匠人学院
jiangren.com.au

Java Spring Boot —— RESTful API

www.jiangren.com.au

RESTful Web Services

REST stands for **RE**presentational **S**tate **T**ransfer. REST is a web standards based architecture and uses HTTP Protocol for data communication. It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in year 2000.

In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource like Text, JSON and XML. JSON is now the most popular format being used in Web Services.

RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.

匠人学院
jiangren.com.au

REST Features

- Resource
- Messages
- Addressing
- Methods
- Stateless
- Caching
- Security

Resources

REST architecture treats every content as a resource. These resources can be Text Files, Html Pages, Images, Videos or Dynamic Business Data. REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource where Text, JSON, XML. The most popular representations of resources are XML and JSON.

Representation of Resources

XML, JSON, HTML etc.

```
<user>
   <id>1</id>
   <name>Mahesh</name>
   <profession>Teacher</profession>
</user>
```

```
{
   "id":1,
   "name":"Mahesh",
   "profession":"Teacher"
}
```

Resources Representation

Good Resources Representation

REST does not impose any restriction on the format of a resource representation. A client can ask for JSON representation whereas another client may ask for XML representation of the same resource to the server and so on.

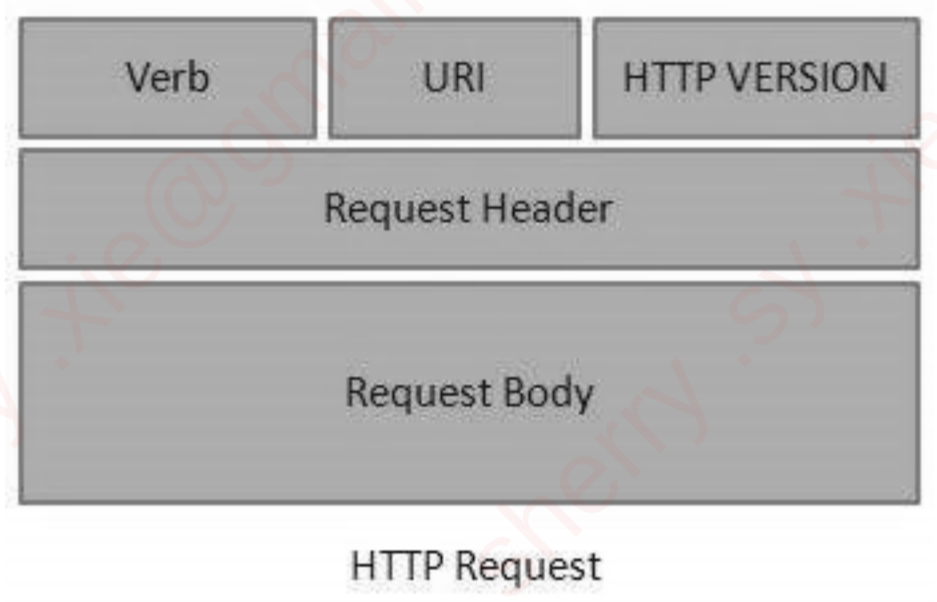
Understandability – Both the Server and the Client should be able to understand and utilize the representation format of the resource.

Completeness – Format should be able to represent a resource completely. For example, a resource can contain another resource. Format should be able to represent simple as well as complex structures of resources.

Linkablity – A resource can have a linkage to another resource, a format should be able to handle such situations.

匠人学院
jiangren.com.au

Messages HTTP Request



HTTP Request

Verb – Indicates the HTTP methods such as GET, POST, DELETE, PUT, etc.

