

# Making Java easy to learn

Java Technology and Beyond



You are here [▶](#) [Home](#) > [java](#) >

## How To Develop REST CRUD API Using Spring Boot ?

java   Spring Boot   Spring Boot REST   by [devs5003](#) - October 15, 2020   11

It will not be an overstatement if I say that we can't develop an enterprise application without using webservises as an integration layer. Generally, we develop webservises in the form of either a producer or a consumer or both. However Producer is very important for us because we develop it in Java only. Also, we have full control of database interaction logic implementation with us. Now you might have interpreted the significance of our article 'How to develop REST CRUD API using Spring Boot?'. On the other hand, Consumer can be Angular Application, ReactJS Application, Android Device, iOS Device and many others or even our favorite java based RestTemplate(Spring Boot REST Client).

Our focus in this article is on developing producer API(REST API) using Spring Boot. Now let's discuss on 'How to develop REST CRUD API using Spring Boot ?' without leaving any theoretical concept which are very essential to know.

### Developing REST API Using Spring Boot

```
@RestController
@RequestMapping("/api")
public class InvoiceRestController { }

@PostMapping("/invoices")
public ResponseEntity<String> saveInvoice(
    @RequestBody Invoice inv)
{ ... }

@GetMapping("/invoices")
public ResponseEntity<?> getAllInvoices()
{ ... }

@GetMapping("/invoices/{id}")
public ResponseEntity<?> getOneInvoice(
    @PathVariable Long id)
{ ... }

@PatchMapping("/invoices/{id}/{number}")
public ResponseEntity<String> updateInvoiceNumberById(
    @PathVariable Long id,
    @PathVariable String number)
{ ... }
```

```
@PostMapping("/invoices")
@GetMapping("/invoices")
@GetMapping("/invoices/{id}")
@DeleteMapping("/invoices/{id}")
@PutMapping("/invoices/{id}")
@PatchMapping("/invoices/{id}/{number}")

@DeleteMapping("/invoices/{id}")
public ResponseEntity<String> deleteInvoice(
    @PathVariable Long id) { ... }

@PutMapping("/invoices/{id}")
public ResponseEntity<String> updateInvoice(
    @PathVariable Long id,
    @RequestBody Invoice invoice)
{ ... }
```

@javatechonline.com

### Table of Contents (Click on links below to navigate) [\[hide\]](#)

- 1 What will you learn from this article?
- 2 What is REST?
- 3 How will you define a development of REST API ?
- 4 What is RestController ?
- 5 Software used in this Project ?
- 6 Coding Steps
  - 6.1 Step#1 : Create Project in STS
  - 6.2 Step#2 : Create Database
  - 6.3 Step#3 : Update application.properties and Write Classes & methods(REST API)
- 7 How to run the application ?
- 8 How to test the application ?
  - 8.1 ♠ Testing saveInvoice() method : [\[http://localhost:8080/api/invoices\]](http://localhost:8080/api/invoices)
  - 8.2 ♦ Testing getAllInvoices() method : [\[http://localhost:8080/api/invoices\]](http://localhost:8080/api/invoices)
  - 8.3 ♠ Testing getOneInvoice() method : [\[http://localhost:8080/api/invoices/{id}\]](http://localhost:8080/api/invoices/{id})
  - 8.4 ♦ Testing updateInvoice() method : [\[http://localhost:8080/api/invoices/{id}\]](http://localhost:8080/api/invoices/{id})
  - 8.5 ♠ Testing deleteInvoice() method : [\[http://localhost:8080/api/invoices/{id}\]](http://localhost:8080/api/invoices/{id})
  - 8.6 ♦ Testing updateInvoiceNumberById() method : [\[http://localhost:8080/api/invoices/{id}/{number}\]](http://localhost:8080/api/invoices/{id}/{number})
- 9 Can we use this REST API in the real project ?
- 10 Summary

## What will you learn from this article?

Once you complete going through all points of this article, You will be able to answer :

- 1) What is REST and REST API in the context of Web services?



TOP

- 2) How to create a Spring Boot REST application that incorporates industry level project design ?
- 3) How to develop CRUD (Create, Retrieve, Update, Delete) operations that can be used by any other even non-java application ?
- 4) How to write bug free CRUD operations, including exceptions & exception handlers?
- 5) Equally important, How to use annotations @RestController, @RequestMapping, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping, @Modifying, @Query, @Transactional, @RestControllerAdvice, @ExceptionHandler, @ControllerAdvice, @ResponseBody, @RequestBody, @PathVariable, @Data, @NoArgsConstructor, @AllArgsConstructor, @Entity, @Component, @Service, @Autowired ?
- 6) How to work with Spring Boot Data JPA repository interface?
- 7) How to write modular & reusable code?
- 8) Moreover, How to implement dynamic code with minimal changes, keeping future change requests in mind?
- 9) How to develop an integration layer to get interoperability ?
- 10) How to test REST Application by supplying & receiving JSON data ?
- 11) Last but not the least you will learn “How to develop REST CRUD API using Spring Boot ?”

