Video – 103(What are APIs) by Salesforce hulk:

* API stands for Application programming interface.
* This is the way to do communication between two different applications.

Video – 104(Need of APIs in Salesforce) by Salesforce hulk:

* Allowing other program to interacts with our developed application.
* There is other use case of API is to send the data over multiple type of device like when we use face book on web application or mobile app. There is no two type of backend of web app and mobile app. Backend is same for both the devices the UI is different only.
* You don’t need to expose the complete code for showing your data. You have just given a window to take their required data.
* Facebook is not only social media platform. It’s providing identity service as well. Whenever you go on any other platform then facebook provide identity service there.

Video – 105(Languages Used in creating APIs in Salesforce):

* JSON and XML are two standard languages for communication between two system.
* Standard languages help to communicate between two different system. There is possibility that System A which developed in PHP and System B which developed in Java can communicate via standard languages.
* JSON:
  + JSON stands for JavaScript Object Notation.
  + It is lightweight language/format used to interact between different systems and software.
  + It is widely used as it is very easy to read and write by humans and also very easy for machines to parse and generate.
  + It is just a text format that is completely language independent which means we can use it inside any programming language which makes it an ideal data-interchange language and hence it can be considered as universal data structure.
  + JSON is built on two structures:
    - Object: A collection of name/value pairs.
    - Array: An ordered list of values.
  + JSON text is always written in key-value pair:
    - {“key” : “Value”}
  + Example:
    - {“Name”: “Pankaj”}
  + Here a value can be string in double quotes, or a number, or true or false or null, or an object or an array.
  + Simple JSON Object:
    - {“key1”: “Value1”, “key2”: “Value2”, “key3”: “Value3”}
  + Example:
    - {“Name”: “Shrey”, “Age”: 23, “City”: “Jainpur”}
  + Simple JSON Array:
    - {“key”: [“value1”, “value2”, “value3”] }
    - Example:
      * {“Name”: [“John”, “Paul”, “Donna”, “Jessica”] }
    - Example 2:
      * {“employee”:
      * [
      * {“name”: “John”, “email”: “[john@gmail.com](mailto:john@gmail.com)”, “age”: 23},
      * {“name”: “Paul”, “email”: “[paul@gmail.com](mailto:paul@gmail.com)”, “age”: 28},
      * {“name”: “Donna”, “email”: “[donna@gmail.com](mailto:donna@gmail.com)”, “age”: 33},
      * {“name”: “Jessica”, “email”: “[jessica@gmail.com](mailto:jessica@gmail.com)”, “age”: 41}
      * ]
      * }
  + JSON Structure can be nested as well.
  + Example:
    - {

“firstName” : “Amit”,

“lastName”: “Bansal”,

“age”: 58,

“address” : {

“streetAddress”: “Plot -6 , Gandhi Nagar”,

“city”: “Jaipur”,

“Stage”:”Rajasthan”,

“postalCode”: “3020202”

}

* + - }
* XML:
  + XML stands for extensible Markup language.
  + It is similar to HTML, designed to store and transport data over Internet.
  + It is both human and machine readable.
  + It does not use predefined tags which means we can create our own tags to define the data.
  + It stores data in plain text format which provides a software and hardware independent way of storing, transporting and sharing data.
  + It separates data from presentation which means that the same data can be used in many different presentation scenarios.
  + Example:

<BusinessCard>

<Name> Pankaj Kumar</Name>

<Title>CEO</Title>

<Company>Cyntexa</Company>

<Email>[pankajvitmca@gmail.com</Email](mailto:pankajvitmca@gmail.com%3c/Email)>

<Phone>6200753817</Phone>

</BusinessCard>

* + Note: XML Tags are Case Sensitive.
  + Example2:

<Breakfast\_menu>

<food>

<name> Belgian Waffles</name>

<price> 120 Rs. </price>

<description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>

<calories>650</calories>

</food>

<food>

<name> French Toast</name>

<price> 150 Rs. </price>

<description> Thick slice made from homemade grains</description>

<calories>850</calories>

</food>

</Breakfast\_menu>

* JSON vs XML:
  + Both JSON and XML can be used to receive or send data from or to a web server.
  + JSON is similar to XML because
    - Both JSON and XML are “self-describing” (human readable)
    - Both JSON and XML are hierarchical (values with values)
    - Both JSON and XML can be parsed and used by lots of programming languages.
  + JSON is not similar to XML because
    - JSON doesn’t use end tag.
    - JSON is shorter.
    - JSON is quicker to read and write.
    - JSON can use arrays.
  + Why JSON is better than XML:
    - XML is much more difficult to parse than JSON.
    - JSON is parsed into a ready to use JavaScript object.
    - JSON has less size than XML because, XML carries end tag as overhead which increases the size of data that is transported whereas JSON there is no end tag which reduces the size of data.
    - JSON easily defines the data as compared to XML.

Video – 106(How Communication happens in APIs in Salesforce):

* There is System A and other System B and we have written an API on System B and it’s the API of system B is exposed to public. So, whenever system A hits on system B, it is called “callout” and API code is called web services which written in system B. so, callout thing is called “request” and whatever data returned my system B is called “response”.
* Webservice is nothing but just code which executes when callout happens from another system.
* We need authentication key for callout the system B. if API requires authentication.
* After authentication of first time by passing username and password, server provides one authentication token. So, when you can use the multiple feature of application without passing user id and password multiple time.