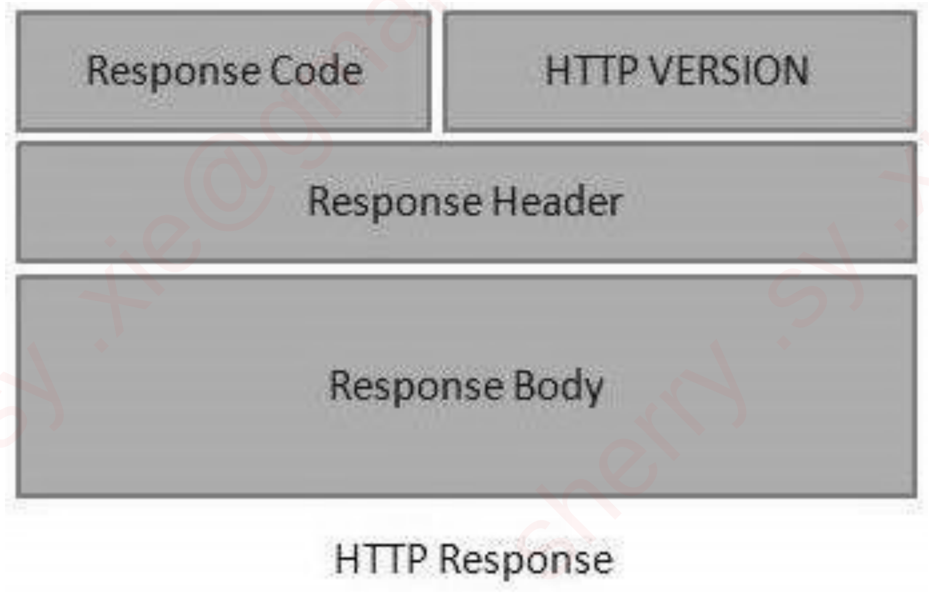
URI – Uniform Resource Identifier (URI) to identify the resource on the server.

HTTP Version – Indicates the HTTP version. For example, HTTP v1.1.

Request Header – Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by the client, format of the message body, cache settings, etc.

Request Body – Message content or Resource representation.

Messages HTTP Response



Status/Response Code – Indicates the Server status for the requested resource. For example, 404 means resource not found and 200 means response is ok.

HTTP Version – Indicates the HTTP version. For example HTTP v1.1.

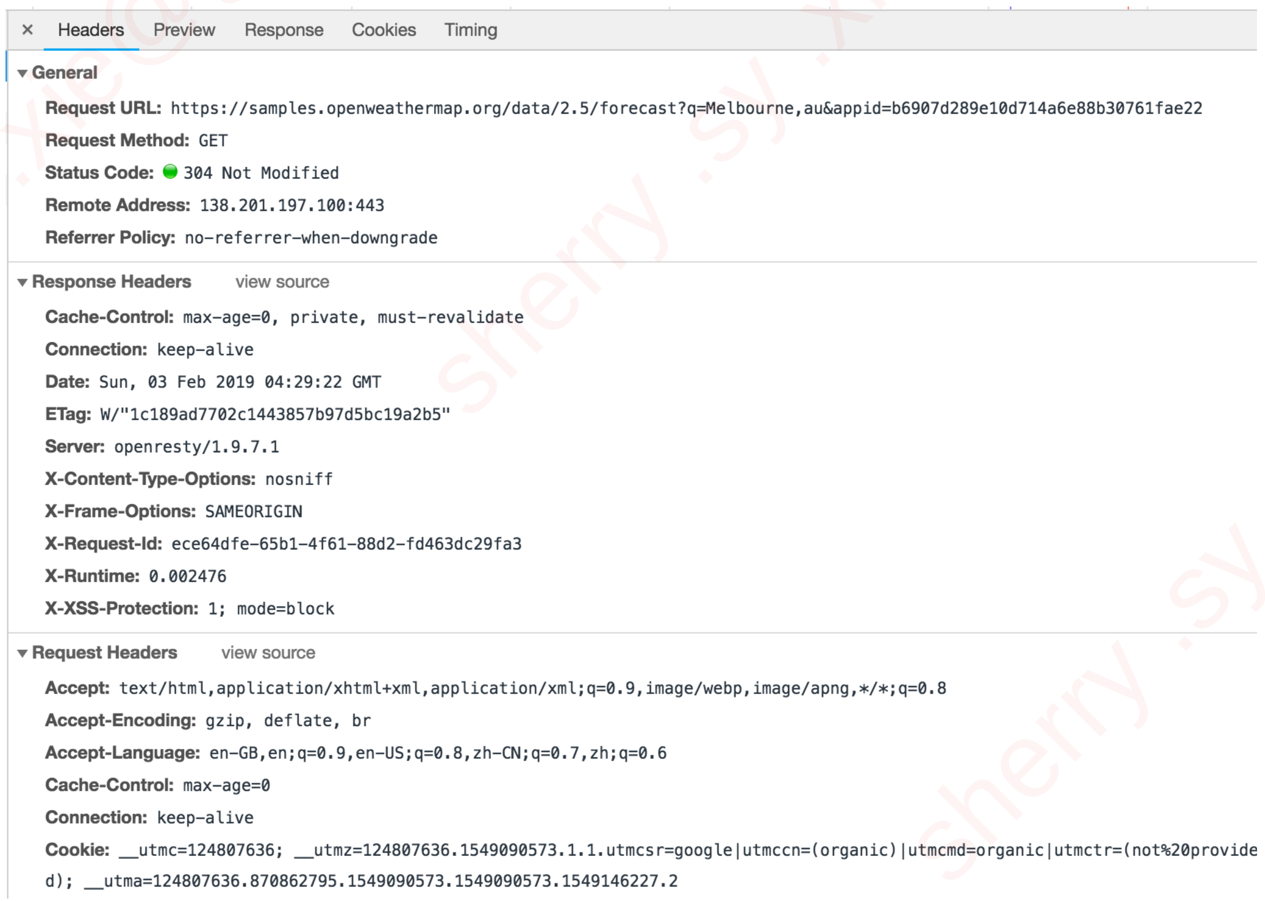
Response Header – Contains metadata for the HTTP Response message as key value pairs. For example, content length, content type, response date, server type, etc.

