**HTTP Status Codes**

This page is created from HTTP status code information found at [ietf.org](https://www.ietf.org/assignments/http-status-codes/http-status-codes.xml) and [Wikipedia](https://en.wikipedia.org/wiki/HTTP_status_code). Click on the **category heading** or the **status code** link to read more.

**1xx Informational**

100 Continue

[101 Switching Protocols](https://www.restapitutorial.com/httpstatuscodes.html)

[102 Processing (WebDAV)](https://www.restapitutorial.com/httpstatuscodes.html)

**2xx Success**

 200 OK

 201 Created

202 Accepted

203 Non-Authoritative Information

 204 No Content

205 Reset Content

206 Partial Content

207 Multi-Status (WebDAV)

208 Already Reported (WebDAV)

226 IM Used

**3xx Redirection**

300 Multiple Choices

301 Moved Permanently

302 Found

303 See Other

 304 Not Modified

305 Use Proxy

306 (Unused)

307 Temporary Redirect

308 Permanent Redirect (experimental)

[**4xx Client Error**](https://www.restapitutorial.com/httpstatuscodes.html)

 400 Bad Request

 401 Unauthorized

402 Payment Required

 403 Forbidden

 404 Not Found

405 Method Not Allowed

406 Not Acceptable

407 Proxy Authentication Required

408 Request Timeout

 409 Conflict

410 Gone

411 Length Required

412 Precondition Failed

413 Request Entity Too Large

414 Request-URI Too Long

415 Unsupported Media Type

416 Requested Range Not Satisfiable

417 Expectation Failed

418 I'm a teapot (RFC 2324)

420 Enhance Your Calm (Twitter)

422 Unprocessable Entity (WebDAV)

423 Locked (WebDAV)

424 Failed Dependency (WebDAV)

425 Reserved for WebDAV

426 Upgrade Required

428 Precondition Required

429 Too Many Requests

431 Request Header Fields Too Large

444 No Response (Nginx)

449 Retry With (Microsoft)

450 Blocked by Windows Parental Controls (Microsoft)

451 Unavailable For Legal Reasons

499 Client Closed Request (Nginx)

[**5xx Server Error**](https://www.restapitutorial.com/httpstatuscodes.html)

 500 Internal Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

505 HTTP Version Not Supported

506 Variant Also Negotiates (Experimental)

507 Insufficient Storage (WebDAV)

508 Loop Detected (WebDAV)

509 Bandwidth Limit Exceeded (Apache)

510 Not Extended

511 Network Authentication Required

598 Network read timeout error

599 Network connect timeout error

POJO class:

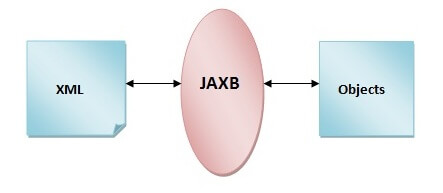
POJO stands for Plain Old Java Object. It is an ordinary Java object, not bound by any special restriction other than those forced by the Java Language Specification and not requiring any classpath. POJOs are used for increasing the readability and re-usability of a program. POJOs have gained the most acceptance because they are easy to write and understand.

Example POJO Class:

|  |
| --- |
| // Employee POJO class to represent entity Employee  public class Employee  {  // default field  String name;    // public field  public String id;    // private salary  private double salary;    //arg-constructor to initialize fields  public Employee(String name, String id,  double salary)  {  this.name = name;  this.id = id;  this.salary = salary;  }    // getter method for name  public String getName()  {  return name;  }    // getter method for id  public String getId()  {  return id;  }    // getter method for salary  public Double getSalary()  {  return salary;  }  } |

# JAXB

**JAXB** stands for Java Architecture for XML Binding. It provides mechanism to marshal (write) java objects into XML and unmarshal (read) XML into object. Simply, you can say it is used to convert java object into xml and vice-versa.



Maven dependency for JAXB

|  |
| --- |
| <dependency>  <groupId>javax.xml.bind</groupId>  <artifactId>jaxb-api</artifactId>  <version>2.2.12</version>  </dependency> |

Simple JAXB Marshalling Example: Converting Object into XML

Let's see the steps to convert java object into XML document.

