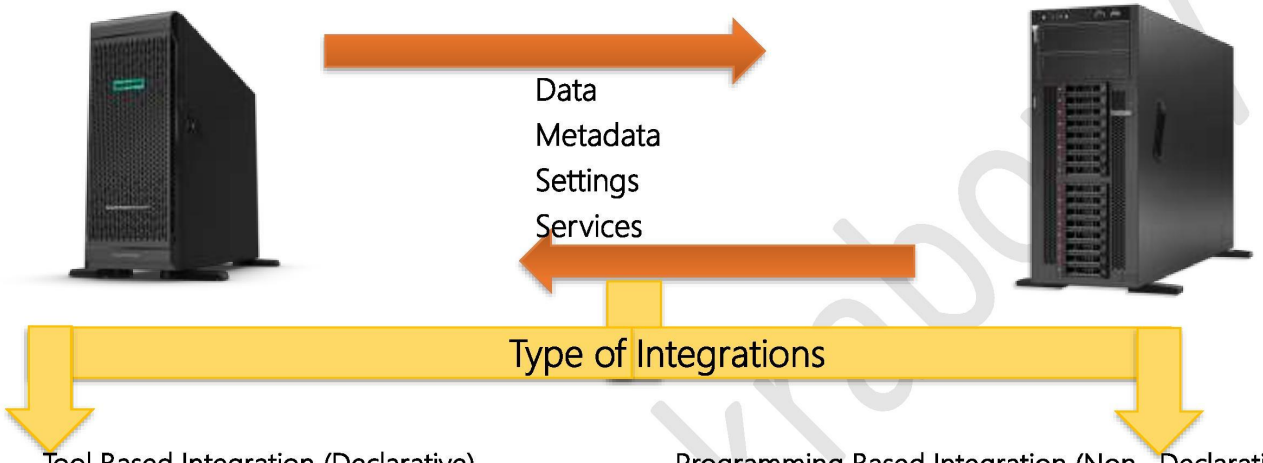
Salesforce Course Material (Theoretical and Practical)

Integration – JSON , XML & HTTP

1

Introduction to SFDC Integration

Any exchange of Data, Metadata, Settings, Configuration or Services between two system is called integration.



Tool Based Integration (Declarative)

- Dataloader
- MuleSoft
- SSO
- Informatica
- Dell Boomi
- DB AMP
- Visual Code
- Eclipse
- ANT

Programming Based Integration (Non –Declarative)

Webservices (WSDL)

- REST
- SOAP

Metadata API (SOAP)
Bulk API (REST)
Chatter API (REST)
Streaming API(REST)
Partner/Enterprise API (SOAP)
Tooling API(SOAP)

Salesforce Integration (4 Types)

Security Integration : Integrate authentication mechanisms across applications to improve the user experience and minimize user administration.

User Interface Integration : Combine the UIs of two or more apps to create composite apps with little or no rework on the UI of each individual app.

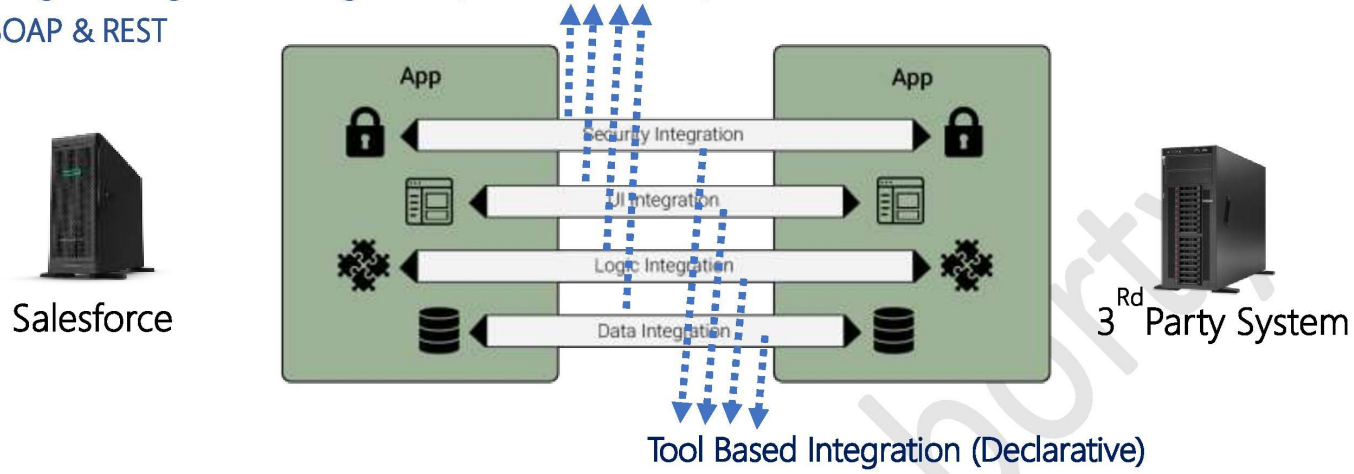
Business Logic Integration : Business logic often spans multiple application systems. Extending related business logic from one app to another helps to implement complete end to end business processes.

Data Integration: Integrating applications at the data layer is a common scenario. For example, multiple apps written in different programming languages can all use an open API and manage related data in one shared database

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

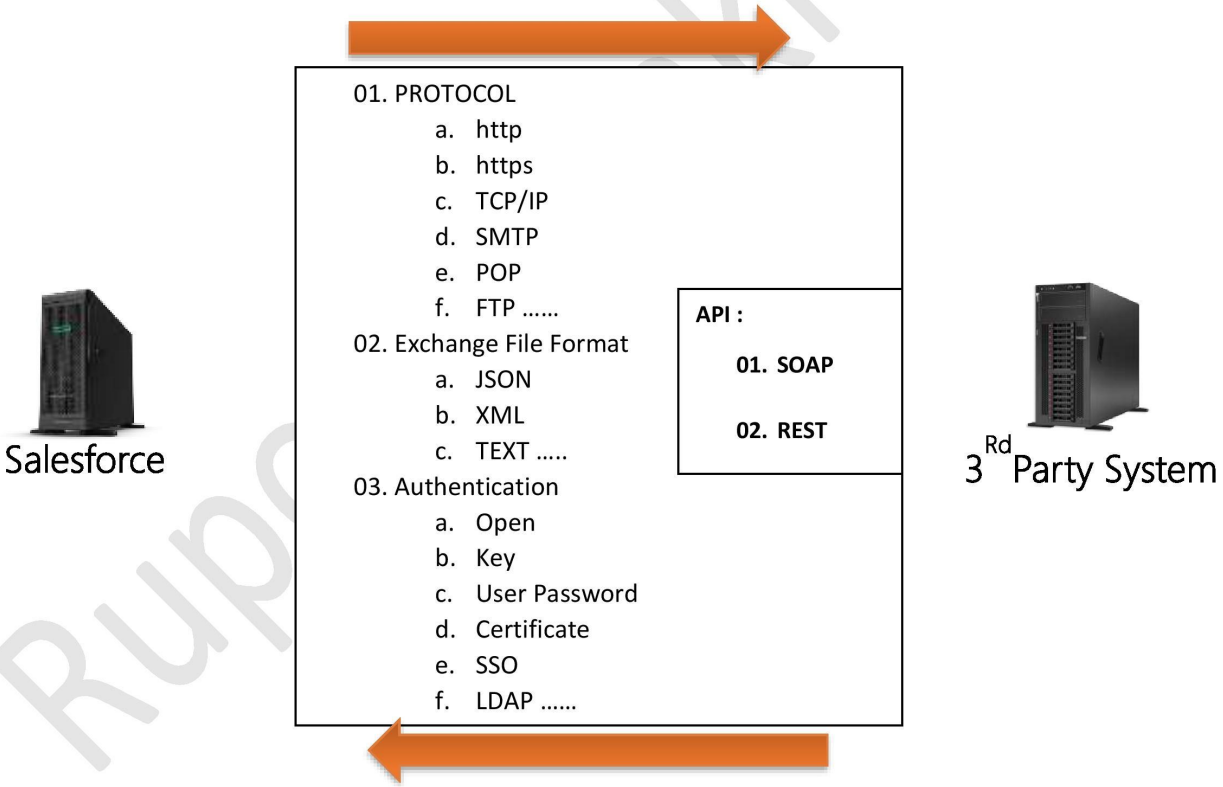
Programming Based Integration (Non –Declarative)

SOAP & REST



Tool Based Integration (Declarative)

Integration Concepts



RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

Data Exchange Format – JSON

JavaScript Object Notation (JSON) is an open-standard file format or data interchange format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications, such as serving as replacement for XML in AJAX system.

