Invoking currency-exchange-service from currency-conversion-service

We have the currency-exchange-service ready, and we have set up a currency-calculation-service (currency-conversion-service). Now we will invoke the currency exchange service from the currency calculation service.

We use **RestTemplate()**constructor to invoke an external service. Let's create a RestTemplate and try to invoke currency-exchange-service.

**Step 1:**Select the **currency-conversion-service** project.

**Step 2:** Open the **CurrencyConversionController.java** and create a new **RestTemplate** that invokes the currency-exchange-service application.

**Step 3:**Invoke the **getForEntity()**method of RestTemplate class.

**getForEntity():** It is a method of **RestTemplate** class that retrieves an entity by using the **HTTPGET** method for the specified URL. It converts and stores the response in the ResponseEntity. It returns the **ResponseEntity**.

**Parameters:**It accepts two parameters:

* **URL:** The URL.
* **responseType:** The type of the return value.

1. ResponseEntity<CurrencyConversionBean>responseEntity=**new** RestTemplate().getForEntity("http://localhost:8000/currency-exchange/from/{from}/to/{to}", CurrencyConversionBean.**class**, uriVariables);

**Step 4:**In the URL parameter, put the URL of **currency-converter-service** that is [http://localhost:8000/currency-exchange/from/{from}/to/{to}](http://localhost:8000/currency-exchange/from/%7bfrom%7d/to/%7bto%7d). It takes values from the variable **{from}** and **{to}** from the request. Whatever comes in the request we sent it to the currency-exchange-service.

**Step 5:**In the above URL, we need to pass two values **"from"** and **"to."**For passing the values, create a **Map**for URI variables. Pass the **uriVariables** in the URI as a parameter.

1. Map<String, String>uriVariables=**new** HashMap<>();
2. uriVariables.put("from", from);
3. uriVariables.put("to", to);

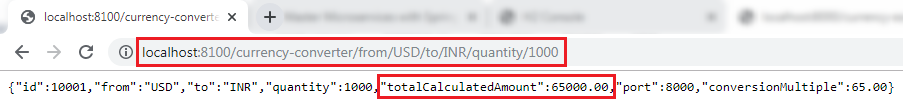
**Step 6:**The response type that we are expecting back is **CurrencyConversionBean,** so store the response in the CurrencyConversionBean.

1. CurrencyConversionBean response=responseEntity.getBody();

**CurrencyConversionController.java**

1. **package** com.javatpoint.microservices.currencyconversionservice;
2. **import** java.math.BigDecimal;
3. **import** java.util.HashMap;
4. **import** java.util.Map;
5. **import** org.springframework.http.ResponseEntity;
6. **import** org.springframework.web.bind.annotation.GetMapping;
7. **import** org.springframework.web.bind.annotation.PathVariable;
8. **import** org.springframework.web.bind.annotation.RestController;
9. **import** org.springframework.web.client.RestTemplate;
10. @RestController
11. **public** **class** CurrencyConversionController
12. {
13. @GetMapping("/currency-converter/from/{from}/to/{to}/quantity/{quantity}") //where {from} and {to} represents the column
14. //returns a bean back
15. **public** CurrencyConversionBeanconvertCurrency(@PathVariable String from, @PathVariable String to, @PathVariableBigDecimal quantity)
16. {
17. //setting variables to currency exchange service
18. Map<String, String>uriVariables=**new** HashMap<>();
19. uriVariables.put("from", from);
20. uriVariables.put("to", to);
21. //calling the currency-exchange-service
22. ResponseEntity<CurrencyConversionBean>responseEntity=**new** RestTemplate().getForEntity("http://localhost:8000/currency-exchange/from/{from}/to/{to}", CurrencyConversionBean.**class**, uriVariables);
23. CurrencyConversionBean response=responseEntity.getBody();
24. //creating a new response bean and getting the response back and taking it into Bean
25. **return** **new** CurrencyConversionBean(response.getId(), from,to,response.getConversionMultiple(), quantity,quantity.multiply(response.getConversionMultiple()),response.getPort());
26. }
27. }

**Step 7:** Run the both services independently. When we run the currency conversion, it returns the response shown below:



The conversion Multiple is multiplied by the quantity and returns the **totalCalculatedAmount** 65000.00. It means $1000 is equal to 65000.00 INR. It also shows the port **8000** that denotes the other service (currency-exchange-service) is running on port 8000.