* Create POJO or bind the schema and generate the classes
* Create the JAXBContext object
* Create the Marshaller objects
* Create the content tree by using set methods
* Call the marshal method

|  |
| --- |
| import javax.xml.bind.annotation.XmlAttribute;  import javax.xml.bind.annotation.XmlElement;  import javax.xml.bind.annotation.XmlRootElement;    @XmlRootElement  public class Employee {  private int id;  private String name;  private float salary;    public Employee() {}  public Employee(int id, String name, float salary) {  super();  this.id = id;  this.name = name;  this.salary = salary;  }  @XmlAttribute  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  @XmlElement  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  @XmlElement  public float getSalary() {  return salary;  }  public void setSalary(float salary) {  this.salary = salary;  }  } |

**@XmlRootElement** specifies the root element for the xml document.

**@XmlAttribute** specifies the attribute for the root element.

**@XmlElement** specifies the sub element for the root element.

Object to XML conversion

|  |
| --- |
| import java.io.FileOutputStream;    import javax.xml.bind.JAXBContext;  import javax.xml.bind.Marshaller;      public class ObjectToXml {  public static void main(String[] args) throws Exception{  JAXBContext contextObj = JAXBContext.newInstance(Employee.class);    Marshaller marshallerObj = contextObj.createMarshaller();  marshallerObj.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, true);    Employee emp1=new Employee(1,"Vimal Jaiswal",50000);    marshallerObj.marshal(emp1, new FileOutputStream("employee.xml"));    }  } |

Employee.xml

|  |
| --- |
| 1. **<?xml** version="1.0" encoding="UTF-8" standalone="yes"**?>** 2. **<employee** id="1"**>** 3. **<name>**Vimal Jaiswal**</name>** 4. **<salary>**50000.0**</salary>** 5. **</employee>** |

## Simple JAXB UnMarshalling Example: Converting XML into Object

|  |
| --- |
| import java.io.File;  import javax.xml.bind.JAXBContext;  import javax.xml.bind.JAXBException;  import javax.xml.bind.Unmarshaller;    public class XMLToObject {  public static void main(String[] args) {  try {  File file = new File("employee.xml");  JAXBContext jaxbContext = JAXBContext.newInstance(Employee.class);    Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();  Employee e=(Employee) jaxbUnmarshaller.unmarshal(file);  System.out.println(e.getId()+" "+e.getName()+" "+e.getSalary());    } catch (JAXBException e) {e.printStackTrace(); }    }  } |

Example 2: In this example, we are going to convert the object into xml having primitives, strings and collection objects.

# JAXB Marshalling Example: Converting Object into XML

* Create POJO or bind the schema and generate the classes
* Create the JAXBContext object
* Create the Marshaller objects
* Create the content tree by using set methods
* Call the marshal method

Questions.java

|  |
| --- |
| import java.util.List;    import javax.xml.bind.annotation.XmlAttribute;  import javax.xml.bind.annotation.XmlElement;  import javax.xml.bind.annotation.XmlRootElement;    @XmlRootElement  public class Question {  private int id;  private String questionname;  private List<Answer> answers;  public Question() {}  public Question(int id, String questionname, List<Answer> answers) {  super();  this.id = id;  this.questionname = questionname;  this.answers = answers;  }  @XmlAttribute  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  @XmlElement  public String getQuestionname() {  return questionname;  }  public void setQuestionname(String questionname) {  this.questionname = questionname;  }  @XmlElement  public List<Answer> getAnswers() {  return answers;  }  public void setAnswers(List<Answer> answers) {  this.answers = answers;  }  } |

Answer.java

|  |
| --- |
| public class Answer {  private int id;  private String answername;  private String postedby;  public Answer() {}  public Answer(int id, String answername, String postedby) {  super();  this.id = id;  this.answername = answername;  this.postedby = postedby;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getAnswername() {  return answername;  }  public void setAnswername(String answername) {  this.answername = answername;  }  public String getPostedby() {  return postedby;  }  public void setPostedby(String postedby) {  this.postedby = postedby;  }    } |

*ObjectToXml.java*

|  |
| --- |
| import java.io.FileOutputStream;  import java.util.ArrayList;    import javax.xml.bind.JAXBContext;  import javax.xml.bind.Marshaller;      public class ObjectToXml {  public static void main(String[] args) throws Exception{  JAXBContext contextObj = JAXBContext.newInstance(Question.class);    Marshaller marshallerObj = contextObj.createMarshaller();  marshallerObj.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, true);    Answer ans1=new Answer(101,"java is a programming language","ravi");  Answer ans2=new Answer(102,"java is a platform","john");    ArrayList<Answer> list=new ArrayList<Answer>();  list.add(ans1);  list.add(ans2);    Question que=new Question(1,"What is java?",list);  marshallerObj.marshal(que, new FileOutputStream("question.xml"));    }  } |

