Spring based REST Services...

Create REST APIs/Web service using Spring

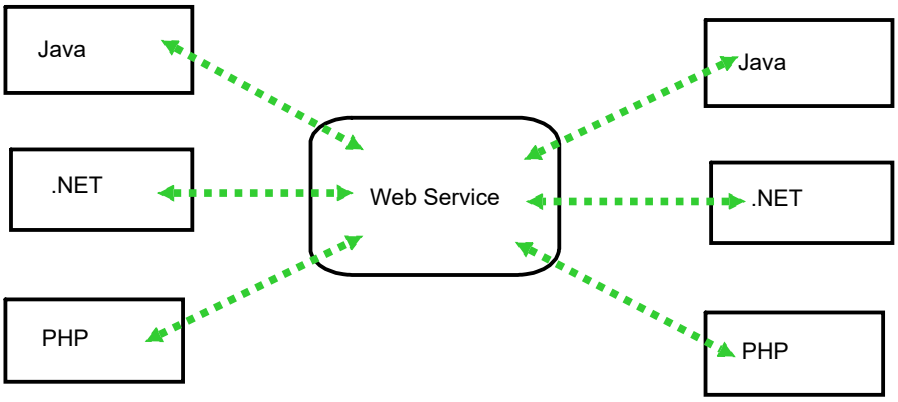
Discuss REST concepts,JSON,HTTP messaging

REST client tool : POSTMAN

Building and consuming CRUD interface with Spring REST

JdbcTemplate and Hibernate

Web Service:



as a collection of standards or protocols for exchanging info between two devices or applications

CORBA : Broker architecture : not as smooth, efficient and abstract as required :

Enterprise requirement does not satisfied with Broker architecture...

JCP : Java Community Program:

instead of broker architecture major vendors agreed upon a set of protocols or web service component to share info

Protocol package:

- SOAP
- WSDL
- UDDI

SOAP : Simple Object Access Protocol

==>W3C Recommendation for comm between applications (XML-based protocol for accessing and consuming web services)

XML : language independent and platform independent

WSDL : "wiz-dull"

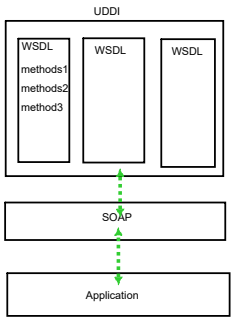
#web services description language :

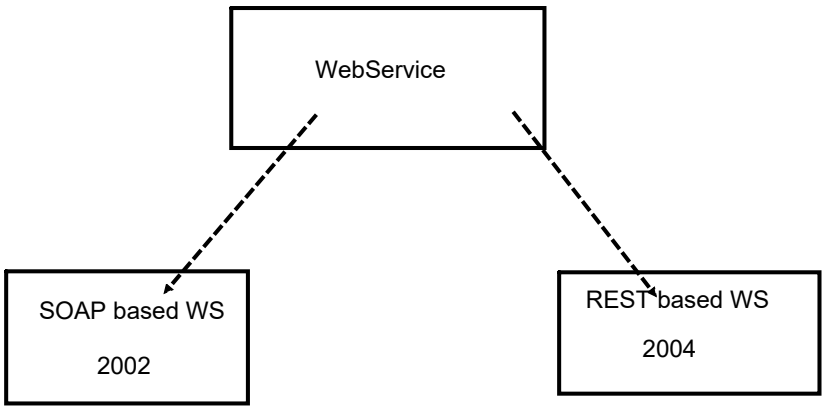
xml document containing info about web services like : method name,parameter,return,how to access

UDDI : Universal Description, Discovery and Integration

Directory of web services interfaces described by WSDL

XML based framework for describing,discovering and integrating web services





SOAP Based :

advantage : security :SOAP defines its own security (WS Security)

Language independent

limitation:

Slow : 1. XML based : need more time to parse

2. defines many standards to be followed , slow and consumes more bandwidth

WSDL dependent : no other mechanism to discover web services other than WSDL document

RESTful web services:

REpresentational State Transfer

#is an architecture style not a protocol

=>can have any implementation as long as it follows REST architecture style

FAST :

Language and platform independent

can use SOAP protocol for implementation

Permits diff data formats

REST V/s SOAP

==> SOAP is protocol

#REST: architectural style

==>SOAP can't use REST

#REST : concept: it can use SOAP, HTTP

==>SOAP uses service interface to expose business logic (exposing object used to call methods to get any service done)

#REST uses URI to expose business logic

==>SOAP : defines standard to be strictly followed

#REST : does not have any strict standards

==>SOAP : resource intensive (uses more bandwidth)

