Spring Cloud 网关

基础知识

网关技术

核心技术：反向代理

- 层次：四层（Envory）、七层（Nginx）
- 核心
  - Web 服务发现（服务提供方）
  - HTTP/TCP 请求转发
    - 内置 HTTP Client
- 应用
  - 授权、认证
  - URL 重定向
  - 安全

Apache Httpd

- 模块（mods）模块
  - url rewrite
  -

Nginx

扩展：

- Tengine
- OpenResty
  - Nginx + Lua
- Kong
  - Nginx + Lua + Admin(Persistence)

Java 网关

Servlet Gateway

1997 Servlet 相当于 Java CGI（Common Gateway Interface）

Servlet 基于线程（Linux 内核小于 2.6 没有线程的概念）

> JSP 动态 Servlet，运行时编译，到达字节码变化效果

CGI 基于进程

Netflix Zuul

省略（可以参考公开课）

基于 Servlet API

- 基于 Servlet
- 基于 Filter

Spring Cloud Gateway

Spring Reactive Web 实现

Spring Boot 2.0 Web 实现

- Servlet 3.1
  - Spring Web MVC
- Reactive Web
  - Spring WebFlux
    - Netty
    - Servlet 3.1

Spring Cloud Gateway 普遍印象：

- 难用
- 性能不佳
- 扩展复杂

整体表现不如 Spring Cloud Zuul

Spring 分布式上的积累、抽象远不及 Java EE 社区

> 小知识：
>
> 最广泛流行 HTTP 1.1 协议下，Reactive Web 是一个伪命题
>
> HTTP 1.1 工作模式：Request/Response（短连接，同步）
>
> > WebSocket HTTP 1.1 协议来建连， Socket 来做长连接
>
> HTTP 2.0 + 工作模式：HTTP 1.1 + Stream（长连接、流式、异步）

Java 网关实战

写手 Servlet 网关

Servlet 生命周期

Servlet 初始化

`Servlet#init(ServletConfig)`

`ServletConfig` 配置：

- 获取 Servlet 名称

```
<servlet>
    <servlet-name>Servlet名称</servlet-name>
    <servlet-init>
        <init-name>cleanupAfterInclude</init-name>
        <init-value>false</init-value>
    </servlet-init>
</servlet>
```

- 获取 Servlet 参数
  - Servlet 3.0+
    - `@WebInitParam`
- 获取 `ServletContext`

Servlet 请求处理

`Servlet#service(ServletRequest,ServletResponse)`

输入（请求），输出（响应）

Servlet 的设计并不只为了 HTTP 协议，其中 `HttpServlet` 才是关注于 HTTP 协议

Servlet API 接口隐形含义，当 service 方法内部处理是线程安全的，Servlet 实例本身是非线程安全的。

一种类型 Servelt 实例在同一个Servlet 上下文 并非只有一个，一个名称对应了一个 Servlet 实例，但是允许不同名称的同一个类 Servlet 的实例。

从 Servlet 2.3 开始，引入了 Filter API

同时，引入了 Servlet 生命周期接口，比如 `ServletContextListener`，它的一个著名实现类 Spring `ContextLoaderListener`

Servlet 销毁

使用 Spring Cloud Gateway 搭建网关

Java 网关设计

Spring Web 5.0 + 核心 API

- Web 服务交换对象 - `ServerWebExchange`
  - 总结：聚合 Request、Response 以及 Session 的对象，并且深度参考 Servlet API
  - 请求 - `getRequest()`：`ServerHttpRequest`
    - 类比 - `HttpServletRequest`
  - 响应 - `getResponse()`：`ServerHttpResponse`
    - 类比 - `HttpServletResponse`
  - 属性 -
    - 全体：`getAttrributes()`：`Map`
    - 个体：`getAttrribute(String)`：`Object`
  - 会话 - `getSession()` - `Mono<WebSession>` - Reactive + `Optional`
    - 类比 - `HttpSession`
  - 认证 - `getPrincipal()` - `Mono<Principal>`
    - 类比 - `javax.servlet.http.HttpServletRequest#getUserPrincipal`
- Web 处理接口 - `WebHandler` `
  - 类比 - `Servlet`
    - Spring Web MVC - `DispatcherServlet`
    - Spring Web Flux - `DispatcherHandler`

> 小知识

场景	Reactor API（推）	Stream API（拉）
单个	Mono	java.util.Optional
多个	Flux	java.util.Collection

分析 Spring Cloud Gatway 设计思路

基本步骤

1. 请求匹配
    1. 前置判断 - 当前请求是否匹配 - `RoutePredicateFactory`
        1. After Route Predicate Factory
        2. Before Route Predicate Factory
        3. Between Route Predicate Factory
        4. Cookie Route Predicate Factory
        5. 路径匹配 - Path
        6. 请求头匹配
2. 过滤处理
    1. `HttpHeadersFilter`
    2. `GlobalFilter`
    3. 自定义业务逻辑
3. 路由规则
    1. `RouteLocatorBuilder`
    2. `RouteLocator`
4. 请求转发
    1. 普通 HTTP 转发
    2. Netty HttpClient

GatewayAutoConfiguration 自动装配的 Bean

- `RouteLocatorBuilder`
- `RouteLocator`
-

JAR-RS 设计（精简）

与 Spring MVC 注解区别

请求映射

请求映射	Spring MVC	JAX-RS
GET 方法	@GetMapping	@GET
POST 方法	@PostMapping	@POST
HTTP 方法	@RequestMapping	@HttpMethod

请求参数处理

JAX-RS 允许在接口上定义，Spring MVC 也允许在接口上定义，不过如果 Java 版本小于 8，或者 Java 8 为开启 -parameters 编译参数时，方法参数的名称可能得不到

JAX-RS 请求参数注解属性没有默认值，与 Spring MVC 相反

`@PathParam` -> `@RequestParam`