- JSON stands for JavaScript Object Notation
- JSON is a lightweight format for storing and transporting data
- JSON is often used when data is sent from a server to a web page
- JSON is "self-describing" and easy to understand.

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON Example

Eg1 : {"firstName":"Rupom", "lastName":"Chakraborty"}

Eg2: {name: "Rupom", age: 36, city: "Hyderabad"};

Eg3 :

```
{
"employees":[
   {"firstName":"Rupom", "lastName":"Chakraborty"},
   {"firstName":"Reyansh", "lastName":"Chakraborty"},
   {"firstName":"Madhura", "lastName":"Chakraborty"}
]
}
```

JSON In Salesforce

Under System Namespace salesforce has 4 JSON class to handle all JASON related request in Salesforce.

01. JSON Class

02. JSONGenerator Class

03. JSONParser Class

04. JSONToken Enum

JSON Class : Contains methods for serializing Apex objects into JSON format and deserializing JSON content that was serialized using the serialize method in this class.

Usage : Use the methods in the System.JSON class to perform round-trip JSON serialization and deserialization of Apex objects.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

JSON Methods :

The following are methods for `JSON`. All methods are static.

- **createGenerator(prettyPrint)**
  Returns a new JSON generator.
- **createParser(jsonString)**
  Returns a new JSON parser.
- **deserialize(jsonString, apexType)**
  Deserializes the specified JSON string into an Apex object of the specified type.
- **deserializeStrict(jsonString, apexType)**
  Deserializes the specified JSON string into an Apex object of the specified type.
- **deserializeUntyped(jsonString)**
  Deserializes the specified JSON string into collections of primitive data types.
- **serialize(objectToSerialize)**
  Serializes Apex objects into JSON content.
- **serialize(objectToSerialize, suppressApexObjectNulls)**
  Suppresses `null` values when serializing Apex objects into JSON content.
- **serializePretty(objectToSerialize)**
  Serializes Apex objects into JSON content and generates indented content using the pretty-print format.
- **serializePretty(objectToSerialize, suppressApexObjectNulls)**
  Suppresses `null` values when serializing Apex objects into JSON content and generates indented content using the pretty-print format.

JSONGenerator Class :_Contains methods used to serialize objects into JSON content using the standard JSON encoding.

Usage: The System.JSONGenerator class is provided to enable the generation of standard JSON-encoded content and gives you more control on the structure of the JSON output.

JSONGenerator Methods

The following are methods for `JSONGenerator`. All are instance methods.

- **close()**
  Closes the JSON generator.
- **getAsString()**
  Returns the generated JSON content.
- **isClosed()**
  Returns `true` if the JSON generator is closed; otherwise, returns `false`.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

- **writeBlob(blobValue)**
  Writes the specified `Blob` value as a base64-encoded string.
- **writeBlobField(fieldName, blobValue)**
  Writes a field name and value pair using the specified field name and BLOB value.
- **writeBoolean(blobValue)**
  Writes the specified Boolean value.
- **writeBooleanField(fieldName, booleanValue)**
  Writes a field name and value pair using the specified field name and Boolean value.
- **writeDate(dateValue)**
  Writes the specified date value in the ISO-8601 format.
- **writeDateField(fieldName, dateValue)**
  Writes a field name and value pair using the specified field name and date value. The date value is written in the ISO-8601 format.
- **writeDateTime(datetimeValue)**
  Writes the specified date and time value in the ISO-8601 format.
- **writeDateTimeField(fieldName, datetimeValue)**
  Writes a field name and value pair using the specified field name and date and time value. The date and time value is written in the ISO-8601 format.
- **writeEndArray()**
  Writes the ending marker of a JSON array (']').
- **writeEndObject()**
  Writes the ending marker of a JSON object ('}').
- **writeFieldName(fieldName)**
  Writes a field name.
- **writeId(identifier)**
  Writes the specified ID value.
- **writeIdField(fieldName, identifier)**
  Writes a field name and value pair using the specified field name and identifier value.
- **writeNull()**
  Writes the JSON null literal value.
- **writeNullField(fieldName)**
  Writes a field name and value pair using the specified field name and the JSON null literal value.
- **writeNumber(number)**
  Writes the specified decimal value.
- **writeNumber(number)**
  Writes the specified double value.
- **writeNumber(number)**
  Writes the specified integer value.
- **writeNumber(number)**
  Writes the specified long value.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

- **writeNumberField(fieldName, number)**
  Writes a field name and value pair using the specified field name and decimal value.
- **writeNumberField(fieldName, number)**
  Writes a field name and value pair using the specified field name and double value.
- **writeNumberField(fieldName, number)**
  Writes a field name and value pair using the specified field name and integer value.
- **writeNumberField(fieldName, number)**
  Writes a field name and value pair using the specified field name and long value.
- **writeObject(anyObject)**
  Writes the specified Apex object in JSON format.
- **writeObjectField(fieldName, value)**
  Writes a field name and value pair using the specified field name and Apex object.
- **writeStartArray()**
  Writes the starting marker of a JSON array ('[').
- **writeStartObject()**
  Writes the starting marker of a JSON object ('{').
- **writeString(stringValue)**
  Writes the specified string value.
- **writeStringField(fieldName, stringValue)**
  Writes a field name and value pair using the specified field name and string value.
- **writeTime(timeValue)**
  Writes the specified time value in the ISO-8601 format.
- **writeTimeField(fieldName, timeValue)**
  Writes a field name and value pair using the specified field name and time value in the ISO-8601 format.

JSONParser Class : Represents a parser for JSON-encoded content.

Usage : Use the `System.JSONParser` methods to parse a response that's returned from a call to an external service that is in JSON format, such as a JSON-encoded response of a Web service callout.

JSONParser Methods

The following are methods for `JSONParser`. All are instance methods.

- **clearCurrentToken()**
  Removes the current token.
- **getBlobValue()**
  Returns the current token as a BLOB value.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

- **getBooleanValue()**
  Returns the current token as a Boolean value.
- **getCurrentName()**
  Returns the name associated with the current token.
- **getCurrentToken()**
  Returns the token that the parser currently points to or `null` if there's no current token.
- **getDatetimeValue()**
  Returns the current token as a date and time value.
- **getDateValue()**
  Returns the current token as a date value.
- **getDecimalValue()**
  Returns the current token as a decimal value.
- **getDoubleValue()**
  Returns the current token as a double value.
- **getIdValue()**
  Returns the current token as an ID value.
- **getIntegerValue()**
  Returns the current token as an integer value.
- **getLastClearedToken()**
  Returns the last token that was cleared by the `clearCurrentToken` method.
- **getLongValue()**
  Returns the current token as a long value.
- **getText()**
  Returns the textual representation of the current token or `null` if there's no current token.
- **getTimeValue()**
  Returns the current token as a time value.
- **hasCurrentToken()**
  Returns `true` if the parser currently points to a token; otherwise, returns `false`.
- **nextToken()**
  Returns the next token or `null` if the parser has reached the end of the input stream.
- **nextValue()**
  Returns the next token that is a value type or `null` if the parser has reached the end of the input stream.
- **readValueAs(apexType)**
  Deserializes JSON content into an object of the specified Apex type and returns the deserialized object.
- **readValueAsStrict(apexType)**
  Deserializes JSON content into an object of the specified Apex type and returns the deserialized object. All attributes in the JSON content must be present in the specified type.
- **skipChildren()**
  Skips all child tokens of type `JSONToken.START_ARRAY` and `JSONToken.START_OBJECT` that the parser currently points to.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

JSONToken Enum : Contains all token values used for parsing JSON content.