Generated XML

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="yes"?>  <question id="1">  <answers>  <answername>java is a programming language</answername>  <id>101</id>  <postedby>ravi</postedby>  </answers>  <answers>  <answername>java is a platform</answername>  <id>102</id>  <postedby>john</postedby>  </answers>  <questionname>What is java?</questionname>  </question> |

# JAXB Unmarshalling Example: Converting XML into Object

# By the help of UnMarshaller interface, we can unmarshal(read) the object into xml document.

In this example, we are going to convert simple xml document into java object.

Let's see the steps to convert XML document into java object.

* Create POJO or bind the schema and generate the classes
* Create the JAXBContext object
* Create the Unmarshaller objects
* Call the unmarshal method
* Use getter methods of POJO to access the data

### Xml Document

*File: question.xml*

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="yes"?>  <question id="1">  <answers>  <answername>java is a programming language</answername>  <id>101</id>  <postedby>ravi</postedby>  </answers>  <answers>  <answername>java is a platform</answername>  <id>102</id>  <postedby>john</postedby>  </answers>  <questionname>What is java?</questionname>  </question> |

POJO classes

File: Question.java

|  |
| --- |
| import java.util.List;    import javax.xml.bind.annotation.XmlAttribute;  import javax.xml.bind.annotation.XmlElement;  import javax.xml.bind.annotation.XmlRootElement;    @XmlRootElement  public class Question {  private int id;  private String questionname;  private List<Answer> answers;  public Question() {}  public Question(int id, String questionname, List<Answer> answers) {  super();  this.id = id;  this.questionname = questionname;  this.answers = answers;  }  @XmlAttribute  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  @XmlElement  public String getQuestionname() {  return questionname;  }  public void setQuestionname(String questionname) {  this.questionname = questionname;  }  @XmlElement  public List<Answer> getAnswers() {  return answers;  }  public void setAnswers(List<Answer> answers) {  this.answers = answers;  }  } |

File: Answer.java

|  |
| --- |
| public class Answer {  private int id;  private String answername;  private String postedby;  public Answer() {}  public Answer(int id, String answername, String postedby) {  super();  this.id = id;  this.answername = answername;  this.postedby = postedby;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getAnswername() {  return answername;  }  public void setAnswername(String answername) {  this.answername = answername;  }  public String getPostedby() {  return postedby;  }  public void setPostedby(String postedby) {  this.postedby = postedby;  }    } |

Unmarshaller class

File: XmlToObject.java

|  |
| --- |
| import java.io.File;  import java.util.List;    import javax.xml.bind.JAXBContext;  import javax.xml.bind.JAXBException;  import javax.xml.bind.Unmarshaller;    public class XmlToObject {  public static void main(String[] args) {    try {    File file = new File("question.xml");  JAXBContext jaxbContext = JAXBContext.newInstance(Question.class);    Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();  Question que= (Question) jaxbUnmarshaller.unmarshal(file);    System.out.println(que.getId()+" "+que.getQuestionname());  System.out.println("Answers:");  List<Answer> list=que.getAnswers();  for(Answer ans:list)  System.out.println(ans.getId()+" "+ans.getAnswername()+" "+ans.getPostedby());    } catch (JAXBException e) {  e.printStackTrace();  }    }  } |

# REST Assured requests

|  |
| --- |
| package com.oracle.cegbu.unifier.rest.restassured.methods;import static io.restassured.RestAssured.given;import java.util.Base64;import io.restassured.builder.RequestSpecBuilder;import io.restassured.http.ContentType;import io.restassured.response.Response;import io.restassured.specification.RequestSpecification;public class RestAssuredRequests {public static Response postCall(String uri, String token, Object bodyPayload ){returngiven().log().all().header("Authorization", "Bearer "+token).header("Content-Type", "application/json").accept(ContentType.JSON).body(bodyPayload).when().post(uri).then().assertThat().statusCode(200).extract().response();}public static Response putCall(String uri, String token, String inputJson ){Response response =given().header("Authorization", "Bearer "+token).header("Content-Type", "application/json").accept(ContentType.JSON).body(inputJson).when().put(uri).then().assertThat().statusCode(200).extract().response();return (response);}public static <T> T deleteCall(String uri, String token, Object bodyPayload, Class<T> responseClass ){returngiven().header("Authorization", "Bearer "+token).header("Content-Type", "application/json").accept(ContentType.JSON).body(bodyPayload).when().delete(uri).then().assertThat().statusCode(200).extract().as(responseClass);}public static Response getCall(String uri, String token ){Response response =given().log().uri().log().parameters().header("Authorization", "Bearer "+token).accept(ContentType.JSON).when().get(uri).then().assertThat().statusCode(200).extract().response();return (response);}public static Response getCall(String uri, String token, String filter){Response response =given().log().uri().log().parameters().header("Authorization", "Bearer "+token).accept(ContentType.JSON).queryParam("filter", filter).when().get(uri).then().assertThat().statusCode(200).extract().response();return (response);}} |

