Table 1: Concepts to be explored

|  |  |  |
| --- | --- | --- |
| Concept or Technology at the concept Level | Example Resource | SEVERITY |
| Idempotency |  |  |
| GraphQL & Netflix Falcor |  |  |
| Open API, Swagger & IDLs in general. What are client-side stubs and server-side skeletons? |  |  |
| REST APIs once and for all and how the benefits that Fielding mentions are achieved by complying to the REST constraints | <https://shorturl.at/gFhV8> |  |
| Maven scopes and why Lombok is provided scope for example |  |  |
| Authorization Methods (RBAC, ABAC, …) details and implementation. Blocked Ips, Password failures and time-outs, … more about security policies. |  |  |
| Soft deletes in DB and how Hibernate’s dynamic mapping helps you do so. |  |  |
| Encryption and Hashing methods: SHA(256), MD5, PBE, base4, etc. |  |  |
| Git internal architecture and more advanced commands like cherrypick, squash, … |  |  |
| Spring application events, the Observer pattern, and annotation-based event listeners |  |  |
| HTTP in more depth: cookies, headers, caching, etc. why is it firewall friendly? Other verbs, … |  |  |
| User impersonation in spring security |  |  |
| Elastic Search |  |  |
| Class loader, Memory, execution cycle, JVM and JVM references, java.lang.ref and its use in caches, Garbage Collection, JRE vs JDK, JIT vs interpreter, GraalVM, etc. class path, compilation class path vs execution class path. How to tune garbage collection and heap and stack memories.  Disk swapping, disk paging, out of memory error, and memory footprint and its effect on performance, how to optimally set heap and stack sizes | <https://www.geeksforgeeks.org/differences-jdk-jre-jvm/>  [https://javarevisited.blogspot.com/2011/04/garbage-collection-in-java.html#](https://javarevisited.blogspot.com/2011/04/garbage-collection-in-java.html%23)  <https://javarevisited.blogspot.com/2019/04/top-5-courses-to-learn-jvm-internals.html>  https://objectcomputing.com/resources/publications/sett/june-2000-collaborating-with-the-java-memory-manager |  |
| Java8, scope of variables in lambdas, method references etc.. |  |  |
| Literals in Java, d, f, … |  |  |
| Transparency in dependent classes like in the context of JPA, or test classes to the code under test. |  |  |
| Sealed access modifier in java 17 |  |  |
| Multi-tenancy vs Row-level security vs ACL |  |  |
| PostgreSQL’s Generalized Search Tree and multi-dimensional indexes |  |  |
| More on Enums, overring its methods, indices, etc. |  |  |
| Excellent resource on memory and synchronization  <https://jenkov.com/tutorials/java-concurrency/java-memory-model.html> |  |  |
| LockSupport. park in concurrency |  |  |
| Fail-fast and fail-safe iterations |  |  |
| I/O buffering and the difference between System.out and System.err in this context |  |  |
| Digest authentication |  |  |
| Remember-me tokens, session management in depth and persisting shopping cards |  |  |
| Can you say that session cache is inline cache? Like write back on eviction? |  |  |
| Redis Gears |  |  |
| CDNs |  |  |
| UDP, and other protocols |  |  |
| Object Storage Database |  |  |
| Database dump |  |  |
| Virtual threads and how they help with asynchrony(vinkat) |  |  |
| Saga, Camunda, Apache Camel, and more  https://www.baeldung.com/cs/saga-pattern-microservices |  |  |
| How Redis simplifies microservices design patterns:  <https://thenewstack.io/how-redis-simplifies-microservices-design-patterns/> |  |  |
| Redis as cache at the network(http) level. Using reverse proxies like Nginx, and headers like cache-control, expires-at, last-modified. Search HTTP caching. Go to this series:  <https://docs.oracle.com/cd/E13183_01/en/alui/devdoc/docs6x/aluidevguide/tsk_pagelets_settingcaching_httpcachecontrol.html>  <https://docs.oracle.com/cd/E13183_01/en/alui/devdoc/docs6x/aluidevguide/tsk_pagelets_settingcaching_httpexpires.html>  <https://docs.oracle.com/cd/E13183_01/en/alui/devdoc/docs6x/aluidevguide/tsk_pagelets_settingcaching_httplastmodified.html> | <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Caching> |  |
| Encryption of users’ personal data and data anonymization |  |  |
| Self-signed JWT in spring security |  |  |
| Iterator in Java |  |  |
| Map reduce |  |  |
| CompletableFuture, ForkJoinPool, ForkJoinTask, and why ForkjoinPool is not good for blocking tasks |  |  |
| CAP theorem |  |  |
| The Bridge and the Decorator patterns |  |  |
| Bitmap and bloom filters |  |  |
| Compound indexes in Mongo DB and SQL |  |  |
| Pattern Matching in java |  |  |
|  |  |  |
|  |  |  |
| Time Series |  |  |
| Search like in books or movie websites |  |  |
| B-tree |  |  |
| Arrays in terms of being objects like enums? and the Array class |  |  |
| Optimistic locks <https://www.baeldung.com/jpa-optimistic-locking>, pessimistic locks <https://www.baeldung.com/jpa-pessimistic-locking>  And when to use which and these vs native support for Isolation in database transactions |  |  |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-configuration-processor</artifactId>  <optional>true</optional> </dependency> |  |  |