## What is REST?

REST stands for **R**epresentational **S**tate **T**ransfer. It transfers state(data) in global format(representational) between two different applications running on different servers. In the process of data transfer, who requests data is called Consumer/Client application and who provides data is called a producer application. REST is an architectural style that follows a set of rules to create webservices. Webservices provide reusable data to multiple applications and interoperability between them on the internet. Web services that conform to the REST architectural style, called RESTful Web services.

## How will you define a development of REST API ?

Developing a Rest API is nothing but creating classes & methods in a specific architectural style so that data can be reused between interoperable applications. More or less we create RestController and respective CRUD operations in the process of development of the REST API.

## What is RestController ?

In Spring Boot REST programming RestController is a mandatory class which acts as a front controller. It contains several methods that return Body and Status as a ResponseEntity object. Body refers to data in form of String, Object, Collections etc. Whereas Status refers to the HttpResponseMessage Status (200, 404, 405, 500 etc.).

## Software used in this Project ?

-STS (Spring Tool Suite) : Version-> 4.7.1.RELEASE  
-MySQL Database : Version ->8.0.19 MySQL Community Server  
-JDK8 or later versions (Extremely tested on JDK8, JDK9 and JDK14)

## Coding Steps

### Step#1 : Create Project in STS

If you are new to Spring Boot, visit [Internal Link](#) to create a sample project in spring boot. While creating project in STS add 4 starters 'MySQL Driver', 'Spring Data JPA', 'Spring Web' and 'Lombok'. You can also add 'Spring Boot DevTools' optionally. If you are new to 'Lombok', kindly visit '[How to configure Lombok](#)' and to know all about it in detail.

### Step#2 : Create Database

You can use database software of your own choice and create a Database accordingly. We have used MySQL database software. Login to your MySQL and execute a Database creation query. In our case DB Name is 'REST\_INVOICE', so our query will be 'CREATE DATABASE REST\_INVOICE'.

### Step#3 : Update application.properties and Write Classes & methods(REST API)



We are considering 'Invoice' as a model to develop the REST API. Invoice will have many fields such as invoiceNumber, invoiceName, invoiceAmount etc.. Below Table will show the list of classes & other files used in the project accordingly.

Package/Location	Class/Interface/file name	Purpose
src/main/resources	application.properties	properties file to declare common properties in the project
com.dev.invoice.rest.entity	Invoice.java	Model/Entity class with Database table mapping
com.dev.invoice.rest.repo	InvoiceRepository.java	Repository Interface which extends JpaRepository interface
com.dev.invoice.rest.service	InvoiceService.java	Service interface with Database related methods
com.dev.invoice.rest.service.impl	InvoiceServiceImpl.java	Service class contains implementations of methods declared in Service interface for Database related operations
com.dev.invoice.rest.entity	ErrorType.java	Helper class used in InvoiceErrorHandler.java
com.dev.invoice.rest.exception	InvoiceNotFoundException.java	To define custom Exception if any Invoice not found
com.dev.invoice.rest.exception.handler	InvoiceErrorHandler.java	To handle the error In case InvoiceNotFoundException is thrown from any controller method
com.dev.invoice.rest.util	InvoiceUtil.java	Utility class to maximize code reusability & minimize code redundancy.
com.dev.invoice.rest.controller	InvoiceRestController.java	Rest Controller the backbone of REST API, accepts all requests coming from client and handover to respective method for processing.

application.properties

```
# DB Connection Properties
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/REST_INVOICE
spring.datasource.username=root
spring.datasource.password=devs
6
# JPA Properties
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

Invoice.java

```

1 package com.dev.invoice.rest.entity;
2
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.Id;
6
7 import lombok.AllArgsConstructor;
8 import lombok.Data;
9 import lombok.NoArgsConstructor;
10
11 @Data
12 @AllArgsConstructor
13 @NoArgsConstructor
14 @Entity
15 public class Invoice {
16
17     @Id
18     @GeneratedValue
19     private Long id;
20     private String name;
21     private Double amount;
22     private Double finalAmount;
23     private String number;
24     private String receivedDate;
25     private String type;
26     private String vendor;
27     private String comments;
28 }

```

#### InvoiceRepository.java

```

1 package com.dev.invoice.rest.repo;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.data.jpa.repository.Modifying;
5 import org.springframework.data.jpa.repository.Query;
6
7 import com.dev.invoice.rest.entity.Invoice;
8
9 public interface InvoiceRepository extends JpaRepository<Invoice, Long>{
10
11     // Update is Non-Select Operation, so @Modifying is used
12     @Modifying
13     @Query("UPDATE Invoice SET number=:number WHERE id=:id")
14     Integer updateInvoiceNumberById(String number,Long id);
15 }

```

#### Service Interface IInvoiceService.java

