Spring Professional: Q & A

Table of Contents

1. Container, Dependency, and IOC

What is dependency injection and what are the advantages?

Dependency injection is a pattern used to create instances of objects that other objects rely on without knowing at compile time which class will be used to provide that functionality.

What is an interface and what are the advantages of making use of them in Java?

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. Interfaces provide abstraction from the specific implementations.

What is meant by "application-context" and how do you create one?

The ApplicationContext is the central interface within a Spring application for providing configuration information to the application.

- `ApplicationContext context = SpringApplication.run( ApplicationConfig.class );`

What is the concept of a "container" and what is its lifecycle?

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction. The Spring container uses dependency injection (DI) to manage the components that make up an application.

Dependency injection using Java configuration

Using `@Bean` annotation.

Dependency injection in XML, using constructor or setter injection

-

Dependency injection using annotations (@Component, @Autowired)

-

Component scanning, Stereotypes and Meta-Annotations

`@Service`, `@Respository`, `@Component`, ...

Scopes for Spring beans. What is the default?

Defaule scope: Singleton. Only once instance per bean.

What is an initialization method and how is it declared in a Spring bean?

The init-method attribute specifies a method that is to be called on the bean immediately upon instantiation. XML: `init-method`, Annotation: `@PostConstruct`.

What is a destroy method, how is it declared and when is it called?

These methods are called at shutdown prior to destroying the bean instance. XML: `destroy-method`, Annotation: `@PreDestroy`.

What is a BeanFactoryPostProcessor and what is it used for?

Applies transformations to bean definitions before objects are actually created. Used to change the bean definition e.g., for reading properties, or registering a custom scope.

What is a BeanPostProcessor and how is the difference to a BeanFactoryPostProcessor? What do they do? When are they called?

- **BeanFactoryPostProcessor**: Can modify bean definitions. Run after bean definitions got loaded and before instantiating the beans.
- **BeanPostProcessor**: Can modify bean instances in any way. May run before or after the initialize step.

Are beans lazily or eagerly instantiated by default? How do you alter this behavior?

Each bean is eagerly instantiated by default in right order with its dependencies injected. Change with `@Lazy("true")` or XML `lazy-init="true"`.

What does component-scanning do?

By using `@ComponentScan("⋯")` and pointing the base package, Spring will auto-discover and wire the components into the Spring container.

What is the behavior of the annotation @Autowired with regards to field injection, constructor injection and method injection?

- **Field:** Optional and circular dependencies, even for private fields. Inherited automatically.
- **Constructor**: Mandatory, immutable dependencies. Concise (pass several params at once).
- **Setter**: Similar to field injection, better testability.

How does the @Qualifier annotation complement the use of @Autowired?

Prevent disambiguities. NoSuchBeanDefinitionException is thrown at start-up in case no unique bean is defined. Use the `@Qualifier("⋯")` annotation and explicit bean ids to refer to these.

What is the role of the @PostConstruct and @PreDestroy annotations? When will they get called?

They are lifecycle methods.

- **@PostConstruct**: Called after construction (after setter injection).
- **@PreDestroy**: Called before destroying the instance.

IMPORTANT	Both do not get invoked for prototype beans.

What is a proxy object and what are the two different types of proxies Spring can create?

Beans get wrapped in **dynamic (JDK) proxys** which are created in the init phase by dedicated BeanPostProcessors. There are *CGLib* and *JDK* proxies.

What is the power of a proxy object and where are the disadvantages?

Behaviour can be added dynamically. Dynamic proxies require interfaces. CGLib works around this limitation using a subclass approach. On the downside, constructor injection is no longer possible.

What are the limitations of these proxies (per type)?

- **JDK proxy**: Requires interfaces.
- **CGLib proxy**:
  - Constructor injection not possible.
  - Cannot override final methods.

How do you inject scalar/literal values into Spring beans?

Using `@Value` after defining a `@PropertySource`.

How are you going to create a new instance of an ApplicationContext?

```
ApplicationContext context = SpringApplication.run( ApplicationConfig.class );
```

TODO: ApplicationContext applicationContext = new ClassPathXmlApplicationContext("/application-context.xml");

What is a prefix?