Response Body – Response message content or Resource representation.

匠人学院
jiangren.com.au

Messages Example

https://samples.openweathermap.org/data/2.5/forecast?q=Melbourne,au&appid=b6907d289e10d714a6e88b30761fae22



× Headers Preview Response Cookies Timing

▼ General

```
Request URL: https://samples.openweathermap.org/data/2.5/forecast?q=Melbourne,au&appid=b6907d289e10d714a6e88b30761fae22
Request Method: GET
Status Code: ● 304 Not Modified
Remote Address: 138.201.197.100:443
Referrer Policy: no-referrer-when-downgrade
```

▼ Response Headers view source

```
Cache-Control: max-age=0, private, must-revalidate
Connection: keep-alive
Date: Sun, 03 Feb 2019 04:29:22 GMT
ETag: W/"1c189ad7702c1443857b97d5bc19a2b5"
Server: openresty/1.9.7.1
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Request-Id: ece64dfe-65b1-4f61-88d2-fd463dc29fa3
X-Runtime: 0.002476
X-XSS-Protection: 1; mode=block
```

▼ Request Headers view source

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en;q=0.9,en-US;q=0.8,zh-CN;q=0.7,zh;q=0.6
Cache-Control: max-age=0
Connection: keep-alive
Cookie: __utmc=124807636; __utmz=124807636.1549090573.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provide
d); __utma=124807636.870862795.1549090573.1549090573.1549146227.2
```

Chrome developer Tools

Addressing

Addressing refers to locating a resource or multiple resources lying on the server. It is analogous to locate a postal address of a person.

Constructing a Standard URI

Use Plural Noun – Use plural noun to define resources.

Avoid using spaces – Use underscore (_) or hyphen (-) when using a long resource name. For example, use authorized_users instead of authorized%20users.

Use lowercase letters – Although URI is case-insensitive, it is a good practice to keep the url in lower case letters only.

Maintain Backward Compatibility – As Web Service is a public service, a URI once made public should always be available. In case, URI gets updated, redirect the older URI to a new URI using the HTTP Status code, 300.

Use HTTP Verb – Always use HTTP Verb like GET, PUT and DELETE to do the operations on the resource. It is not good to use operations name in the URI.

Addressing Examples

```
GET http://localhost:8080/UserManagement/rest/UserService/getUser/1  BAD
DELETE http://localhost:8080/UserManagement/rest/UserService/deleteUser/1
BAD