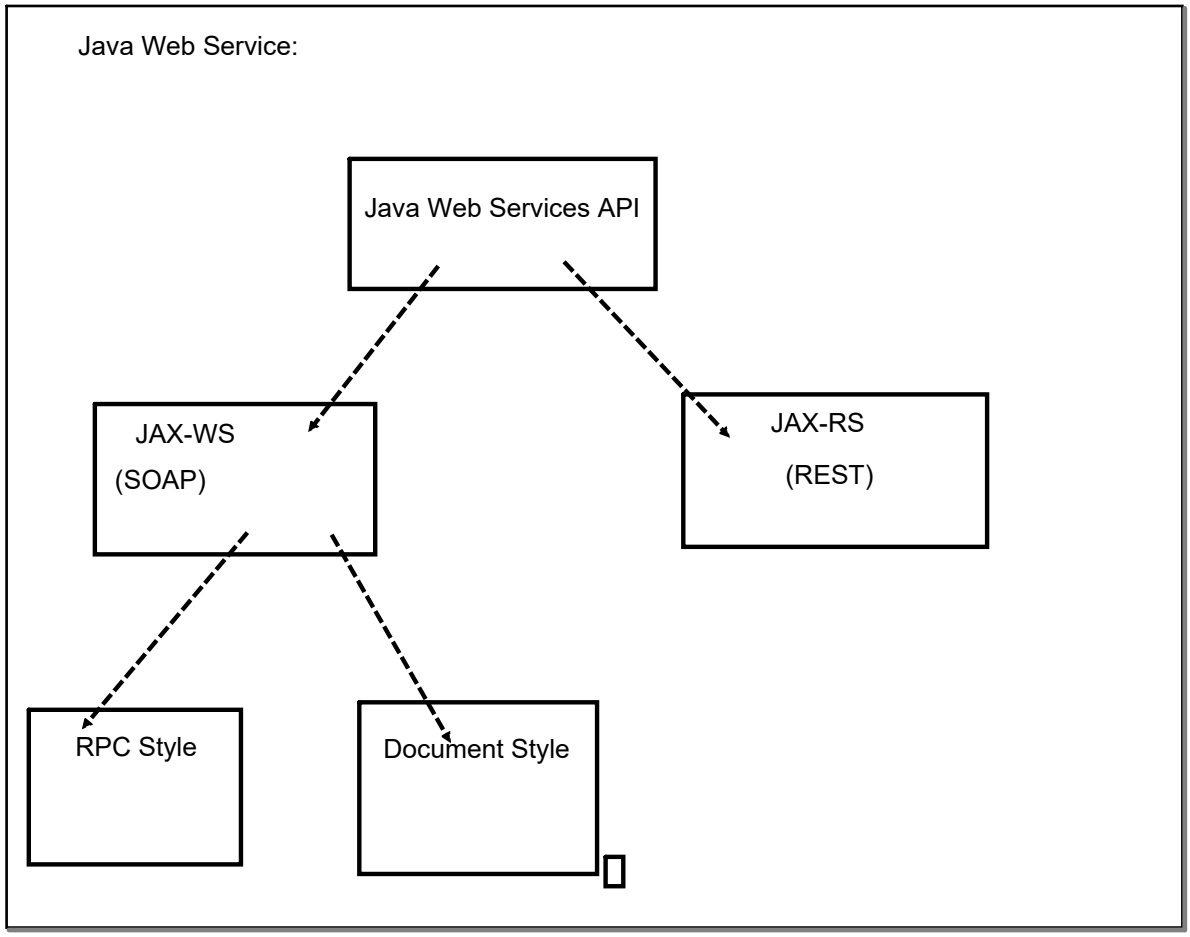
#REST : lightweight implementation

==>SOAP : defines its own security

#REST: inherits the security measures from underlying framework

==>SOAP: permits only XML format

#REST: multiple formats (JSON,XML,PLAIN TEXT,HTML...)


Java Web Service:
Java Web Services API
JAX-WS
(SOAP)
JAX-RS
(REST)
RPC Style
Document Style

Key Elements of RESTful web Services:

1. Resource: actual service client app needs to consume
   web application : http://demo.app.com (maintain employee records)
   TO access record of an particular employee (with id 2) :
   need to issue a command / URI (http://demo.app.com/employee/2)

2. Request Verbs : (GET,PUT,POST,DELETE) : intention,
what do you want to do with res
http://demo.app.com/employee
   GET : get all employee records
   POST: add a new employee record
   PUT : update a employee record

3. Request Header : additional info req to sent with request
4. Request Body : Data to be sent with request
5 Response Body : main body of response (containing response data)
6. Response Status Code : returned along with response

RESTful Principles and Constraints:

any RESTful service must comply with the following characteristics

1. RESTful Client-Server

Both must comply with RESful key elements (RESTful way)

2. Stateless:

Server does not maintains state /track of requests..

