulefly API

The Schedulefly API is a web service that allows you to programatically access Schedulefly data.

The following operations are supported. For a formal definition, please review the Service Description.

- getAdminScheduledShifts
  Returns the scheduled shifts for the given date range. Use mm/dd/yyyy.
- getAllStaff
  Returns all active and inactive Employees in your account. For a list of just active staff use getStaff().
- getJobOpenings
  Returns Job posts created in Schedulefly.
- getJobOpenings2
  Returns Job posts created in Schedulefly. This one allows you to pass a param to include or not include applications. Pass 1 to include them. 0 to not.
- getScheduledShifts
  Returns the scheduled shifts for the given date range. Use mm/dd/yyyy.
- getScheduledShiftsForEmployee
  Returns the scheduled shifts for the given date range. Use mm/dd/yyyy.
- getStaff
  Returns all active Employees in your account. For a list of all staff (active and inactive) use getAllStaff().
- getStaffCategories
  Returns all Employee Categories (schedule names) in your account.
- getStaffDetails
  Returns the Employee for the given ID.
- getStaffInfo
  Returns the Employee who made the request.

XML/JSON formatted-RPC APIs

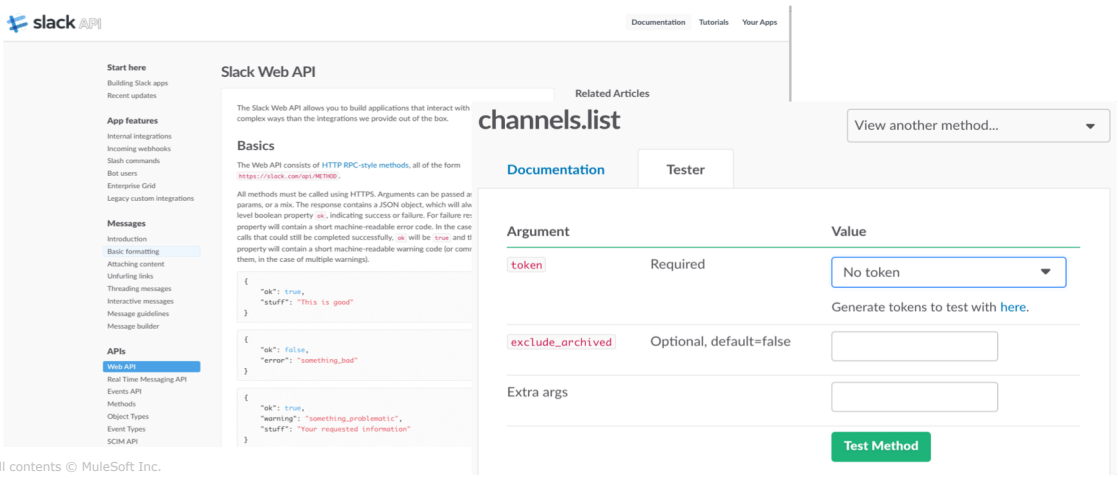
MuleSoft

- Use HTTP as the communication protocol
- Return data in XML/JSON format
  - Tightly coupled to the RPC (Remote Procedure Call) type
- Require extensive documentation to be used
  - Require developers to know the name of the procedures and specific parameters and the order of the parameters
- Require separate commands for separate actions
- Are stateless
- Are easier for developers to use compared to SOAP APIs

Walkthrough 1-2: Review and make a call to an RPC API

MuleSoft

- Examine the Slack RPC API
- Make a call to an RPC endpoint



REST APIs

MuleSoft

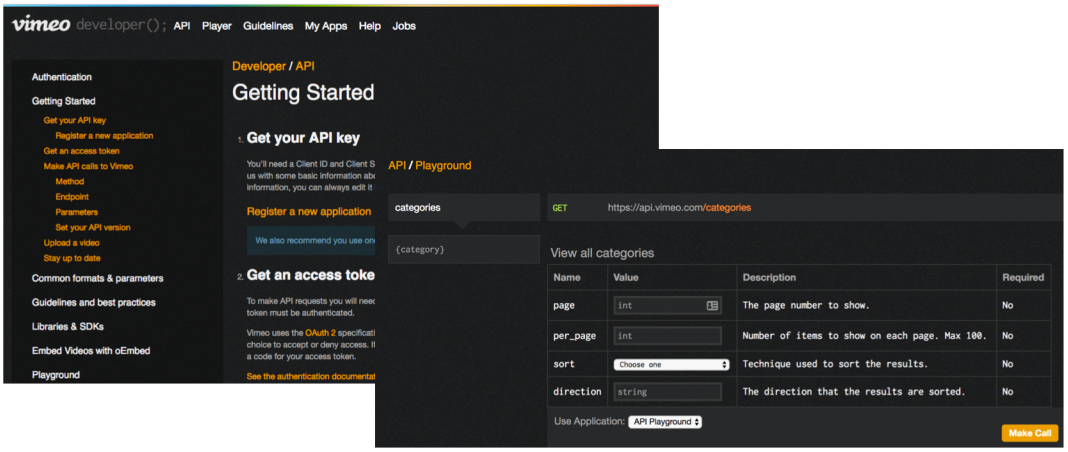
- Typically used over HTTP
- Supports any content type (XML, JSON, YAML…)
- Requires single command for multiple actions
- Stateless
- Are less difficult for new developers to use compared to SOAP APIs