Table 2: Skills to be acquired

|  |  |  |
| --- | --- | --- |
| Skill | EXample Resource | SEVERITY |
| GraphQL |  |  |
| gRPC and async gRPC |  |  |
| Writing Solid Tests | * <https://www.youtube.com/watch?v=rUbjV3VY1DI> * <https://www.youtube.com/watch?v=jqwZthuBmZY&list=PL82C6-O4XrHcg8sNwpoDDhcxUCbFy855E> * <https://www.youtube.com/watch?v=hR0bbk2tsF0&list=WL&index=2> * <https://youtu.be/-H5sud1-K5A?si=NqF-ucjVHweeEsEQ> * <https://youtu.be/2bTAb-2vhBk?si=h4xgDcYlykHPEl_r> * <https://docs.spring.io/spring-boot/reference/testing/index.html#testing> |  |
| Spring cloud config |  |  |
| Serialization in Java,  More of Jackson like @JsonSerialize |  |  |
| Concurrency in Java |  |  |
| Multi-module Maven Projects, Parent POMs, BOMs, and aggregate POMs. |  |  |
| Hibernate Dynamic Mapping (@Filter, @Where, etc.) | <https://www.baeldung.com/hibernate-dynamic-mapping> |  |
| @Async in Spring |  |  |
| Using abstraction in spring beans and how they implemented the service layer using it. |  |  |
| MapStruct |  |  |
| Caching in Spring, Second level cache in hibernate, caching in REST, Materialized Views, Application Cache, and caches in general. |  |  |
| Calculated columns and using @Formula in hibernate |  |  |
| Handling Blobs in Database and file storages |  |  |
| Configuration management in spring |  |  |
| How they write things like orElseThrow, buildAndThrow for builders, etc. |  |  |
| Sending http requests in java |  |  |
| Aggregate operators, reduction, parallelism in java collections |  |  |
| Floyd’s Cycle finding algorithm |  |  |
| Annotation processors |  |  |
| Date and Time |  |  |
| Intellij shortcuts |  |  |
| Validation technique in redirect uri in oauth2 authz server in  RegisteredClientBuilder |  |  |
| Spring data projection and query by example |  |  |
| Azizkhani search option, EntityMapper,  EntityId in VMs etc. |  |  |
| Java.text and the Message.format class |  |  |
| Instance.io |  |  |
| JsonPath |  |  |
| Parameter objects for executing commands in aggregate root:  Execute (Command cmd) |  |  |
| @Bean in Lite mode |  |  |
|  |  |  |
| Bean definition profiles vs properties profiles vs build profiles |  |  |
| https://appwrite.io/blog/post/enums-api-design |  |  |
| <https://docs.spring.io/spring-boot/docs/2.0.0.M4/reference/html/boot-features-developing-auto-configuration.html>  <https://www.youtube.com/watch?v=6u6PJXTb1cQ>  <https://medium.com/@mahammadkhalilov/understanding-spring-boots-internal-magic-factories-conditional-configuration-and-ace04f399059> |  |  |
| String formats, the Formatter class, etc |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table 3: Questions to be answered

|  |  |
| --- | --- |
| Question | SEVERITY |
| Why isn’t declaring operations in REST APIs as query parameters or using sub-resources like order/cancel REST compliant? |  |
| Why Should PUT requests be idempotent and how do you treat update operations that are not? |  |
| Synchronous vs Asynchronous vs Reactive vs Non-Blocking vs Blocking |  |
| CGLIB vs Dynamic Proxies vs AspectJ |  |
| Difference between key-value and document stores |  |
| How not to ruin git history when renaming files |  |
| What are the use cases of inner classes? |  |
| Read-only mode in hibernate and its special behavior when dealing with associations especially mapped collections. And remove operations |  |
| How are Java SE codes and other libraries integrated in your application? |  |
| How are arrays implemented behind the scenes in Java |  |
| Can you set transient entities for associations in JPA entities when persisting the parent? Most likely not, but how can you just set an id for it and you will be good? |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table 4: Here and There stuff