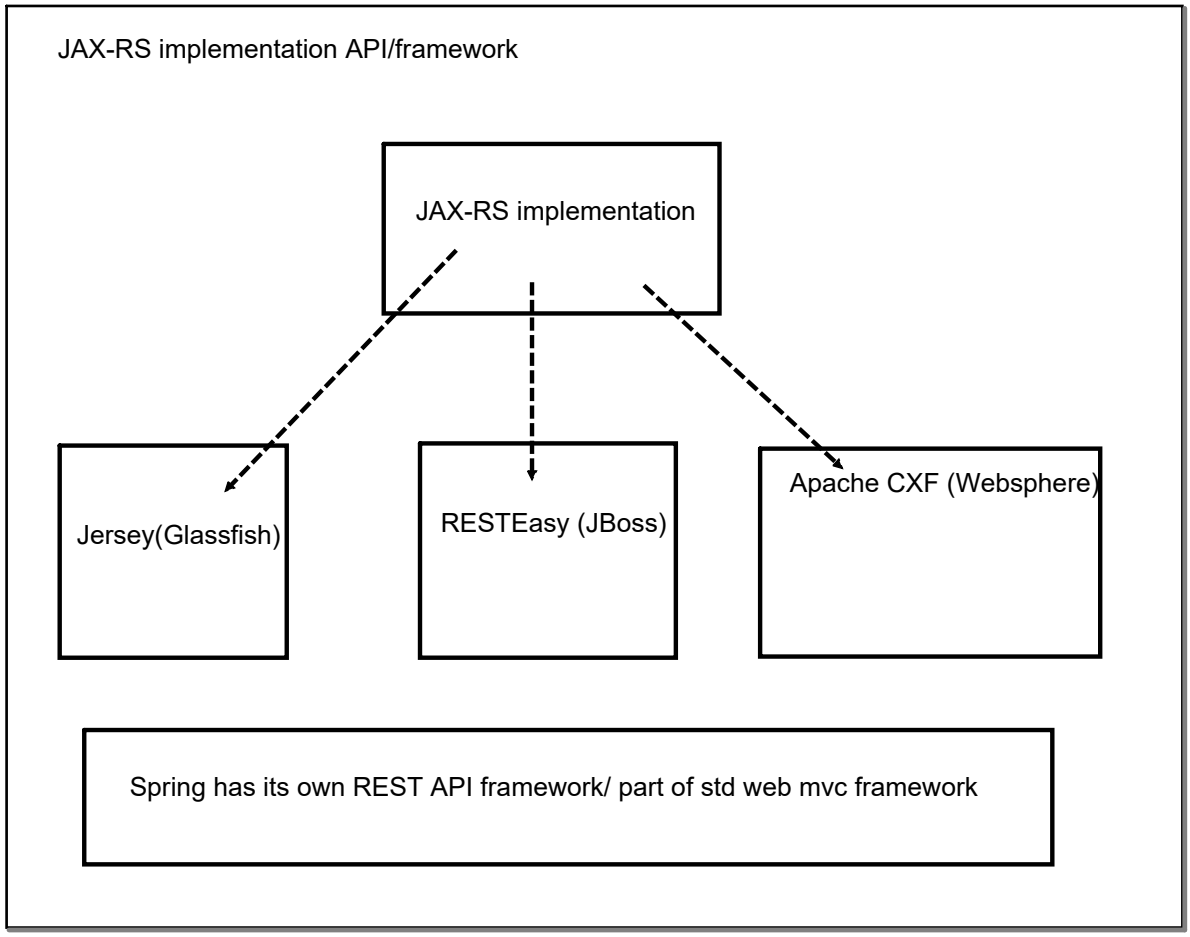
each request is new for server

3. Cache shall be implemented on client to save network traffic

4. Layered System : can have a middlleware layer : additional services :

#must be transparent (not to disturb client-server interaction)

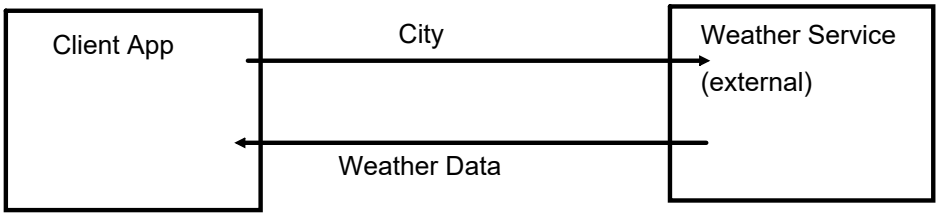
5.Uniform contract / interface


JAX-RS implementation API/framework
JAX-RS implementation
Jersey(Glassfish)
RESTEasy (JBoss)
Apache CXF (Websphere)
Spring has its own REST API framework/ part of std web mvc framework