In the `@PropertySource` path, you can use `classpath:` (default), `file:` and `http:` prefixes.

What is the lifecycle on an ApplicationContext?

Initialization → Use → Destruction

What does the "@Bean annotation do?

To declare a bean, simply annotate a method with the @Bean annotation.

How are you going to create an ApplicationContext in an integration test or a JUnit test?

1. Run the test with `@RunWith(SpringJUnit4ClassRunner.class)`
2. Implement `ApplicationContextAware` class.
3. Override `setApplicationContext` method to get the context.

What do you have to do, if you would like to inject something into a private field?

Either @Autowired directly on the field, provide a setter or constructor injection.

What are the advantages of JavaConfig? What are the limitations?

- Pros
  - Is centralized in one (or a few) places
  - Write any Java code you need
  - Strong type checking enforced by compiler (and IDE)
  - Can be used for all classes (not just your own)
- Cons
  - More verbose than annotations

What is the default bean id if you only use "@Bean"?

The method/parameter/field name.

Can you use @Bean together with @Profile?

Yes, if done so the bean will only be created if the profile is active.

What is Spring Expression Language (SpEL for short)?

The Spring Expression Language (SpEL) is a powerful expression language that supports querying, manipulating as well as evaluating logical and mathematical expressions

What is the environment abstraction in Spring?

Environment object used to obtain properties from runtime environment e.g., JVM System Properties or Java Properties Files.

What can you reference using SpEL?

Variables, Functions.

How do you configure a profile. What are possible use cases where they might be useful?

Profiles can represent purpose: “web”,“offline” or environment: “dev”, “qa”, “uat”, “prod”.

How many profiles can you have?

0..*

How do you enable JSR-250 annotations like @PostConstruct?

They must be in a @Configuration class. Import javax.annotation.

Why are you not allowed to annotate a final class with @Configuration

Because Spring wont’t be able to create a CGlib proxy.

Why must you have a default constructor in your @Configuration annotated class?

Spring needs this to be able to instantiate it.

Why are you not allowed to annotate final methods with @Bean?

Because Spring proxies the method to be able to return one cached bean instance (singleton) for every method call.

What is the preferred way to close an application context?

Call `ApplicationContext.close()`.

How can you create a shared application context in a JUnit test?

1. Run the test with `@RunWith(SpringJUnit4ClassRunner.class)`
2. Implement `ApplicationContextAware` class.
3. Override `setApplicationContext` method to get the context.

NOTE	Annotate test method with @DirtiesContext to force recreation of the cached ApplicationContext if method changes the contained beans.

What does a static @Bean method do?

Static beans are created first. It ensures property-sources are read before any @Configuration bean using @Value is initialized.

What is a ProperySourcesPlaceholderConfigurer used for?

Resolves ${...} placeholders within bean definition property values and @Value annotations against the current Spring Environment and its set of PropertySources.

What is a namespace used for in XML configuration

They allow hiding of actual bean definitions.

If you saw one of the <context/> elements covered in the course, would you know what it does?

Define component scanning and property sources.

What is @Value used for?

To inject scalar/literal values into Spring beans.

What is the difference between $ and # in @Value expressions?

`$` to read properties and `#` for SpEL expressions.

2. Aspect oriented programming

What is the concept of AOP? Which problem does it solve?

Aspect-Oriented Programming (AOP) enables modularization of cross-cutting concerns. A cross-Cutting Concern is ageneric functionality that is needed in many places in your application e.g., Logging and Transaction Management.

What is a pointcut, a join point, an advice, an aspect, weaving?

- **Join Point**: A point in the execution of a program such as a method call or exception thrown.
- **Pointcut**: An expression that selects one or more Join Points.
- **Advice**: Code to be executed at each selected Join Point.
- **Aspect**: A module that encapsulates pointcuts and advice.

- **Weaving**: Technique by which aspects are combined with main code.

How does Spring solve (implement) a cross cutting concern?

Java-based AOP framework with AspectJ integration. Uses dynamic proxies for aspect weaving. Focuses on using AOP to solve enterprise problems.

Which are the limitations of the two proxy-types?