**Rest of the modules will focus on REST APIs

Walkthrough 1-3 Review and make a call to a REST API

MuleSoft

- Examine the Vimeo REST API
- Make a call to a REST API endpoint to retrieve information

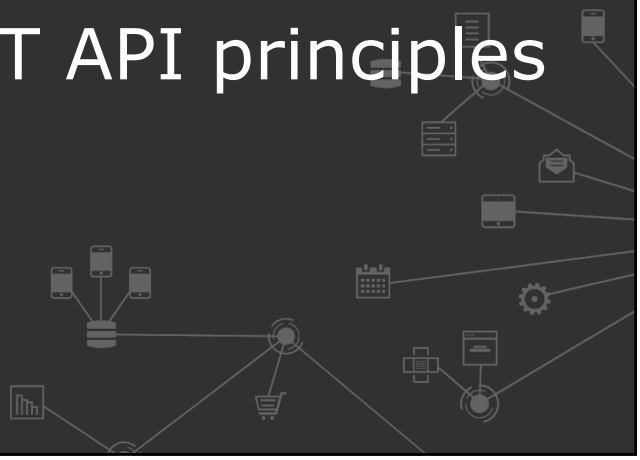


To summarize the three common formats of Web APIs

MuleSoft

SOAP	RPC	REST
- Tightly coupled with XML format - Stateless or stateful - Calls are sent through POST method - Requires methods and resources to be defined using WSDL	- Tightly coupled to RPC type (XML or JSON) - Stateless - Calls are sent through GET or POST - Separate URI for each action/method - Requires users to know the name of procedure, and order of parameters in the method call	- Supports any content type - Stateless - Calls are sent through all HTTP methods (CRUD) - Single resource for multiple actions - Does not expose methods while returning data

Introducing REST API principles



Enforcing REST principles in an API

MuleSoft

- Separation of concerns between client and server
  - Changes to the client application should not affect the server application and vice versa
  - The applications can grow and scale independently
- Uniform interface
  - Allows client and server to communicate using a single language, independent of the architecture of each of them
  - Provides standard means of communication by decoupling implementation
  - Allows independent evolution of services without tight coupling to the API layer
- Stateless
  - The API should not rely on server to store the client session data to make a call
  - The API should provide data in that call itself
  - To reduce memory, session information is stored in the client

Enforcing REST principles in an API

MuleSoft

- Client-side caching
  - The API should encourage clients to cache data on their side by specifying this information in responses
    - This is a result of increased request overhead due to the REST principle of statelessness
  - Helps reduce number of interactions with the API
  - Provides API consumers with tools to deliver fast and efficient applications
- Layered system
  - Creates scalable, modular applications promoting loose coupling
    - Each layer has a specific functionality and responsibility
  - Promotes flexibility, and security through multiple layers
  - Reduce cost, increase delivery speed and maintainability

Enforcing REST principles in an API (cont)

MuleSoft

- Code on demand
  - Allows for code or applets to be sent via the API for implementing within a client application
    - For example, a client application can send a request to a server to receive code that can be executed locally to utilize the resources that the client application has access to
  - Increases time efficiency to add new features, extensibility and configurability
  - Optional principle to implement in APIs (since it reduces visibility)

Planning to design REST APIs

Introducing API design-first approach (Spec-Driven Development)

MuleSoft

- Spec-Driven Development
  - Separates design and development into two separate processes
    - Increases agility and helps approach design iteratively
- Consumer driven
  - Start by figuring out what API consumers really want from your API
  - Think about design that best serves them
  - Define the interface between infrastructure and the API consumers first
- Focus exclusively on design to model APIs cleanly and consistently
  - API design should not be constrained by existing limitations in systems or by legacy infrastructure
    - These constraints should come into the picture while implementing the API
  - Get feedback from API consumers on usability and functionality along the way
    - Include tools to discover and learn the API

API consumer focused design considerations

MuleSoft

- Audience
  - Who is the API for?
- Actions
  - Which actions do they need access to?
- Interactions
  - Explain how the API will interact with existing backend services or applications
- Maintenance
  - How are you going to maintain the API?
- Versioning
  - How are you going to version the API?

API consumer focused design considerations

MuleSoft

- Documentation
  - How are you going to document the API?
- Access
  - How will API consumers access/interact with the API? Will they have to use an API key or an access token?
  - What are the security considerations?
- Support
  - How are you going to manage support?

Summary

Summary

MuleSoft

- Three common web API formats
  - SOAP
  - RPC
  - REST
- REST API architecture allows for various constraints and governance
- Highly recommended to take an API-first design approach with the following considerations in mind
  - Take an approach of exposing data in the system through APIs
  - Design for the API consumer
    - Prototype designs and work with key stakeholders before implementation
  - Design for maintainability and long-term usability