Video – 107(How to Create an API in Salesforce) by Salesforce hulk:

* Types of APIs:
  + 1. REST(Representational State Transfer) API
  + 2. SOAP(Simple object access protocol) API – it stands for simple object access protocol. In this protocol, client side need to know the metadata of server side as well. When you need to develop for public use then go for REST API. If it’s needed to know metadata of server in client then go for SOAP API.
  + 3. Metadata API

Video – 108(REST API in Salesforce Part – 1) by Salesforce Hulk:

* HTTP Methods
  + Get – When you want to get only the information then use “Get” method there.
  + Post – When you want to save some information on server then use “Post” method there.
    - Note:
    - 1. There can be only one Post, one get, one put or one patch or one delete Method in API class.
    - 2. We can use any class name and method name.
  + Put -
  + Patch –
  + Delete
* Instructor of this course written following line of code in this video:
  + CreateLeadAPI.apxc class

@RESTResource(urlMapping='/CreateLead/')

global class CreateLeadAPI {

@HTTPPost

global static String creatingLeadRecord(){

String fName,lName,comp;

RestRequest req = RestContext.request;

RestResponse res = RestContext.response;

String arr = req.requestBody.toString();

System.debug('Body is:'+arr);

return 'Hey';

}

}

* Mock Callouts through following tools for checking our developed APIs
  + Workbench – it is specific to Salesforce platform. It is also used to import or export of data from Salesforce or to Salesforce.
  + Postman –

Video – 109(REST API in Salesforce Part – 2) by Salesforce Hulk:

* logged In through workbench -> Go on Utilities -> REST Explorer
* Choose Post as HTTP method:
  + Passed - /services/apexrest/**CreateLead/** for Rest API call on above class “CreateLeadAPI”. It’s called following line of code

@RESTResource(urlMapping='/CreateLead/')

global class CreateLeadAPI {

@HTTPPost

global static String creatingLeadRecord(){

String fName,lName,comp;

RestRequest req = RestContext.request;

RestResponse res = RestContext.response;

String arr = req.requestBody.toString();

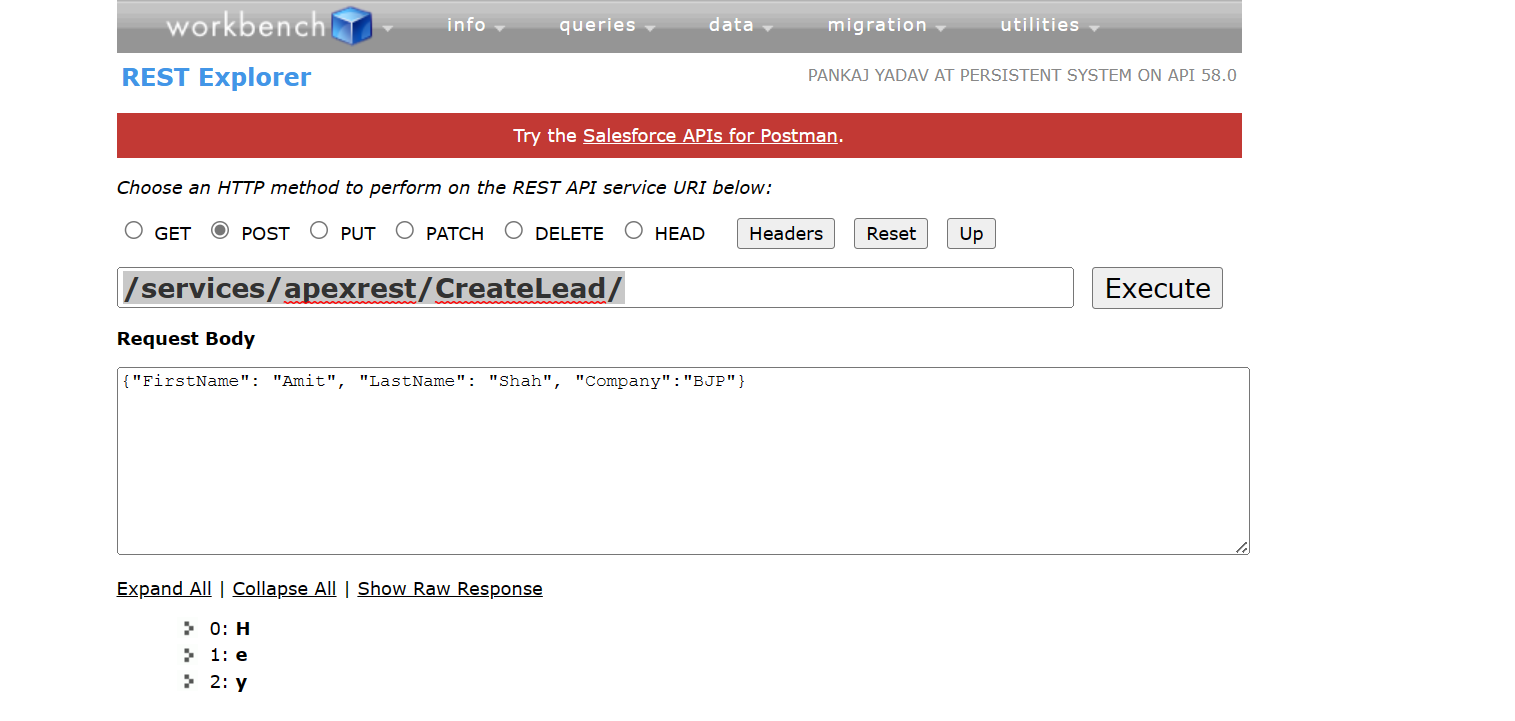
System.debug('Body is:'+arr);

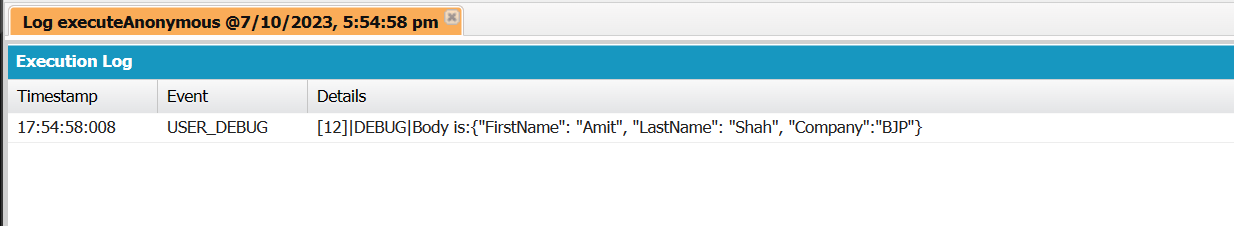
return 'Hey';

}

}

* + After click on Execute, it has called above method from CreateLeadAPI class.
  + It’s return ‘Hey’. And passed the body into the Salesforce system. As you can see in log screenshot below. Just like below screenshot





* Write code where you do mock callout from workbench(calling the apex code from external environment) and it will create lead record inside Salesforce org.