If the target object to be proxied implements at least one interface then a JDK dynamic proxy will be used. All of the interfaces implemented by the target type will be proxied. If the target object does not implement any interfaces then a CGLIB proxy will be created. However, final methods cannot be advised, as they cannot be overridden.

How many advice types does Spring support. What are they used for?

- **@Before**: Proxy → BeforeAdvice → Target
- **@AfterReturning**: Proxy → Target(success) → AfterReturningAdvice
- **@AfterThrowing**: Proxy → Target (exception thrown) → AfterThrowingAdvice
- **@After**: Proxy → Target (successful or exception) → AfterAdvice
- **@Around**: Proxy → AroundAdvice → Target → AroundAdvice

What do you have to do to enable the detection of the @Aspect annotation?

Use `@ComponentScan` and apply `@EnableAspectJAutoProxy` to the configuration.

Name three typical cross cutting concerns.

Logging, Transaction Management, Security, Caching, Error Handling, Performance Monitoring, Custom Business Rules.

What two problems arise if you don't solve a cross cutting concern via AOP?

Code tangling and scattering.

What does @EnableAspectJAutoProxy do?

Configures Spring to apply @Aspect to you beans.

What is a named pointcut?

You can refer to the pointcut by its name, so if you have multiple advices referring to the same pointcut you only need to change it in one place and leave the references untouched.

How do you externalize pointcuts? What is the advantage of doing this?

Externalize into a `aop.xml` file. All pointcuts at the same place, separated from Java code.

What is the JoinPoint argument used for?

Provides context information on the intercepted point e.g., method name and signature.

What is a ProceedingJoinPoint?

Inherits from JoinPoint and adds the proceed() method which actually invokes the method. Method invocation is up to the Advice.

Which advice do you have to use if you would like to try and catch exceptions?

@AfterThrowing, @After or @Around advice.

What is the difference between @EnableAspectJAutoProxy and <aop:aspectj-autoproxy>?

XML vs. Java configuration.

3. Data Management: JDBC, Transactions, JPA, Spring Data

What is the difference between checked and unchecked exceptions?

- **Checked Exceptions**: Must be declared in method signatures and handled the whole chain up.
- **Unchecked Exceptions**: Can be thrown up the call hierarchy to the best place to handle it. No need to catch.

Why do we (in Spring) prefer unchecked exceptions?

Intermediate methods must declare exception(s) from all methods below (form of tight-coupling). Unchecked exceptions do not have to be catched → slighter.

What is the data access exception hierarchy?

It hides whether you are using JPA or anything else and provides several unchecked exceptions which are consistent between different technologies.

How do you configure a DataSource in Spring? Which bean is very useful for development/test databases?

Spring provides an `EmbeddedDatabaseBuilder` to conveniently define a new (empty) in-memory database. HSQL, H2 and Derby are supported.

What is the Template design pattern and what is the JDBC template?

Define the outline or skeleton of an algorithm. Leave the details to specific implementations later.

Hides away large amounts of boilerplate cod The JdbcTemplate provides full access to the standard JDBC constructs and handles SQLExceptions.

What is a callback? What are the three JdbcTemplate callback interfaces described in the notes? What are they used for? (You would not have to remember the interface names in the exam, but you should know what they do if you see them in a code sample).

- `RowMapper`: Maps a single row of a ResultSet to an object and can be used for single- and multiple- row queries.
- `RowCallbackHandler When there is no return object e.g., for streaming or conversion stuff.
- `ResultSetExtractor`: You are responsible for iterating the ResultSet. Useful e.g. for mapping entire ResultSet to a single object.

Can you execute a plain SQL statement with the JDBC template?

Yes, with the `queryFor⋯` or `update⋯` methods.

Does the JDBC template acquire (and release) a connection for every method called or once per template?

Depends on the definition of the datasource. If connection pooling is activated, the template may use any connection from the pool.

Is the JDBC template able to participate in an existing transaction?

No. JdbcTemplate does not provide transactions out of the box. TransactionManager has to be defined.

What is a transaction? What is the difference between a local and a global transaction?

Transactions are a set of tasks which take place as a single, indivisible action. **Local Transactions** are managed by underlying resource: App ⇒ Database. **Global (distributed) Transactions** are managed by separate, dedicated transaction manager. App ⇒ TRX manager ⇒ PSQL, RabbitMQ, …

Is a transaction a cross cutting concern? How is it implemented in Spring?