```

1 package com.dev.invoice.rest.service;
2
3 import java.util.List;
4
5 import com.dev.invoice.rest.entity.Invoice;
6
7 public interface IInvoiceService {
8
9     /**
10      * Takes Invoice Object as input and returns PK generated
11      */
12     Long saveInvoice(Invoice inv);
13
14     /**
15      * Takes existing Invoice data as input and updates values
16      */
17     void updateInvoice(Invoice e);
18
19     /**
20      * Takes PK(ID) as input and deletes Invoice Object data
21      */
22     void deleteInvoice(Long id);
23
24     /**
25      * Takes id as input and returns one row as one object
26      */
27     Invoice getOneInvoice(Long id); //used in RestController
28
29     /**
30      * select all rows and provides result as a List<Invoice>

```



```

31     */
32     List<Invoice> getAllInvoices();
33
34     /**
35     * Takes Id as input, checks if record exists returns true, else false
36     */
37     */
38     boolean isInvoiceExist(Long id);
39
40     /**
41     * Takes 2 fields as input, updates Invoice data as provided where clause
42     * like 'UPDATE Invoice SET number=:number WHERE id=:id'
43     */
44     Integer updateInvoiceNumberById(String number, Long id);
45 }

```

## InvoiceServiceImpl.java

```

package com.dev.invoice.rest.service.impl;
2
import java.util.List;
import java.util.Optional;
5
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
8
9
import com.dev.invoice.rest.entity.Invoice;
import com.dev.invoice.rest.exception.InvoiceNotFoundException;
import com.dev.invoice.rest.repo.InvoiceRepository;
import com.dev.invoice.rest.service.IInvoiceService;
import com.dev.invoice.rest.util.InvoiceUtil;
15
@Service
16
public class InvoiceServiceImpl implements IInvoiceService {
18
19     @Autowired
20     private InvoiceRepository repo;
21
22     @Autowired
23     private InvoiceUtil util;
24
25     @Override
26     public Long saveInvoice(Invoice inv) {
27         util.CalculateFinalAmountIncludingGST(inv);
28         Long id = repo.save(inv).getId();
29         return id;
30     }
31
32     @Override
33     public void updateInvoice(Invoice inv) {
34         util.CalculateFinalAmountIncludingGST(inv);
35         repo.save(inv);
36     }
37
38     @Override
39     public void deleteInvoice(Long id) {
40         Invoice inv = getOneInvoice(id);
41         repo.delete(inv);
42     }
43
44     public Optional<Invoice> getSingleInvoice(Long Id) {
45         return repo.findById(Id);
46     }
47
48     @Override
49     public Invoice getOneInvoice(Long id) {
50
51         Invoice inv = repo.findById(id)
52             .orElseThrow(() -> new InvoiceNotFoundException(
53                 new StringBuffer().append("Product ").append(id)
54                     .append("' not exist")
55                     .toString());
56
57         return inv;
58     }
59
60
61     @Override
62     public List<Invoice> getAllInvoices() {
63         List<Invoice> list = repo.findAll();
64         //JDK 1.8 List Sort (using Comparator)
65         list.sort((ob1, ob2) -> ob1.getId().intValue() - ob2.getId().intValue());
66         //list.sort((ob1, ob2) -> ob1.getAmount().compareTo(ob2.getAmount())); //ASC
67         //list.sort((ob1, ob2) -> ob2.getAmount().compareTo(ob1.getAmount())); // DESC
68
69         return list;

```



TOP

```

69     }
70
71     @Override
72     public boolean isInvoiceExist(Long id) {
73
74         return repo.existsById(id);
75     }
76
77     @Override
78     @Transactional
79     public Integer updateInvoiceNumberById(
80         String number, Long id)
81     {
82         if(!repo.existsById(id)) {
83             throw new InvoiceNotFoundException(
84                 new StringBuffer()
85                     .append("Invoice ")
86                     .append(id)
87                     .append("' not exist")
88                     .toString());
89         }
90         return repo.updateInvoiceNumberById(number, id);
91     }
92
93 }

```

#### ErrorType.java

```

package com.dev.invoice.rest.entity;
2
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;
6
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ErrorType {
11
12     private String time;
13     private String status;
14     private String message;
15
16 }

```

#### InvoiceNotFoundException.java

```

package com.dev.invoice.rest.exception;
2
//Custom Exception
public class InvoiceNotFoundException extends RuntimeException{
5
6     private static final long serialVersionUID = 1L;
7
8     public InvoiceNotFoundException() {
9         super();
10    }
11
12    public InvoiceNotFoundException(String message) {
13        super(message);
14    }
15
16 }

```

#### InvoiceErrorHandler.java