GET http://localhost:8080/usermanagement/api/v1/userservice/users/1 GOOD
DEL http://localhost:8080/usermanagement/api/v1/userservice/users/1 GOOD
```

HTTP Methods

The following HTTP methods are most commonly used in a REST based architecture.

- **GET** – Provides a read only access to a resource.
- **POST** – Used to create a new resource.
- **PUT** – Used to update an existing resource or create a new resource.
- **DELETE** – Used to remove a resource.
- **OPTIONS** – Used to get the supported operations on a resource.

HTTP Methods

No.	HTTP Method	URI	Operation
1	**GET**	/books	Get list of books
2	**GET**	/books/1	Get book with Id 1
3	**PUT**	/books/2	Update book with Id 2
4	**POST**	/books	Create a book
5	**DELETE**	/books/1	Delete Book with Id 1
6	**OPTIONS**	/books	List the supported operations in web service

Statelessness

As per the REST architecture, a RESTful Web Service should not keep a client state on the server. This restriction is called Statelessness. It is the responsibility of the client to pass its context to the server and then the server can store this context to process the client's further request. For example, session maintained by server is identified by session identifier passed by the client.

```
<book>
   <id>1</id>
   <name>Game of throne</name>
</book>
```

Stateless means every time, you hit the RESTful Service, it should return the same value, regardless the State of the user journey.

Idempotency : 幂等性

匠人学院
jiangren.com.au

Statelessness Pros vs Cons

Pros:

- Web services can treat each method request independently.
- Web services need not maintain the client's previous interactions. It simplifies the application design.
- As HTTP is itself a statelessness protocol, RESTful Web Services work seamlessly with the HTTP protocols.

Cons

- Web services need to get extra information in each request and then interpret to get the client's state in case the client interactions are to be taken care of.

Solutions:

Server Side Session, Business Process Engine in the background etc.

Caching

Caching refers to storing the server response in the client itself, so that a client need not make a server request for the same resource again and again. A server response should have information about how caching is to be done, so that a client caches the response for a time-period or never caches the server response.

No	Header	Description
1	Date	Date and Time of the resource when it was created.
2	Last Modified	Date and Time of the resource when it was last modified.
3	Cache-Control	Primary header to control caching.
4	Expires	Expiration date and time of caching.
5	Age	Duration in seconds from when resource was fetched from the server.

Caching Cache-Control

Cache-Control Header

No	Header	Description
1	Public	Indicates that resource is cacheable by any component.
2	Private	Indicates that resource is cacheable only by the client and the server, no intermediary can cache the resource.
3	no-cache/no-store	Indicates that a resource is not cacheable.
4	max-age	Indicates the caching is valid up to max-age in seconds. After this, client has to make another request.
5	must-revalidate	Indication to server to revalidate resource if max-age has passed.

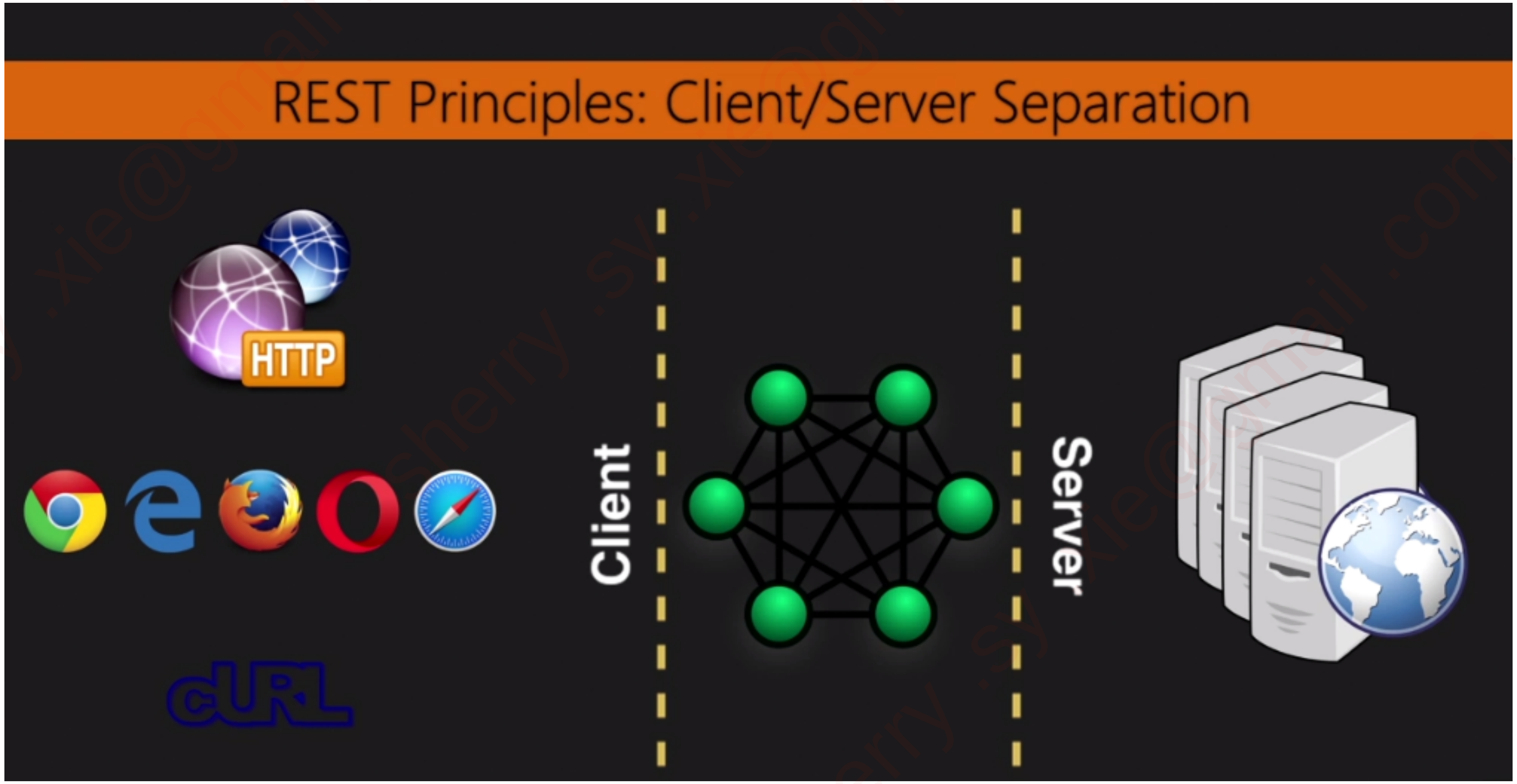
匠人学院
jiangren.com.au

Caching in Spring Boot Code

