Deployment

- TCP vs UDP (broadcast)
- HTTP (1.0 vs 2.0)
  - HTTP methods:
  - GET vs POST ??? request body or not
  - idempotence ??? what + when to use + why??
- Web Resource
  - static resources
    - html / css / js / txt / mp4 / jpg
  - dynamic resources
    - jsp pages / Servlet program
- HTTP status code

- postman

- Web Server
  - Tomcat:
    - a web server provided by Apache organization supporting jsp and Servlet
    - lightweight javaWeb Container (free)
  - Jboss:
    - open source following JavaEE rules
    - pure Java EJB server (free)
  - GlassFish:
    - developed by Oracle, robust commercial server
  - Resin:
    - developed by CAUCHO, excellent performance, developed in java
  - WebLogic:
    - developed by Oracle, following JavaEE rules, adaptive for large-scale project

- Tomcat Version Servlet/JSP Version JavaEE Version Runtime Environment
  - 4.1 2.3/1.2 1.3 JDK 1.3
  - 5.0 2.4/2.0 1.4 JDK 1.4
  - 5.5/6.0 2.5/2.1 5.0 JDK 5.0
  - 7.0 3.0/2.2 6.0 JDK 6.0
  - 8.0 3.1/2.3 7.0 JDK 7.0
  - run foreground vs run background
  - program architecture (usage of each folder)
  - how to boost

```
(base) shimiharu@shimeiqingdeMacBook-Pro Documents % jps
4056 Bootstrap
7566 Jps
```

    - where to print logs
    - Linux experience

- postman

https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods

https://developer.mozilla.org/en-US/docs/Glossary/Idempotent

https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status

- payload, response payload
- persist layer

- why do we care about Idempotency

> Right, that’s correct — idempotency is not just an HTTP concept, it’s a broader system-level idea.
>
> Why is it important? Because in real-world systems, networks can have **fluctuations**, or a **user might click a button multiple times**, sending **duplicate requests**.
>
> Also, under **high traffic**, the **message queue** might fail to detect and filter out those duplicates.
>
> For example: a duplicate **order creation**.
>
> In such systems, implementing **idempotency** helps us avoid problems like **duplicate orders**, which can cause pressure on

> both the **backend server** and the **database**.

> In real systems, even though we prefer to use idempotent HTTP methods (like GET, PUT, DELETE),
>
> **we still cannot avoid using** `POST` in some cases — for example, when we need to **insert new records** into the database.
>
> So the question is:
>
> Can we still design a system that supports idempotency, even when using POST?
>
> The answer is yes — we can design an **idempotent system using POST**, by implementing additional logic such as **idempotency keys**.

> 在高并发或网络不稳定的环境下，幂等性是确保请求安全的关键设计。
>
> 前端的限流和按钮防抖**只能预防用户误操作**，
>
> 真正的保证必须由后端逻辑通过幂等机制实现，避免重复处理同一请求。

Solution:

> We can have a centralized Redis service.
>
> Every time the backend wants to insert an order into Oracle DB, it should first:
>
> 1. **Check Redis** to see whether the **unique order ID** already exists.

> 2. If it does exist → this is a **duplicate request**, so we **ignore it** or **return cached result**.
> 3. If it does **not** exist → this is the **first request**, so we:
>   - Process it,
>   - Write the order to the database,
>   - And **save the unique ID to Redis**.
>
> If a duplicate is detected, maybe we also need to **regenerate the UI**, like reset the form or block the button.

理论上，HTTP 规范并**不禁止**在GET请求中带 body，但**实际上几乎所有服务器、浏览器、代理都会忽略它。**

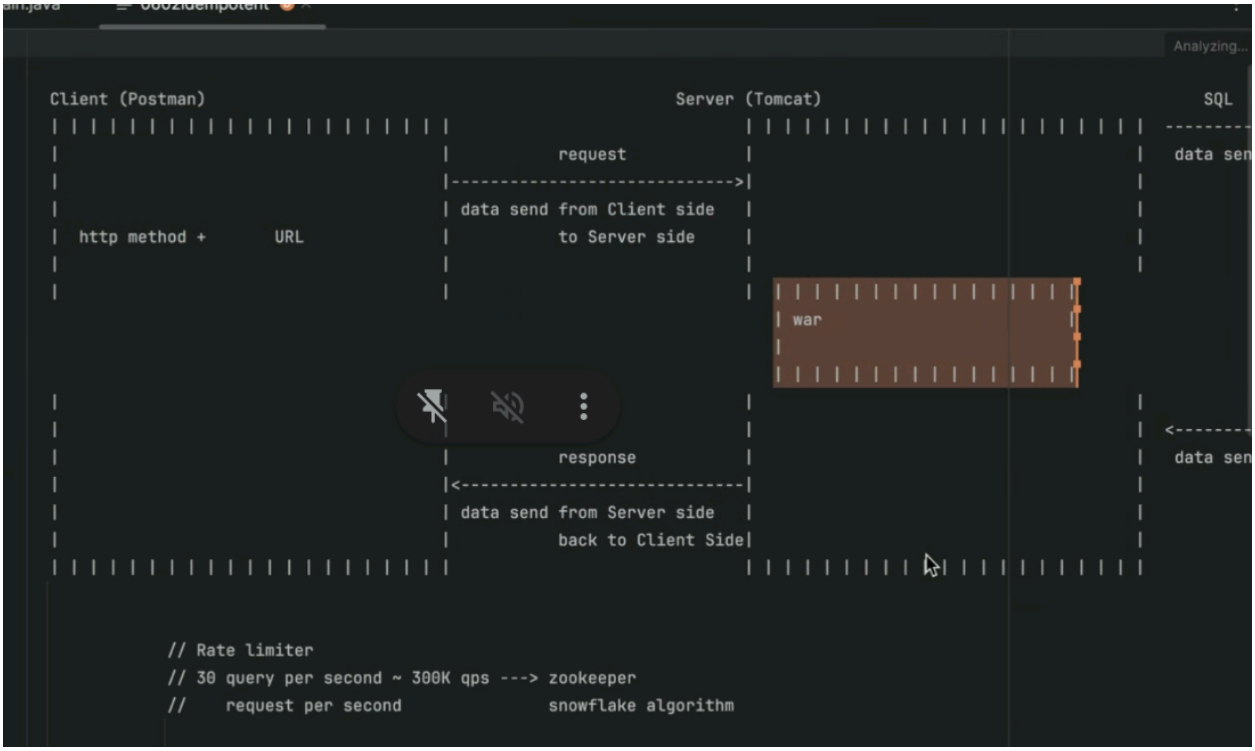
How are we going to send request from client to the server?