Enum Value	Description
END_ARRAY	The ending of an array value. This token is returned when ']' is encountered.
END_OBJECT	The ending of an object value. This token is returned when '}' is encountered.
FIELD_NAME	A string token that is a field name.
NOT_AVAILABLE	The requested token isn't available.
START_ARRAY	The start of an array value. This token is returned when '[' is encountered.
START_OBJECT	The start of an object value. This token is returned when '{' is encountered.
VALUE_EMBEDDED_OBJECT	An embedded object that isn't accessible as a typical object structure that includes the start and end object tokens START_OBJECT and END_OBJECT but is represented as a raw object.
VALUE_FALSE	The literal “false” value.
VALUE_NULL	The literal “null” value.
VALUE_NUMBER_FLOAT	A float value.
VALUE_NUMBER_INT	An integer value.
VALUE_STRING	A string value.
VALUE_TRUE	A value that corresponds to the “true” string literal.

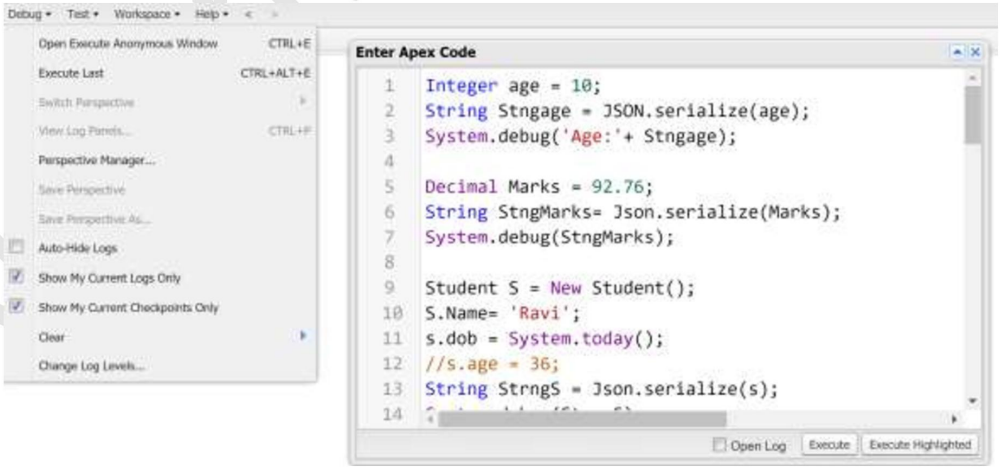
Serialization is a process of converting an object into a string.

Deserialization is a process of converting a String into an Object Class.

Practical/Lab Task 1

JSON Serialization

01. Login to Developer Force , Open Developer Console
02. Under Debug ‘Open Execute Anonymous Window’
03. Type the Programs given below to see the out put in Logs under Debug.



01. Serialize Integer to String :

Integer age = 10;

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

```
String Stngage = JSON.serialize(age);
System.debug('Age:'+ Stngage);
```

02. Serialize Decimal to String :

```
Decimal Marks = 92.76;
String StngMarks= Json.serialize(Marks);
System.debug(StngMarks);
```

03. Serialize Pretty Object Class to String :

```
Student S = New Student();
S.Name= 'Ravi';
s.dob = System.today();
//s.age = 36;
String StrngS = Json.serialize(s);
System.debug(StrngS);
String StrngS1 = Json.serializePretty(s,true);
System.debug(StrngS1);
```

04. Serialize Array to String :

```
Student[] St = New Student[] {s,s,s};
String StrngSt = Json.serialize(st);
System.debug(StrngSt);
```

05. Serialize sObject to String

```
Account[] Acc = [Select Name, Id , Fax from Account];
String StrngAcc = Json.serialize(acc);
System.debug(StrngAcc);
```

06. Serialize Pretty Integer to String

```
Integer age1 = 10;
String Stngage1 = JSON.serializePretty(age1);
System.debug('Age:'+ Stngage1);

JSON_Eg1 j =new JSON_Eg1();
j.create('Rupom','1234','Energy');
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

Practical/Lab Task 2

JSON Deserialization

01. Deserialize String to Integer Class :

```
String A = '10';
Integer Age = (Integer)Json.deserialize(A,Integer.class);
System.debug(Age);
```

02. Deserialize String to Decimal Class :

```
String B = '1000';
Decimal Salary = (Decimal)Json.deserialize(B,Decimal.Class);
System.debug(Salary);
```

03. Deserialize String to sObject Class :

```
String V = '{"Name":"Wipro","Phone":"222","Billingcity":"Hyd"}';
Account Ac = (Account)Json.deserialize(v,Account.class);
System.debug(Ac);   //Insert AC;
```

04. Deserialize String to sObject Class :

```
String V1 = '{"Name":"Wipro","Phone":"222","Billingcity":"Hyd","Email":"abc@abc.com"}';
Account Ac1 = (Account)Json.deserialize(V1,Account.class);
System.debug(Ac1);
```

05. Deserialize Strict String to sObject Class :

```
Account Ac2 = (Account)Json.deserializeStrict(V1,Account.class);
System.debug(Ac2);

String jString='{"Company":"Wipro","Lastname":"Rupom","Email":"ab@ab.com","Salary":25000}';
Map<String,object> Mp = (Map<String,Object>)json.deserializeUntyped(jstring);
System.debug(MP);
System.debug(MP.keyset());
System.debug(MP.values());

String Company = (String)MP.get('Company');
Decimal Salary= (Decimal)MP.get('Salary');
System.debug(Company);
System.debug(Salary);
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

Practical/Lab Task 3

JSON Serialization

Receiving a Data(Account Data) in String, Saving as Account and Giving Response (Serializing) back as JSON String

01. Create a Class

```
public class JSON_Eg1 {
Public Void Create(String Name,String Phone,String Industry)
{
    system.JSONGenerator js = Json.createGenerator(true);
    js.writeStartObject();
    Try
    {
      Account A = New Account();
       a.Name = Name;
       a.phone= phone;
       a.Industry = Industry;
       Insert A;
       js.writeStringField('Status', 'Success');
       js.writeIdField('Id', A.id);
       js.writeNumberField('StatusCode', 100);
    }
    catch(Exception e)
    {
      js.writeStringField('Status', 'Failed');
      js.writeStringField('Status',e.getMessage());
      js.writeNumberField('StatusCode', 200);
    }
    js.writeEndObject();
    String result = js.getAsString();
    System.debug(result);
}
}
```

02. Execute Under in Developer Console 'Open Execute Anonymous Window'

```
JSON_Eg1 j =new JSON_Eg1();
j.create('Rupom','1234','Energy');
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

Practical/Lab Task 4

JSON Deserialization – Token Concept

Receiving a Data in JSON String and Deserializing in Visual Force Page to see the desterilize output

01.Create a Class

```
public class JSON_Example2 {
    Public String jsonString {set;get;}
    Public Map<String,Object> ResultMap {set;get;}

    Public JSON_Example2()
    {
       ResultMap = new Map<String,Object>();
       jsonString = '{"Name":"Rupom","Marks":[10,20,30], "City":"Hyd"}';
       System.JSONParser jp = JSON.createParser(jsonString);

        While(jp.nextToken()!=null)
        {
          ResultMap.put(jp.getText(), jp.getCurrentToken()) ;
        }
    }
}
```

02. Write a Visual Force Page to see the output

```
<apex:page controller="JSON_Example2" >
    <apex:form>
        <apex:dataTable value="{!ResultMap}" var="a" cellpadding="10" rules="rows" frame="border">
            <apex:column value="{!a}" headerValue="Key"/>
            <apex:column value="{!ResultMap[a]}" headerValue="Token"/>
        </apex:dataTable>
    </apex:form>
</apex:page>
```

03. Output : ---------->

Key	Token
10	VALUE_NUMBER_INT
20	VALUE_NUMBER_INT
30	VALUE_NUMBER_INT
Hyderabad	VALUE_STRING
Marks	FIELD_NAME
Name	FIELD_NAME
Rupom	VALUE_STRING
[	START_ARRAY
]	END_ARRAY
city	FIELD_NAME
{	START_OBJECT
}	END_OBJECT

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

Practical/Lab Task 5

JSON Deserialization – Token Concept

Receiving a Data in JSON String and Deserializing to consume selected data (Firstname, Account, Salary)

01.Create a Class