```
public User getUser(@PathVariable int id, final HttpServletResponse
response) {
  String headerValue = CacheControl.maxAge(10,
TimeUnit.SECONDS)
      .getHeaderValue();

  response.addHeader("Cache-Control", headerValue);
  return new User(id, "user@test.com", "testPassword", 30);
}
```

REST Client/Server Separation



Swagger 2 构建 RESTful APIs文档

swagger2 官网：https://swagger.io/

- 接口非常多，细节又复杂，如果由程序员高质量的输出一个文档，经常耗时长而且效果也不好，抱怨声不绝与耳。
- 随着时间的推移，文档需要与代码保持同步。但由于大部分程序员都还有着文档不重要的思想，于是总有这样那样的原因导致程序员不愿意或遗忘了更新文档。

Swagger 是一系列 RESTful API 的工具，通过 Swagger 可以获得一种交互式文文档，客户端 SDK 的自自 动生生成等功能。
Swagger 的目目标是为 REST APIs 定义一一个标准的、与语言言无无关的接口口，使人人和计算机在看不不到源码或者看 不不到文文档或者不不能通过网网络流量量检测的情况下，能发现和理理解各种服务的功能。当服务通过 Swagger 定义， 消费者就能与远程的服务互动通过少量量的实现逻辑。类似于低级编程接口口，Swagger 去掉了了调用用服务时的很 多猜测。

匠人学院
jiangren.com.au

Swagger 2 构建 RESTful APIs文档

作用范围	API	使用位置
协议集描述	@Api	用于 Controller 类上
协议描述	@ApiOperation	用在 Controller 的方法上
非对象参数集	@ApiImplicitParams	用在 Controller 的方法上
非对象参数描述	@ApiImplicitParam	用在 @ApiImplicitParams 的方法里边
响应集	@ApiResponses	用在 Controller 的方法上
响应信息参数	@ApiResponse	用在 @ApiResponses 里边
描述返回对象的意义	@ApiModel	用在返回对象类上
对象属性	@ApiModelProperty	用在出入参数对象的字段上