# Sample JSON

|  |
| --- |
|  |

# POSTMAN:

|  |
| --- |
| //Json response parsing      var serviceRes **=** pm.response.json();  ////validation-1 ::Retrieve and verify code value =1      var code **=** serviceRes.code      console.**info**("Code value:: "**+**code);      pm.expect(code).to.equals(1) |

|  |
| --- |
| pm.**test**("Body matches string", **function** () {          pm.expect(pm.response.text()).to.include("string\_you\_want\_to\_search");      }); |

|  |
| --- |
| pm.**test**("Status code is 200", **function** () {      pm.response.to.have.status(200);  });  pm.**test**("Successful POST request", **function** () {      pm.expect(pm.response.code).to.be.oneOf([201, 202]);  }); |

|  |
| --- |
| pm.**test**("Response time is less than 200ms", **function** () {      pm.expect(pm.response.responseTime).to.be.below(200);  }); |

## Running collections on the command line with Newman

Newman is a command-line Collection Runner for Postman. It enables you to run and test a Postman Collection directly from the command line.

Newman is built on Node.js. To run Newman, make sure you have Node.js installed. You can [download and install](https://nodejs.org/en/download/current/) Node.js on Linux, Windows, and macOS

After you install Node.js, Newman is just a command away. Install Newman from npm globally on your system, which allows you to run it from anywhere.

|  |
| --- |
| $ npm install -g newman |

The easiest way to run Newman is to run it with a collection.

|  |
| --- |
| **$ newman run mycollection.json -e environment.json -d testdatafile.json -r htmlextra --reporter-htmlextra-export ./HtmlReports/ProdLadBE\_Poker\_Automation\_Results.html** |

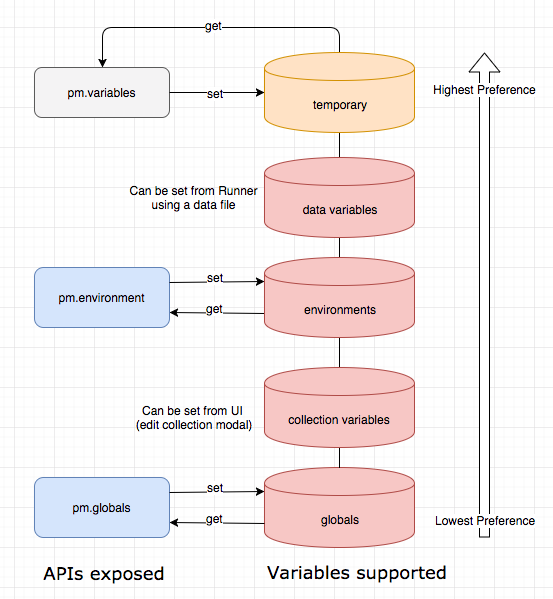
For example, a data file such as the one shown below runs 2 iterations, with each iteration using a set of variables.

|  |
| --- |
| [{  "url": "http://127.0.0.1:5000",  "user\_id": "1",  "id": "1",  "token\_id": "123123",  },  {  "url": "http://postman-echo.com",  "user\_id": "2",  "id": "2",  "token\_id": "899899",  }] |

|  |
| --- |
| $ newman run mycollection.json -d data.json |

Here's an example of the CSV file for the above set of variables:

|  |
| --- |
| url, user\_id, id, token\_id  http://127.0.0.1:5000, 1, 1, 123123123  http://postman-echo.com, 2, 2, 899899 |