```
public class Json_Example4
{
    Public String JsonString {set;get;}
    Public Json_Example4()
    {
jsonString = '{"LastName":"Chakraborty","FirstName":"Rupom", "Account":{"Name":"Dell","Phone":"04022"},"Salary":2000}';
    //01. Create Parser Object
    System.JSONParser jp = JSON.createParser(jsonString)  ;
    While(jp.nextToken()!= null) //02. Create Loop
    {
    //03. Get the field anme and check for first name
      if(jp.getText()=='FirstName')
      {
        jp.nextToken(); // 04. if current field is name then move to next position
        string value = jp.getText(); //05. Read the string Value
        System.debug(value);
      }
      If(jp.getText()=='Account') //06. Check for field name Account
      {
        jp.nextToken(); // 07. if current field is name then move to next position
        Account acc = (Account)jp.readValueAs(Account.class); //08. Read value as Account
        System.debug(acc);
      }
        If(jp.getText()=='salary') //08.Checl for Salary
        {
          jp.nextToken(); // 07. if current field is name then move to next position
          Decimal Salary = (Decimal)jp.getDecimalValue();
          System.debug(Salary);
        }
    }
    }
}
```

02.Execute Under in Developer Console 'Open Execute Anonymous Window'

```
JSON_Example4 je = New JSON_Example4();
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

Practical/Lab Task 6

JSON Serialization and Deserialization

Serializing and Deserializing a attachment File (Blob)

01. Write a Class

```
public class testjsonserdeser {
 public static void Json_Serialization_Deserialisation (){
        Document D=[select id,name,body from document where name='test'];
    String  J = Json.serialize(D); // Standard Serilzation
           System.JsonGenerator JG=Json.createGenerator(true);
             jg.writeStartObject();
                 jg.writeBlobField('Attachment',D.body);
             jg.writeEndObject();
     String jg1 = jg.getAsString(); // You have not created the final String Output

        Blob Bs = (Blob)Json.deserialize(Jg1,Blob.class); //Standard DeSerilization

        System.JsonParser jp=json.createparser(jg1);  // Parse the output string tha
          while(jp.nextToken()!= null) //You have used jg Its jp ---the Json parser
            {
        if(jp.getText() == 'Attachment') // Where is Test ????  in serilization you
            {
               jp.nextToken();
               Blob B = (Blob)jp.getBlobValue();
               //System.debug('body of the attachment is'+jp.readValueAs());  -----
               System.debug(B);
               }
    }
    }
}
```

02. Execute Under in Developer Console 'Open Execute Anonymous Window'

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

Data Exchange Format – XML

- Xml (eXtensible Markup Language) is a mark up language.
- XML is designed to store and transport data.
- XML is designed to be self-descriptive.
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.
- XML uses DTD(Document Type Definition) or Schema to describe data
- XML allows users to define own tags and own document Structure

XML Data Format Example :

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Rupom</to>
  <from>OnlineTechnologies</from>
  <heading>Integration</heading>
  <body>Don't forget weekend integration classbody>
</note>
```

XML document with an External DTD

```
<?xml version="1.0"?>
<!DOCTYPE employee SYSTEM "employee.dtd">
<employee>
<firstname>Rupom</firstname>
<lastname>Chakraborty</lastname>
<email>vimal@javatpoint.com</email>
</employee>
```

DTD

- A DTD is a Document Type Definition.
- A DTD defines the structure and the legal elements and attributes of an XML document.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

XML document with an internal DTD

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

XSD Schema

An XML Schema describes the structure of an XML document. The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Complex element definitions:

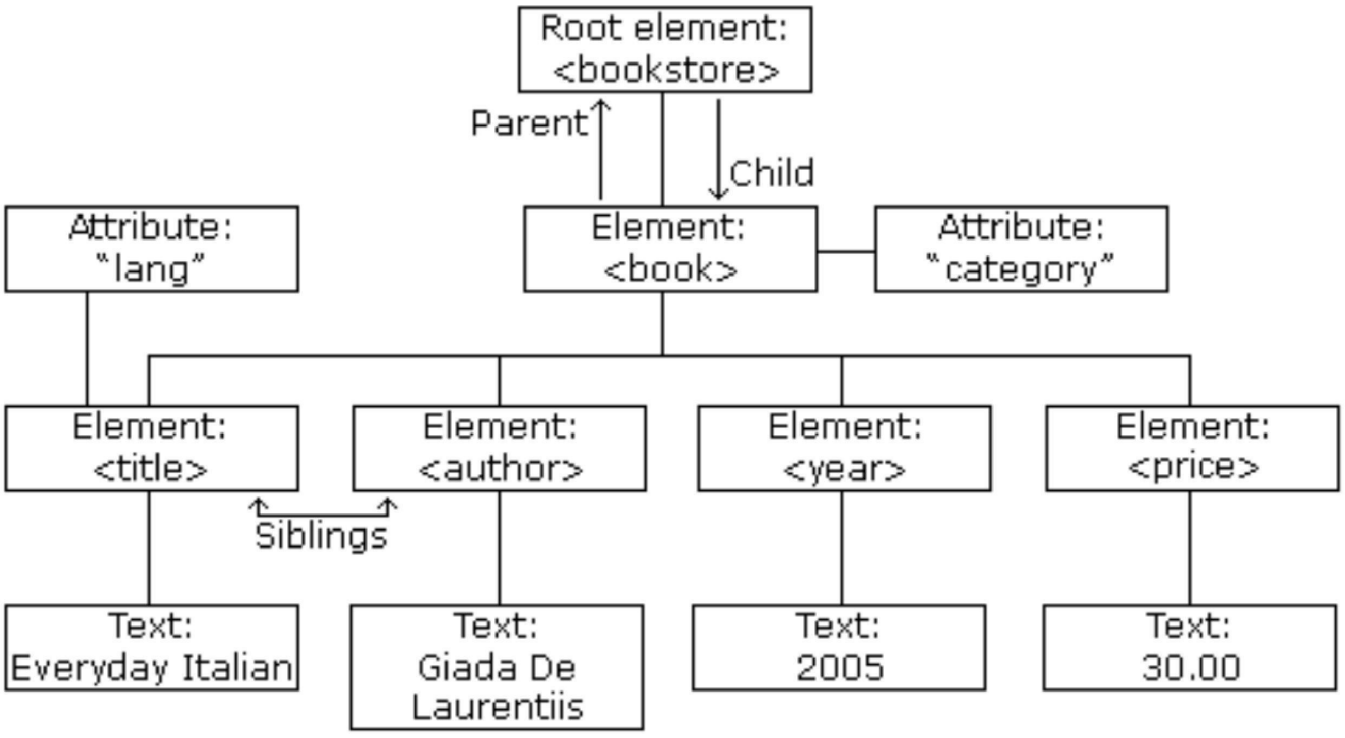
Eg 1:

```
<employee>
  <firstname>Rupom</firstname>
  <lastname>Chakraborty</lastname>
</employee>
```

Eg2:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Tree Structure



RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

XML in Salesforce

Dom Namespace : The Dom namespace provides classes and methods for parsing and creating XML content.

- **Document Class**
  Use the Document class to process XML content. You can parse nested XML content that's up to 50 nodes deep.
- **XmlNode Class**
  Use the XmlNode class to work with a node in an XML document.

Syntax:

```
DOM.Document xmldoc = new DOM.Document();
```

Document Class

Use the Document class to process XML content. You can parse nested XML content that's up to 50 nodes deep.

Usage

One common application is to use it to create the body of a request for HttpRequest or to parse a response accessed by HttpResponse.

Document Methods

The following are methods for Document. All are instance methods.

- **createRootElement(name, namespace, prefix)**
  Creates the top-level root element for a document.
- **getRootElement()**
  Returns the top-level root element node in the document. If this method returns null, the root element has not been created yet.
- **load(xml)**
  Parse the XML representation of the document specified in the *xml* argument and load it into a document.
- **toXmlString()**
  Returns the XML representation of the document as a String.

XmlNode Class

Use the XmlNode class to work with a node in an XML document.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

XmlNode Methods

The following are methods for XmlNode. All are instance methods.

- **addChildElement(name, namespace, prefix)**
  Creates a child element node for this node.
- **addCommentNode(text)**
  Creates a child comment node for this node.
- **addTextNode(text)**
  Creates a child text node for this node.
- **getAttribute(key, keyNamespace)**
  Returns *namespacePrefix:attributeValue* for the given key and key namespace.
- **getAttributeCount()**
  Returns the number of attributes for this node.
- **getAttributeKeyAt(index)**
  Returns the attribute key for the given index. Index values start at 0.
- **getAttributeKeyNsAt(index)**
  Returns the attribute key namespace for the given index.
- **getAttributeValue(key, keyNamespace)**
  Returns the attribute value for the given key and key namespace.
- **getAttributeValueNs(key, keyNamespace)**
  Returns the attribute value namespace for the given key and key namespace.
- **getChildElement(name, namespace)**
  Returns the child element node for the node with the given name and namespace.
- **getChildElements()**
  Returns the child element nodes for this node. This doesn't include child text or comment nodes.
- **getChildren()**
  Returns the child nodes for this node. This includes all node types.
- **getName()**
  Returns the element name.
- **getNamespace()**
  Returns the namespace of the element.
- **getNamespaceFor(prefix)**
  Returns the namespace of the element for the given prefix.
- **getNodeType()**
  Returns the node type.
- **getParent()**
  Returns the parent of this element.
- **getPrefixFor(namespace)**
  Returns the prefix of the given namespace.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

- **getText()**
  Returns the text for this node.
- **insertBefore(newChild, refChild)**
  Inserts a new child node before the specified node.
- **removeAttribute(key, keyNamespace)**
  Removes the attribute with the given key and key namespace. Returns true if successful, false otherwise.
- **removeChild(childNode)**
  Removes the given child node.
- **setAttribute(key, value)**
  Sets the key attribute value.
- **setAttributeNs(key, value, keyNamespace, valueNamespace)**
  Sets the key attribute value.
- **setNamespace(prefix, namespace)**
  Sets the namespace for the given prefix.

Practical/Lab Task 7

XML Serialization – Creating DOM file

01. Write a Class :

```
public class XML_Example1
{
   Public String xmlString {set;get;}

   Public XML_Example1()
   {
    DOM.Document doc = New DOM.Document();
     //Dom.XmlNode root  = doc.createRootElement('Employee',null,null);
     Dom.XmlNode root  = doc.createRootElement('Employee','https://www.flipkart.com/','Rupom');
     Dom.XmlNode name = root.addChildElement('Name',null,null);
     Dom.XmlNode LName = root.addChildElement('LastName',null,null);
      LName.addTextNode('Chakraborty');
     Dom.XmlNode FName = root.addChildElement('FirstName',null,null);
       FName.addTextNode('Rupom');
    xmlString = doc.toXmlString();   //JS.getAsString
   }
}
```

02. Write Visual Force Page

```
<apex:page controller="XML_Example1">
   {!xmlString}
</apex:page>
```

03. Output XML File :

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

```
<?xml version="1.0" encoding="UTF-8"?>
    <Rupom:Employee xmlns:Rupom="https://www.flipkart.com/">
        <Name />
        <LastName>Chakraborty</LastName>
        <FirstName>Rupom</FirstName>
    </Rupom:Employee>
```

Practical/Lab Task 8

XML Serialization – Creating DOM file dynamically from Account (4-Records)

Out Put:

```
<?xml version="1.0" encoding="UTF-8"?>
<Acc>
        <Account Name="Ravi Tech1" Phone="2222" />
        <Account Name="Oracle India Pvt Ltd" Phone="986600" />
        <Account Name="Rupom Tech" Phone="1234" />
        <Account Name="Ravi Tech" Phone="2222" />
</Acc>
```

01. Write a Class :

```
public class XML_Example2 {

   Public string xmlstring {set;get;}

   Public XML_Example2 ()
   {
      List<Account> Acc = [select id,name,phone,industry,sic,billingcity from Account Limit 4];
      DOM.Document doc  = new DOM.Document();
      DOM.XmlNode root = doc.createRootElement('Acc',null,null);

      For(Account A: Acc)
      {
         DOM.XmlNode ACNT = root.addChildElement('Account',null,null);
         ACNT.setAttribute('Name', A.Name);
         ACNT.setAttribute('Phone', A.phone);
         //ACNT.setAttribute('Industry', A.industry);
      }
      xmlstring = doc.toXmlString() ;
   }
}
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

02. Write VF Page to Get Output

```
<apex:page controller="XML_Example2">
  {!xmlstring}
</apex:page>
```

Practical/Lab Task 9

XML Serialization – Creating DOM file dynamically from Account (10-Records) and Corresponding Contacts

Output :

```
<?xml version="1.0" encoding="UTF-8"?>
<Accounts>
        <Account Name="Ravi Tech1" Phone="2222" />
        <Account Name="Microsoft" Phone="789000" />
        <Account Name="Wipro" Phone="789000" />
        <Account Name="Oracle India Pvt Ltd" Phone="986600" />
        <Account Name="Microsoft" Phone="789000" />
        <Account Name="Rupom Tech" Phone="1234"> <Contacts Count=2 /> </Account>
                <Contact FirstName="Rahul" LastName="Gandhi" />
                <Contact FirstName="Sonia" LastName="Gandhi" />
        <Account Name="Ravi Tech" Phone="2222"><Contacts Count=1  /></Account>
                <Contact FirstName="Nitin" LastName="Gadkari" />
        <Account Name="ICICI Bank" Phone="040000"> <Contacts  Count=1 /></Account>
                <Contact FirstName="Narendra" LastName="Modi" />
        <Account Name="HDFC" Phone="2323"><Contacts Count=1 /></Account>
                <Contact FirstName="Raj Nath" LastName="Singh" />
        <Account Name="Dell" Phone="56456" />
</Accounts>
```

01. Write a Class

```
public class XMLExample3_AccCon
{
   Public String xmlString  {set;get;}
   Public XMLExample3_AccCon()
   {
    List<Account> ACCM = [select name,phone,(Select Firstname,LastName from contacts) from Account limit 10];

    Dom.Document doc = new Dom.Document();
    Dom.XmlNode root = doc.createRootElement('Accounts',null,null);

    for(Account A : ACCM)
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

```
{
   Dom.XmlNode acc = root.addChildElement('Account',null,null);
   acc.setAttribute('Name', A.Name);
   acc.setAttribute('Phone', A.phone);
   if(A.contacts.size()>0)
   {
    Dom.XmlNode contacts = acc.addChildElement('Contacts',null,null);
    for(Contact C:A.contacts )
    {
      Dom.XmlNode con = root.addChildElement('Contact',null,null);
     con.setAttribute('FirstName', c.firstname);
     con.setAttribute('LastName', c.lastname);
    }
   }
  }
  xmlString = doc.toXmlString();
 }
}
```

02. Write a VF page to get Output

```
<apex:page controller="XMLExample3_AccCon">
  {!xmlString}
</apex:page>
```

Practical/Lab Task 10

XML Deserialization – Create Parse Data from received XML string File

Output

01. Write a Class

```
public class XML_Example4_DS
{
   Public string xmlString {set;get;}
   Public string parsedData  {set;get;}

   Public Void CallMe()
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

```
    {
        Dom.Document doc = new Dom.Document();
        doc.load(xmlString);
        Dom.XmlNode root = doc.getRootElement();
        parsedData = root.getName();
        List<Dom.XmlNode> Childs = root.getChildElements();
        for(Dom.XmlNode c :Childs )
        {
          parsedData =  parsedData + '\n'+ c.getName();
          parsedData =  parsedData + ': '+c.getText();
        }

    }
}
```

02. Write a VF page

```
<apex:page controller="XML_Example4_DS" sidebar="false">
  <apex:form >
      <apex:pageBlock title="XML Parsing Example">
        <apex:panelGrid columns="3" width="600">
          <apex:inputTextarea rows="8" cols="80" value="{!xmlString}" />

          <apex:panelGrid columns="1">
            <apex:commandButton value="Click to Convert" action="{!CallMe}" reRender="one"/>
          </apex:panelGrid>

          <apex:inputTextarea rows="8" cols="80" value="{!parsedData}" id="one" />
        </apex:panelGrid>
      </apex:pageBlock>
  </apex:form>
</apex:page>
```

Practical/Lab Task 11

XML Deserialization – Create Parse Data from received XML File(Blob) and Create Account

01. Open Notepad , Type the below XML data and Save the file as XML

```
<Accounts>
    <Account Name="Yahoo" Industry="Energy" sic="abc"/>
    <Account Name="Google" Industry="Finance"/>
    <Account Name="Bajaj" Industry="Construction"/>
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