@RESTResource(urlMapping='/CreateLead/')

global class CreateLeadAPI {

@HTTPPost

global static String creatingLeadRecord(){

String fName,lName,comp;

String fieldName,fieldValue;

RestRequest req = RestContext.request;

RestResponse res = RestContext.response;

String jsonBody = req.requestBody.toString();

JSONParser parser = JSON.createParser(jsonBody);

while(parser.nextToken() != null){

if(parser.getCurrentToken() != JSONToken.END\_OBJECT){

fieldName = parser.getCurrentName();

fieldValue = parser.getText();

if(fieldName == 'FirstName'){

fName = fieldValue;

}

else if(fieldName == 'LastName'){

lName = fieldValue;

}

else if(fieldName == 'Company'){

comp = fieldValue;

}

}

}

Lead l = new Lead(firstName=fName, LastName=lName, Company=comp);

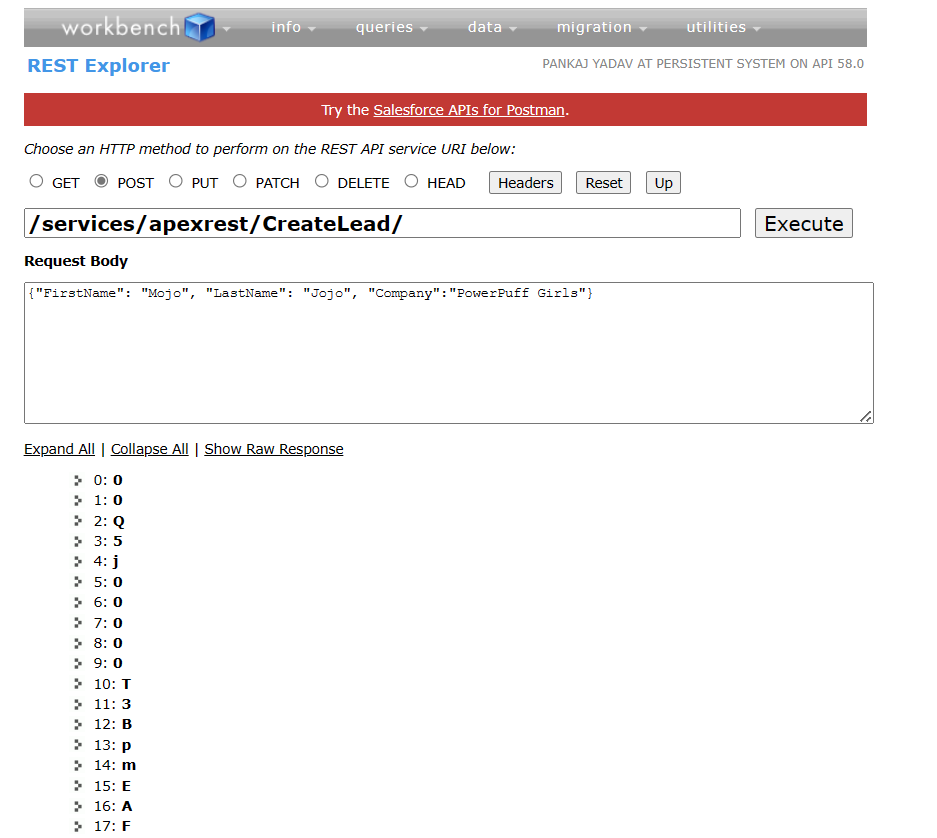
insert l;

return l.Id;

}

}

* I did callout from workbench like following screenshot and return as recordId after insertion of record.



Video – 110(REST API in Salesforce Part – 3) by Salesforce Hulk:

* Choose Post method and pass following parameter
  + https://persistentsystem-b5-dev-ed.salesforce.com/services/apexrest/CreateLead/
* Following parameter as header
  + Authorization as bearer accesstoken(made through connected ap)
  + Authorization as “Bearer 00D5j00000AdmN2ARIAQEBpScUoqas27OzRhpOtNS7psecrby1kWNt6HEHNebM6fp8D89NpOWxy6P.FMXJNtN3riRxd.6Y8IiElS0eqMLZDlI3r”
  + Content-type as application/json
* Pass following thing in body as raw and json