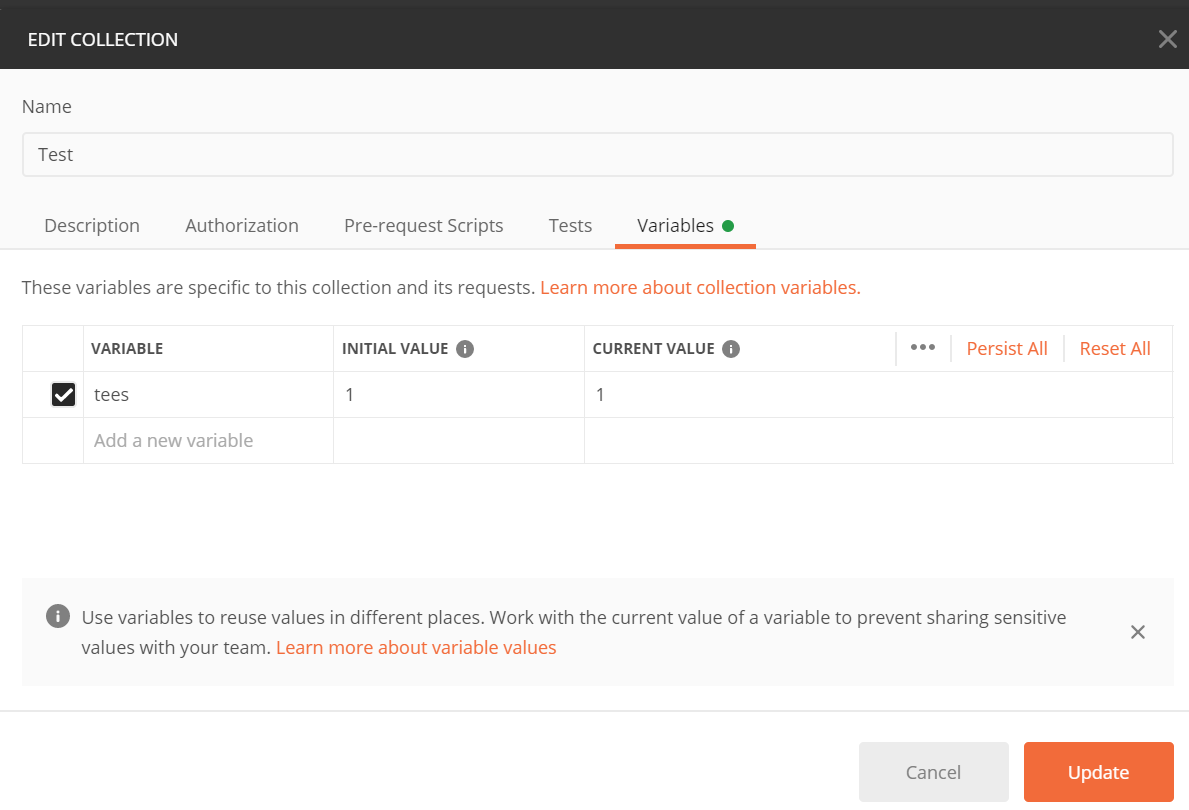
### Defining global and environment variables

You can create and edit environment variables by selecting **Environments** on the left of Postman, or using the **Environment quick look** External link icon at the top right.

You can choose an environment in the drop-down list at the top right:

You can also activate an environment in the left sidebar, by selecting the check-mark button to make the environment active.

### Defining collection variables



### Defining variables in scripts

|  |
| --- |
| pm.globals.set("variable\_key", "variable\_value");  pm.collectionVariables.set("variable\_key", "variable\_value");  pm.environment.set("variable\_key", "variable\_value");  pm.variables.set("variable\_key", "variable\_value"); |

You can use unset to remove a variable:

|  |
| --- |
| pm.environment.unset("variable\_key"); |

## Accessing variables

|  |
| --- |
| {{username}} |

## you could have a request body that accesses a variable by wrapping its reference in double-quotes:

|  |
| --- |
| { "customer\_id" : "{{cust\_id}}" } |

### Using variables in scripts

|  |
| --- |
| //access a variable at any scope including local  pm.variables.get("variable\_key");  //access a global variable  pm.globals.get("variable\_key");  //access a collection variable  pm.collectionVariables.get("variable\_key");  //access an environment variable  pm.environment.get("variable\_key"); |

## ****Generate Advanced HTML Reports, While Using Newman****

We will use **htmlextra reporter** for Newman and this is an **npm dependency** that we need to install separately. This reporter helps to provide a very decent report with a lot of detailed information.

1. To install the **htmlextra reporter**, run command:

**npm install -g newman-reporter-htmlextra**