Rakuten Interview Questions Cheat Sheet

by woshiamiaojiang via cheatography.com/193748/cs/40392/

code review consideration

Code style
Tests
Documentation
iplementation Semantics
API Semantics
Functionality and Requirements:
Code Clarity and Readability:
Code Style and Formatting:
Error Handling:
Testing and Test Coverage:
Performance and Scalability
Security
Concurrency and Thread Safety
Documentation
Logging and Debugging
easy to Maintainability and Extensibility
Code Comments

pro and con of microservices vs monolithic

Microservices Architecture:
Scalability
Flexibility
Modularity
Rapid Development:
Fault Isolate
Continuous Delivery:
Monolithic Architecture:
not complexity
Performance
Easier Debugging
Development Speed

microservice architecture system design

Decomposition of Services:
Service Boundaries:
API Design:
Data Management:
服务配置中心
服务发现
服务负载均衡
容错机制 Fault Tolerance and Resilience:
timeout, circuit breaker, downgrade, rate limit, retry
日志与监控，告警 skywalking / Prometheus & Grafana Application performance monitor tool for distributed systems
数据一致性问题
Logging and Monitoring:
Security:
Deployment and Orchestration:
Continuous Integration and Deployment (CI/CD):

Linux中 JVM常用命令

top
top -Hp PID 查看占用CPU最高的进程的线程情况
确定线程后，计算线程ID对应的十六进制值： printf "%x\n" <java_thread_id>
将该线程堆栈内容输出： jstack <java_pid> | grep <线程id十六进制值> -A 30 【-A 30表示向下打印30行】
ps：ps命令用于列出当前运行的进程。
ps aux | grep java
jps：它通常用于查找正在运行的Java进程。
jstack：jstack命令用于生成Java线程转储，以便分析Java应用程序中的线程状态和堆栈信息。你需要提供Java进程的PID。
jstack <PID>

Linux中 JVM常用命令 (cont)

jmap：jmap命令用于生成Java进程的内存映像。它可以用于查看堆内存使用情况和分析内存泄漏。
jmap -heap <PID>
jstat：jstat命令用于监视Java应用程序的性-能统计信息，如垃圾回收统计和类加载统计。
jstat -gc <PID>
jcmd：jcmd命令提供了对Java进程的广泛管理和诊断功能。你可以使用它来执行各种操作，如线程转储、堆转储、性能监视等。
jcmd <PID> <命令>
jconsole：jconsole是Java自带的可视化监控工具，用于监视和管理Java应用程序的性能和资源使用情况。

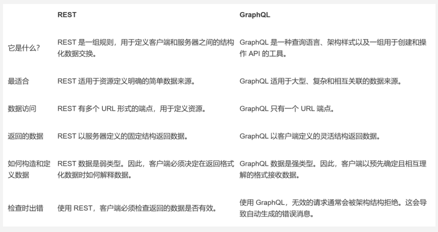
API to handle large amount of traffic

rate limit in the api gateway
pagnation
cache
load test jemeter
Scalable Architecture:
Use Efficient Protocols:
Caching:
Throttling and Rate Limiting:
Asynchronous Processing:
Content Delivery Networks (CDNs):
Authentication and Authorization:
Monitoring and Analytics:
Error Handling and Status Codes:
Pagination and Filtering:
API Versioning:
Security:

API to handle large amount of traffic (cont)

Protect against common security threats, such as SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).
Employ security headers and encryption (e.g., TLS/SSL) for data protection.
Documentation and Developer Support:
Testing and Load Testing:
Failover and Redundancy:
Rate Monitoring and Billing:
Feedback and Improvement:

difference between Rest and GraphQL API



	REST	GraphQL
它是什么?	REST 是一组规则，用于定义客户端和服务器之间的结构化数据交换。	GraphQL 是一种查询语言、架构样式以及一组用于创建和操作 API 的工具。
最适合	REST 适用于资源定义明确的简单数据来源。	GraphQL 适用于大型、复杂和相互关联的数据来源。
数据访问	REST 有多个 URL 形式的端点，用于定义资源。	GraphQL 只有一个 URL 端点。
返回的数据	REST 以服务器定义的固定结构返回数据。	GraphQL 以客户端定义的灵活结构返回数据。
如何构造和定义数据	REST 数据是弱类型。因此，客户端必须决定在返回格式化数据时如何解释数据。	GraphQL 数据是强类型。因此，客户端以预先确定且相互理解的格式接收数据。
检查时出错	使用 REST，客户端必须检查返回的数据是否有效。	使用 GraphQL，无效的请求通常会被架构结构拒绝。这会导致自动生成的错误消息。

