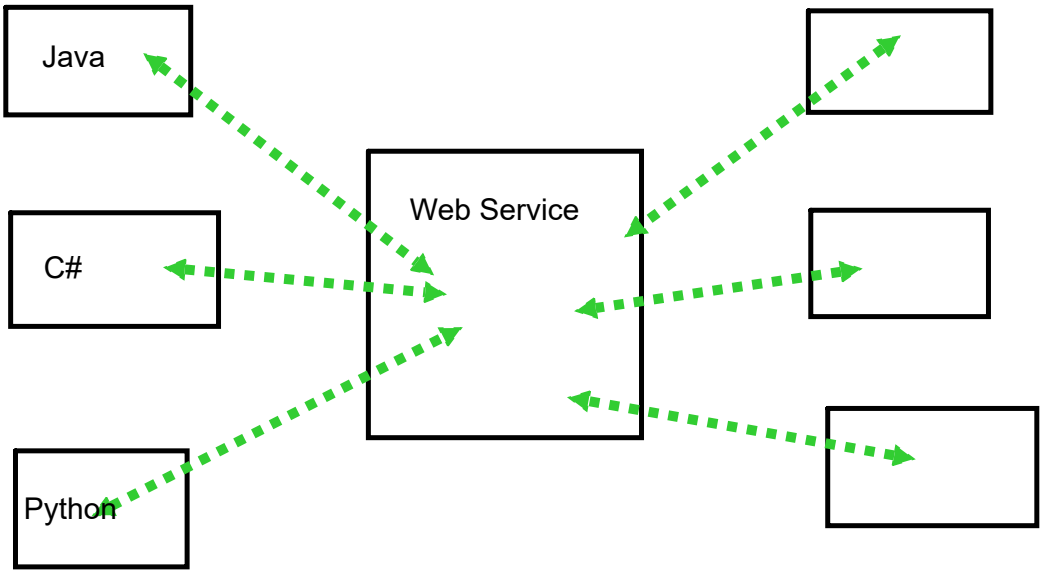
REST API

Web Service : shareable web service over web, platform independent



JCP : java community program

#protocol package

Web Service Component

SOAP : Simple Object Access protocol

WS : Object

XML based

UDDI : Universal Description, Discovery and Integration

is a Directory of web service containing WSDL Documents

WSDL : WEb Service Description Language

XML Document : details about WS

method name

param

return

how to access

SOA : SERVICE ORIENTED ARCHITECTURE

SOAP WS

Resource intensive (slow)

extra baggage (WSDL document)

extended client base : mobile/embedded system

SOAP has its own security framework

REST WS

Not a protocol (flexible)

architectural style

capable of using diff protocols

supports multiple data formats

usage : WS exposed as URI (not as method)

Security : depends on protocol or underlying framework

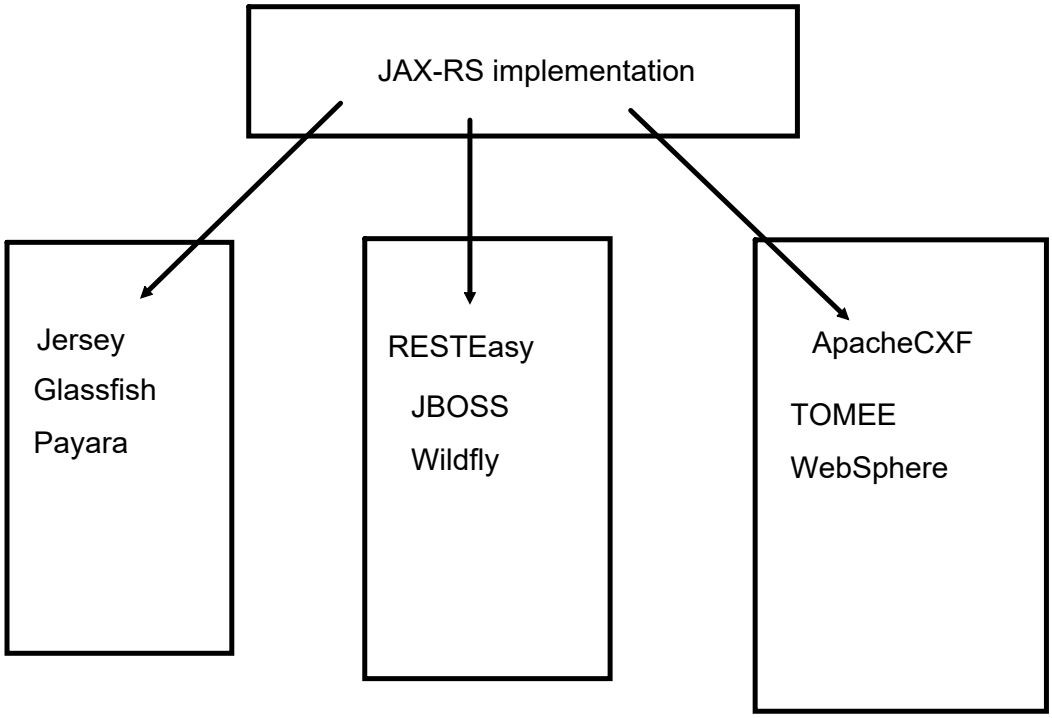
REST : Stateless :

