**58.1. Spring REST - HTTP Overview**

**REST over HTTP**:

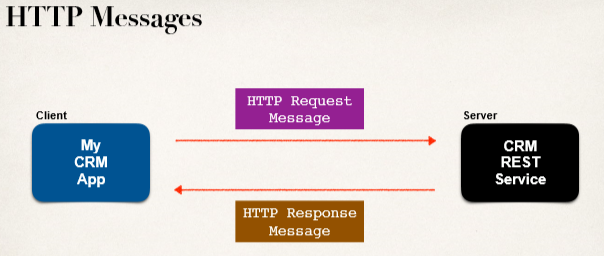
HTTP is a server/client protocol: the server has the file, and the client wants it. In regular web surfing, the client is a web browser such as Mozilla or Internet Explorer. The URL for a document identifies the server, which the browser contacts and requests the document from. The server returns either in error ("file not found") or success (in which case the document is attached).

* Most common use of REST is over HTTP
* Leverage HTTP methods for CRUD operation

|  |  |
| --- | --- |
| HTTP Method | CRUD Operation |
| POST | **C**reate a new entity |
| GET | **R**ead a list of entities or a single data |
| PUT | **U**pdate an existing entity |
| DELETE | **D**elete an existing entity |

**HTTP Message**:

HTTP messages are how data is exchanged between a server and a client. There are two types of messages: requests sent by the client to trigger an action on the server, and responses, the answer from the server.

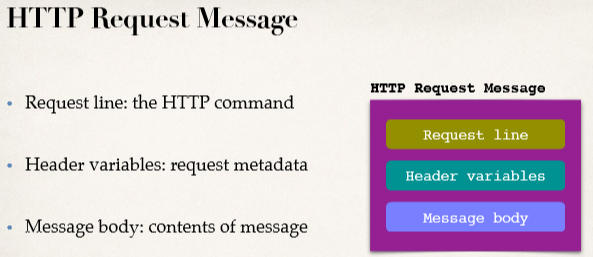


**HTTP Request Message**:

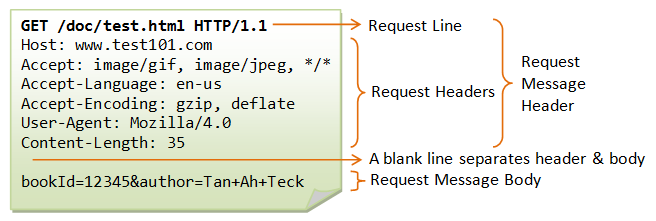
An HTTP request has three parts: the request line, the header variables, and message body of the request.

1. Request line: the HTTP command
2. Header variables: request metadata
3. Message body: contents of message

**Diagram of HTTP Request Message**:



**HTTP Request Message Example**:

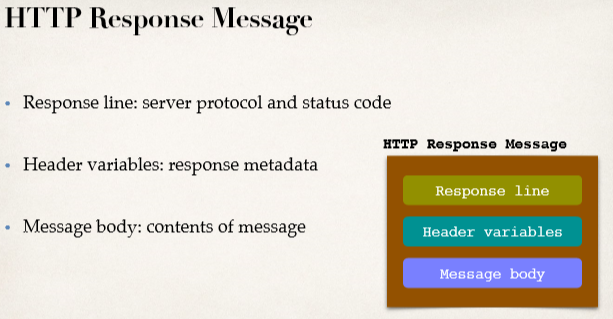


**HTTP Response Message**:

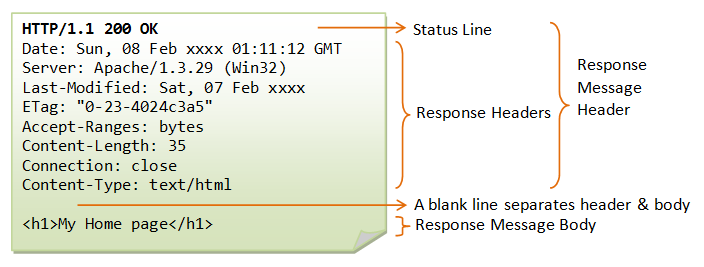
The server's response also has three parts: the response line, header variables, and an message body.