API

when you build your api , What technology and what kind of protocol do you use?
Programming Languages
Web Frameworks
Data Storage
Protocol:
HTTP/HTTPS
WebSocket
GraphQL
gRPC
Authentication and Authorization: APIs often require authentication and authorization mechanisms to secure access to data. Common methods include API keys, OAuth 2.0, JWT (JSON Web Tokens), and OAuth 2.0.
Documentation
Testing

API (cont)

Versioning
Rate Limiting and Throttling
Monitoring and Logging
Security
Load Balancing

high traffic API checklist

1. Performance Testing
2. Security Testing
3. Error Handling
4. Rate Limiting
5. Caching
6. Logging and Monitoring
7. Documentation
8. Versioning
9. Backup and Recovery
10. Deployment Strategy
11. Rollback Strategy
12. API Throttling

why hashmap expand at the threadhold of 0.75?

Minimizing Collisions: A lower load factor would lead to fewer collisions because it would create more buckets (and therefore a smaller number of entries in each bucket). However, if the load factor is set too low, the HashMap would require frequent resizing, which is an expensive operation.
Avoiding Frequent Resizing: Resizing a HashMap involves creating a new, larger array and copying all the existing elements into the new array. This is a computationally expensive operation and can lead to performance degradation if it occurs frequently. A load factor of 0.75 strikes a balance between reducing collisions and minimizing the number of resizes.

why hashmap expand at the threadhold of 0.75? (cont)

Memory Efficiency: While a lower load factor would reduce collisions, it would also increase memory usage because it would require more buckets. A higher load factor would reduce memory usage but increase the likelihood of collisions. A load factor of 0.75 aims to strike a balance between these considerations, providing a reasonable compromise between memory efficiency and performance.
Average-Case Performance: A load factor of 0.75 is chosen to ensure that, on average, buckets remain reasonably small, keeping the lookup, insertion, and removal operations close to constant time (O(1)) in typical scenarios. It provides good performance for a wide range of use cases.

hashmap thread safe

在 jdk1.7 中，头插法，在多线程环境下，扩容时会造成环形链或数据丢失。
在 jdk1.8 中，尾插法，在多线程环境下，会发生数据覆盖的情况。
If you need high concurrency and performance, consider using ConcurrentHashMap.
If you have an existing HashMap and want to make it thread-safe with minimal changes, you can wrap it using Collections.synchronizedMap().
If you require a legacy thread-safe solution, Hashtable is an option.

byte type and what byte type in Java

Size: A byte is 8 bits, which means it can represent 2^8 (256) distinct values. In the case of signed integers, the range is typically from -128 to 127.
Signed: In Java, byte is a signed data type

byte type and what byte type in Java (cont)

Default Value: The default value of a byte variable is 0.

byte myByte = 42; // Assigning a positive value

tuning weblogic or something about tomcat

configJVM
configThread Pool
using NIO
config Connection Pools
CDN or Nginx offload static content

SpringBoot authrization

SpringBoot authrization (cont)

```
>        @ExceptionHandler(MethodArgumentNotValidException.class)
        ResponseEntity<String> handleValidationException(MethodArgumentNotValidException ex) {
            String errorMessage = ex.getBindingResult().getFieldError().getDefaultMessage();
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(errorMessage);
        }
        @ExceptionHandler(Exception.class)
        ResponseEntity<String> handleInternalServerError(Exception ex) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Internal Server Error");
        }
    }
    // POST endpoint with request body validation and authorization check
    @PostMapping("/api/post")
    @Secured("ROLE_USER")
    ResponseEntity<String> postEndpoint(
            @Valid @RequestBody RequestData requestData,
            @AuthenticationPrincipal User authenticatedUser
    ) {
        // You can access the authenticated user's information if needed
        String username = authenticatedUser.getUsername();
        // Process the request data (e.g., save it to a database)
        String message = requestData.getMessage();
        return ResponseEntity.ok("Received POST request with message: " + message);
    }
    // GET endpoint with authorization check
```