Uniform Interface Contract : used HTTP key verbs (HTTP method)

Request :

1. URI based
2. service type (POST, GET, PUT, DELETE....)

Java API support for REST (JAX-RS)



Spring has its own REST API : web-mvc framework

not a JAX-RS implementation
#alternate to it

keep it in sync with existing Spring feature

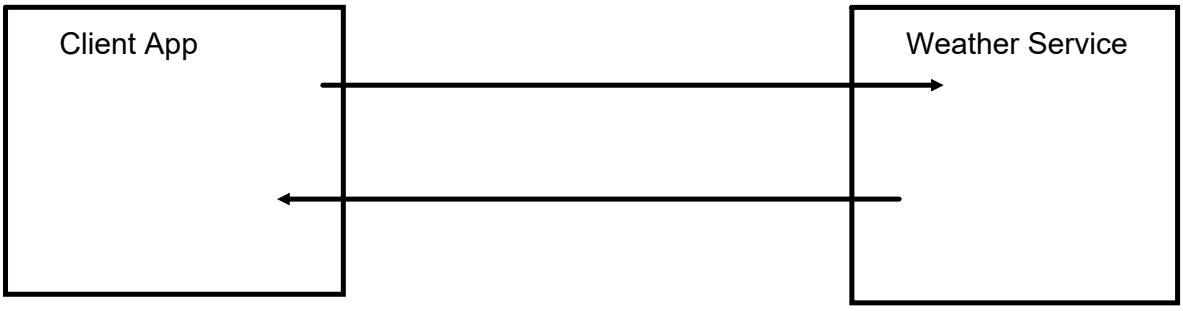
Jersey/REST Easy : servlet filter
ApacheCXF :
JAX-RS / JAX-WS

Spring
@RequestMapping(attributes... : method/produces/consumes)

JAX-RS
@Path
@POST
@PathParam
@QueryParam
@Produce
@Consume

REST Service

Use Case



need to provide weather report

need to fetch data from external source

1. How to connect
2. What language
3. What is data format

REST :

1. make rest call over HTTP using the URI (REST endpoint) and HTTP key verbs
2. REST is language independent
3. multiple data format (JSON format popular)

JSON : Javascript Object Notation

simple syntax to define JSON object

```
{
"id" : 4,
"name" : "First",
"email : "first@mail.com"
}
```

Curly braces to define Object

Object member key-value

Support multiple data forms

Number : no quotes

String : double quotes

Boolean : true/false

Array

Nested JSON Object

null

JSON as medium of data trans: Req/resp

need to interconvert JSON to Traget Object (POJO)

Java-JSON data binding : process if converting

Mapping

Serialization/DEserialization

Marshalling/Unmarshalling

#Jackson Project (special project : JSON databinding API)

com.fasterxml.jackson.databind

interconversion is based purely on field names (must match)

Jackson API uses getter/setter methods

java class must have appropriate get/set

does not maps directly to fields

Maven Dependency scope:

visibility of project in relation to life cycle

compile : default scope : needed for build,test, run

provided : needed for build and test (need for run as well, but not to exported), to be provided by runtime env

runtime: not needed for build, but needs to be part of classpath for test, running

test : not needed for build and run : needed to compile and run the unit test

system : similar to provided : not retrieved for remote loacation , to be fetched from project subdirectory

<scope>system</scope>

<systemPath> ${basedir}\... </systemPath>

import : dependency is to be replaced by effective list of dependencies from other pom artifact id

```
<groupId>other.pom.group.id</groupId>
<artifactId>other-pom-artifact-id</artifactId>
<scope>import</scope>
<type>pom</type>
```

```
<section>
    <dependency1>
    <dependency1>
    <dependency1>
</section>
```