1. Response line: server protocol and status code
2. Header variables: response metadata
3. Message body: contents of message

**Diagram of HTTP Response Message**:



**HTTP Response Message Example**:



**Details**:

* <http://lwp.interglacial.com/ch02_02.htm>
* <https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html>

**HTTP Response - Status Codes**:

The status code is a 3-digit number. The first line of the response message (i.e., the status line) contains the response status code, which is generated by the server to indicate the outcome of the request. We have status code in different ranges.

|  |  |
| --- | --- |
| **Code Range** | **Description** |
| 100 - 199 | Information (Request received; server is continuing the process) |
| 200 - 299 | Successful (The request was successfully received, understood, accepted and serviced) |
| 300 - 399 | Redirection (Further action must be taken in order to complete the request) |
| 400 - 499 | Client error (The request contains bad syntax or cannot be understood) |
| 500 - 599 | Server error (The server failed to fulfill an apparently valid request) |

**Some commonly encountered status codes are:**

* 100 Continue: The server received the request and in the process of giving the response.
* 200 OK: The request is fulfilled.
* 301 Move Permanently: The resource requested for has been permanently moved to a new location. The URL of the new location is given in the response header called Location. The client should issue a new request to the new location. Application should update all references to this new location.
* 302 Found & Redirect (or Move Temporarily): Same as 301, but the new location is temporarily in nature. The client should issue a new request, but applications need not update the references.
* 304 Not Modified: In response to the If-Modified-Since conditional GET request, the server notifies that the resource requested has not been modified.
* 400 Bad Request: Server could not interpret or understand the request, probably syntax error in the request message.
* 401 Authentication Required: The requested resource is protected, and require client’s credential (username/password). The client should re-submit the request with his credential (username/password).
* 403 Forbidden: Server refuses to supply the resource, regardless of identity of client.
* 404 Not Found: The requested resource cannot be found in the server.
* 405 Method Not Allowed: The request method used, e.g., POST, PUT, DELETE, is a valid method. However, the server does not allow that method for the resource requested.
* 408 Request Timeout:
* 414 Request URI too Large:
* 500 Internal Server Error: Server is confused, often caused by an error in the server-side program responding to the request.
* 501 Method Not Implemented: The request method used is invalid (could be caused by a typing error, e.g., "GET" misspell as "Get").
* 502 Bad Gateway: Proxy or Gateway indicates that it receives a bad response from the upstream server.
* 503 Service Unavailable: Server cannot response due to overloading or maintenance. The client can try again later.
* 504 Gateway Timeout: Proxy or Gateway indicates that it receives a timeout from an upstream server.

**MIME Content Type**:

MIME is the abbreviation for Multipurpose Internet Mail Extensions. It’s a standard originally developed to extend e-mails to be able to support more formats like non-ASCII text and attachments in form of image, audio, video or executable files. The MIME Type is part of the header of he MIME and specifies the type of media contained in an e-mail. It is often also referred to as media type or MIME content type.

The usage of MIME is not limited to e-mail though. Actually, it is used on the internet to determine the type of a file. It works similarly to the file extension on a computer. Web servers and browsers contain a list of such MIME Types that help them to identify and thus interpret all kinds of files, independent of the operating system and hardware used by the user.

MIME Types are structured in a certain way, containing a type and a subtype. For example, the MIME Type of a .jpg image is image/jpeg with “image” being the type and “jpeg” being the subtype. A slash (/) is used to separate type from subtype.

**MIME Content Types:**

* The message format is described by MIME content type.
  + **M**ultipurpose **I**nternet **M**ail **E**xtensions.
* Basic Syntax: **type / sub-type**
  + Example: "text/html", "text/plain", "application/json", "application/xml"

**Client Tool**:

We need a client tool to send HTTP request to the REST web service API, get the response and view them in the client. Here we will use **POSTMAN**.

58.1. Spring REST - HTTP Overview