```
        <Account Name="Honda" Industry="Manufacturing"/>
</Accounts>
```

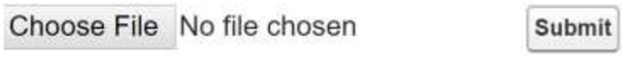
02. Write a Class

```
public class XML_Eg5_ReadXMLfromFile
{
   Public Blob body {set;get;}

   Public Void CallMe()
   {
     List<Account> accounts = new List<Account>();
     String xlmString = body.toString();
     body = Null;

     DOM.Document Doc = new DOM.Document();
     Doc.load(xlmString) ;
     DOM.XmlNode root = doc.getRootElement();
     List<DOM.XmlNode> Childs = root.getChildElements();
     for(DOM.XmlNode C :Childs )
     {
       Account a = New Account();
       a.Name = c.getAttributeValue('Name',Null);
       a.Industry = c.getAttributeValue('Industry',Null);
       accounts.add(a);
     }
     Insert accounts;
   }

}
```

03. Write VF

```
<apex:page controller="XML_Eg5_ReadXMLfromFile">
 <apex:form >
   <apex:inputFile value="{!body}" fileName="Example" />
   <apex:commandButton value="Submit" action="{!CallMe}"/>
  </apex:form>
</apex:page>
```

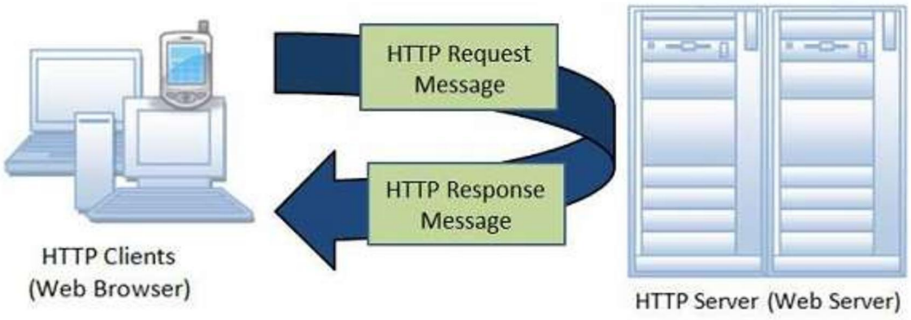
04. Preview the VF and Choose file and Click submit



05. Check Accounts to the record creation

Protocol – HTTP

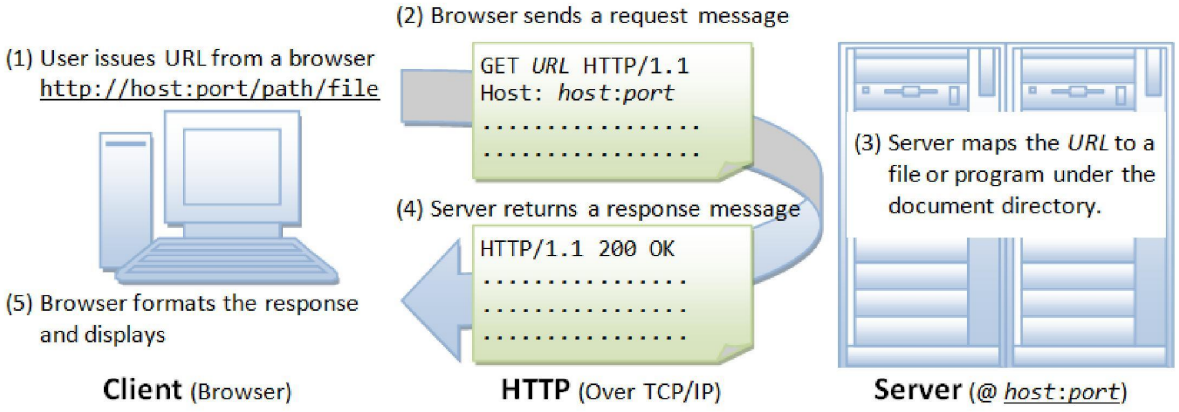
The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (i.e. internet) since 1990. HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.



HTTP is connectionless: The HTTP client, i.e., a browser initiates an HTTP request and after a request is made, the client waits for the response. The server processes the request and sends a response back after which client disconnect the connection. So client and server knows about each other during current request and response only. Further requests are made on new connection like client and server are new to each other.

HTTP is stateless: As mentioned above, HTTP is connectionless and it is a direct result of HTTP being a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages.

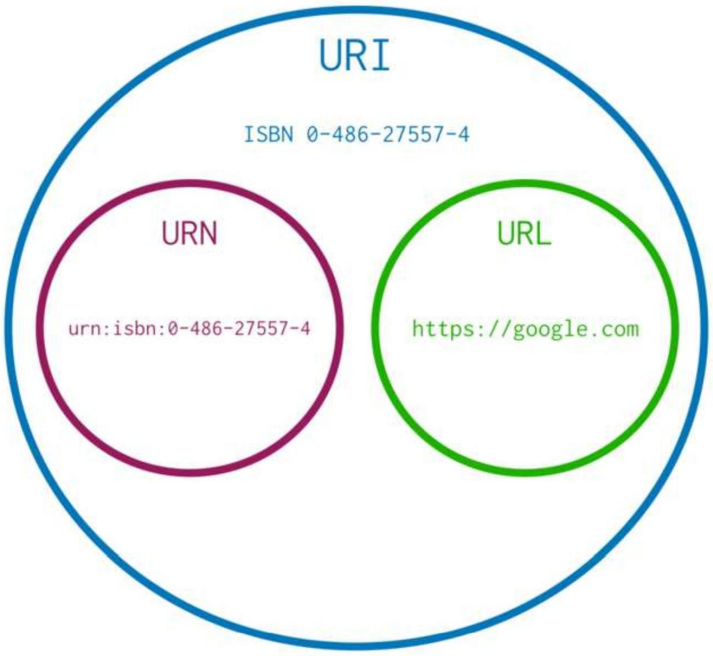
HTTP is media independent: It means, any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type using appropriate MIME-type.



RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

URLs, URIs, and URNs

- URL is a Uniform Resource Locator, tells you the how and where of something
  - [Scheme]://[Domain]:[Port]/[Path]?[QueryString]#[FragmentId]
  - http://www.wrox.com/remtitle.cgi?isgn=0470114878
- URN is a Uniform Resource Name, is simply a unique name
  - urn:[namespace identifier]:[namespace specific string]
  - urn:isbn:9780470114872
- URI is a Uniform Resource Identifier, is URL or URN



RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

HTTP Protocol Format

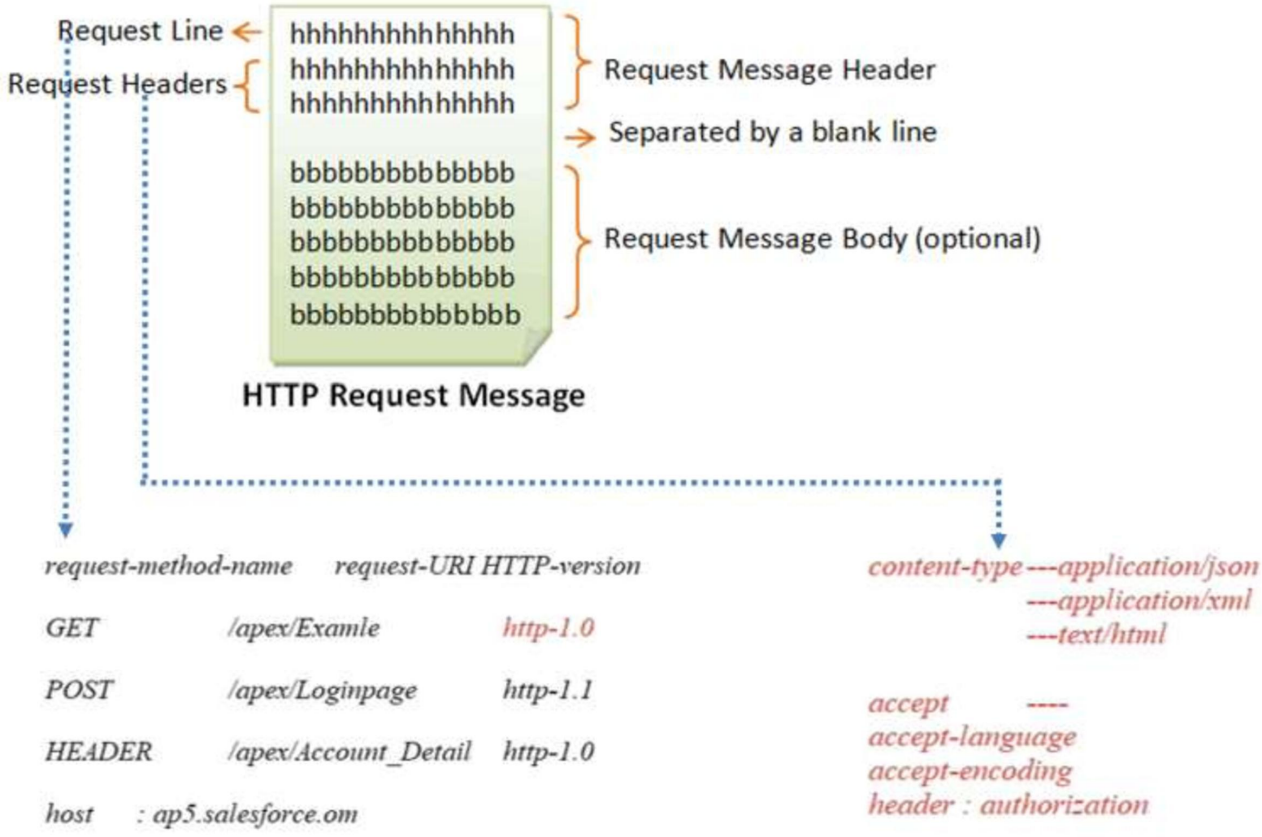
```
http: 80
https: 443
ftd  : 21
smtp :25
```

