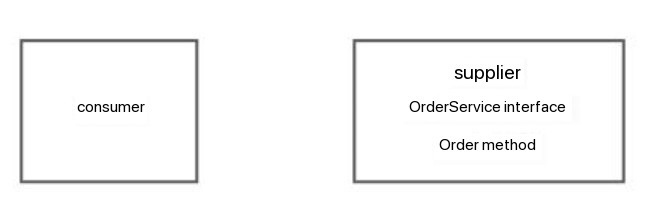
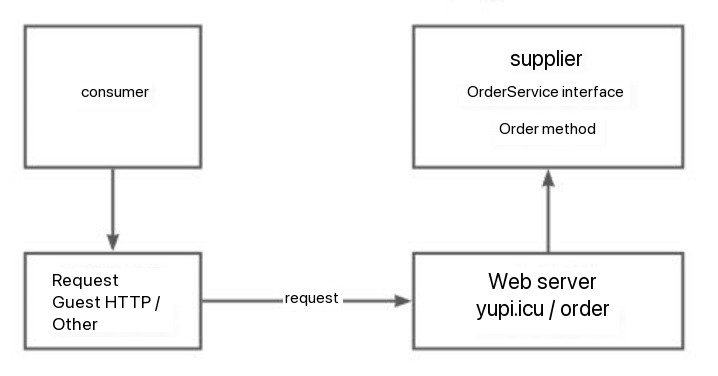
# RPC framework implementation ideas

First of all, two roles, service consumer and service supplier:



If the consumer wants to call the supplier, it needs the supplier to start a web service, and then call it by requesting the client to send an HTTP or other protocol request.

For example, after requesting the xxx.com/order address, the supplier will call the order method of orderService:

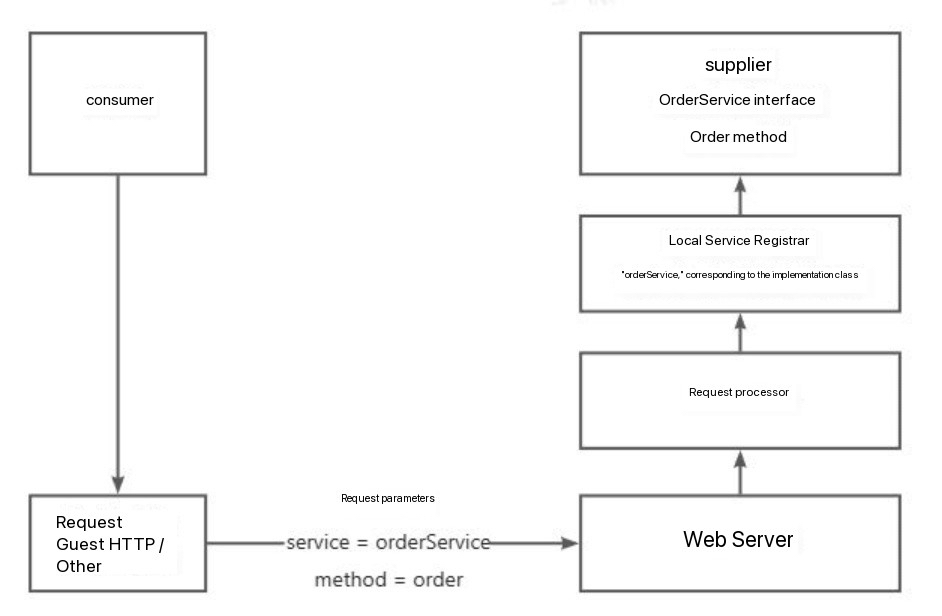


But if the supplier provides multiple services and methods, must a separate interface be written for each interface and method? Does the consumer need to write a piece of HTTP call logic for each interface?