Business Problem:

=>in my client app need to provide weather report of the city

=>need to get weather data form external service



1. How to connect to external Weather service???
2. What programming language do we use???
3. What is data format???

Ans 1: We can call REST API call using URI (REST endpoints) over HTTP

Ans 2: REST is language independent

Ans 3: Flexibility of multiple data format:
default standard to use JSON (popular and modern)

eg:
openweathermap.org

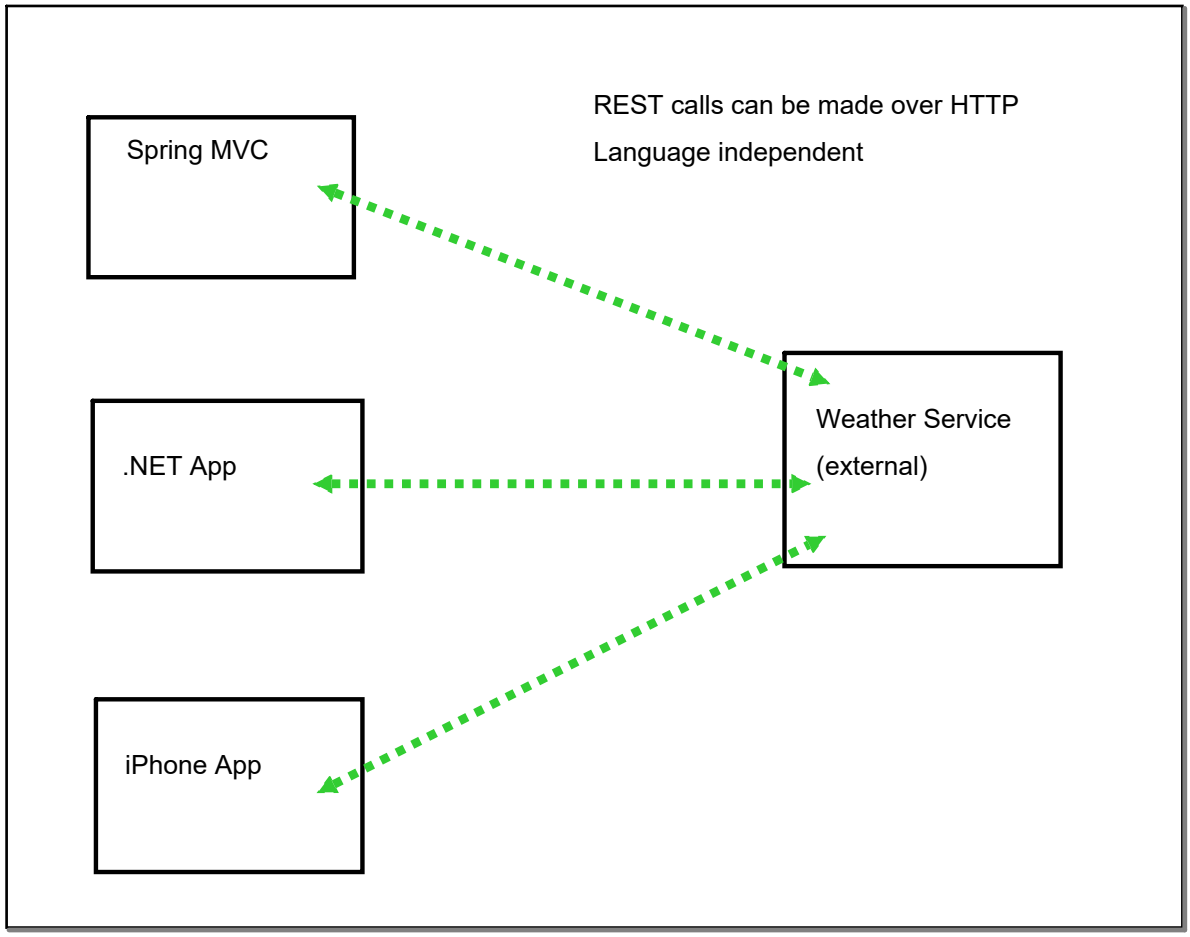
REST endpoint : api.openweather.org/data/2.5/weather?q={cityname}

Weather data as JSON (JavaScript Object Notation)

eg:

```
{
    "temp" : 25,
    "temp_min": 11,
    "temp_max": 30
    "humidity" : 81
...

}
```


