1. Problem related to foreign keys of database model:

* When this foreign key is also included as part of primary key, you cannot remove this foreign key nor drop the primary key.

->remove the schema and regenerate all tables; if not necessary, do not include foreign keys as part of primary key.

* You cannot remove a table when one of the columns is used as foreign keys in other tables. Similarly, you cannot truncate a table when foreign keys in other tables have been instantiated.
* When instantiate an entity with a foreign key, errors usually appear when the foreign key doesn't have a valid value or empty.

2. Thymeleaf related problem

* You cannot give two th:each properties in one div, which means you cannot loop through two lists at the same time.
* You can use Thymeleaf to check if the user is authenticated by adding sec:authorize="isAuthenticated()" to a div, and the div will not show up if it’s false. Same for sec:authorize="hasAuthority('ADMIN')", the div will only show up when the user has authority of admin.

**E.g.** <a sec:authorize="isAuthenticated()" th:href="@{/restaurant/name/}" class="mdl-navigation\_\_link" href="">Restaurant</a> (line 50 of navbar.html)

* If you want use model attributes from controller in JavaScript, th:inline="javascript" has to be added to the script tag and use special format to use the attributes: **var** url = /\*[[${url}]]\*/'url'; (see line 33-41 of account.html)

3. RequestMethod.Post related

* Everything get post to the controller via HTTP is String, it’s up to the server to decide which type you want to decode. If you specify @Valid int tableId, the controller will try to decode the value to int and will introduce an error if it fails.
* Hidden input can be used to submit action where there is no input field. You can also put a hidden input field to normal form.
* RedirectAttributes can be used to pass data from post function after redirecting. This is specifically for Spring Framework. After initializing, you can use it like this: redir.addFlashAttribute("message"," This username is already registered!"); and in the redirected GET method, you can get the information by specifying @ModelAttribute("message") final String message .

-Those flash attributes are passed via the session

-They will be destroyed immediately after being used

-They are not visible in URL

-You are not restricted to String, but may pass arbitrary objects.

4. URL appending

* If you want to go to another base on the current page, you need add an "/"after.

e.g. current url:"www.mybase.com/home/" href="eidt" will lead you to "www.mybase.com/home/edit". If you current url is "www.mybase.com/home" href="edit" will lead you to “www.mybase.com/edit”.

5. Role checking in server side (Spring security)

* You may check if the logged in user has a role of admin in controller final:

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

SimpleGrantedAuthority AUTHORITY\_ADMIN = new SimpleGrantedAuthority("ADMIN");

org.springframework.security.core.userdetails.User user = auth.getPrincipal(); if(user.getAuthorities().contains(AUTHORITY\_ADMIN)){

model.addAttribute("isAdmin",true);

}

6. Database and JPA mapping related

* "table","order" and other similar keywords cannot be used as table name or column name, SQL will fail.
* @Transient annotation can be used for an entity field which will only be used during its life time, but will not mapping to database when it’s saved.

7. Problem on making a shared model

* Initially, we want to make a jar file and include it in our project, but we failed to use/refer to it. Later, we choose to create a local Maven dependency, and compile it in the Gradle dependencies.
* How to make a Maven dependency with your models?
* Create a maven project with your-group-id and your-artificial-id;
* Create a package: your-group-id.model under java folder, put all your java files under that folder.
* Do maven build (install).
* In the Gradle project where you want to use those models, add mavenLocal() under repositories {…} and add compile(your-group-id: your-artificial-id:1.0-SNAPSHOT') under dependencies {…}.
* After build your project, you will find your-group-id.model under your-artificial-id- 1.0-SNAPSHOT.jar under the corresponding dependencies within External Libraries.

8. Combining eureka discovery with spring security.

* According to the tutorial from spring.io/blog, the author disables component scan in the main application and create all necessary beans inside main. However, this may cause an issue, because it will also disable the security configuration which is usually a separate config class. In this case, the whole project will use default spring security configuration (all pages need login with default username and password).
* It’s not necessary to create Beans for Controllers and Services in the MainApp.java. Controllers as singletons will be created by the container automatically. One Controller can use multiple Services by adding notation @Autowired.
* By configuring web application as a client of eureka server, you only need physical address of your eureka registration server. For all the Restful API calls, you only need the name of the corresponding server. The eureka server will find a registered server with the right name you wanted. E.g. String serviceUrl = "http://menu-api-server ".