SpringBoot authrization (cont)

```
>    @GetMapping("/api/get")
    @Secured("ROLE_ADMIN")
    ResponseEntity<String> getEndpoint() {
        // Process the GET request
        return ResponseEntity.ok("Received GET request");
    }
}
```

DB system

what database system:
ralational : mysql , postgre, oracle
nosql
列式column-based ：HBase、Cassandra、ClickHouse
键值：Redis、Memcached
文档 doc：MongoDB
时序time-series ：InfluxDB、Prometheus
搜索：Elasticsearch
graph : neo4j

multithreading environment

Concurrency: progress on multiple tasks at the same time.
Parallelism: Parallelism is a specific form of concurrency。 where multiple threads or processes execute tasks simultaneously on multiple CPU cores or processors.
Thread Safety: multiple threads access shared data simultaneously.
Race Conditions
Deadlocks
Thread Communication: wait/notify (in Java)

```
@SpringBootApplication
public class Spring Boo tAp -
iEx ample {
        public static void
main(S tring[] args) {
                Spr ing App -
lic ati on.r un (Sp rin gBo -
otA piE xam ple.class, args);
       }
       // DTO (Data Transfer
Object) class for request body
validation
        @Va lidated
        static class Reques -
tData {
                @No tBl ank -
(me ssage = " Message cannot be
blank")
                private String
message;
                public String
getMes sage() {
                        return
message;
               }
                public void
setMes sag e(S tring message) {
                        thi -
s.m essage = message;
               }
       }
       // Custom exception
handler for 400 Bad Request and
500 Internal Server Error
        @Re stC ont rol ler -
Advice
        static class Custom -
Exc ept ion Handler {
```

alert on production on errors

tell your team you are dealing with it
Gather Info
Assess the severity of the error.
identify the specific component or service that is causing the error.
Rollback (if applicable)
Prioritize the alert based on its impact and urgency
Notify relevant memebers
Incident Response Plan
Root Cause Analysis
Temporary Fixes
Testing and Deploy the Fix
Monitor
Post-Incident Review
Documentation and Knowledge Sharing

java 1.7 and 1.8?

Lambda Expressions (Java 1.8):
Functional Interfaces (Java 1.8):
Stream API (Java 1.8):
Default Methods (Java 1.8):
Default methods allow interfaces to provide method implementations.
Method References (Java 1.8):
New Date and Time API (Java 1.8):
PermGen Removal (Java 1.8):introduced the Metaspace memory area.
Garbage-First (G1) collector as the default collector

how to keep track of performance metrics

Select Monitoring Tools:Prometheus, Grafana
Instrument Your Code:
Set Up Alerts:
Monitor System Resources: CPU usage, memory usage, disk I/O, and network traffic.
Application-Level Metrics: Collect application-specific metrics related to your business logic and functionality.
Custom Metrics:
Logging and Tracing:
Dashboards:
Data Retention Policy:
Regularly Review and Analyze Metrics:
Scaling Events:
Load Testing:
Documentation:
Continuous Improvement:
Security Monitoring:
Compliance Monitoring:

a database design perspective

constructed type, sime-constructed, non-constructed
db choose
assuption of total data size
write qps and read qps
1. Data Modeling:
2. Indexing:
3. Query Optimization:
4. Partitioning and Sharding:
5. Caching:
6. Connection Pooling:
7. Data Compression and Archiving:
8. Backup and Recovery:
9. Monitoring and Tuning:

a database design perspective (cont)