{

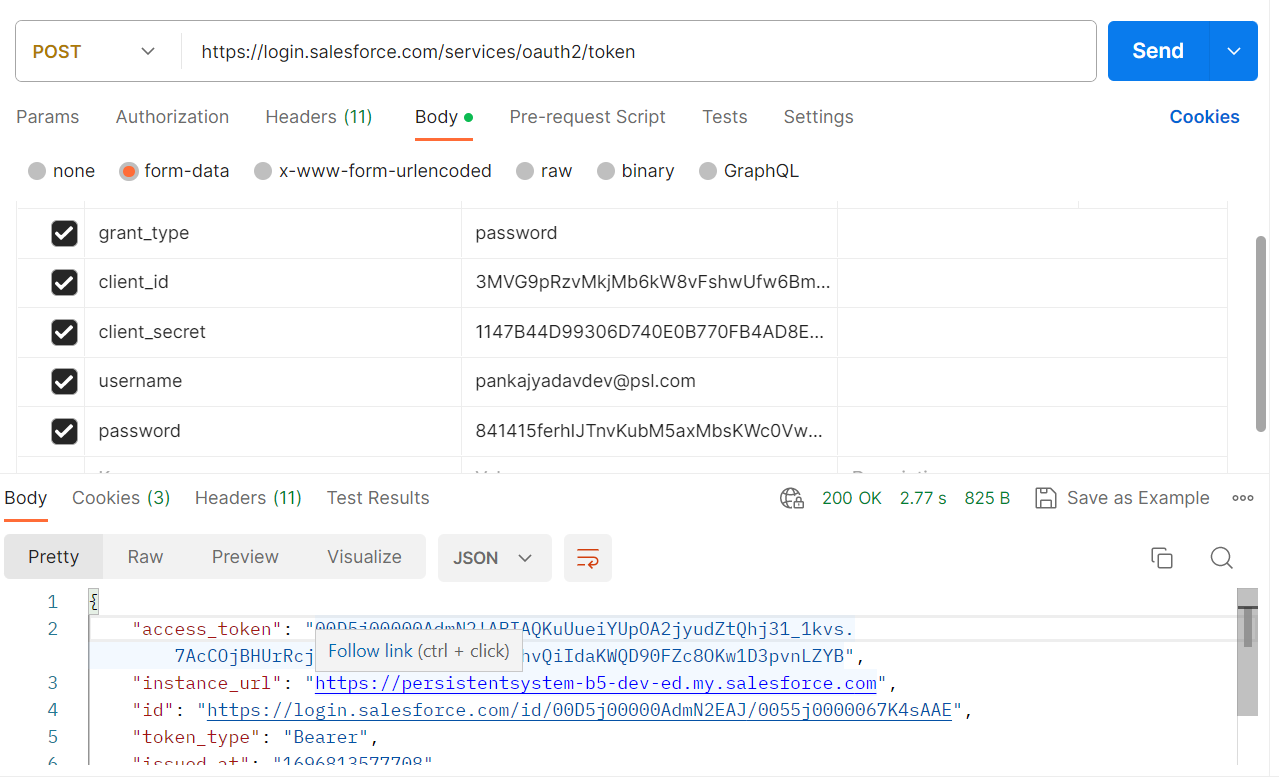
    "FirstName": "Nirmala",

    "LastName": "Sitaraman",

    "Company":"BJP"

}

* Create a method in which we gonna pass a objectapi name,record-id of that object and all response will get you as json format.
* How to get access token from Salesforce org for testing of rest api code



Video – 111(REST API in Salesforce Part – 4) by Salesforce Hulk:

* Instructor has start write get method code in above CreateLeadAPI class.

@RESTResource(urlMapping='/CreateLead/')

global class CreateLeadAPI {

@HTTPGet

global static String showLead(){

String objName;

Id i;

RestRequest req = RestContext.request;

RestResponse res = RestContext.response;

System.debug(req.headers.toString());

objName = req.headers.get('ObjectName');

i = req.headers.get('Id');

String qr = 'Select Name from '+objName+' where Id= :i'+' Limit 1';

sObject s = Database.query(qr);

String jsonResponse = '{\"Name\":\"'+s.get('Name')+'\"}';

return jsonResponse;

}

}

* Get are most and widely used method in API and web services. This method is used to retrieve the data from the service at specified resource.
* Since a get request is only requesting the data and not modifying any resources. So, it consider as safe and independent method.
* You can do some changes by passing some data through get method but it’s not best practices to pass data from get method and modify the database through it.

Note: you can’t pass data from body in get method.

* What is the difference between passing data from header and body ?
  + When you pass data from body then it’s encrypted and when it arrives in other side then it’s decrypted. It’s secure
  + But when you pass data from header than anyone in the channel can sneak what we are passing.

Video – 112(REST API in Salesforce Part – 5) by Salesforce Hulk:

* Post request are used to set data to the API server to create or update a resource. The data sent to the server is stored in the request body of the http request.
* Post request are non-idempotent as it mutate data on the backend server by creating or updating the resource.
* Put request are used to send data to the API to create and update a resource.
* The difference between put and post request is that put request are idempotent that is calling the same put request multiple times will produces the same result in contrast, calling the post request repeatedly may have side effect of same resource multiple times.
* Patch method – It is similar to post and put but only difference with patch is that you apply partial modification to the resource.
  + It is also non-idempotent.
* Delete – It is used to delete the resource at specified URL.
* When we call API of external system then it’s called as apex callout.
* When you do callout on external system then we do request for access token and when ever we do callout to external system. We have to pass access token there.
* Create a class called accountManagement in apex which you have following functionality first, if we pass the record id then it should return us the record information in the json format. Second, using the api, we should be able to create a new account record. If you want to update modify any field in account record then it should be doing it by passing the record id and field name and field value which you want to modify. Fourth, if we pass a record id, we should be able to delete record id. If we pass the json array, which included multiple account information then we should be able to create all those account in one goal.
* Note: when external system callout your api then it’s called inbound integration.
* Note : when you callout on external system api then it’s called as outbound integration.

Video – 113(REST API in Salesforce Part – 6) by Salesforce Hulk:

* In this video, instructor talks about how to apex callout means how to consume another system API.
* He is explaining apex callout to external system API by making a real time mini project that is currency converter.
* When you want to get resource from another system which doesn’t exist in your system then you need to do external callout to the system.

public class CurrencyConverterClass {

public static Decimal convertCurrencyToINR(Decimal usd){

Decimal inr = 0.0;

HTTP h = new HTTP();

HTTPRequest req = new HTTPRequest();

req.setEndpoint('https://free.currconv.com/api/v7/convert?q=USD\_INR');

req.setMethod('GET');

HTTPResponse res = h.send(req);

Map<String,Object> jsonBody = (Map<String,Object>)Json.deserializeUntyped(res.getBody());

Map<String,Object> mMap = (Map<String,Object>)jsonBody.get('results');

Map<String,Object> mMap2 = (Map<String,Object>)mMap.get('USD\_INR');

Decimal conversionRate = (Decimal) mMap2.get('Val');

inr = usd \* conversionRate;

return inr;

}

}

Video – 114(REST API in Salesforce Part – 7) by Salesforce Hulk:

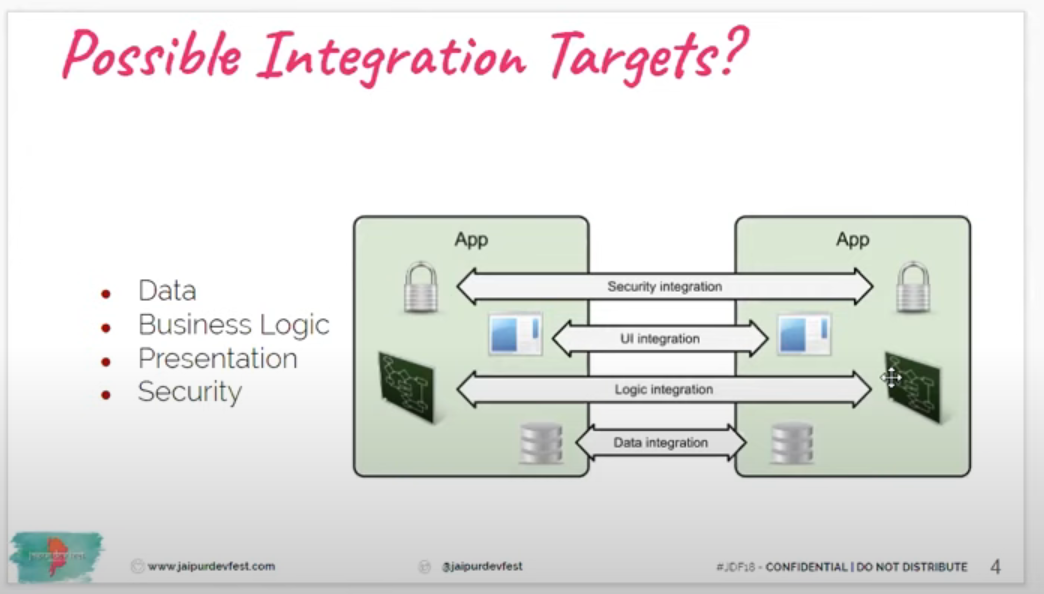
* Whenever we do callout to external system then we have to configure remote site setting to external system endpoint for do the callouts.
* Whenever we call external system callout then we need API key from external system for callout.
* In this video, instructor explain above code and takes a lot of time figure out some errors.