```

package com.dev.invoice.rest.exception.handler;
2
import java.util.Date;
3
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
8

```



↑  
TOP

```

9
10 import com.dev.invoice.rest.entity.ErrorType;
11 import com.dev.invoice.rest.exception.InvoiceNotFoundException;
12
13 //ControllerAdvice
14 @ControllerAdvice
15 public class InvoiceErrorHandler {
16     /**
17      * In case of InvoiceNotFoundException is thrown
18      * from any controller method, this logic gets
19      * executed which behaves like re-usable and
20      * clear code (Code Modularity)
21      * @param nfe
22      * @return ResponseEntity
23      */
24     //ResponseBody
25     @ExceptionHandler({InvoiceNotFoundException.class})
26     public ResponseEntity<ErrorType> handleNotFound(InvoiceNotFoundException nfe){
27
28         return new ResponseEntity<ErrorType>(
29             new ErrorType(
30                 new Date(System.currentTimeMillis()).toString(),
31                 "404- NOT FOUND",
32                 nfe.getMessage()),
33             HttpStatus.</yoastmark>NOT_FOUND);
34     }
35 }

```

## Utility class InvoiceUtil.java

```

1 package com.dev.invoice.rest.util;
2
3 import org.springframework.stereotype.Component;
4
5 import com.dev.invoice.rest.entity.Invoice;
6
7 @Component
8 public class InvoiceUtil {
9
10     public Invoice CalculateFinalAmountIncludingGST (Invoice inv) {
11         var amount=inv.getAmount();
12         var gst= 0.1;
13         var finalAmount=amount+(amount*gst);
14         inv.setFinalAmount(finalAmount);
15         return inv;
16     }
17
18     public void copyNonNullValues(Invoice req, Invoice db) {
19
20         if(req.getName() !=null) {
21             db.setName(req.getName());
22         }
23
24         if(req.getAmount() !=null) {
25             db.setAmount(req.getAmount());
26         }
27
28         if(req.getNumber() !=null) {
29             db.setNumber(req.getNumber());
30         }
31
32         if(req.getReceivedDate() !=null) {
33             db.setReceivedDate(req.getReceivedDate());
34         }
35
36         if(req.getType() !=null) {
37             db.setType(req.getType());
38         }
39
40         if(req.getVendor() !=null) {
41             db.setVendor(req.getVendor());
42         }
43
44         if(req.getComments() !=null) {
45             db.setComments(req.getComments());
46         }
47     }
48 }

```

## InvoiceRestController.java



