Working with RestTemplate
=========================

```
 Methods associated with RestTemplate
            xxxForEntity(,,,) => overloaded method
```

Instead of calling xxxForEntity as per the requeset methods we can use single exchange(,,) for all operations

```
Note:: @RequestMapping(value="" method = RequestMethod.POST/GET)
                        or
         @GetMapping(value="") and @PostMapping(value="")
```

Syntax of exchange()
====================

```
 public <T> ResponseEntity<T> exchange(
                                          String url,
                                          HttpMethod method,
                                          HttpEntity<?> requestEntity,
                                          Class<T> responseType,
                                          Object... uriVariables)throws
RestClientException
```

```
url           =>  The URL
method        =>  HttpMethod(GET,POST,...)
requestEntity =>  headers+body
responseType  =>  required response type
urivairables  =>  path variable values
```

Output:: ResponsEntity<T>

```
Note: This method is an alternative to getForEntity(), postForEntity(),......
        RestTemplate supports synchronous communication.
        RestTemplate introduced in Spring3.X version
       Internally RestTemplate uses java.net connection to send HttpRequest
```

WebClient
=========

 It is introduced from Spring5.X
 It supports for both Synchronous and Asynchronous request
 To use WebClient,SpringBoot has provided a starter called "SpringWebFlux".

```
 <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-webflux</artifactId>
 </dependency>
```

Code
=====

```
@Service("service")
public class CurrencyService {

      private static final String REST_END_POINT =

      "http://localhost:8888/SpringRest-CurrencyConverter-Providerapp/api/
currency/getCurrencyExchangeCost/from/USD/to/INR";
```

```
    public void invokeRestApiSync(String from, String to) {
        // Sending Synchronous request
        WebClient webClient = WebClient.create();
        System.out.println("***Synchronous: Rescall started ******");
        CurrencyResponse response = webClient.
                get().
                uri(REST_END_POINT, from, to).
                accept(MediaType.APPLICATION_JSON).
                retrieve().
                bodyToMono(CurrencyResponse.class).
                block();

        System.out.println(response);
        System.out.println("****Synchronous: Restcall ended ******");
    }

    public void invokeRestApiASync(String from, String to) {

        // Sending Synchronous request
        WebClient webClient = WebClient.create();
        System.out.println("***ASynchronous: Restcall started ******");
        webClient.
            get().
            uri(REST_END_POINT, from, to).
            accept(MediaType.APPLICATION_JSON).
            retrieve().
            bodyToMono(CurrencyResponse.class).
            subscribe(CurrencyService::myResponse);

        System.out.println("****ASynchronous: Restcall ended ******");
    }

    public static void myResponse(CurrencyResponse response) {
        System.out.println(response);
        //use repsonse object as per the needs[push to Apache-kafka]
    }

}
```

refer:: SpringRest-CurrencyConverter-Providerapp and SpringRest-WebClient-GetRequest

Sending POST request
====================

```
@Service("service")
public class ERailClientApp {

    private static final String REST_END_URL = "http://localhost:8888/SpringRest-TicketBooking-ProviderApp/api/ticket/register";

    public void invokeRestApi() {

        WebClient client = WebClient.create();

        PassengerInfo body = new PassengerInfo();
        body.setFirstName("nitin");
        body.setLastName("manjunath");
        body.setJourneyDate("22/06/2023");
```

```
            body.setFrom("bengaluru");
            body.setTo("pune");
            body.setTrainNumber("BNG-PUN-1234");

            Ticket response =
client.post().uri(REST_END_URL).accept(MediaType.APPLICATION_JSON)
                        .body(BodyInserters.fromValue(body)).retrieve().bodyToMono(
Ticket.class).block();

            System.out.println(response);

      }
}
                  refer::SpringRest-TicketBooking-ProviderApp and SpringRest-
WebClient-PostRequest
```

Develop a REST API with HATEOAS
==============================

1. SpringBoot has provided a starter file to work with HATEOS

```
            eg: <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-hateoas</artifactId>
               </dependency>
```

eg#1

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Book extends RepresentationModel {

      private String isbn;
      private String name;
      private Double price;
      private String author;
}
```

RestController
=============

```
public class BookController {

      @GetMapping(value = "/getBook/{isbn}", produces = "application/json")
      public ResponseEntity<Book> getBook(@PathVariable("isbn") String isbn) {

            Book book = new Book(isbn, "Spring", 234.5, "RodJhonson");

            Link link =
WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getAllBoo
ks())
                        .withRel("allBooks");

            book.add(link);

            return new ResponseEntity<Book>(book, HttpStatus.OK);

      }

      @GetMapping(value = "/allBooks")
      public List<Book> getAllBooks() {

            List<Book> bookList = new ArrayList<Book>();

            bookList.add(new Book("ISBN-111", "Spring", 350.5, "RodJhonson"));
            bookList.add(new Book("ISBN-222", "Hibernate", 350.5, "GavinKing"));
            bookList.add(new Book("ISBN-333", "Servlet", 350.5, "KeitySeirra"));

            return bookList;

      }

}
```

```
Input: http://localhost:9999/getBook/10
Output:
{
    "isbn": "10",
    "name": "Spring",
    "price": 234.5,
    "author": "RodJhonson",
    "_links": {
        "allBooks": {
            "href": "http://localhost:9999/allBooks"
        }
    }
}
```

.