```
          userinfo          host        port
        ┌────────┴────────┐ ┌──┴──────┐ ┌┴┐
http://john.doe:password@www.example.com:123/forum/questions/?tag=networking&order=newest#top
└──┘   └──────────────────────────────────┘└──────────────┘└──────────────────────────┘└──┘
scheme          authority                      path                query              fragment
```

https://online86-dev-ed.lightning.force.com/lightning/setup/SetupOneHome/home

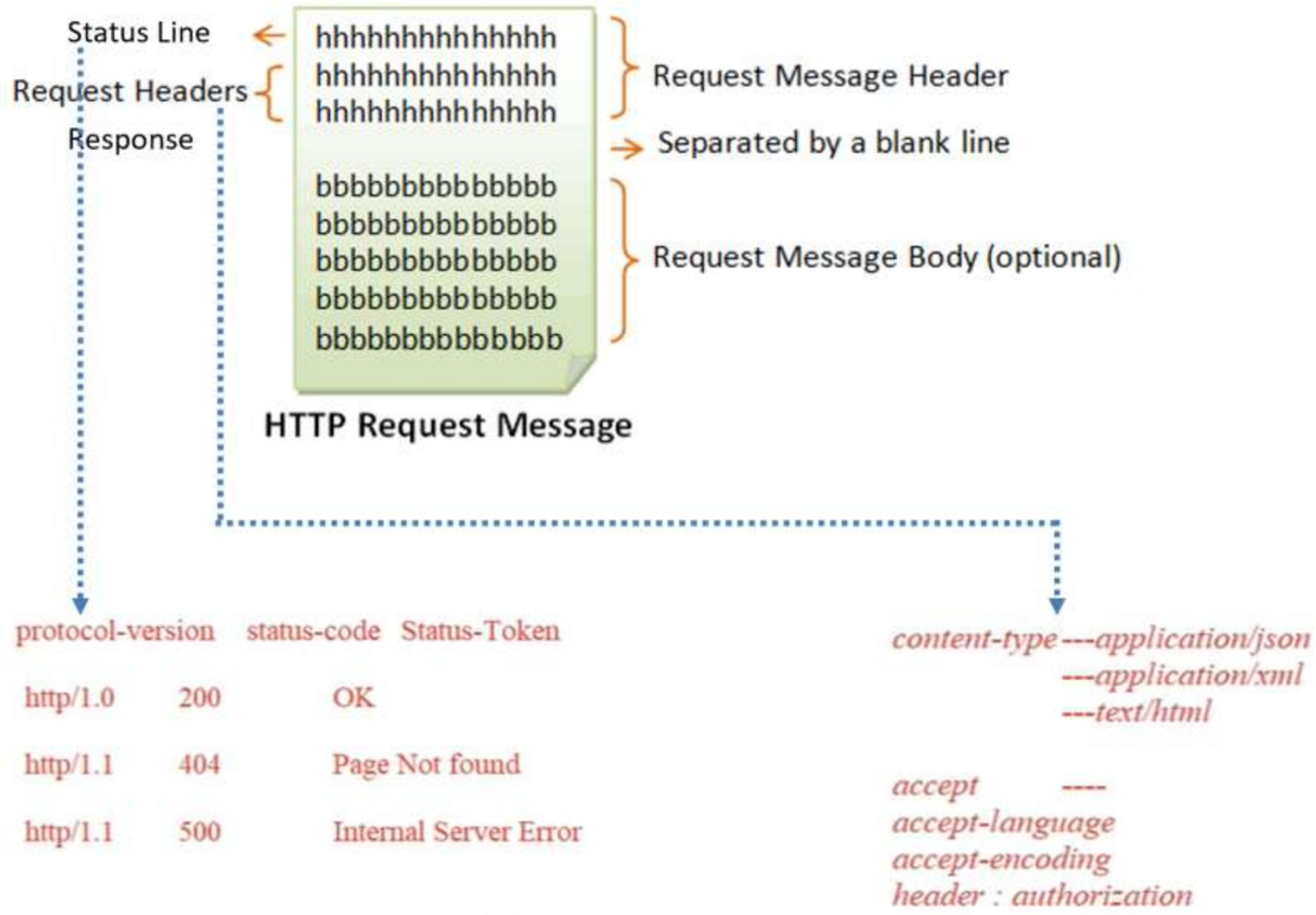
Protocol | Host Name | Path

HTTP Message Structure (Request)



RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

HTTP Message Structure (Response)



Http Class

Use the Http class to initiate an HTTP request and response.

Http Methods

The following are methods for Http. All are instance methods.

- **send(request)**
  Sends an HttpRequest and returns the response.
-
- **toString()**
  Returns a string that displays and identifies the object's properties.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

HttpRequest Class

Use the HttpRequest class to programmatically create HTTP requests like GET, POST, PUT, and DELETE.

Usage

Use the XML classes or JSON classes to parse XML or JSON content in the body of a request created by HttpRequest.

HttpRequest Methods

The following are methods for HttpRequest. All are instance methods.

- **getBody()**
  Retrieves the body of this request.
- **getBodyAsBlob()**
  Retrieves the body of this request as a Blob.
- **getBodyDocument()**
  Retrieves the body of this request as a DOM document.
- **getCompressed()**
  If true, the request body is compressed, false otherwise.
- **getEndpoint()**
  Retrieves the URL for the endpoint of the external server for this request.
- **getHeader(key)**
  Retrieves the contents of the request header.
- **getMethod()**
  Returns the type of method used by HttpRequest.
- **setBody(body)**
  Sets the contents of the body for this request.
- **setBodyAsBlob(body)**
  Sets the contents of the body for this request using a Blob.
- **setBodyDocument(document)**
  Sets the contents of the body for this request. The contents represent a DOM document.
- **setClientCertificate(clientCert, password)**
  This method is deprecated. Use setClientCertificateName instead.
- **setClientCertificateName(certDevName)**
  If the external service requires a client certificate for authentication, set the certificate name.
- **setCompressed(flag)**
  If true, the data in the body is delivered to the endpoint in the gzip compressed format. If false, no compression format is used.
- **setEndpoint(endpoint)**
  Specifies the endpoint for this request.

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

- **setHeader(key, value)**
  Sets the contents of the request header.
- **setMethod(method)**
  Sets the type of method to be used for the HTTP request.
- **setTimeout(timeout)**
  Sets the timeout in milliseconds for the request.
- **toString()**
  Returns a string containing the URL for the endpoint of the external server for this request and the method used, for example, Endpoint=http://YourServer, Method=POST

HttpCalloutMock Interface

Enables sending fake responses when testing HTTP callouts.

HttpCalloutMock Methods

The following are methods for HttpCalloutMock.

- **respond(request)**
  Returns an HTTP response for the given request. The implementation of this method is called by the Apex runtime to send a fake response when an HTTP callout is made after Test.setMock has been called.

- Http
- HttpRequest
- HttpResponse

Requesting :-

Where/from Whom ???????
url (endpoint url)

Type of Request ?????
GET
POST
PUT
DELETE

01.

