Design Report ---webflux

1. Finished features
2. Base webFlux features
3. Functionan request handler
4. Persisting data in mysql
5. API Design and Implementation

|  |  |  |
| --- | --- | --- |
| router | method | function |
| /trainer/list | Get | Get all trainer info |
| /trainer/show/{id} | Get | Get a trainer info |
| /trainer/add | Post | Save a trainer info |
| /trainer/update/{id} | Put | Update a trainer info |
| /trainer/delete/{id} | Delete | Delete a trainer info |

1. Router design

public RouterFunction<ServerResponse> route(TrainerHandler handler) {

return RouterFunctions

.route(GET("/trainer/list").and(accept(MediaType.APPLICATION\_JSON)), handler::findAll)

.andRoute(GET("/trainer/show/{id}").and(accept(MediaType.APPLICATION\_STREAM\_JSON)), handler::findById)

.andRoute(POST("/trainer/add").and(accept(MediaType.APPLICATION\_JSON)), handler::save)

.andRoute(DELETE("/trainer/delete/{id}").and(accept(MediaType.APPLICATION\_JSON)), handler::delete)

.andRoute(PUT("/trainer/update/{id}").and(accept(MediaType.APPLICATION\_JSON)), handler::update);

}

1. Persistence design

Due to the use of mongoDB for data persistence in the database, storage and query errors, the teacher's example also appeared to be unable to operate the database.

So I use Mysql as a data persistence database. webFlux can't integrate mysql, so I added a layer of data object conversion layer between webFlux and mysql, converted the object of webFlux operation (Mono, Flux) into data entity in mysql, or converted mysql entity into webFlux operable object.Thereby achieving the function of persisting data.