理解前后端程序---（以“应用分布”板块为例）

前端

首先锁定CenterLeft1.vue文件夹，核心如下：

```
8   const { data } = api.common.getStatistics.useQuery();
9
10  const showedProtocols = [
11    { type: ["HTTP", "HTTPS"], label: "Web 类应用" },
12    { type: ["IMAP", "POP3"], label: "邮件类应用" },
13  ];
14  const icons = ["💖", "🔀"];
15
16  const numberData = computed(() => {
17    const { application = [], totalPacket = 1 } = data.value ?? {};
18    return showedProtocols.map(({ label, type }) => {
19      const n =
20        Array.isArray(type) ?
21          type.reduce((acc, cur) => acc + (application.find((v) => v.type === cur)?.packet ?? 0), 0)
22        : application.find((v) => v.type === type)?.packet ?? 0;
23      return {
24        config: {
25          number: [(n / totalPacket) * 100],
26          toFixed: 1,
27          content: "{nt}%",
28          style: { fontSize: 23 },
29        },
30        text: label,
31      };
32    });
33  });
```

第八行涉及一个data的获取，下面的函数则实现的是data的计算，我先探究data的获取

```
70  export const api = mixinRouters({
71    common: commonApi,
72    security: securityApi,
73  });
```

这里我个人的理解是`api`由`mixinRouters`生成，过程是向其传入`common`和`security`，然后`mixinRouters`函数会将其中的函数层层拆开，全部交给`mixinVueQuery`函数，`mixinVueQuery`函数则将`vueQuery`相关函数融进原对象，实现对原对象的扩展升级。

然后我产生了一点问题：

```ts
const mixinVueQuery = <F extends RequestFn>(
  fn: F,
  path: string[],
): F & {
  useQuery: (...args: Parameters<F>) => ReturnType<typeof useQuery<Awaited<ReturnType<F>>>>;
  useMutation: () => Merge<
    Omit<ReturnType<typeof useMutation<Awaited<ReturnType<F>>>>, "mutateAsync">,
    { mutateAsync: (...args: Parameters<F>) => ReturnType<F> }
  >;
  invalidateQueries: (...args: Parameters<F>) => Promise<void>;
} => {
  const useQueryFn = (...args: Parameters<F>) =>
    useQuery({
      queryKey: [path.join("."), ...args],
      queryFn: () => fn(...args) as never,
    });
  const useMutationFn = () => {
    const originalResult = useMutation({
      mutationFn: (args: Parameters<F>) => fn(...args) as never,
    });
    const originalMutateAsync = originalResult.mutateAsync;
    const mutateAsync = (...args: Parameters<F>) => originalMutateAsync(args);
    return Object.assign(originalResult, { mutateAsync });
  };
  const invalidateQueries = async (...args: Parameters<F>) => {
    await useQueryClient().invalidateQueries({ queryKey: [path.join("."), ...args] });
  };

  return Object.assign(fn, {
    useQuery: useQueryFn,
    useMutation: useMutationFn,
    invalidateQueries,
  }) as never;
};
```

因为有`const { data } = api.common.getStatistics.useQuery();`所以最后调用的是`useQuery`吗？我对`useQuery`的作用感到费解，我暂时理解成这是一个便于我`getStatistics`的手段（虽然我对其作用完全不理解）

好，接下来看`common.getStatistics`的部分。

```ts
const commonApi = {
  getMetadata: (): Promise<Metadata> => request.get("/api/metadata"),

  getStatistics: (): Promise<Statistics> => request.get("/api/statistics"),

  getPacketSizeDistribution: (): Promise<PacketSizeDistribution> =>
    request.get("/api/packet-size-distribution"),

  getSrcCardinalityDistribution: (): Promise<SrcCardinalityDistribution> =>
    request.get("/api/src-cardinality-distribution"),

  getTimeFlowChange: (): Promise<TimeFlowChange> => request.get("/api/time-flow-change"),
```

注意到`getStatistics`时，向`/api/statistics`发送了get，我好奇：这个地址具体是什么？是localhost:8080吗？

后端

在ApiService.java中我们可以清楚地看到`getStatistics`函数。

```java
public Statistics getStatistics() {  1 个用法  liangjiachen +1
    Statistics statistics = new Statistics();
    statistics.setTotalFlow(PcapngReader.getTotalFlow() / 100000);
    statistics.setAverageFlowPerMinute(PcapngReader.getTotalFlow() / 100000 / 60);
    statistics.setTotalPacket(PcapngReader.getTotalPacket());
    statistics.setAveragePacketPerMinute(PcapngReader.getTotalPacket() / 60);
    statistics.setTotalAddress(PcapngReader.getSrcPacket().size() + PcapngReader.getDestPacket().size());
    statistics.setTotalCommunicationPair(PcapngReader.getVisitedPair().size());
    List<Protocol> protocols = new ArrayList<>();
    for (String key : PcapngReader.getProtocol().keySet()) {
        Protocol protocol = new Protocol();
        protocol.setType(key);
        protocol.setPacket(PcapngReader.getProtocol().get(key));
        protocols.add(protocol);
    }
    statistics.setProtocol(protocols);
    List<Application> a = new ArrayList<>();
    for (String key : PcapngReader.getApplication().keySet()) {
        Application application = new Application();
        application.setType(key);
        application.setPacket(PcapngReader.getApplication().get(key));
        a.add(application);
    }
    statistics.setApplication(a);
    return statistics;
}
```

其功能也容易看出，大概是对数据进行了统计，其中`PcapngReader`我将其理解为网络数据包的阅读器（顾名思义），让我们可以对网络数据包信息进行访问等处理。

ExampleController.java中有其使用：

```java
@GetMapping("/statistics")  liangjiachen
public Statistics getStatistics() {
    return apiService.getStatistics();
}
```

可以看到，监测的是`/statistics`的get请求（为什么不是`/api/statistics`呢）这里`/statistics`的地址又是什么？

以上就是我学习过程中的一些问题和思考。