```
HttpRequest  request = new HttpRequest()
request.setEndPoint ('www.flipkart.com') ;
request.setMethod('GET');
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

02.
url: https://www.icicibank.com/NewCustomer
POST

```
HttpRequest rq = new HttpRequest();
rq.setEndPoint('https://www.icicibank.com/NewCustomer');
rq.setMethod(POST);
```

RESPONSE

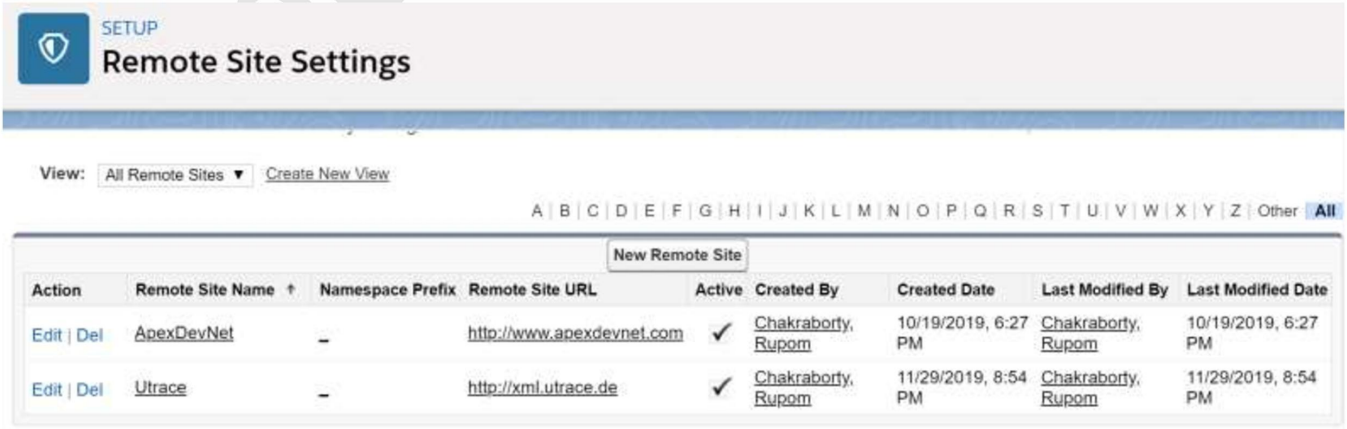
```
HttpResponse response = new Httpresponse();
String Status = response.getStatusCode();
String body = response.getBody();
```

```
HttpRequest rq = new HttpRequest();
rq.setEndPoint('https://www.icicibank.com/NewCustomer');
rq.setMethod(POST);

Http p = new Http();
HttpResponse response =p.send(rq)
String Status = response.getStatusCode();
String body = response.getBody();
```

Practical/Lab Task 12

Request IP address to Utrace URL and get Response in XML to consume

01. Add the UTRACE URL to Remote Site Settings



RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

02. Create a Class

```
public class Http_URL_URTACE
{
   Public String ipAddress          {set;get;}
   Public String Body               {set;get;}
   Public string status             {set;get;}
   Public String data               {set;get;}
   Public Map<String,Object> resMap    {set;get;}

   Public Void CallService()
   {
      resMap = new Map<String,String>();

      Http p = new Http();
      String endpoint = 'http://xml.utrace.de/?query='+ ipAddress;

      HttpRequest request = New HttpRequest();
      request.setEndpoint(endpoint);
      request.setMethod('GET');

      HttpResponse response = p.send(request);
      Body = response.getBody();
      status = 'Code: ' + response.getStatusCode();
      data = response.getHeader('Content-Type');

                  Object Size = response.getHeader('Content-Length');
                  resMap.put('Status',status);
                  resMap.put('Length',Size);
                  resMap.put('Content-Type',data);


      Dom.Document doc = new dom.Document();
      doc.load(Body);
      DOM.XmlNode root = doc.getRootElement(); //Results
      List<DOM.XmlNode> result = root.getChildElements(); //result
      DOM.XmlNode parent = result[0];
      List<DOM.XmlNode> childs = parent.getChildElements(); //all
      for(DOM.XmlNode c:childs)
      {
         resMap.put(c.getName(), c.getText()) ;
      }
   }
}
```

03. Create VF to Read the Response

```
<apex:page controller="Http_URL_URTACE" >
  <Style>
    .box{
      Width:600px;
      height:300px;
```


RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

```
        margin-left:50px;
        padding:20px;
      }
  </Style>


  <apex:form >
      <apex:outputPanel id="op" styleClass="box">
        <apex:inputText value="{!ipAddress}">
        <apex:commandButton value="TraceMe" action="{!CallService}" reRender="op"/>
        </apex:inputText>
        <br/><br/><br/>
        <apex:dataTable value="{!resMap}" var="a" cellpadding="10" rules="rows" width="800"
frame="border">
          <apex:column value="{!a}" headerValue="Property"/>
          <apex:column value="{!resMap[a]}" headerValue="Value"/>
        </apex:dataTable>

      </apex:outputPanel>
  </apex:form>
</apex:page>
```

Practical/Lab Task 12

Request Postal Code address to postalpincode URL and get Response in JSON to consume

01. Add the PostalPinCode (https://api.postalpincode.in/pincode) URL to Remote Site Settings

02. Create the class

```
public class HTTP_PostalCode {
  Public List<String> Places {set;get;}
  Public String Pincode          {set;get;}

  Public Void RequestData()
  {
    Places = New List<String> ();
    //Endpoint : https://api.postalpincode.in/pincode/500090
    //Method: Get
    //Response : JSON
    String URL = 'https://api.postalpincode.in/pincode/' +Pincode;
    HttpRequest request = new HttpRequest();
```

```
        request.setEndpoint(URL);
        request.setMethod('GET');
        Http p = new Http();
        HttpResponse response = p.send(request);
        String Body = response.getBody();

        System.JSONParser jp = Json.createParser(Body);
        While(jp.nextToken()!= null)
        {
          if(jp.getText()=='Name')
          {
             jp.nextToken();
             Places.add(jp.getText());
          }
        }
      }
    }
```

03. Create VF page to request and get response

```
<apex:page controller="HTTP_PostalCode" >
  <apex:form id="fm">
       Pincode :  <apex:inputText value="{!Pincode}"/>
    <br/> <br/>
    <apex:commandButton value="Go" action="{!RequestData}" reRender="fm"/>
      <br/> <br/>
      <br/> <br/>
    <apex:dataList value="{!Places}" var="a">
      {!a}
    </apex:dataList>
  </apex:form>
</apex:page>
```

04. Output :

RUPOM CHAKRABORTY - Capital Info Solutions
Shyamala Plaza, Behind Mythrivanam, Amerpet, Hyderabad, Telangana State, INDIA. Ph: 8686864286

LAB TASK HOME WORKS

01. Write an API (JSON) to display Title against the key words search from PLOS Repository.

Intro : PLOS is a central repository for collection of all biological research works done by various researchers.

Endpoint : http://api.plos.org/search?q=title:Leg

Method : GET

Response : JSON

Key Word : RNA, DNA, EYE, LEG Etc... (Body Parts) . keyword is the last word after Title: on the end point link.

API Details : please refer to http://api.plos.org/solr/examples/ get more info on JSON response and call format.

02. Write an API (XML) to display Country Population against the year search from World Bank finance repository

01. Intro : World Bank is a the leader of finance banking storing various economic data about countries in their repository.
02. Endpoint : http://api.worldbank.org/v2/country/all/indicator/SP.POP.TOTL?date=2015
03. Method : GET
04. Response : XML
05. Key Word : 2019, 2018, 2017, 2016, 2015 (Years) at the Date = {YYYY}
06. API Details : please refer to http://api.plos.org/solr/examples/ get more info on JSON response and call format.