|  |  |
| --- | --- |
| Item | SEVERITY |
| Ipconfig/(preferred) DNS/IP settings on windows/gateway/network mask/etc.  [\\192.0.0.1](file:///\\192.0.0.1) in windows file explorer. |  |
| Triggers/with(CTE) clause/exists/ |  |
| Docker/Docker compose in spring |  |
| How does Gitlab and nexus work locally especially Gitlab auth. |  |
| Spring Common Logging spring-jcl |  |
| Camunda framework |  |
| Spel |  |
| <servers>, <mirrors> and setting.xml in Maven |  |
| How to log people’s actions and how to use Elasticsearch and Redis for logging purposes. How can Kafka be used in this context?  How did people in Vira use AOP to do so with @Async? |  |
| Marker annotations and how to use them for debugging |  |
| FreeMarker Java template engine |  |
| SSL, TLS, SSL redirect, Socket Programming  <https://auth0.com/blog/the-tls-handshake-explained/> |  |
| LDAP vs DB vs FTP |  |
| Schema in @Table |  |
| The Idea of having something like BaseInformation |  |
| Named queries in hibernate and using them to call stored procedures |  |
| Kent Beck, Extreme Programming Explained: Embrace Change |  |
| The difference between @Service, @Repository,… |  |
| Jhipster |  |
| Good DSA document: <https://www.wscubetech.com/resources/dsa> |  |
| How do you preserve order of events when publishing to/consuming from message brokers? How do you make sure publishing or consuming an event more than once won’t cause a problem? How do you make sure orders are preserved in these scenarios? How can using multiple threads or multiple instances of an app make problems in that regard? |  |
| Heap/memory profiler to detect memory leaks |  |
| Call-backs in java <https://www.baeldung.com/java-callback-functions> |  |
| 418 error code |  |
| <https://github.com/rieckpil/blog-tutorials/tree/master>  good spring boot examples |  |
| Aliasing problem with mutable objects and how value objects solve it |  |
| What is this? PostgreSQLContainer<SELF extends PostgreSQLContainer<SELF>> extends JdbcDatabaseContainer<SELF>  The same generic use happens in the enum class I guess |  |
| Learn Spring DI in more details:  <https://docs.spring.io/spring-framework/reference/core/beans/dependencies/factory-collaborators.html>  https://docs.spring.io/spring-framework/reference/core/beans.html |  |
| The effect of the volatile and synchronized key word on the instruction orders in compiler |  |
| https://vladmihalcea.com/hibernate-hbm2ddl-auto-schema/ |  |
| When to use checked vs unchecked exceptions |  |
| Write conflicts in databases that use optimistic locks or MongoDB or databases with multi-version CC and using retries and the possibility of using an exception translator that maps those exceptions to something that might be useful |  |
| Read Preference, Write Concern, and WriteResultChecking Policy in MongoDB  Also Entity Call-backs, MongoTemplate#useEstimatedCount, and Entity Events |  |
| Pool-Induced Deadlocks |  |
| flatmap |  |
| Type casting, long, int why BigDecimal.of(long i) accepts int |  |
| Hi/Lo id generation |  |
| Testing package private classes |  |
| @Role in spring context |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

……………………………………………………………………………………….…………………………………………………………………………

Interesting:

[https://docs.spring.io/spring-framework/reference/core/beans/java/bean-annotation.html#beans-java-declaring-a-bean](https://docs.spring.io/spring-framework/reference/core/beans/java/bean-annotation.html%23beans-java-declaring-a-bean)

public interface BaseConfig {

@Bean

default TransferServiceImpl transferService() {

return new TransferServiceImpl();

}

}

@Configuration

public class AppConfig implements BaseConfig {

}

You can also declare your @Bean method with an interface (or base class) return type, as the following example shows:

@Configuration

public class AppConfig {

@Bean

public TransferService transferService() {

return new TransferServiceImpl();

}

}

However, this limits the visibility for advance type prediction to the specified interface type (TransferService). Then, with the full type (TransferServiceImpl) known to the container only once the affected singleton bean has been instantiated. Non-lazy singleton beans get instantiated according to their declaration order, so you may see different type matching results depending on when another component tries to match by a non-declared type (such as @Autowired TransferServiceImpl, which resolves only once the transferService bean has been instantiated).

|  |  |
| --- | --- |
|  | If you consistently refer to your types by a declared service interface, your @Bean return types may safely join that design decision. However, for components that implement several interfaces or for components potentially referred to by their implementation type, it is safer to declare the most specific return type possible (at least as specific as required by the injection points that refer to your bean). |

……………………………………………………………………………………….…………………………………………………………………………

Although field injection is discouraged in production code, field injection is actually quite natural in test code. The rationale for the difference is that you will never instantiate your test class directly. Consequently, there is no need to be able to invoke a public constructor or setter method on your test class.

……………………………………………………………………………………….…………………………………………………………………………