Spring - Mar 2

System

→ dont want everyone can access DB to do CRUD

→ server to provide API (permission)

→ multiple servers make net become bigger and bigger, like a graph

→ run out of control

→ cuz dependences become so complicated. When server shuts down, it will impact lots of other servers who depend on its service(APIs)

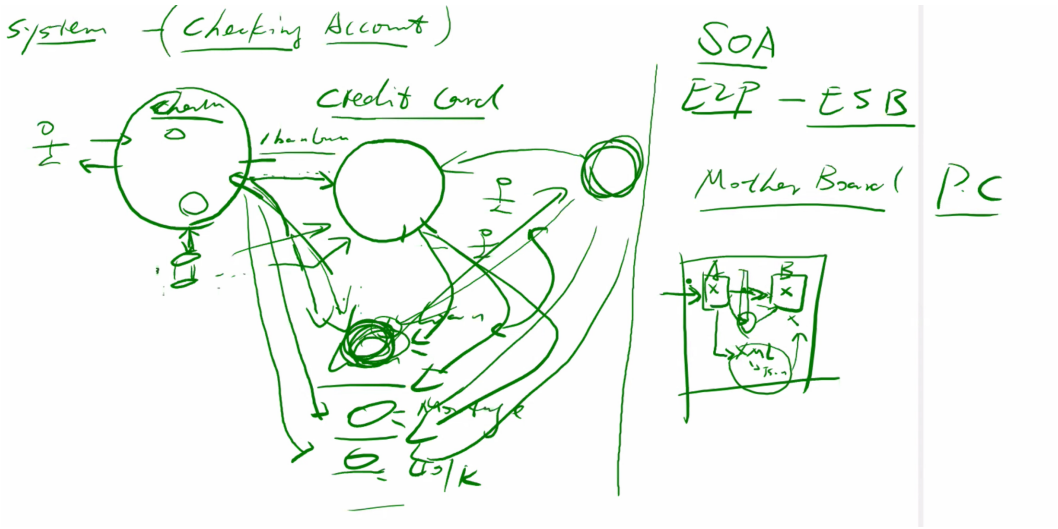
SOA(Service Oriented Architectures)

- we provide the services, as a basic unit
- manage the services

EIP (enterprise integration pattern) → ESB(Enterprise Service Bus)

ESB: it dont need nodes know each other (lose couple)

> instead, ESB looks like a controller, and it links every part together and every part dont know each other
>
> e.g
>
> ESB seems like a mother board of PC
>
> nodes dont care about where output data is going to
>
> ESB controls data flows
>
> When we want to change a data flow, we dont need change A or B node. Just to change ESB



MicroServices vs. SOA

MicroServices

focus on split a very big service into servlet small services to MAKE whole system become more elastic

SOA

at very beginning, designed the services, designed input/output that they can talk each other. All the services together, they can do something

SOA dont care about the services are light or heavy

MicroServices

- **scalability**
- **flexibility → development**

  We have services, they could be very small(one/two api), which will be easy to add or delete. When the service become very busy, we can easily add more server to handle the increasing data flow

One solution → MicroServices

Spring Cloud

- Distributed/versioned configuration
- Service registration and discovery
- Routing API gateway

https://spring.io/projects/spring-cloud