10. Hardware Considerations:
11. Security:
12. Scalability:

asyncrhonous processing

用户行为采样，前端把数据放进消息中间件中就返回。然后线程池 每10S取一批处理做数据清洗，batch放进clickhouse中。
批量下单，多个任务completableFuture同时下单，全部完成后，统计成功与失败个数。
定时任务，统计数据，CountDownLatch，单个线程等待其他线程池完成后作汇总任务。
线程池的任务都是异步的。

microservice arthitecture

1. Service Independence
2. Single Responsibility
3. Decentralization
4. API-Based Communication
5. Polyglot Persistence
6. Independent Data Management
7. Resilience
8. Scalability
9. Continuous Deployment
10. Monitoring and Observability
11. Docker and Containerization
12. Orchestration and Service Discovery
13. Event-Driven Architectures
14. Incremental Development
15. Domain-Driven Design (DDD)
16. Security

hashmap Treeify and untreeify 时机

Treeify (treeifyBin() method):
Balancing Act: The threshold of 8 is chosen to strike a balance between performance and memory usage. It's an empirical value that, based on extensive testing and analysis, provides a good trade-off for most use cases. When the number of entries in a bucket exceeds 8, the linked list is treeified to improve lookup performance. When the number falls below 6, the tree structure is untreeified to save memory and improve iteration performance.

testing performance test and api test

performance testing include:
Load Testin
Stress Testing
Scalability Testing
Volume Testing
Endurance Testing
Spike Testing
API Testing:
Functional Testing
Integration Testing
Security Testing
Performance Testing
Load Testing
Mocking

kafka rabbit

What is the ideal backend architecture design

API网关 做限流
微服务，服务拆分，无状态，自动扩缩容
NGINX 负载均衡
ELK 日志查询
prometheus & grafana 监控告警
mysql 分库分表 高性能，一主多从，读写分离。搜索走ES。
redis cluster 高可用 缓存 加快响应
异地多活 multi center of services in different cities
全链路压测
定期安全检查
CI CD 自动化部署
perfect documentation of the whole system
message queue for asynchronous communication
数据库数据备份

CI CD

jenkins pipeline
第1步.开发(IDE)提交代码 push events 到项目仓库服务器（gitlab）;
第2步.jenkins开始执行Pipeline代码文件，开始从(gitlab)仓库git clone代码;
第3步.jenkins启动Pipeline里面第一个stage(阶段);
第4步.图里面第一个Stage从仓库检出（-Checkout）代码;
第5步.接着进入第二Stage构建（Build）检出的代码;
第6步.然后进入测试（Test）的阶段，执行各种自动化测试验证; Sonar
第7步.然后测试结束，到运维的部署（Deploy）阶段;

CI CD (cont)

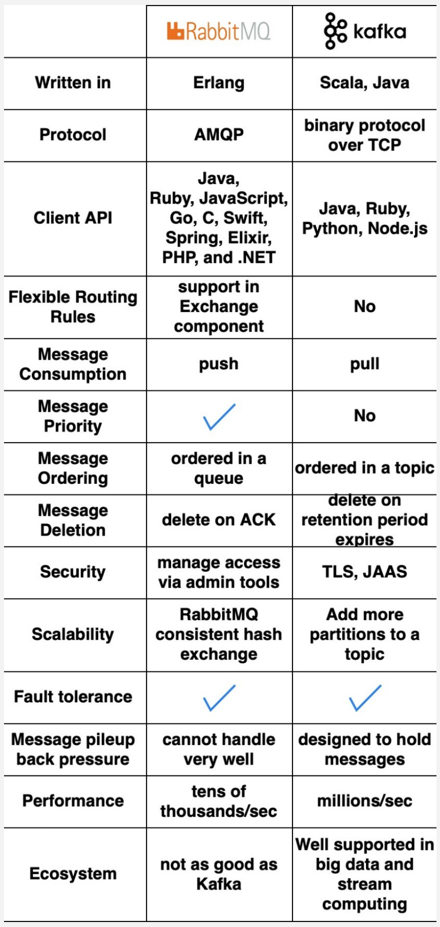
第8步.部署结束，输出报告，整个自动化流程工作完成;

cookie track

你登录某个电商网站，电商网站放了tracking cookie在你的浏览器中。
然后登陆了其他网站，这个网站和google ads合作，把你的cookie给了google,谷歌看到了这个唯一ID，知道了是电商网站放的，给你推送了那个电商网站的广告。

redis于rabbitmq和kafka优势

redis作为消息中间件相对于rabbitmq和kafka有什么优势吗？
Redis的优势：
低延迟
轻量级
多数据类型支持
Redis的限制：
消息持久性：如果你需要保证消息不会丢失，特别是在高负载或故障情况下，Kafka可能更适合。
扩展性：扩展性有限。Kafka和RabbitMQ在这方面表现更出色，可以轻松处理高吞吐量的数据流。
消息顺序性：Redis的发布/订阅模型不保证消息的顺序性
消息保持时间：Redis的消息通常是短暂的



