Day 17

Making HTTP Calls from Spring

- Spring application may need to access external web services
  - Eg travel information, foreign exchange
- External web services can be
  - message or RPC
  - content can JSON or XML

`RestTemplate`

- Class for making HTTP request
- Allow you to
  - Set the HTTP method to use
  - Configure HTTP headers
  - Set payload to be POSTed to the endpoint
    - Eg. URL encoded form, JSON, file upload
  - Read HTTP response status code, headers and payload
- Major classes
  - `RestTemplate`
  - `RequestEntity`
  - `ResponseEntity`

Example - GET

Create an instance of `RestTemplate`

Make the call and retrieve the result as String

```
RestTemplate template = new RestTemplate();

ResponseEntity<String> resp = template.getForEntity(url, String.class)

System.out.printf("Status code: %d\n", resp.getStatusCodeValue());
System.out.printf("Payload: %s\n", resp.getBody());
```

Holds the result of the invocation.
The payload is a string; matches the second parameter of `getForEntity()`

Get the status code and the payload of the request

Example - GET with `exchange()`

```
RequestEntity<String> req = RequestEntity
    .get(url)
    .accept(MediaType.APPLICATION_JSON)
    .build();

RestTemplate template = new RestTemplate();

ResponseEntity<String> resp = template.exchange(req, String.class);

try (InputStream is = new ByteArrayInputStream(resp.getBody().getBytes())) {
    JsonReader reader = Json.createReader();
    JsonObject data = reader.readObject();
}
```

Configure the `GET` invocation

`exchange()` is a generic method for making HTTP calls

Example - GET with Query Parameters

Build a URL with query parameter.
Query parameters will be URL safe

```
String url = UriComponentBuilder
        .fromUriString(url)
        .queryParam("name", "fred")
        .queryParam("email", "fred@gmail.com")
        .toUriString();
```

Add query parameters

```
RequestEntity req = RequestEntity.get(url).build();

RestTemplate template = new RestTemplate();

ResponseEntity<String> resp = template.exchange(req, String.class);
```

Example - POSTing a Url Encoded Form

```
MultiValueMap<String, String> form = new LinkedMultiValueMap<>();
form.add("name", "fred");
form.add("email", "fred@gmail.com");
```

Add form fields

```
RequestEntity<MultiValueMap<String, String>> req = RequestEntity
    .post(url)
    .contentType(MediaType.APPLICATION_FORM_URLENCODED)
    .body(form, MultiValueMap<String, String>.class)
```

Specify the content type and the form

```
RestTemplate template = new RestTemplate();

ResponseEntity<String> resp = template.exchange(req, String.class);
```

Example - POSTing JSON Data

```
JsonObject json = Json.createObject()
    .add("name", "fred")
    .add("email", "fred@gmail.com").build();

RequestEntity<String> req = RequestEntity
    .post(url)
    .contentType(MediaType.APPLICATION_JSON)
    .headers("Accept", MediaType.APPLICATION_JSON)
    .body(json.toString(), String.class)

RestTemplate template = new RestTemplate();

ResponseEntity<String> resp = template.exchange(req, String.class);
```

Build the JSON object

Specify the content type as JSON
Add additional headers