Yes, implemented using AOP:

1. Target object wrapped in a proxy (Around advice)
2. Proxy implements the following behavior
    a. Transaction started before entering the method
    b. Commit at the end of the method
    c. Rollback if method throws a RuntimeException (default, can be overridden)
3. Transaction context bound to current thread.
4. All controlled by configuration

How are you going to set up a transaction in Spring?

There are 3 steps: declare a PlatformTransactionManager bean and the transactional methods (annotationm, XML or programatic). @EnableTransactionManagement defines a BPP for @Transactional beans.

What does @Transactional do? What is the PlatformTransactionManager?

@Transactional defines that all queries in a method will be executed in the same transaction. The PlatformTransactionManager hides implementation details.

What is the TransactionTemplate? Why would you use it?

Template class that simplifies programmatic transaction demarcation and transaction exception handling.

What is a transaction isolation level? How many do we have and how are they ordered?

4 levels:

- **READ_UNCOMMITTED**: Lowest level - allows dirty reads.

- **READ_COMMITTED**: Does not allow dirty reads (only committed information can be accessed)
- **REPEATABLE_READ**: Does not allow dirty reads. Non-repeatable reads are prevented.
- **SERIALIZABLE**: Prevents non-repeatable reads and dirty-reads and also phantom reads.

What is the difference between @EnableTransactionManagement and <tx:annotation-driven>?

No difference.

How does the JdbcTemplate support generic queries? How does it return objects and lists/maps of objects?

`query(…)`, `queryForList`, `queryForMap(…)`

What does transaction propagation mean?

Transaction propagation is calling a @Transactional method within another @Transactional method. There are 7 levels of propagation e.g., `REQUIRED` and `REQUIRES_NEW`.

What happens if one @Transactional annotated method is calling another @Transactional annotated method on the same object instance?

The device will not be calles twice (limitation of AOP). If you call a method with a @Transactional annotation from a method with @Transactional within the same instance, then the called methods transactional behavior will not have any impact on the transaction.

Where can the @Transactional annotation be used? What is a typical usage if you put it at class level?

On class the @Transactional applies to all methods declared in the interface(s). You can also declare it on both, class- and method- level but with different settings e.g., timeouts.

What does declarative transaction management mean?

Specify transaction behavior declaratively down to individual method level. Separates transaction demarcation from transaction implementation

What is the default rollback policy? How can you override it?

By default, a transaction is rolled back if a RuntimeException has been thrown. Default settings can be overridden with rollbackFor and noRollbackFor attributes.

What is the default rollback policy in a JUnit test, when you use the SpringJUnit4ClassRunner and annotate your @Test annotated method with @Transactional?

Runs the test method in a transaction and rolls back afterwards. Using the @Commit annotation, the transaction won't be rolled back.

Why is the term "unit of work" so important and why does JDBC AutoCommit violate this pattern?

Some updates may belog to one common logical unit of work e.g., removing and adding an amount when transfering money. This is an all-or-nothing operation.

What does JPA mean - what is ORM? What is the idea behind an ORM?

- **JPA (Java Persistence API)**: Designed for operating on domain objects: Pojos, no interfaces. It is a common API for object-relational mapping.
- **ORM(Object Relational Mapping)**: Technique for converting data between incompatible type systems in object-oriented programming languages.

What is a PersistenceContext and what is an EntityManager. What is the relationship between both?

The EntityManager manages a unit of work and persistent objects therein: the PersistenceContext.

Why do you need the @Entity annotation. Where can it be placed?

The @Entity annotation marks an entity object which is mapped to a table. Optionally define table name. To be plasses on class-level.

What do you need to do in Spring if you would like to work with JPA?

1. Define an `EntityManagerFactory` bean.
2. Define a `DataSource` bean
3. Define a `TransactionManager` bean
4. Define Mapping Metadata
5. Define DAOs

Are you able to participate in a given transaction in Spring while working with JPA?

Yes. Transparently participate in Spring-driven transactions:

- Use a Spring FactoryBean for building the EntityManagerFactory
- Inject an EntityManager reference with @PersistenceContext

What is the PlatformTransactionManager?