	RabbitMQ	kafka
Written in	Erlang	Scala, Java
Protocol	AMQP	binary protocol over TCP
Client API	Java, Ruby, JavaScript, Go, C, Swift, Spring, Elixir, PHP, and .NET	Java, Ruby, Python, Node.js
Flexible Routing Rules	support in Exchange component	No
Message Consumption	push	pull
Message Priority	✓	No
Message Ordering	ordered in a queue	ordered in a topic
Message Deletion	delete on ACK	delete on retention period expires
Security	manage access via admin tools	TLS, JAAS
Scalability	RabbitMQ consistent hash exchange	Add more partitions to a topic
Fault tolerance	✓	✓
Message pileup back pressure	cannot handle very well	designed to hold messages
Performance	tens of thousands/sec	millions/sec
Ecosystem	not as good as Kafka	Well supported in big data and stream computing

how java's hash map internally work?

Hashing:
hashCode()
first 16 and last 16 do exclusive or % array.length()
the key-value pair is placed in that bucket.
no other key-value pair with the same hash code in the bucket, the new pair is added directly.
Retrieving Values:
Load Factor and Rehashing:
HashMap has a load factor (typically 0.75) that determines when it should resize the internal array.

How about maintenance and production?

Prometheus + ELK
Multi-Site High Availability
update OS patches
Backups and Disaster Recovery
auto-scale
periodic security audits and vulnerability assessments
Maintain an incident response plan
Continuously assess the capacity of your system and anticipate future growth
Regular testing, including load testing, security testing, and disaster recovery testing
Keep documentation up to date
Change Management:
Maintain SLAs and performance targets for your application or service
Encourage a culture of continuous improvement within your operations team

memory cache and qps:

memory cache and qps:
single Redis qps benchmark is around 100 thousand.
Reids Culster can reach around 1 million.
Cassandra: tens of thousands to hundreds of thousands

figure out JVM memory leak

dump easy gc
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/path/to/dumpfile
Analyze Heap Dumps:
Once you have a heap dump, you can analyze it using tools like Eclipse Memory Analyzer (MAT) or VisualVM's Heap Dump Analyzer. These tools can help you identify which objects are causing the memory leak.

method of web security

Input Validation:
Authentication and Authorization:
Secure Password + salt
HTTPS Encryption
Web Application Firewall (WAF):
Security Scanning and Testing:
Penetration Testing
Rate Limiting:
Monitor, log, alert
1. 身份验证（Authentication）：
2. 授权（Authorization）：
3. 使用HTTPS：
4. 输入验证和数据验证：
5. 防止DDoS攻击：
6. 安全标头（Security Headers）： CORS标头、X-Content-Type-Options、X-Frame-Options和Content-Security-Policy

method of web security (cont)

Risk control system blacklist
重要接口做接口加密：前后端约定加密规则，时间戳+请求参数 生成 sign

experience about mem cache

cache mysql query result
spring security + cas + redis : user's session
bitmap 实现 bloomfilter
bitmap 作为用户签到的统计
hyperloglog 统计pv、uv
zset实现延迟队列
distributed lock
setnx + expire+ lua脚本把前两个命令封装原子性。问题：单点故障或者分区网络问题，造成锁被他人拿取。无法支持锁的重入，主从模式可能造成锁丢失，锁无法自动续期。
redlock：极端情况下，会造成两个线程同时获得锁。为了避免该种情况发生，要求宕机的redis在超过锁超时时间后再重启。使用锁的时间要小于锁超时时间。
zk:临时顺序节点，每个节点监听前面节点的释放
redis rate limit: 三种
string, set , expire 实现fixed window counter algorithm
zset，value保持唯一，可以用UUID生成，而score可以用timestamp表示，因为score我们可以用来计算当前时间戳之内有多少的请求数量。而zset数据结构也提供了range方法让我们可以很轻易的获取到2个时间戳内有多少请求。 Sliding window log algorithm