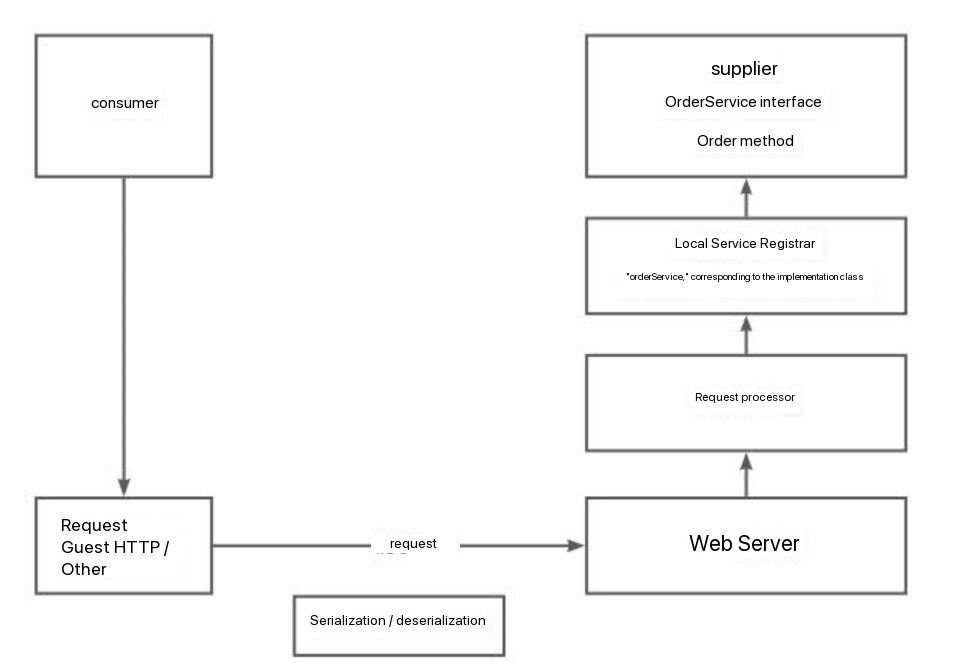
In fact, a unified service calling interface can be provided, and the request processor can perform different processing and call different services and methods according to the client's request parameters.

You can maintain a local service register in the service supplier program to record the mapping between services and corresponding implementation classes.

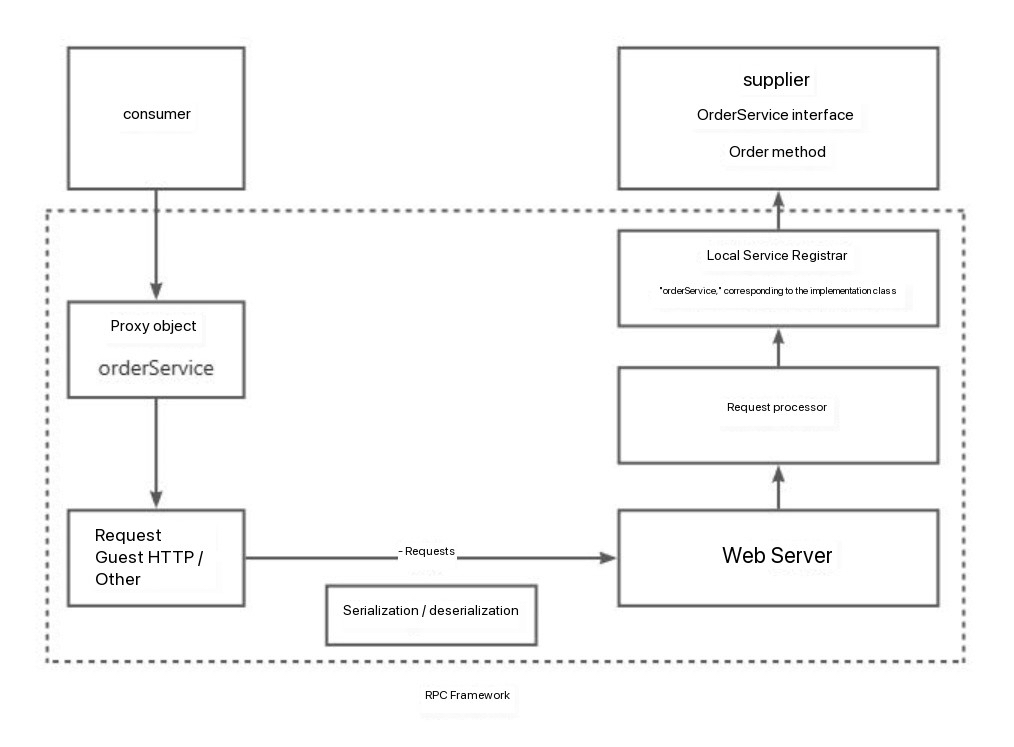
For example, if a consumer wants to call the order method of the orderService service, it can send a request with the parameters service=orderService, method=order. Then the request processor will find the corresponding service implementation class from the service registry according to the service, and use Java The reflection mechanism calls the method specified by method.



It should be noted that since Java objects cannot be transmitted directly over the network, the transmitted parameters must be serialized and deserialized.



At this point, the simplest RPC framework architecture diagram was born:



The dotted box in the figure above is the modules and capabilities that the RPC framework needs to provide.