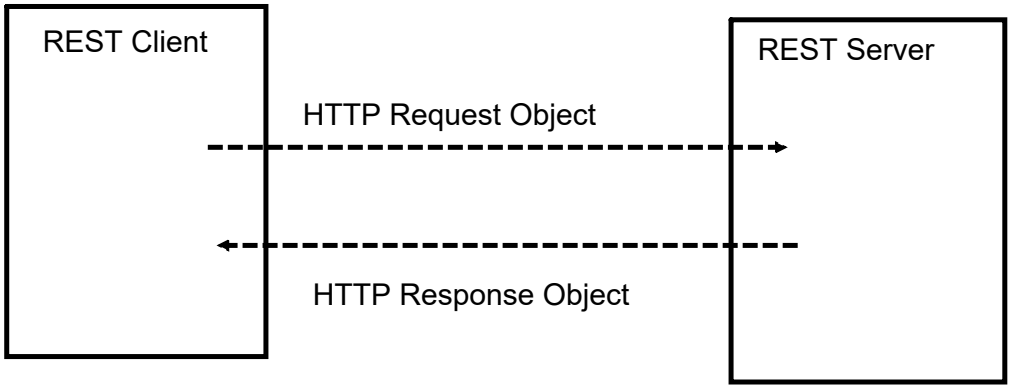
Spring uses jackson project behind the scene

REST over HTTP

Client intention are described by HTTP Verb key

POST

interaction between REST client and SERVER take place through Object



REquest

Request Line : HTTP Command (POST/PUT)

Header Variable : req metadata (added by client container)

Message Body : contents of Message

Response Object

Response Line : server protocol , status code

Header Variable : response metadata
(encoding/ MIME type)

Message Body : content

100 - 199 : Informational

200-299 : Success

300-399 : Redirection

400-499 : Client Error

500-599 : Server Error

REST service that exposes an URI (REST endpoint) working on HTTP GET request and responding a single string

Create Controller (REST Controller)

@RestController (inherited from @Controller)

auto deals with JSON data

Student Entity : need to work with student records as REST Services
implementing the exception situations in Spring Way

#Handle the exception and return error as JSON object

eg:

```
{
 "status" : 404,
 "message" : "student id not found",
 "timestamp" : 4546457
}
```

Create a custom exception class : StudentNotFoundException

specific handler : priority

general handler (Exception / RuntimeException)

create a custom error response class

return a custom message body

actually return a custom response object

ResponseEntity is a wrapper around HTTP Response Object

status code

header variable

message body

==Global Exception Handler

common handler for all controllers

#reusability

#Centralize exception handling

#uniform exception handling approach

priority

try-catch

exception handler method

global exception handler

Create a class (container) : decorate with @ControllerAdvice (inherited from @Component)

==> implements Spring AOP

==> add exception handler inside it

Spring way of DB interaction

Full CRUD REST API

Traditional JDBC

ALT : spring-jdbc api (built on top of Traditional JDBC)

REST : endpoints

REST standards

(uniform endpoint : diff them using HTTP method)

1. Get a list of students (api/students : GET)
2. Get a single student (api/students/{studentId} : GET)
3. Add a new students (api/students : POST)
4. Update student (api/students : PUT)
5. Delete student (api/students/{studentId} : DELETE)

spring-jdbc :

reduce the boiler-plate code

low level activity

1. define the connection param (configure the datasource)
2. register the SQL query template

spring-jdbc classes provide template based classes

JdbcTemplate (generalized/popular)

NamedParameterJdbcTemplate

SimpleJdbcTemplate

SimpleJdbcCall

JdbcTemplate:

execute queries (query templates)