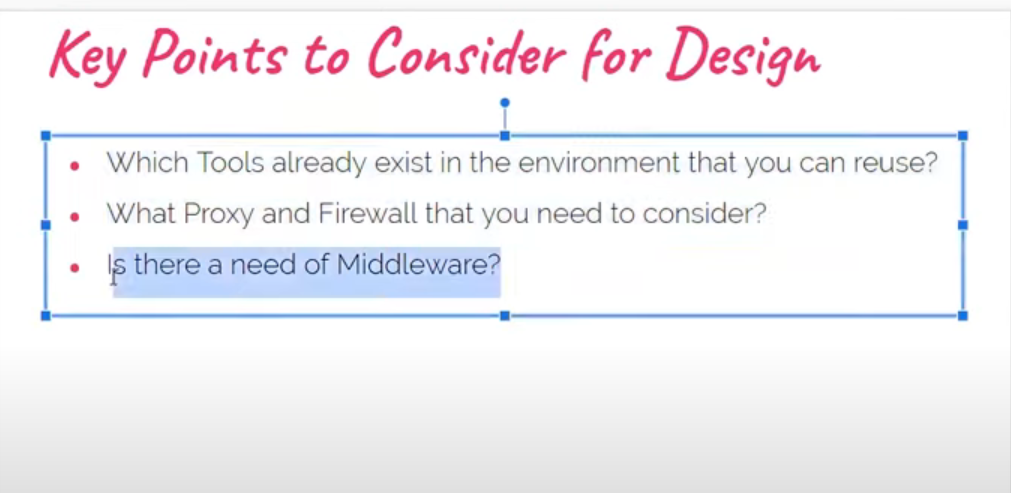
Video – 115(REST API in Salesforce Part – 8) by Salesforce Hulk:

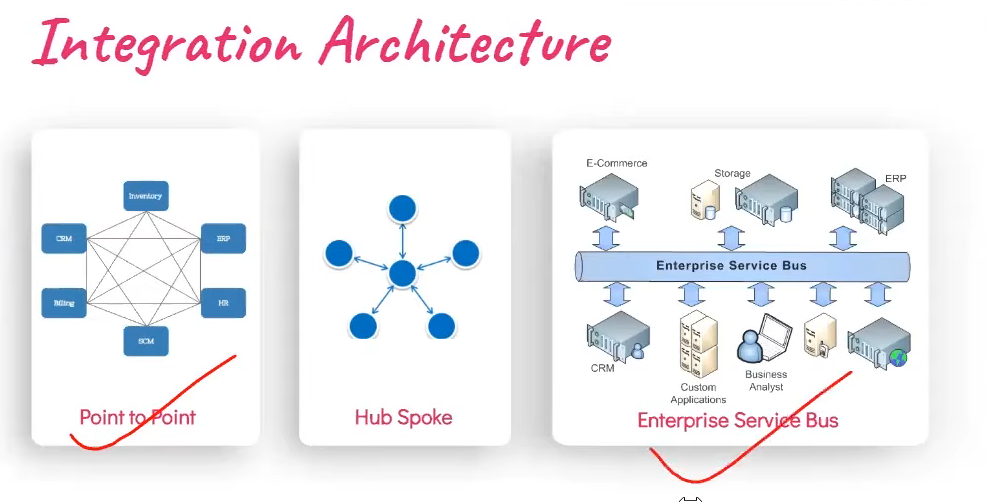
* Assignment given by instructor – Create a field in opportunity object labeled as amount in INR. Whenever an opportunity is created with its amount (USD). The amount in INR should automatically get calculated based on current conversion rate. It should be stored inside amount in INR field.
* Scenario for Practice – whenever customer makes a payment, let say through stripe, the commission get automatically get calculated on the opportunity record and email should sent to the customer included the amount and the commission. In above case, where will you create API for the system ?
  + Answer – you will create API in SFDC and Callout will happen from Stripe.

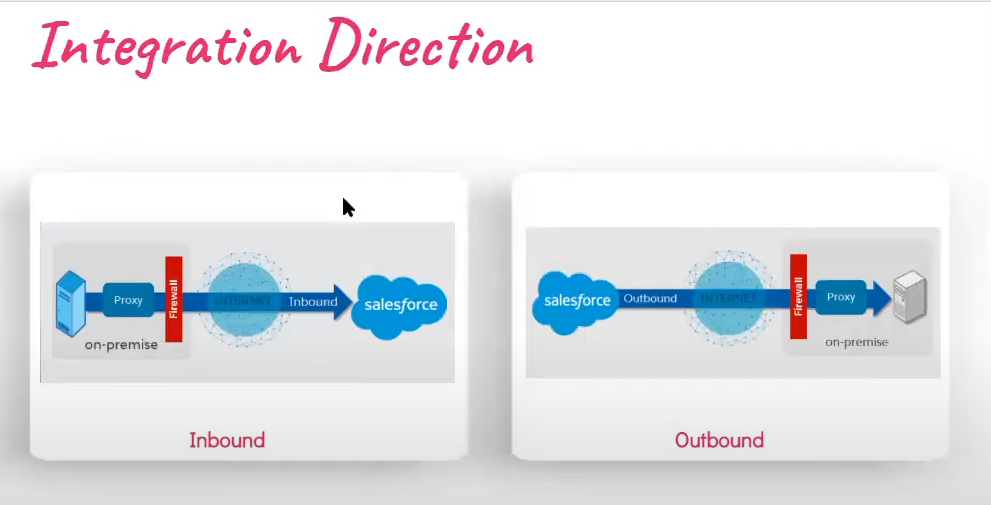
Video – 116(REST API in Salesforce Part – 9) by Salesforce Hulk:

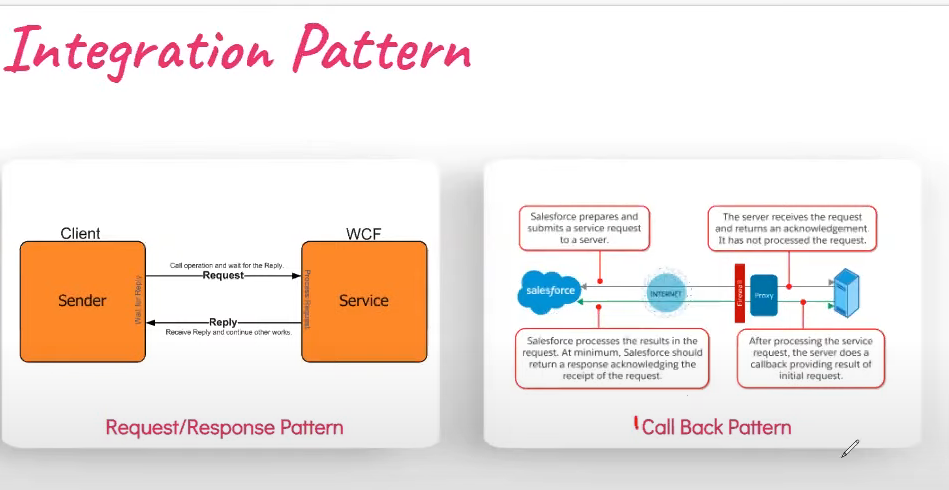
* Integration is the act of bringing together smaller components into a single system that functions as one.

