

While the Spring 2.5 certification examination contains 50 multiple-choice questions, the following questions are representative of the content of the questions on the certification examination.

What is an application context?

- Spring's bootstrap object that loads all the high-level (entry-point) objects that are required by your application.

How many types of application context are there?

- Three (four if one considers `JavaConfig`) - classpath, file-system, and web

How do you create an application context?

- Use operator `new` and pass the configuration file(s) to the constructor.

What is one of the uses of the application context reference?

- To gain access the beans of the application

How is a bean defined in the context XML configuration.

- `<bean id=".." class="..">`

What is dependency injection?

- A pattern (design) whereby collaborators are passed into an object from the outside - the object **does not** allocate them for itself.

Why is it useful?

- Different collaborators for different environments can be injected - test, production, test-database, real database. This allows classes to be tested independent of its collaborators (makes it easy to pass in stub or mock objects).

How is dependency injection performed in the bean-file XML.

- There are nested elements within each bean definition.

How is constructor and setter dependency injection specified a configuration file?

- `<constructor-arg...>` and `<property...>`

What is the advantage of constructor dependency injection?

- Constructor: mandatory, mutable, natural Java approach

What are the advantages of setter dependency injection?

- Setter: natural for Java Beans, avoids overly long constructors, can be inherited, easier to understand in that properties have names, constructor args don't have names.

How else can a bean be defined in the bean-file XML?

- Use a factory – (think of the `factory-method` and `FactoryBean` interface).

How does a Spring `FactoryBean` work?

- It is created by Spring automatically, any setters are called *on the factory* then its `getObject()` method is called to allocate the actual bean.

What are the various actions performed when an application context is created?

- Five steps:
 1. Read bean file XML (or process annotations). Build internal "map" of beans and determine dependency tree.
 2. Post process the *definitions* - e.g. resolve `${variables}`
 3. Allocate objects - dependency inject constructor args and invoke setter.
 4. Perform any initialisation - 3 approaches: `@PostConstruct`, `init-method`, `InitializingBean`
 5. Bean post-processors - e.g. wrap with proxy, `@Required`

If one invokes `getBean` specifying the same bean id multiple times, what is returned?

- Same bean every time (singleton)

How can this be changed?

- Specify an alternative scope

What are the 5 scopes defined in Spring?

1. singleton
2. prototype
3. session - only valid in a Web application context
4. request - ditto
5. custom

What is an anonymous bean? When are they used?

- Two examples:
 1. Nested or inner beans
 2. Configuration beans

What is a nested bean? Why is it useful?

- Scopes the bean within the bean that uses it. One less bean id in the set of bean names.

How can you reuse XML configuration if several beans need to be configured in a similar way?

- Bean inheritance.

What is lazy initialization? How do you achieve it? When is it useful?

- Bean is not allocated until it is actually needed - dependency injected - or requested via `getBean()`. specify `lazy="true"` (false by default). Only useful if bean genuinely may never be used in most runs of the application.

When do you use Spring in testing?

- Integration Testing

What is a cross-cutting concern?

- A requirement that cuts across all the natural modules of your application.

Examples?

- Tracing, security, transactions, business rules ...

Two problems with cross-cutting concerns

- Code tangling - method doing too many unrelated tasks
- Code scattering - code duplication leading to maintenance headache

What is a join point?

- A point in the runtime execution of an application at which advice may be triggered in order to implement a cross-cutting concern.

What is a pointcut?

- An expression that matches zero or more join points.

What is an advice?

- Code that implements a cross-cutting concern, and which will be executed at join points matched by its associated pointcut expression .

What is an aspect?

- A Java class encapsulating one or more advices.

How many types of advice are there:

- Before, AfterReturning, AfterThrowing, After, Around

How do you find out about the current content when the aspect is invoked?

- Joinpoint object

What is a named pointcut? Why use one?

- A pointcut expression that has a name so it can be reused.

1. Annotation: `@Pointcut` applied to a dummy method - name of the dummy method is the name of the pointcut expression.
2. XML `aop:pointcut id=".." expression=".."`

What are Spring's JDBC abstraction classes?

- `JdbcTemplate` and `SimpleJdbcTemplate`

What's the difference between them?

1. `SimpleJdbcTemplate` is easier to use
2. `SimpleJdbcTemplate` supports Java 5 features (autoboxing, generics)
3. `SimpleJdbcTemplate` has fewer methods (less common methods removed compared to `JdbcTemplate`)

How many do you need in an application?

- It is thread safe, so often just one is enough – the point is not to create one for every query.

Name 5 ways to query for data using `JdbcTemplate`?

1. Return a single value – `queryForInt(sql, ...)` (`queryForLong(sql,`

- ...), queryForObject(sql, ...))
- 2. Return a row as a Map or a set of rows as a list of Maps – queryForMap(sql, ...), queryForList(sql, ...)
- 3. Returns a row as an object or a set of rows as a list of objects – queryForObject(sql, RowMapper, ...), query(sql, RowMapper, ...)
- 4. Process every row in a flexible, generic way – query(sql, RowCallbackHandler, ...)
- 5. Process entire result-set in one go (Spring does not iterate over the result-set for you) – query(sql, ResultSetExtractor, ...)

How do you write to the database?

- update(sql, ...) – returns the number of rows modified (zero or more).

How does Simple/JdbcTemplate handle SQL exceptions

- Wraps them as unchecked DataAccessExceptions

Why do we use transactions?

- Data integrity – our database server is multi-user, need to ensure concurrent clients do not get in each other's way.

What are the ACID properties of a transaction?

- *Atomic* - a single "unit of work"
- *Consistent* - database modifications do not leave database in an illegal state
- *Isolated* - clients do not see each other's work (locking)
- *Durable* - the database keeps data once committed (persistence) and recovers to the end of the last committed transaction after system failure (logs, archives).

How do you define transactions in Spring?

- Annotate methods in the service layer with @Transactional
- Define in XML

How do they work?

- Your beans are proxied when they are created (using a bean post-processor) and an Around aspect is used to allow the transaction manager to be used to start and commit/rollback transactions around the method.

Other questions involving Spring with Hibernate, Spring Remoting, Spring Web Services, Spring Security, Spring with JMS and Spring with JMX appear on the examination. The Core Spring class (www.springsource.com/training/spr001) covers each of these topics in detail. These topics are also addressed in the Spring 2.5 technical documentation available off of the www.springframework.org web site. SpringSource does not provide a mock or practice examination.