```
package com.dev.invoice.rest.controller;
2
import java.util.List;
4
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
18
import com.dev.invoice.rest.entity.Invoice;
import com.dev.invoice.rest.exception.InvoiceNotFoundException;
import com.dev.invoice.rest.service.IInvoiceService;
import com.dev.invoice.rest.util.InvoiceUtil;
23
@RestController
@RequestMapping("/api")
// @CrossOrigin(origins = "http://localhost:4200") //Required in case of Angular Client
public class InvoiceRestController {
28
    @Autowired
    private IInvoiceService service;
31
    @Autowired
    private InvoiceUtil util;
34
    /**
    * Takes Invoice Object as input and returns save Status as ResponseEntity<String>
    */
    @PostMapping("/invoices")
    public ResponseEntity<String> saveInvoice(@RequestBody Invoice inv){
    40        ResponseEntity<String> resp = null;
    41        try{
    42            Long id = service.saveInvoice(inv);
    43            resp = new ResponseEntity<String>("Invoice '"+id+"' created",HttpStatus.CREATED); //201-created
    44        } catch (Exception e) {
    45            e.printStackTrace();
    46            resp = new ResponseEntity<String>("Unable to save Invoice",
    47                HttpStatus.INTERNAL_SERVER_ERROR); //500-Internal Server Error
    48        }
    49        return resp;
    50    }
    51
    52
    53
    54    /**
    55    * To retrieve all Invoices, returns data retrieval Status as ResponseEntity<?>
    56    */
    57    @GetMapping("/invoices")
    58    public ResponseEntity<?> getAllInvoices() {
    59        ResponseEntity<?> resp=null;
    60        try {
    61            List<Invoice> list= service.getAllInvoices();
    62            resp= new ResponseEntity<List<Invoice>>(list,HttpStatus.OK);
    63        } catch (Exception e) {
    64            e.printStackTrace();
    65            resp = new ResponseEntity<String>("Unable to get Invoice",
    66                HttpStatus.INTERNAL_SERVER_ERROR);
    67        }
    68        return resp;
    69    }
    70
    71
    72    /**
    73    * To retrieve one Invoice by providing id, returns Invoice object & Status as ResponseEntity<?>
    74    */
    75    @GetMapping("/invoices/{id}")
    76    public ResponseEntity<?> getOneInvoice(@PathVariable Long id){
    77        ResponseEntity<?> resp= null;
    78        try {
    79            Invoice inv= service.getOneInvoice(id);
    80            resp= new ResponseEntity<Invoice>(inv,HttpStatus.OK);
    81        }catch (InvoiceNotFoundException nfe) {
    82            throw nfe;
    83        }catch (Exception e) {
    84            e.printStackTrace();
    85            resp = new ResponseEntity<String>("Unable to find Invoice",
    86                HttpStatus.INTERNAL_SERVER_ERROR);
    87        }
    88        return resp;
    89    }
    90
    91
    92
    93
    94
    95
    96
    97
    98
    99
    100
    101
    102
    103
    104
    105
    106
    107
    108
    109
    110
    111
    112
    113
    114
    115
    116
    117
    118
    119
    120
    121
    122
    123
    124
    125
    126
    127
    128
    129
    130
    131
    132
    133
    134
    135
    136
    137
    138
    139
    140
    141
    142
    143
    144
    145
    146
    147
    148
    149
    150
    151
    152
    153
    154
    155
    156
    157
    158
    159
    160
    161
    162
    163
    164
    165
    166
    167
    168
    169
    170
    171
    172
    173
    174
    175
    176
    177
    178
    179
    180
    181
    182
    183
    184
    185
    186
    187
    188
    189
    190
    191
    192
    193
    194
    195
    196
    197
    198
    199
    200
    201
    202
    203
    204
    205
    206
    207
    208
    209
    210
    211
    212
    213
    214
    215
    216
    217
    218
    219
    220
    221
    222
    223
    224
    225
    226
    227
    228
    229
    230
    231
    232
    233
    234
    235
    236
    237
    238
    239
    240
    241
    242
    243
    244
    245
    246
    247
    248
    249
    250
    251
    252
    253
    254
    255
    256
    257
    258
    259
    260
    261
    262
    263
    264
    265
    266
    267
    268
    269