Spring's PlatformTransactionManager is the base interface for the abstraction of the transaction strategy.

What does @PersistenceContext do?

Inject an EntityManager reference with @PersistenceContext.

What are disadvantages or ORM? What are the benefits?

+ Allows converting data between incompatible type systems. + Transaction management + Hides technologies

- Performance overhead

What is an "instant repository"? (hint: recall Spring Data)

Spring Data provides "instant" repositories. Define a "CrudRepository" and Spring Data will automatically inject a backing implementation and add common CRUD methods.

How do you define an “instant” repository?

Simply extend the `CrudRepository<…,…>` interface. No need to specify any method. Standard methods will be provided.

What is @Query used for?

Used to define a custom query for repository methods e.g.,

```
@Query("select u from User u where u.firstname like %?1")
List<User> findByFirstnameEndsWith(String firstname);
```

4. Spring MVC and the Web Layer

MVC is an abbreviation for a design pattern. What does it stand for and what is the idea behind it?

MVC (Model, View & Controller). Structure to provide separation of concerns.

Do you need spring-mvc.jar in your classpath or is it part of spring-core?

No. it is contained in the spring-webmvc artifact.

What is the DispatcherServlet and what is it used for?

A “front controller” which coordinates all request handling activities.

Is the DispatcherServlet instantiated via an application context?

No, each DispatcherServlet has its own context to be separated from the root context where the "working" beans live.

What is the root application context? How is it loaded?

The root context is created by the ContextLoaderListener based on the files named in the contextConfigLocation. This context is intended to contain the beans that compose the core logic of your app.

What is the @Controller annotation used for? How can you create a controller without an annotation?

Represents a component that receives HttpServletRequest and HttpServletResponse instances just like a HttpServlet but is able to participate in an MVC workflow. Alternative to annotation: Extend from the MultiActionController class but is deprecated since spring 4.3.

What is the ContextLoaderListener and what does it do?

Can be used to instantiate the application context. Allows to integrate Spring into almost every web framework.

What are you going to do in the web.xml. Where do you place it?

Declare a ContextLoaderListener. Located in the WEB-INF folder.

How is an incoming request mapped to a controller and mapped to a method?

request → DispatcherServlet → controller → RequestMapping method

What is the @RequestParam used for?

Extracts parameter from the request and performs type conversion.

What are the differences between @RequestParam and @PathVariable?

- **@RequestParam**: Extracts parameter from the request.
- **@PathVariable**: Extracts path variables e.g., `/accounts/{accountId}`

What are some of the valid return types of a controller method?

1. ModelAndView (Class)
2. Model (Interface)
3. Map
4. String
5. void
6. View
7. HttpEntity
8. HttpHeaders

What is a View and what's the idea behind supporting different types of View?

A View renders web output. Allows using many built-in views such as JSPs, XSLT, templating approaches (Velocity, FreeMarker), etc.

How is the right View chosen when it comes to the rendering phase?

The controller either returns a ViewResolver which resolves to the correct view name or a string identifying the logical view name.

What is the Model?

A model holds data for view.

Why do you have access to the model in your View? Where does it come from?

After delegating the request to the controller, which returns a model, after resolving the view, the model is then delegated to the View. The view has to have access to the model since it visualizes the information from the model.

What is the purpose of the session scope?

The session bean scope allows to have stateful beans e.g., for single users for the time they are being logged in.

What is the default scope in the web context?

singleton

Why are controllers testable artifacts?

Controllers are simple Spring beans, though testable. Each request mapping can be tested separately. Expected views and model content can be verified.

What does the InternalResourceViewResolver do?

In Spring MVC, `InternalResourceViewResolver` is used to resolve “internal resource view” (in simple, it’s final output, jsp or html page) based on a predefined URL pattern.

5. Security

What is the delegating filter proxy?

Proxy for a standard Servlet Filter, delegating to a Spring-managed bean that implements the Filter interface.

What is the security filter chain?

Spring Security maintains a filter chain internally where each of the filters has a particular responsibility and filters are added or removed from the configuration depending on which services are required.

In the notes several predefined filters were shown. Do you recall what they did and what order they occurred in?