experience about mem cache (cont)

list的leftPop方法来获取令牌，scheduledJob定时任务rightPush。Token bucket algorithm
简单窗口
O(1)
O(1)
容易实现，适用于一些简单的流控场景，流量比较均匀，或者允许临界突变
滑动窗口
O(1)
O(M)-M为子窗口数适用大多数场景，可以通过调节采样子窗口数来平衡开销
漏桶
O(1)
O(1)
要求输出速率恒定的场景，能够平滑流量
令牌桶
O(1)
O(1)
与漏桶类似，区别在于允许一定的突发流量
滑动日志
O(log(N)-取决于O(N)-N为时间窗口内
要求完全精确的控制，保证任意T时刻内流量选择的数据结构允许的最大请求量 不超过N,高时间和空间复杂度，性能最差

common linux command

File and Directory Operations:
ls: List files and directories in the current directory.
cd: Change the current working directory.
pwd: Print the current working directory.
mkdir: Create a new directory.

common linux command (cont)

touch: Create an empty file.
cp: Copy files or directories.
mv: Move or rename files or directories.
rm: Remove files or directories.
find: Search for files and directories.
grep: Search for patterns in text files.
File Viewing and Manipulation:
cat: Concatenate and display file contents.
less or more: View files one page at a time.
head and tail: Display the beginning or end of a file.
nano or vi/vim: Text editors for creating or editing files.
File Permissions:
chmod: Change file permissions.
chown: Change file ownership.
chgrp: Change file group ownership.
User and Group Management:
useradd: Add a new user.
userdel: Delete a user.
passwd: Change user password.
groupadd: Add a new group.
groupdel: Delete a group.
groups: List user's groups.
Process Management:
ps: List running processes.
top or htop: Real-time system monitoring tools.
kill: Terminate processes.
pkill and killall: Kill processes by name.
System Information:
uname: Display system information.
df: Display disk space usage.
free: Display memory usage.
uptime: Show system uptime.

common linux command (cont)

lscpu and lshw: Display CPU and hardware information.
Package Management (Package-Based Distributions):
apt-get or apt: Advanced Package Tool for Debian-based systems (e.g., Ubuntu).
yum or dnf: Package manager for Red Hat-based systems (e.g., CentOS, Fedora).
Networking:
ifconfig or ip: Display network interface information.
ping: Send ICMP echo requests to a host.
netstat or ss: Display network statistics.
ssh: Securely connect to remote servers.
scp: Securely copy files between hosts.
File Compression and Archiving:
tar: Archive files and directories.
gzip or gunzip: Compress or decompress files.
zip and unzip: Create and extract ZIP archives.
File Transfer:
wget and curl: Download files from the internet.
ftp or sftp: Transfer files to/from remote servers.

In java, how to test private methods?

Reflection: While not recommended for production code, you can use Java's reflection API to access and invoke private methods for testing purposes. This allows you to bypass access restrictions and test the private methods directly. Here's an example of how you can use reflection to test a private method:
MyClass myClass = new MyClass();
Method privateMethod = MyClass.class.g-etDeclaredMethod("privateMethod", int.class, int.class);

In java, how to test private methods? (cont)

privateMethod.setAccessible(true); // Allow access to private method
int result = (int) privateMethod.invoke(myC-lass, 3, 4);
System.out.println(result); // Output: 7
使用PowerMock测试私有方法:
Object result = Deencapsulation.invoke(m-ockClass, methodName, parameter1, parameter2....)

large amount of traffic

load balancer
rate-limit in api gateway
microservices auto-scale
Circuit Breaker Pattern Hystrix
whole-chain pressure test
sharding mysql
Monitoring and Alerts
redis cache cache warming
CDNs
Database Optimization
:

how many transaction request are you getting

my teams's system should be around 2k qps
my user bebehavior tracking system should be around 500 qps.