```
compile group: 'io.springfox', name: 'springfox-swagger2', version: '2.9.2'
compile group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.9.2'
```

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

  @Bean
  public Docket createRestApi() {
    ApiInfo apiInfo = new ApiInfoBuilder()
        .title("使用Swagger2 Docs")
        .description("API Document ")
        .termsOfServiceUrl("http://localost:8080")
        .contact(new Contact("Fiona", "test.com", "fiona@test.com"))
        .version("1.0.0")
        .build();

    return new Docket(DocumentationType.SWAGGER_2)
        .apiInfo(apiInfo)
        .select()
        .apis(RequestHandlerSelectors.basePackage("com.fiona.demo"))
        .paths(PathSelectors.any())
        .build();
  }
}
```

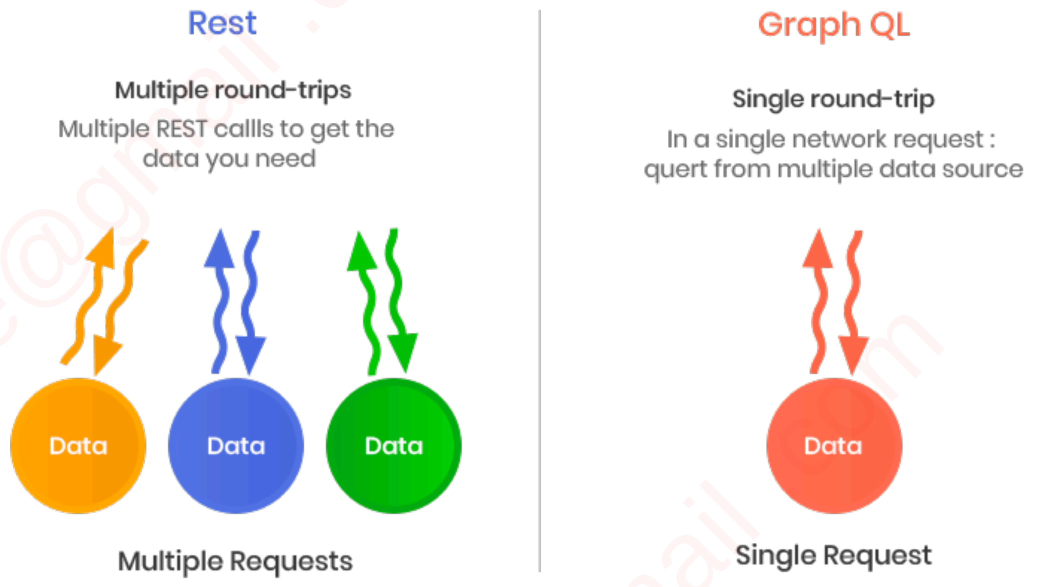
http://localhost:8080/swagger-ui.html

```
@RequestMapping("/hello")
@ApiOperation(value="hello world", notes="")
public String hello(String name) {
  return "Hello " + name;
}
```

```
@ApiOperation(value="get user information", notes="")
@ApiImplicitParams({
    @ApiImplicitParam(name = "userId", value = "Id of user",
required = true, dataType = "int"),
})
```