- **SecurityContextIntegrationFilter**:
  - Establishes SecurityContext and maintains between HTTP requests (formerly: HttpSessionContextIntegrationFilter)
- **LogoutFilter**:
  - Clears SecurityContextHolder when logout requested
- **UsernamePasswordAuthenticationFilter**:
  - Puts Authentication into the SecurityContext on login request (formerly: AuthenticationProcessingFilter)
- **ExceptionTranslationFilter**:
  - Converts SpringSecurity exceptions into HTTP response or redirect
- **FilterSecurity Interceptor**:
  - Authorizes web requests based on on config attributes and authorities

Are you able to add and/or replace individual filters?

Yes. Implement a custom `Filter` bean and add it before/after an existing one or replace one using `http.addFilterAfter (…)`.

Is it enough to hide sections of my output (e.g. JSP-Page)?

No, backend must enforce it, too.

Why do you need the intercept-url?

To secure individual URLs e.g., by allowing role-based access.

Why do you need method security? What type of object is typically secured at the method level (think of its purpose not its Java type).

To manage security aspects on a more fine-grained scale. Allows securing data manipulation etc.

Is security a cross cutting concern? How is it implemented internally?

Yes. Implemented using AOP.

What do @Secured and @RolesAllowed do? What is the difference between them?

Both specify the security roles permitted to access methods in an application. The difference is that `@RolesAllowed` is a Java annotation (JSR-250) and `@Security` is a Speing Security annotation.

What is a security context?

A security context contains the authenticated principal's information.

In which order do you have to write multiple intercept-url's?

Patterns are always evaluated in the order they are defined. Thus it is important that more specific patterns are defined higher in the list than less specific patterns e.g., the more specific `/secure/super/` pattern must appear higher than the less specific `/secure/` pattern. If they were reversed, the `/secure/` pattern would always match and the `/secure/super/` pattern would never be evaluated.

How is a Principal defined?

Create a `User` and `Role` entity and implement a `UserDetailsService`.

What is authentication and authorization? Which must come first?

- **Authentication**: Verifying that a principal's credentials are valid.
- **Authorization**: Deciding whether a principal is allowed to perform an operation.

In which security annotation are you allowed to use SpEL?

`@PreAuthorize`, `@PreFilter`, `@PostAuthorize`, `@PostFilter`, `access` within a intercept-url.

Does Spring Security support password hashing? What is salting?

Yes. Can encode passwords using a hash e.g., sha, md5, bcrypt. A salt is random data that is used as an additional input to a one-way function that "hashes" a password or passphrase. It makes brute force attacks harder.

6. REST

What does REST stand for?

Representational state transfer (REST)

What is a resource?

Any information that can be named can be a resource: a document or image, a temporal service. Alternatively: "whatever thing is accessed by the URL you supply".

What are safe REST operations?

OPTIONS, GET, HEAD

What are idempotent operations? Why is idempotency important?

OPTIONS, GET, HEAD, PUT, DELETE

Is REST scalable and/or interoperable?

It is scalable e.g., through load balancing and highly interoperable since REST uses simple HTTP which is commonly implemented.

What are the advantages of the RestTemplate?

Simplifies and abstracts REST handling.

Which HTTP methods does REST use?

PORT, GET, PUT, PATCH, DELETE

What is an HttpMessageConverter?

Converts between HTTP request/response and object.

Is REST normally stateless?

Yes. Requires client to keep track of the state.

What does @RequestMapping do?

@RequestMapping tells Spring what method to execute when processing a particular request.

Is @Controller a stereotype? Is @RestController a stereotype?

Both are.

What is the difference between @Controller and @RestController?

@RestController is a stereotype annotation that combines @ResponseBody and @Controller.

When do you need @ResponseBody?

Use converters for response data by annotating method with @ResponseBody. No ViewResolver/View involved anymore.

What does @PathVariable do?

Extracts path variables from the request.

What is the HTTP status return code for a successful DELETE statement?

404 (Not Found)

What does CRUD mean?

Create, Read, Update , Delete

Is REST secure? What can you do to secure it?

Use SSL/TLS or JWT.

Where do you need @EnableWebMVC?