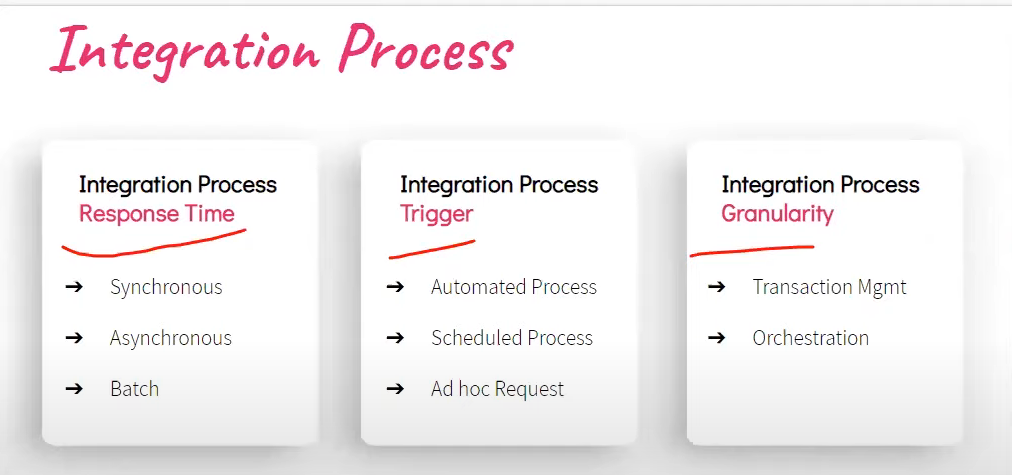


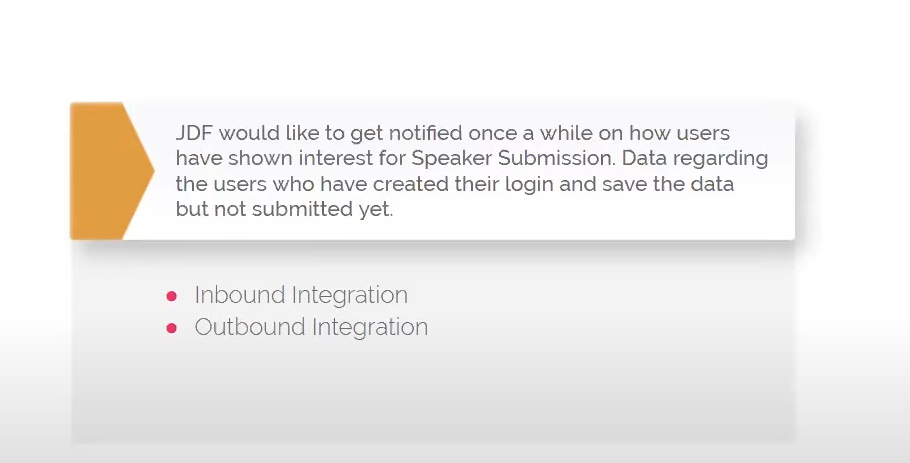




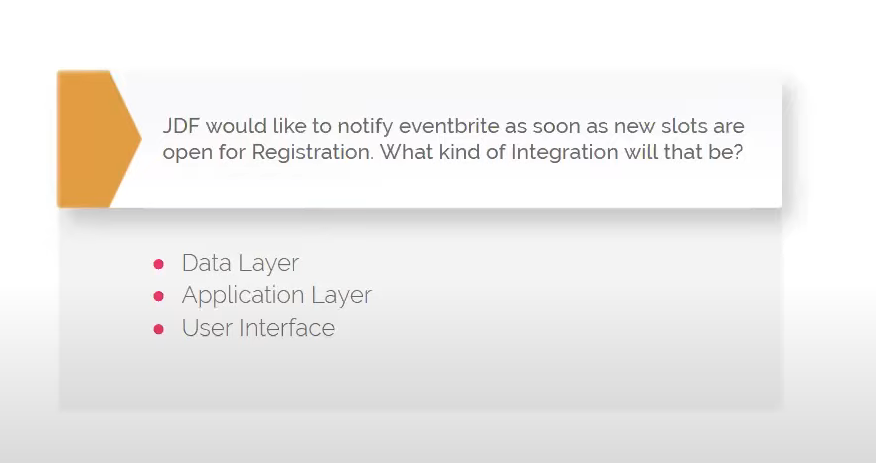






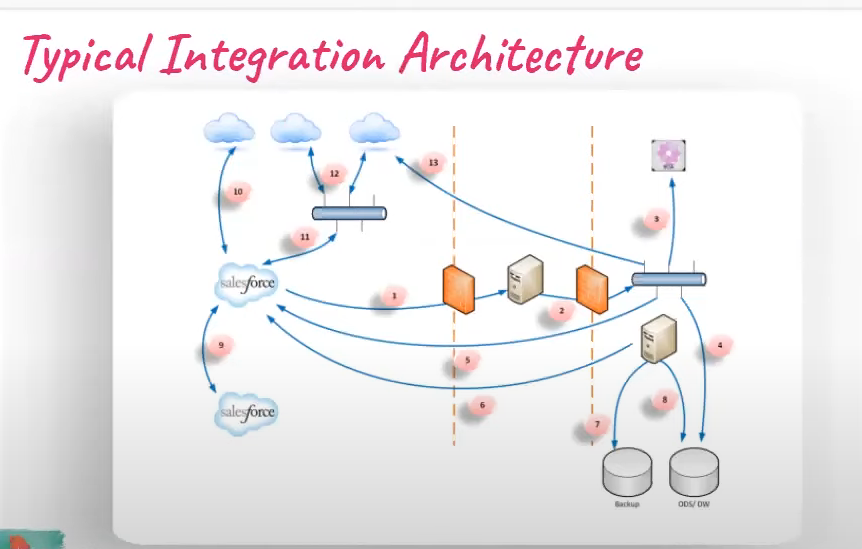


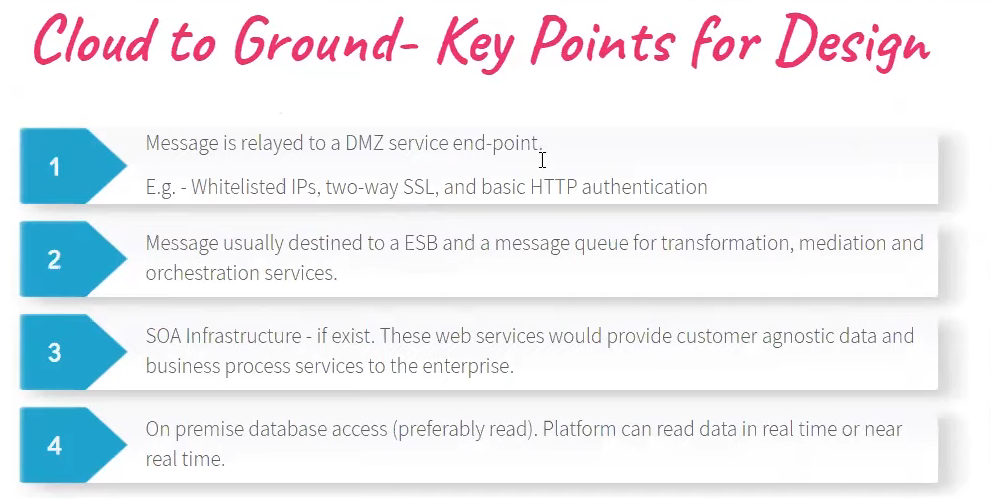
Answer: Inbound Integration



Answer: Data Layer



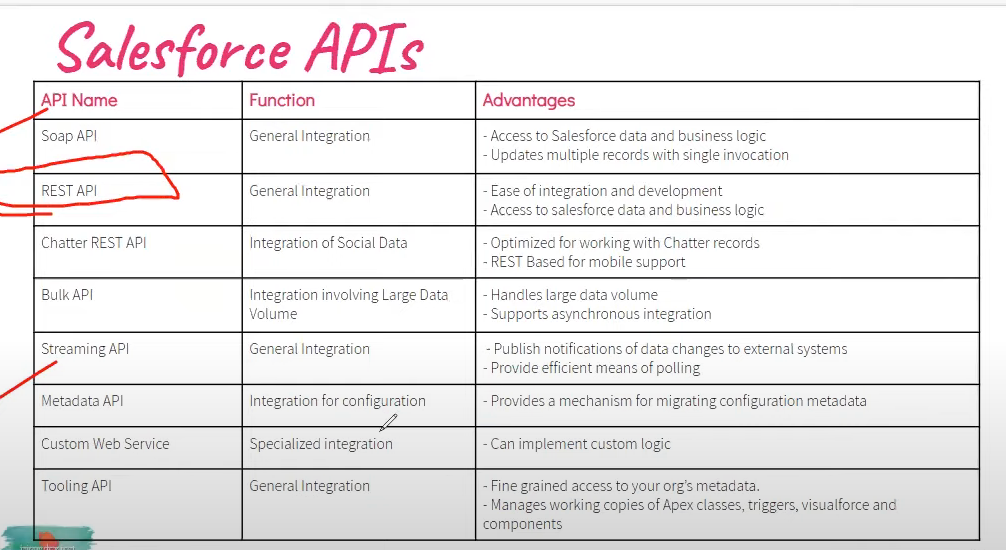


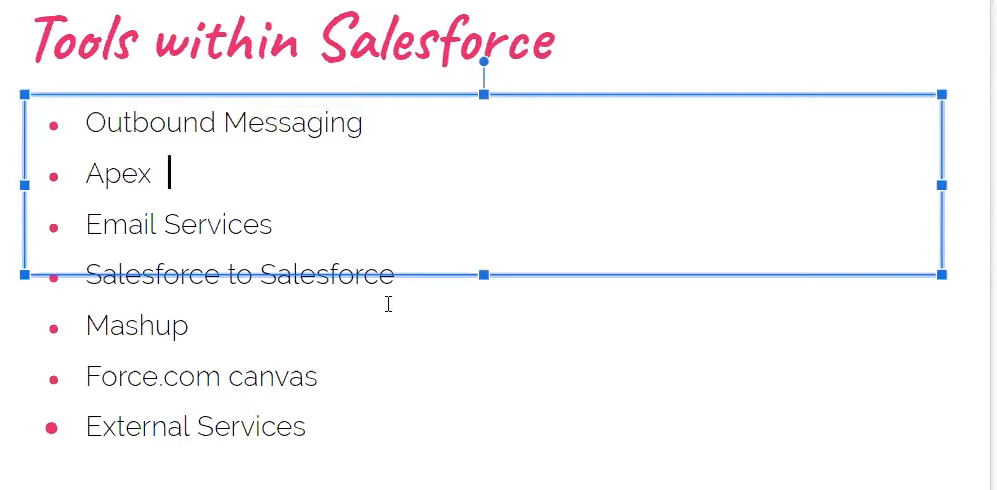


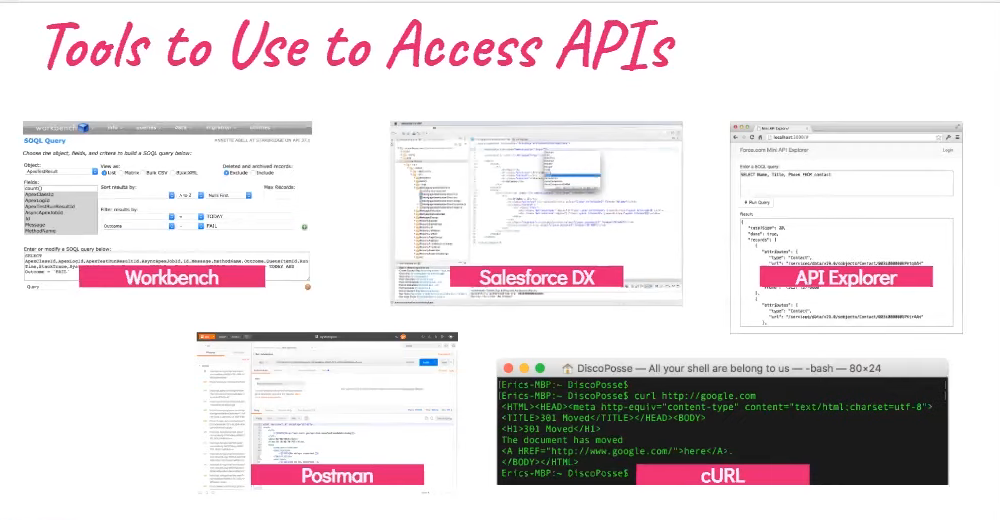
While talking about these slide, he told that these slides are not for developers, these slides are for architect. he started talks about type of architect. There are two level of architect

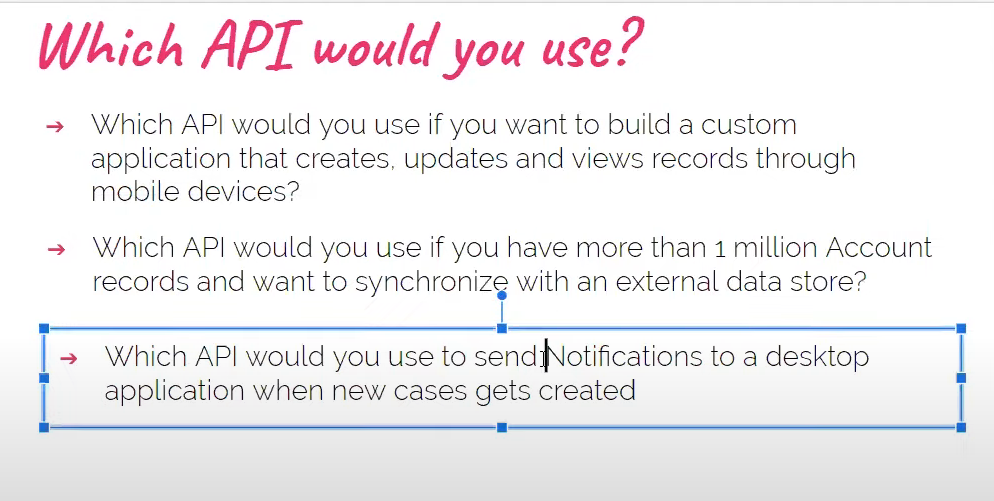
1. System Architect
2. Integration Architect

There are 14 certification, you need to take to become a architect in which 7 for system architect and 7 for Integration architect and then there is a test for CTA(Certified technical architect) and cost of CTA is 6000 Dollar means 4 Lac in indian rupees.









Answer of 1 – REST API

Answer of 2 – Bulk API

Answer of 3 – Outbound Message

Video – 117(SOAP API in Salesforce Part – 9) by Salesforce Hulk:

* It stands for simple object access protocol.
* Suppose there is two system, system A and system B then whenever you are calling the system B but before calling system B, you should be aware about system B metadata and WSDL file stuff like that.
* WSDL stands for Web Services Description Language. It's an XML-based file that describes a web service to a client application. WSDL was developed jointly by IBM and Microsoft.
* WSDL documents describe:
  + The location of the web service
  + The methods available in the web service
  + The various operations that the web service provides