REST calls can be made over HTTP
Language independent
Spring MVC
.NET App
iPhone App
Weather Service
(external)

REST API

RESTful API

REST Webservices

RESTful Webservice

REST Service

RESTful Service

JSON:

Lightweight data format for storing and exchanging data :... (plain text yet complies with object notation)

Language Independent.... (no JAVASCRIPT required)

JSON object:

#defined inside Curley braces

#Object member : name-value pair (delimited by colon)

#name : always in double quotes

JSON value types

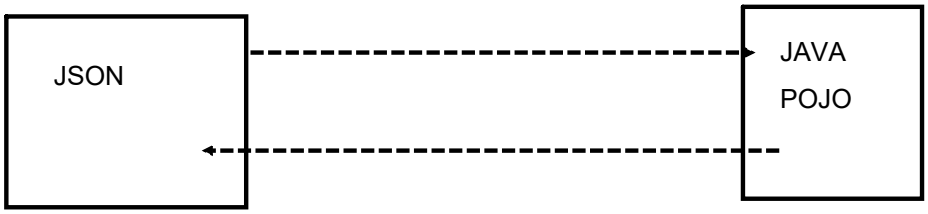
1. Number : no quotes
2. String : double quotes
3. Boolean: true/false
4. Nested JSON object
5. Array
6. null

eg:

```
{
    "id" : 4,
    "firstName" : "First",
    "active" : true
    "courses" : null,
    "languages" : ["java","C#","Python"],
    "address" : {
                    "street" : 6,
                    "city" : "xyz"
                }
}
```

Java JSON Data Binding

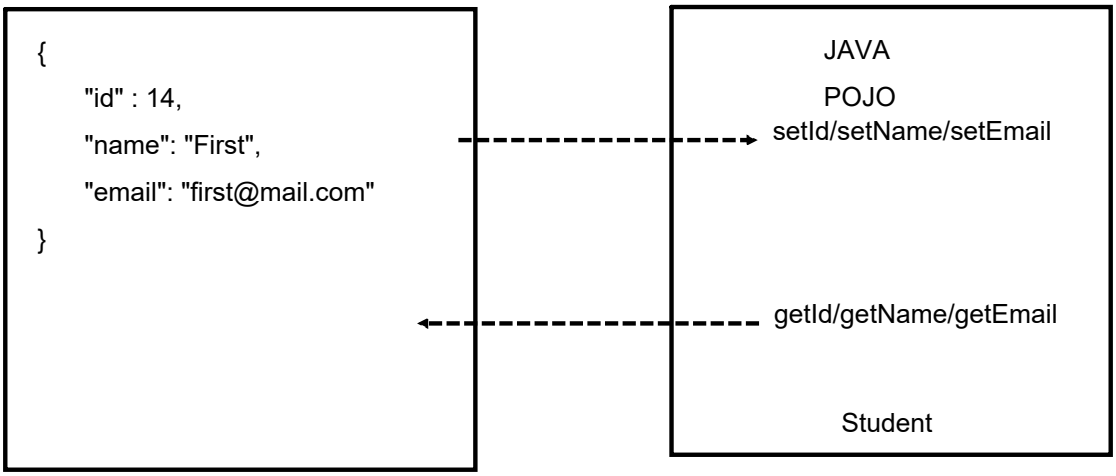
Data Binding is the process of converting JSON data to JAVA POJO and vice versa



Mapping
Serialization/Deserialization
Marshalling/UnMArshalling

Spring uses Jackson Project for data binding (behind the scene)
com.fasterxml.jackson.databind

Jackson mode of working



#Default , Jackson will call appropriate getter/setter method

It does not access the internal private field directly

REST over HTTP

#Common use of REST is over HTTP

Leverage HTTP Methods(HTTP verbs) (intentions)