    270
    271
    272
    273
    274
    275
    276
    277
    278
    279
    280
    281
    282
    283
    284
    285
    286
    287
    288
    289
    290
    291
    292
    293
    294
    295
    296
    297
    298
    299
    300
    301
    302
    303
    304
    305
    306
    307
    308
    309
    310
    311
    312
    313
    314
    315
    316
    317
    318
    319
    320
    321
    322
    323
    324
    325
    326
    327
    328
    329
    330
    331
    332
    333
    334
    335
    336
    337
    338
    339
    340
    341
    342
    343
    344
    345
    346
    347
    348
    349
    350
    351
    352
    353
    354
    355
    356
    357
    358
    359
    360
    361
    362
    363
    364
    365
    366
    367
    368
    369
    370
    371
    372
    373
    374
    375
    376
    377
    378
    379
    380
    381
    382
    383
    384
    385
    386
    387
    388
    389
    390
    391
    392
    393
    394
    395
    396
    397
    398
    399
    400
    401
    402
    403
    404
    405
    406
    407
    408
    409
    410
    411
    412
    413
    414
    415
    416
    417
    418
    419
    420
    421
    422
    423
    424
    425
    426
    427
    428
    429
    430
    431
    432
    433
    434
    435
    436
    437
    438
    439
    440
    441
    442
    443
    444
    445
    446
    447
    448
    449
    450
    451
    452
    453
    454
    455
    456
    457
    458
    459
    460
    461
    462
    463
    464
    465
    466
    467
    468
    469
    470
    471
    472
    473
    474
    475
    476
    477
    478
    479
    480
    481
    482
    483
    484
    485
    486
    487
    488
    489
    490
    491
    492
    493
    494
    495
    496
    497
    498
    499
    500
    501
    502
    503
    504
    505
    506
    507
    508
    509
    510
    511
    512
    513
    514
    515
    516
    517
    518
    519
    520
    521
    522
    523
    524
    525
    526
    527
    528
    529
    530
    531
    532
    533
    534
    535
    536
    537
    538
    539
    540
    541
    542
    543
    544
    545
    546
    547
    548
    549
    550
    551
    552
    553
    554
    555
    556
    557
    558
    559
    560
    561
    562
    563
    564
    565
    566
    567
    568
    569
    570
    571
    572
    573
    574
    575
    576
    577
    578
    579
    580
    581
    582
    583
    584
    585
    586
    587
    588
    589
    590
    591
    592
    593
    594
    595
    596
    597
    598
    599
    600
    601
    602
    603
    604
    605
    606
    607
    608
    609
    610
    611
    612
    613
    614
    615
    616
    617
    618
    619
    620
    621
    622
    623
    624
    625
    626
    627
    628
    629
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000
    1001
    1002
    1003
    1004
    1005
    1006
    1007
    1008
    1009
    1010
    1011
    1012
    1013
    1014
    1015
    1016
    1017
    1018
    1019
    1020
    1021
    1022
    1023
    1024
    1025
    1026
    1027
    1028
    1029
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1734
    1735
    1736
    1737
    1738
    1739
    1740
    1741
    1742
    1743
    1744
    1745
    1746
    1747
    1748
    1749
    1750
    1751
    1752
    1753
    1754
    1755
    1756
    1757
    1758
    1759
    1760
    1761
    1762
    1763
    1764
    1765
    1766
    1767
    1768
    1769
    1770
    1771
    1772
    1773
    1774
    1775
    1776
    1777
    1778
    1779
    1780
    1781
    1782
    1783
    1784
    1785
    1786
    1787
    1788
    1789
    1790
    1791
    1792
    1793
    1794
    1795
    1796
    1797
    1798
    1799
    1800
    1801
    1802
    1803
    1804
    1805
    1806
    1807
    1808
    1809
    1810
    1811
    1812
    1813
    1814
    1815
    1816
    1817
    1818
    1819
    1820
    1821
    1822
    1823
    1824
    1825
    1826
    1827
    1828
    1829
    1830
    1831
    1832
    1833
    1834
    1835
    1836
    1837
    18
```

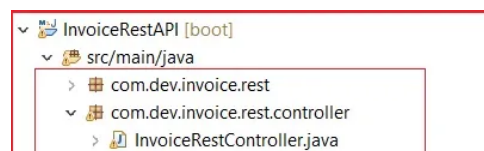


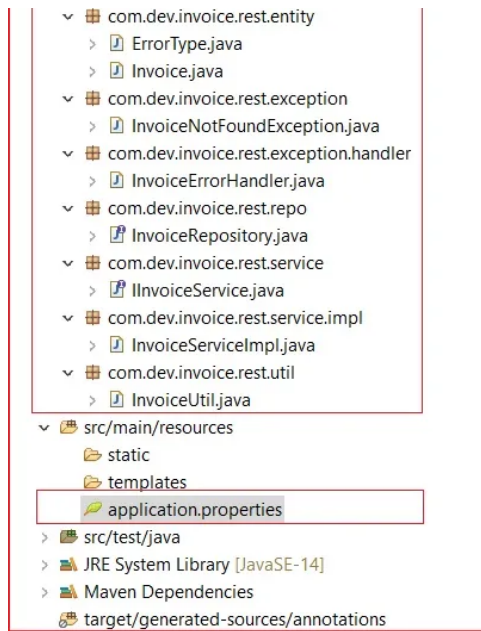
```