  + Information about the service's protocols, such as SOAP
* This way of communication doesn’t use much but in some case, it’s used. It’s kind of old system of communication between two systems.
* For working on SOAP, instructed told me to download Eclipse IDE. It is step – 1 for working with SOAP API.
* Step – 2: Need to download WSDL files of our Salesforce org and Salesforce store about the metadata in XML format and need to convert into WSDL file.
  + For finding metadata of Salesforce org, need to go into setup -> Go on “API”
  + There are multiple type of WSDL available inside Salesforce org:
    - Enterprise WSDL
    - Partner WSDL
    - Apex WSDL
    - Metadata WSDL
    - Tooling WSDL
* SOAP was much more secure in old days than REST API because whoever have WSDL file in their system. Only those can communicate with the system.
* Step – 3: Generate Enterprise WSDL
  + Save the generated file somewhere in your system
* Step – 4: Generate Partner WSDL
  + Save the generated file somewhere in your system
  + Save the file with .xml extension
* Step – 5: Generate Metadata WSDL
  + Save the generate file

Video – 121(Metadata API in Salesforce):

* Till now, we transfer data from one system to another system.
* Now through metadata API, you can create metadata inside Salesforce through external system.
* Even you can create object inside Salesforce from another system through metadata API.
* Suppose, you get a task like create 10 objects and there are 100s of fields in ever object and need to do it same for 5 orgs and you don’t want to create through point and click tool.
  + In this case, you can use metadata API.