REST request component

1. URI
2. HTTP verb

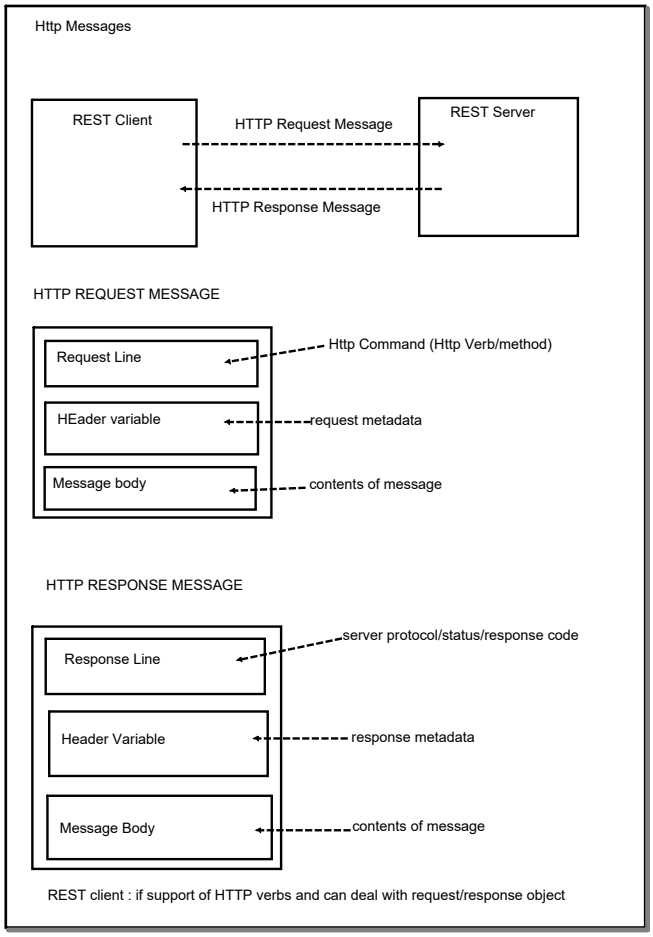
HTTP method

POST : Create a new entity

GET : Read a list of entities or single entity

PUT : Update an existing entity

DELETE : Delete an existing entity


Http Messages
REST Client
HTTP Request Message
REST Server
HTTP Response Message
HTTP REQUEST MESSAGE
Request Line
Http Command (Http Verb/method)
HEader variable
request metadata
Message body
contents of message
HTTP RESPONSE MESSAGE
Response Line
server protocol/status/response code
Header Variable
response metadata
Message Body
contents of message
REST client : if support of HTTP verbs and can deal with request/response object

HTTP RESPONSE-status code: universal to exchange request status between client and server

Code ranges

100-199 : Informational

200-299 : Successful

300-399: Redirection

400-499: Client error

500-599 : Server Error

MIME Content Type

#message format is described by MIME type

Multipurpose Internet Mail-Extension

Basic Syntax : type/sub-type

eg:

text/html , text/plain

application/json, application/xml

Client Tool

POSTMAN

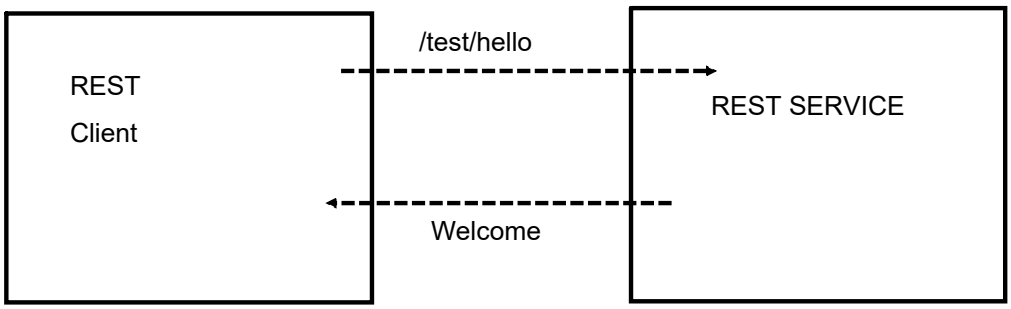
Spring REST Support:

Spring Web MVC provides the complete support for Spring REST

1. Create a Controller (Rest Controller) @RestController (Handle REST request and responses)

2.add dependency of jackson -databind :

Spring REST will use jackson automatically : classpath/pom.xml



Create a Service : Return a list of students

REST endpoint : /api/students (GET)

method: return List<Student>-->JSON (array of JSON Object): jackson will do databinding behind the scene

One more Service : Return a record of a specific student (based on id received from client)

REST endpoint : /api/students/{studentId} (GET)

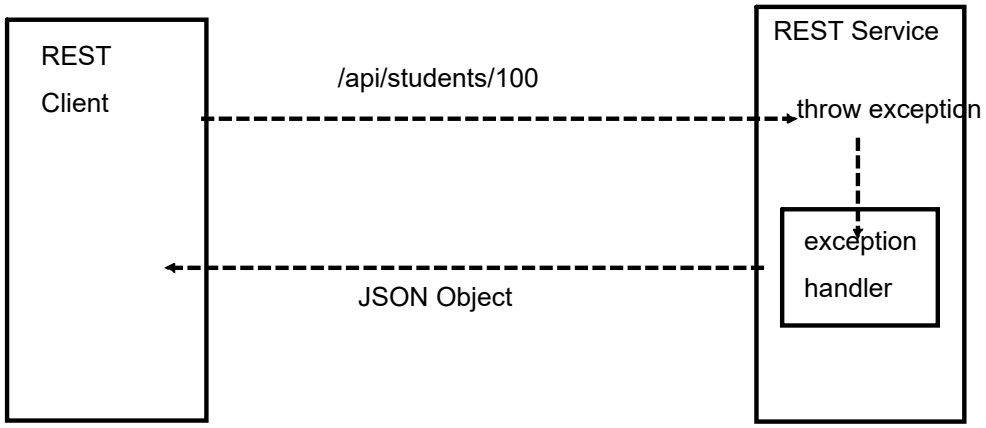
return single Student object

Postman

In case of exception :