unit testing integration

```
Integration Testing in Java:
Setup Testing Enviro nment:
Integr ation testing often
requires setting up a test
enviro nment that resembles the
production enviro nment. This
includes config uring databases,
external services, and any other
depend encies.
Choose a Testing Approach:
```

unit testing integration (cont)

> Embedded Containers: You can use embedded containers like Spring Boot's @SpringBootTest for testing Spring-based applications. These containers provide a controlled environment for integration testing.
Docker Containers: Another approach is to use Docker containers to spin up test versions of external services and databases. Tools like Testcontainers can help with this.
Write Integration Tests: Write test cases that focus on testing the interactions between different components or services. These tests typically cover scenarios like API calls, database operations, and messaging between components.
java
Copy code
import org.junit.jupiter.api.Test;
import org.springframework.beans.fact-ory.annotation.Autowired;
import org.springframework.boot.test.con-text.SpringBootTest;
import org.springframework.boot.test.w-eb.client.TestRestTemplate;
import org.springframework.boot.web.serv-er.LocalServerPort;
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RAN-DOM_PORT)
public class MyIntegrationTest {
  @LocalServerPort
  private int port;
  @Autowired
  private TestRestTemplate restTemplate;
  @Test
  public void testApiEndpoint() {

unit testing integration (cont)

>   String url = "http://localhost:" + port + "/api/some-endpoint";
    String response = restTemplate.getFo-rObject(url, String.class);
    // Perform assertions on the response
  }
}
Run Integration Tests: Similar to unit tests, use your build tool or IDE to run integration tests. Integration tests may take longer to execute than unit tests because they often involve external dependencies.
Clean Up: Ensure that your tests clean up any resources or data created during testing to leave the environment in a clean state.

asynchronized API features in Java?

CompletableFuture: asynchronous tasks, combining them, and handling their results or exceptions using a fluent API.
Async Servlets: This allows your web server to handle more requests with fewer threads, improving scalability.
Java NIO (New I/O)
Java Concurrency Utilities: JUC CountD-ownLatch 单线程等待其他线程做统计，Cy-clicBarrier 多线程等待，semaphore
Reactive Programming: Libraries like Reactor and RxJava enable reactive programming
Java 9+ CompletableFuture Enhancements: combining multiple futures, dealing with timeouts

self introduction

Hi, my name is Andrew. I am a java backend engineer with 5 years experience. I am specilized in developing scalable and reliable applications using Java, Spring Framework, MySQL, Redis, Kafka and etc.
My last job is in QTrade, it's a company doing the business of helping financial companies with compliance.
My team is the infrastructure team. My responsibility is solving online issues and optimizing system. Building some middleware like Canal to symchronize data from MySQL to Redis and ElasticSearch. Building some basic system like user behavior tracking and reporting system.
Before QTrade, I worked in SF Technology. In SF technology, I was in the SF mini program team. My responsibility is developing new features and functions for SF mini program. SF mini program is a popular built-in 3rd-party app which has around 200 thousand qps.
I apply to this position because I have huge interest in Japanese culture. When I was a child, I watched a lot of Japanese cartoons. And I learned some basic Japanese before. Rakuten is an international company with multi-culture. Its tech stack is almost the same as I am using.
Thank you. That's the whole introduction about myself as a 5-years java backend engineer.

day to day activities & contribution to the team

monring stand up to follow up the shedule, talk about potential issues
if there's urgent online issues ,will solve it first

day to day activities & contribution to the team (cont)

communicate the details of the requirement with product manager, finish the code part , self test , write api documentation, communicate with frontend engineer, as scheduled which in agile it's called spur
before release the production version, there will be code review
contributions: solve online issues and optimize the system, build some middle-ware, guide some junior engieers

question time

what is a work day like in this team?
what are the tech stacks that the team is currently using?
what difficulty and challenge the team is facing?
what's the manager's plan for me to do?
what do I need to learn in advance?

question time (copy)

what is a work day like in this team?
what are the tech stacks that the team is currently using?
what difficulty and challenge the team is facing?
what's the manager's plan for me to do?