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 related

* "table","order" and other similar keywords cannot be used as table name or column name, SQL will fail.

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 groupId = com.xxx and Artificial Id, under java folder, create pakage: com.xxx.models, put all your java files under that folder. Then do maven build.

8. combine eureka discovery with spring security.

* According to the tutorial from spring.io/blog, the author disable component scan in the main application and create all 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).