Adding this annotation to an @Configuration class imports the Spring MVC configuration from WebMvcConfigurationSupport, e.g.:

Name some common http response codes. When do you need @ResponseStatus?

- 401 (Created) - POST
- 200 (OK) - GET, PUT, PATCH , DELETE

- 404 (NOT FOUND) - GET, PUT, PATCH , DELETE

@ResponseStatus on a controller method, exception or exception handler.

Does REST work with transport layer security (TLS)?

Yes.

Do you need Spring MVC in your classpath?

Yes. It is included in the Spring Web project but not in the Spring Core.

7. Spring Boot

What is Spring Boot?

An opinionated runtime for Spring Projects which handles most low-level setup with support for different project types like web and batch.

What are the advantages of using Spring Boot?

Easy to create Spring-powered, production-grade applications and services with minimum fuss. Provides embedded server and auto configuration capabilities.

Why is it "opinionated"?

Spring Boot uses sensible defaults, "opinions", mostly based on the classpath contents.

How does it work? How does it know what to configure?

Checks contents of the classpath.

What things affect what Spring Boot sets up?

?

How are properties defined? Where?

Porperties are defined by dependencies e.g., `spring-boot-starter-parent` or the `application.properties` in the classpath root.

Would you recognize common Spring Boot annotations and configuration properties if you saw them in the exam?

Common annotations:

- @SpringBootApplication
- @EnableAutoConfiguration

What is the difference between an embedded container and a WAR?

JAR: Executable JAR file with an embedded container. WAR: Output WAR files and deploy it in any other servlet container.

What embedded containers does Spring Boot support?

Tomcat & Jetty.

What does @EnableAutoConfiguration do? What about @SpringBootApplication?

- `@EnableAutoConfiguration`: Causes Spring Boot to automatically create beans it thinks you need, usually based on classpath contents.
- `@SpringBootApplication`: Aggregates `@EnableAutoConfiguration`, `@Configuration`, and `@ComponentScan`.

What is a Spring Boot starter POM? Why is it useful?

Starter POMs declare the properties to use. It resolves a lot of jars and no Version is needed as it is defined by the parent to ensure compatibility.

Spring Boot supports both Java properties and YML files. Would you recognize and understand them if you saw them?

-

Can you control logging with Spring Boot? How?

Yes, via. the `application.properties`.

```
logging.level.root=WARN
logging.level.org.springframework.web=DEBUG
logging.level.org.hibernate=ERROR
```

8. Microservices

What is a microservices architecture?

Microservice architecture is a method of developing software applications as a suite of **independently deployable**, **small**, **modular** services in which each service runs a unique process and **communicates through a well-defined, lightweight mechanism** to serve a business goal.

What are the advantages and disadvantages of microservices?

- Harder to build
- Network overhead + Better maintainability and extensibility + Scaling out easier

What sub-projects of Spring Cloud did we cover in the course? Spring Cloud is a large umbrella project – only what we covered in the course will be tested.

- Spring Cloud Netflix (Service Discovery, Load Balancing)
- Spring Cloud Config (Dynamic Reconfiguration)
- Spring Cloud Connectors (Iaas Integration)
- Spring Cloud Security (Utility)
- Spring Cloud CLI (Utility)
- Spring Cloud Data Flow (Data Integration)

Would you recognize the Spring Cloud annotations and configuration we used in the course if you saw it in the exam?

- `@EnableEurekaServer`: Eureka service registry.
- `@EnableDiscoveryClient`: Service implementation.
- `@LoadBalanced`: On a rest template to inject a smart, load-balanced template.

What Netflix projects did we use?

- Eureka (Service Discovery)
- Ribbon (Load-Balancer)

What is Service Discovery? How is this related to Eureka?

Service Discovery describes the dynamic lookup of service consumers at a common registry. Eureka itself is a service registry, implemented by Netflix.

How do you setup Service Discovery?

Using the `@EnableDiscoveryClient` annotation and a service discovery configuration e.g.,

```
spring:
    application:
        name: accounts-microservice
eureka:
    client:
        serviceUrl:
            defaultZone: http://localhost:8761/eureka/
```

How do you access a RESTful microservice?

`@EnableDiscoveryClient` injects a smart rest template that accessed the relevant service through a URL.