Postman:

- params
- auth type
- ...

User agent

- Request Header (Github)
- cache control



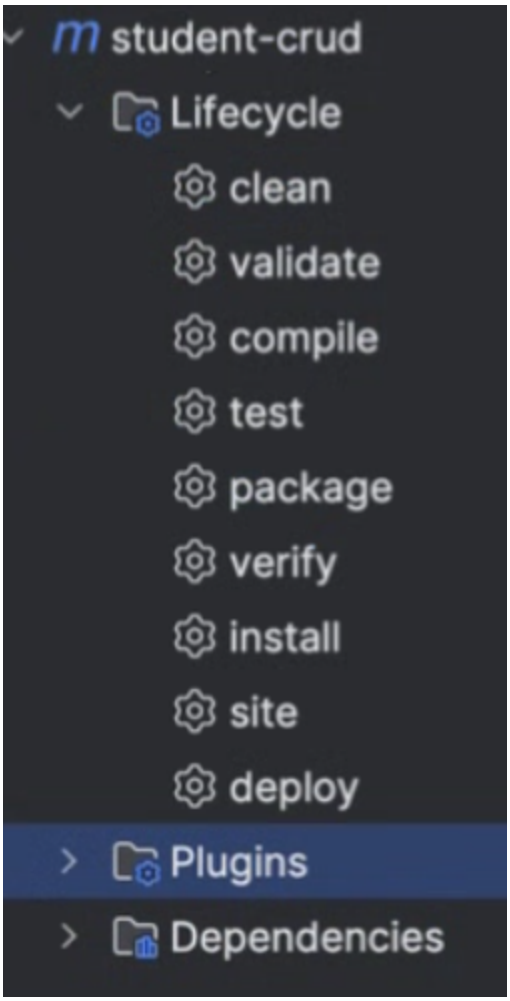
How do urls mapping to content of war?

endpoint

servlet

// use servlet(can also use httpServlet) + JDBC(do crud operation in one table of my sql databse) write the java program, deploy war package into tomcat, do crud using postman

Maven Project



What is these life cycle?

🧭 Complete Maven Lifecycle (Based on IntelliJ View)

📦 Example Project: `student-crud`

A simple Java web app to manage student data. Standard Maven structure:

- `src/main/java`: main source
- `src/test/java`: unit tests
- `pom.xml`: dependencies (JUnit, Servlet API, etc.)

hase	Description	In `student-crud`	How It Works

`clean`	Deletes previous build output	Deletes `target/` folder	Triggered by `mvn clean`, uses `maven-clean-plugin`
`validate`	Validates project structure and pom	Checks `pom.xml` for errors, missing tags	Maven parses the POM, checks project integrity
`compile`	Compiles main source code	Compiles `Student.java`, `Service.java` to `target/classes`	Uses `maven-compiler-plugin`
`test`	Compiles and runs unit tests	Executes `StudentServiceTest.java` with JUnit	Uses `maven-surefire-plugin`
`package`	Packages compiled code	Creates `student-crud.jar` or `.war` in `target/`	Uses `maven-jar-plugin` or `maven-war-plugin`
`verify`	Verifies integration test results	Optional stage, often empty unless configured	Can run tools like `failsafe-plugin` or QA checks
`install`	Installs package to local Maven repo	Puts jar/war in `~/.m2/repository`	Makes the artifact reusable locally
`site`	Generates project documentation	Generates site, Javadoc, dependency info	Uses `maven-site-plugin`
`deploy`	Deploys package to remote repo	Pushes to company Nexus repo (if configured)	Controlled via `distributionManagement` in `pom.xml`

🔄 Typical Command Execution

```
mvn clean install
```

This runs:

- `clean`: clears old builds
- `validate`
- `compile`
- `test`
- `package`

- `install`

🧠 Notes

- Each phase runs all previous phases
- Plugin bindings define what "work" each phase does
- You can customize lifecycle behavior in `pom.xml`