iterate over ResultSet

retrieve value

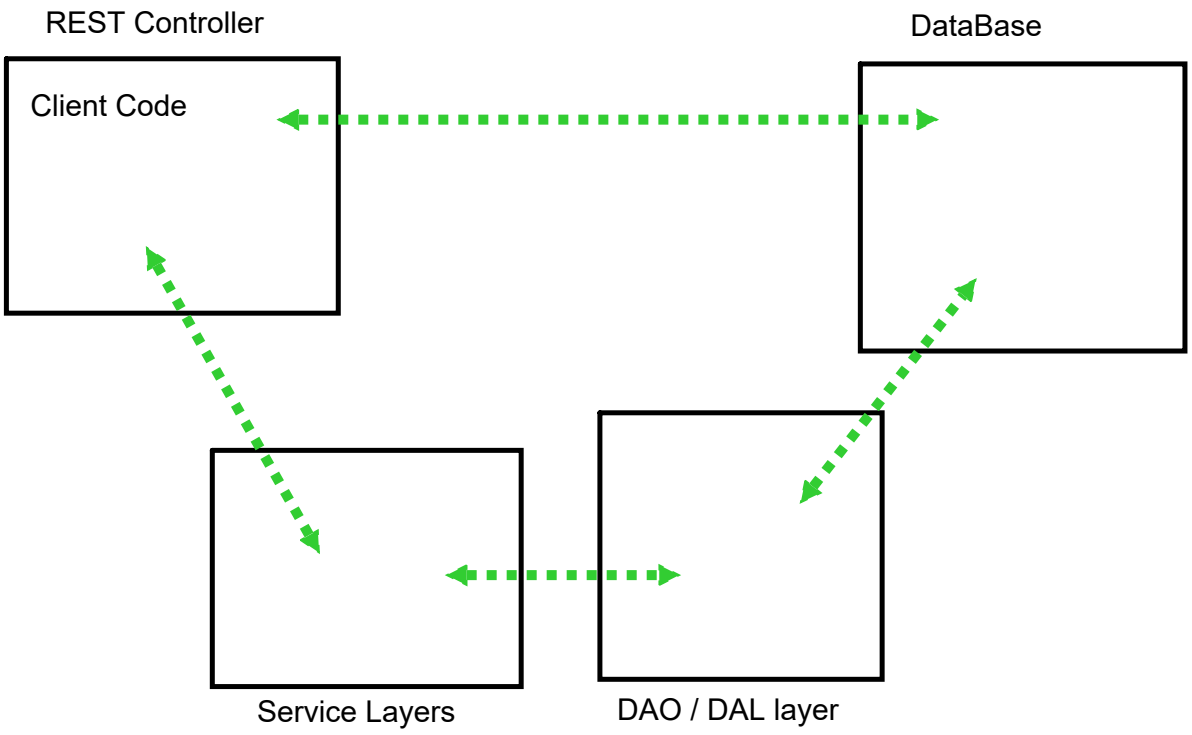
call procedures

catches traditional sql exceptions and translates them to exceptions defined in org.springframework.dao

Thread Safe Classes

STEPS:

1. maven web project
2. add new dependency (spring-jdbc / mysql-connector)
3. config the DataSource
4. develop DB implementation (DAO implementation)



Loosly coupled layers

- DAO : flexible JDBCTemplate | Hibernate | MOCK (Testing)
- Service : flexible Simple | Logging | MOCK | Secure

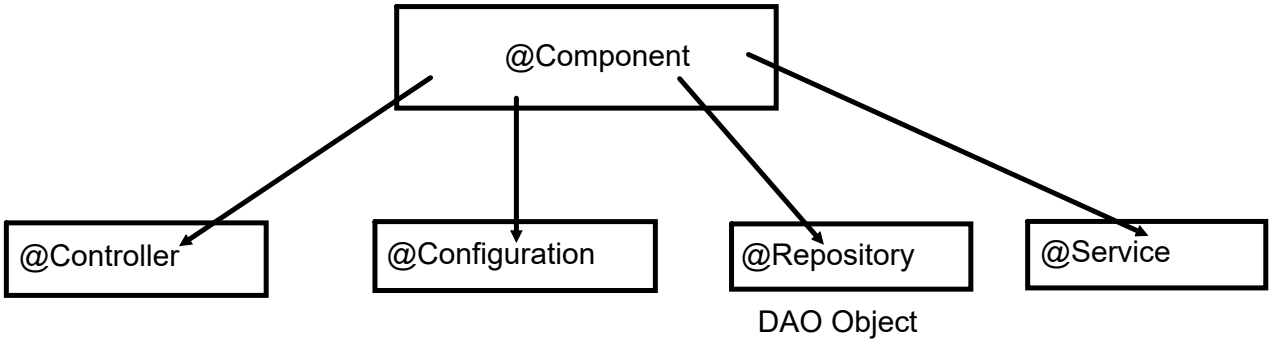
Steps:

1. Add Dependency
2. config the datasource

   # define a properties file to outsource connection param
3. Add DAO layer

   # add an interface (loose coupling)

   # add implementation classes (JDBCTemplate)



@ Repository :help us to converting the SQL Exceptions into springframework.dao exception

@Service : nothing exclusive right now (future prospects)

Properties file

```
# created under resources
# providing ref in config file
# retrieving values from file
```