92  /**
93   * To delete one Invoice by providing id, returns Status as ResponseEntity<String>
94   */
95   @DeleteMapping("/invoices/{id}")
96   public ResponseEntity<String> deleteInvoice(@PathVariable Long id){
97
98       ResponseEntity<String> resp= null;
99       try {
100           service.deleteInvoice(id);
101           resp= new ResponseEntity<String> (
102               "Invoice '"+id+"' deleted",HttpStatus.OK);
103
104       } catch (InvoiceNotFoundException nfe) {
105           throw nfe;
106       } catch (Exception e) {
107           e.printStackTrace();
108           resp= new ResponseEntity<String> (
109               "Unable to delete Invoice", HttpStatus.INTERNAL_SERVER_ERROR);
110       }
111
112       return resp;
113   }
114
115   /**
116   * To modify one Invoice by providing id, updates Invoice object & returns Status as ResponseEntity<String>
117   */
118   @PutMapping("/invoices/{id}")
119   public ResponseEntity<String> updateInvoice(@PathVariable Long id, @RequestBody Invoice invoice){
120
121       ResponseEntity<String> resp = null;
122       try {
123           //db Object
124           Invoice inv= service.getOneInvoice(id);
125           //copy non-null values from request to Database object
126           util.copyNonNullValues(invoice, inv);
127           //finally update this object
128           service.updateInvoice(inv);
129           resp = new ResponseEntity<String> (
130               "Invoice '"+id+"' Updated",
131               HttpStatus.RESET_CONTENT); //205- Reset-Content(PUT)
132
133       } catch (InvoiceNotFoundException nfe) {
134           throw nfe; // re-throw exception to handler
135       } catch (Exception e) {
136           e.printStackTrace();
137           resp = new ResponseEntity<String> (
138               "Unable to Update Invoice",
139               HttpStatus.INTERNAL_SERVER_ERROR); //500-ISE
140       }
141       return resp;
142   }
143
144   /**
145   * To update one Invoice just like where clause condition, updates Invoice object & returns Status as ResponseEntity
146   */
147   @PatchMapping("/invoices/{id}/{number}")
148   public ResponseEntity<String> updateInvoiceNumberById(
149       @PathVariable Long id,
150       @PathVariable String number
151   )
152   {
153       ResponseEntity<String> resp = null;
154       try {
155           service.updateInvoiceNumberById(number, id);
156           resp = new ResponseEntity<String> (
157               "Invoice '"+number+"' Updated",
158               HttpStatus.PARTIAL_CONTENT); //206- Reset-Content(PUT)
159
160       } catch (InvoiceNotFoundException pne) {
161           throw pne; // re-throw exception to handler
162       } catch (Exception e) {
163           e.printStackTrace();
164           resp = new ResponseEntity<String> (
165               "Unable to Update Invoice",
166               HttpStatus.INTERNAL_SERVER_ERROR); //500-ISE
167       }
168       return resp;
169   }
170 }

```

Finally your project structure would look like below screen.





## How to run the application ?

Further to run the application for testing purpose, right click on Project and then select Run As >> Spring Boot App. You can also package it into a jar & run it accordingly. Additionally, to test the app you need to have a Client application/software.

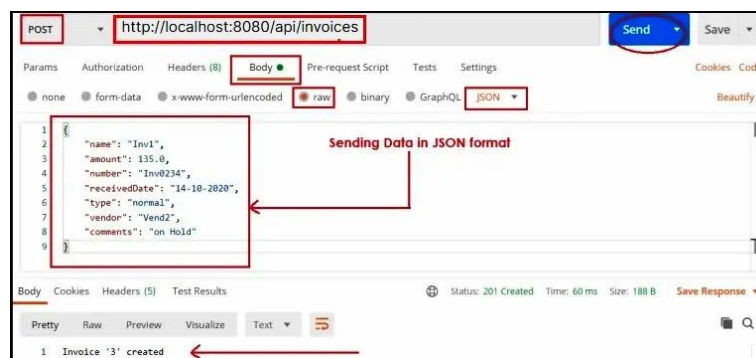
## How to test the application ?

As discussed in introduction part, there are multiple ways to test the REST application. At this point, we will suggest you to use the most popular tool 'POSTMAN'. You can also download it from [here](#).

### ♠ Testing saveInvoice() method : [http://localhost:8080/api/invoices]

Open Postman software, Select method 'POST' from dropdown, enter below URL, select 'Body' then click on 'raw', select 'JSON' from dropdown. All the above selections are highlighted in the below screenshot. Now Enter data in JSON format. If you want to know more about json, visit a separate article on '[how to write data in JSON format](#)'. Once you enter the JSON data, click on 'Send' button & check the successful message in the lower box.

http://localhost:8080/api/invoices



### ♦ Testing getAllInvoices() method : [http://localhost:8080/api/invoices]

Select method 'GET' from dropdown, then enter below URL. Now click on 'Send' button & check the list of all invoices as a JSON format in lower box.

http://localhost:8080/api/invoices

### ♠ Testing getOneInvoice() method : [http://localhost:8080/api/invoices/{id}]

{id} represents dynamic data. Suppose we want to retrieve Invoice whose id is 2. Select method 'GET' from dropdown, then enter below URL. Then click on 'Send' button & check the invoice with given id as a JSON format in lower box.



<http://localhost:8080/api/invoices/2>

### ♦ Testing updateInvoice() method : [<http://localhost:8080/api/invoices/{id}>]

Select method 'PUT' from dropdown, enter below URL, select 'Body' then click on 'raw', select 'JSON' from dropdown. Then Enter data in JSON format to be modified. Then click on 'Send' button and check the successful message in lower box.

<http://localhost:8080/api/invoices/2>

### ♠ Testing deleteInvoice() method : [<http://localhost:8080/api/invoices/{id}>]

Suppose you want to remove an Invoice where id is 2. Then use below pattern URL. Select method 'DELETE' from dropdown, then enter below URL. Then click on 'Send' button and check the successful message in lower box.