#Handle the exception and return error as customized JSON object

#Spring way of handling exception



#Create a user defined Exception class
#Create a custom error response class (JSON error object)
#Update REST code to throw the exception
#Add an exception handler (spring way exception handling)

Custom error JSON

```
{
    "status" : 404,
   "message" : "Student id not found-100",
      "timeStamp" : 4565767865
}
```

Spring way of exception handling:

Add a exception handler method : decorated with @ExceptionHandler
#any name
#shall have parameter of an appropriate exception class
#ResponseEntity : wrapper for custom HTTP response JSON object
HTTP status code, HTTP header , Response body

Global exception handler : a class to handle exception from all methods of all controllers

Persistant programming (DB)

Spring way of interacting with DB

#standard : spring-jdbc(api)

#built on top of core jdbc

#abstract out the basic low level formalities

(Template classes)

JdbcTemplate

NamedParameterJdbcTemplate

SimpleJdbcTemplate

SimpleJdbcInsert and SimpleJdbcCall

translates the jdbc exception into org.springframework.dao (exception)

thread-safe

#config go into spring config files and inject as bean in dao classes

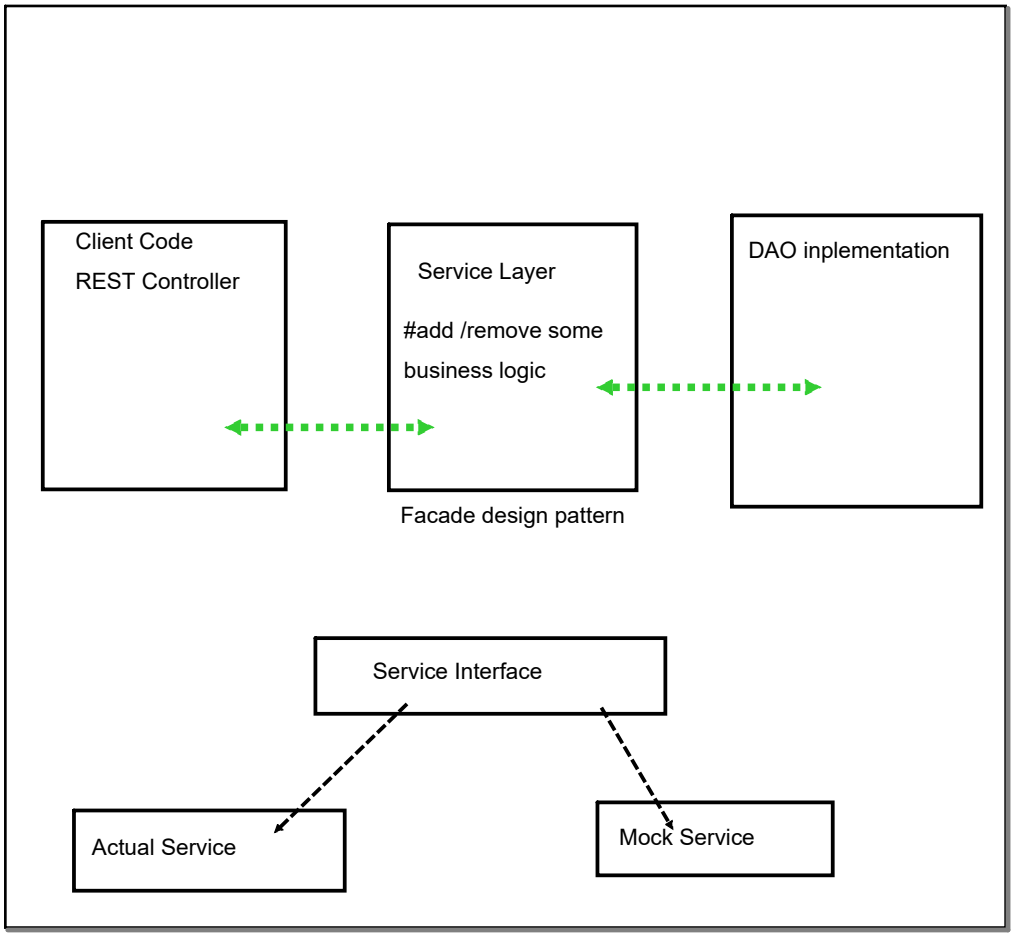
REST CRUD implementation using JdbcTemplate backed up by MySQL