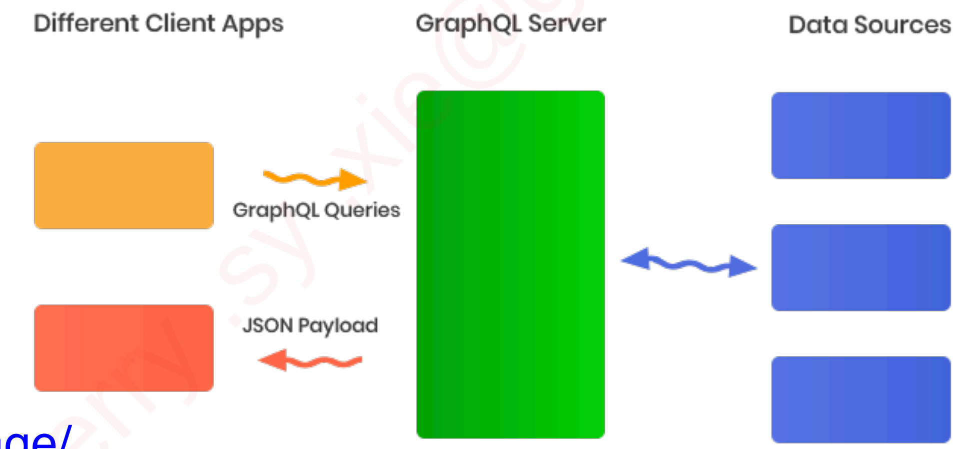
GraphQL

GraphQL 是一个用于 API 的查询语言，是一个使用基于类型系统来执行查询的服务端运行时（类型系统由你的数据定义）。GraphQL 并没有和任何特定数据库或者存储引擎绑定，而是依靠你现有的代码和数据支撑。



Rest
Multiple round-trips
Multiple REST callls to get the data you need
Data
Data
Data
Multiple Requests
Graph QL
Single round-trip
In a single network request : quert from multiple data source
Data
Single Request

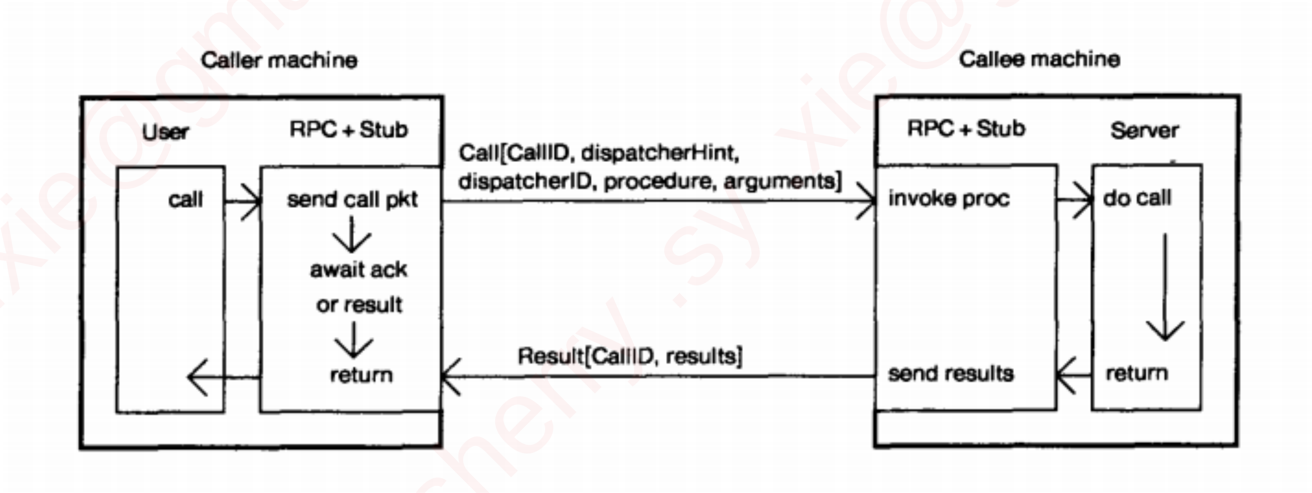
1. **Round-Trips are lesser in GraphQL**
2. **Client Specific Queries**
3. **Self Discoverable**
4. **GraphQL is Hierarchical**



https://graphql.cn/
https://graphql.cn/blog/graphql-a-query-language/

其他接口通信方式

- RPC：远程过程调用Remote Procedure Call



- Protocol: TCP, HTTP
- Format: xml, JSON, protocol buffer..
- Tools: Thrift, gPRC

参考阅读：https://waylau.com/remote-procedure-calls/

匠人学院
jiangren.com.au

Q & A