<http://localhost:8080/api/invoice/2>

### ♦ Testing updateInvoiceNumberById() method : [<http://localhost:8080/api/invoices/{id}/{number}>]

Suppose you want to modify name field of an Invoice where id is 1. Then use below pattern URL. Select method 'PATCH' from dropdown, enter below URL, select 'Body' then click on 'raw', select 'JSON' from dropdown. Then Enter data in JSON format to be modified. Finally click on 'Send' button & check the successful message in lower box.

<http://localhost:8080/api/invoices/1/Inv02345>

## Can we use this REST API in the real project ?

Of course. You have to change all occurrences of Entity Name as per your real project, then you may use it accordingly.

## Summary

Almost every REST application will have these operations we learnt in this article. In other words, no REST API can be developed without these operations. So, you have learnt the mandatory concepts of 'How to develop REST CRUD API using Spring Boot ?'. Furthermore, you can go through other article on [How to consume REST API using RestTemplate](#) using Spring Boot.

Me gusta

Tagged [@ControllerAdvice](#) [@DeleteMapping](#) [@deletemapping example in spring boot](#) [@ExceptionHandler](#) [@GetMapping](#) [@getmapping annotation used on @getmapping spring boot example](#) [@Modifying](#) [@PatchMapping](#) [@patchmapping example](#) [@PathVariable](#) [@PostMapping](#) [@PutMapping](#) [@Query](#) [@RequestBody](#) [@RequestMapping](#) [@ResponseBody](#) [@RestController](#) [@RestControllerAdvice](#) [@Transactional](#) [api rest spring boot](#) [api rest spring boot mysql](#) [Building a RESTful Web Service - Spring](#) [create rest api in java](#) [create rest api in java with spring boot](#) [crud api](#) [CRUD example spring boot with mysql and postman](#) [crud interface java](#) [crud rest api example](#) [exception handling spring boot rest api](#) [how to create a rest api in java](#) [How to create a REST API using Java Spring Boot](#) [How to Create a REST API With Spring Boot](#) [How to develop CRUD REST API using Spring Boot ?](#) [How to develop REST API using Spring Boot ?](#) [How to develop REST CRUD API using Spring Boot ?](#) [how to implement rest api in java](#) [java spring boot rest api](#) [rest api example java](#) [rest api using spring boot](#) [rest api with spring boot](#) [rest crud](#) [rest crud operations](#) [restful web services in spring boot](#) [restful web services spring boot](#) [spring boot crud example](#) [spring boot crud operations](#) [spring boot crud rest api](#) [spring boot rest api](#) [spring boot rest api best practices](#) [spring boot rest api crud example](#) [spring boot rest api crud example with mysql database](#) [spring boot rest api example](#) [spring boot rest api tutorial](#) [spring boot rest controller exception handling](#) [spring boot rest example](#) [spring boot rest web service](#) [spring boot rest web service example](#) [spring boot rest web services](#) [spring boot restful web services crud example](#) [spring boot restful web services example step by step](#) [spring boot restful web services json example](#) [spring boot web services](#) [spring rest api crud example](#) [spring rest api tutorial](#) [spring rest example](#) [spring rest web services example](#) [spring web services](#) [spring web services rest](#)

< [Previous article](#)

[What is Java JSON](#)

[Next article](#) >

[How to reduce boilerplate code in REST API Spring Data REST](#)

## 11 thoughts on “How to develop REST CRUD API using Spring Boot ?”

Pingback: [How to reduce boilerplate code in REST API : Spring Data REST - Making Java easy to learn](#)



October 26, 2020 at 2:16 AM

ok

Devs5003

Reply

Pingback: [How to write REST Consumer API using Spring Boot : RestTemplate - Making Java easy to learn](#)



TOP



October 30, 2020 at 3:30 AM

ok

**Devs5003**

Reply

Pingback: [How to handle Exceptions & Errors in Spring Boot? | Making Java easy to learn](#)Pingback: [How to develop a Reactive CRUD REST API with Spring WebFlux? | Making Java easy to learn](#)Pingback: [How to work with JSON in Java? | Making Java easy to learn](#)Pingback: [Microservices Architecture | Making Java easy to learn](#)

April 11, 2021 at 10:58 PM

**Lehel**

I want to say that this post is awesome, great written and include almost all necessary information.

I would like to see more posts like this .

Reply



April 18, 2021 at 1:59 AM

**Delila Dorey**

This is a topic that's near to my heart... Cheers! Exactly where are your contact details though?

Reply



April 27, 2021 at 3:47 PM

**Devs5003**

Always click on 'Contact Us' to connect with us.

Reply

### Leave a Reply

Name \*

Email Address \*

Website

☐ Yes, add  
me to your  
mailing list

Post Comment

Comment \*

### FOLLOW US



### RECENTLY PUBLISHED POSTS

- » [Spring Boot Interview Questions](#)
- » [Java Interface](#)
- » [Spring Security Without WebSecurityConfigurerAdapter](#)
- » [Spring Boot Batch Example CSV to MySQL Using JPA](#)
- » [Spring Batch Tutorial](#)
- » [Java 14 Features](#)

[↑  
TOP](#)

- » **Java Features After Java 8**
- » **Java Core Tutorial**
- » **Spring Cloud Annotations With Examples**
- » **How To Add JDK 18 Support in Eclipse**

## DO YOU HAVE A QUERY ?

**Submit your Query**

Email \*

**Subscribe to receive Updates !**