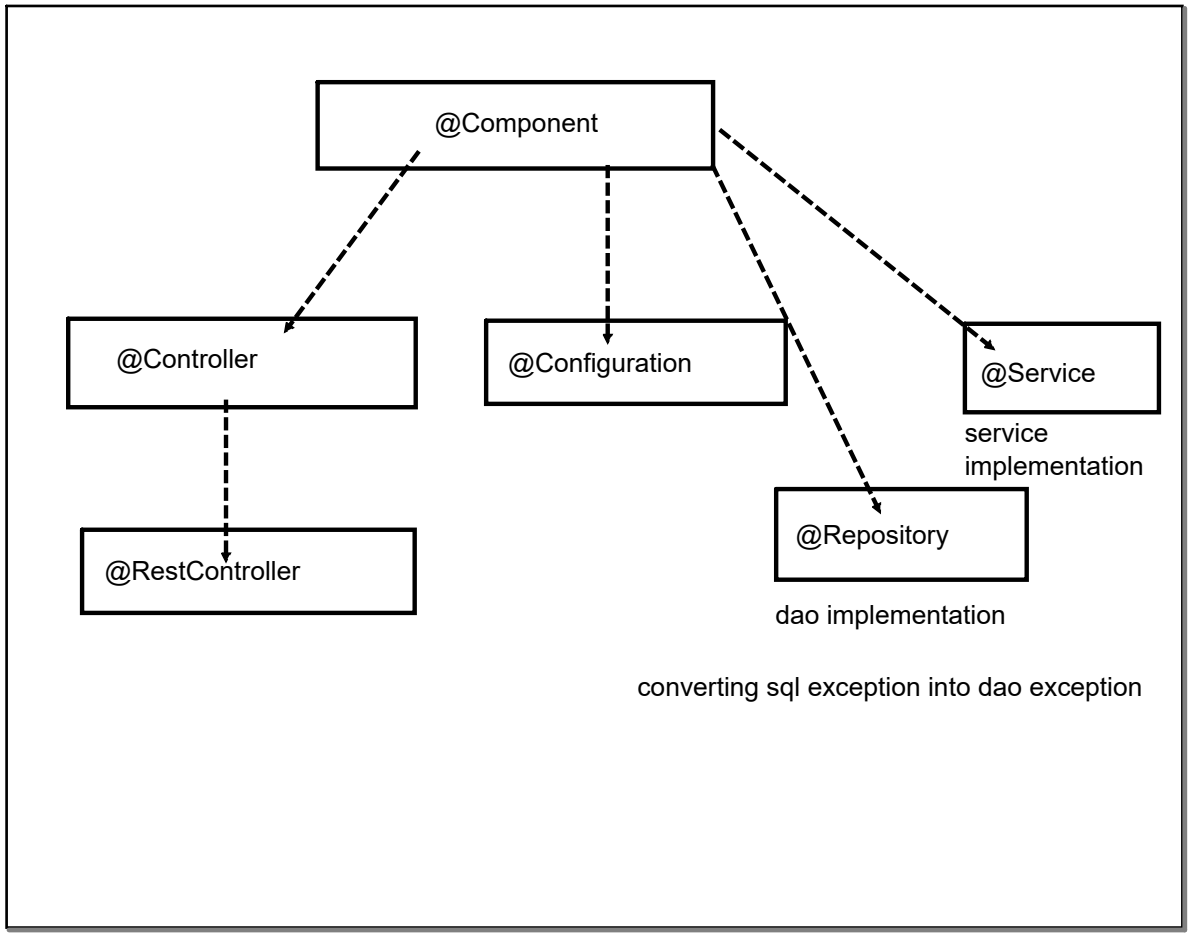
Dependencies
#spring-jdbc
#mysql-connector-java

Config the jdbc template:
#datasource (inside spring config file(

dao classes:
facade design style

MySQL env to create a database
initialize the config values for database
#maintain config literal values in properties
#inject them in config file...


Client Code
REST Controller
Service Layer
#add /remove some
business logic
DAO inplementation
Facade design pattern
Service Interface
Actual Service
Mock Service


@Component
@Controller
@Configuration
@Service
service
implementation
@Repository
@RestController
dao implementation
converting sql exception into dao exception

ORM : Hibernate for DB interaction...

Spring -security