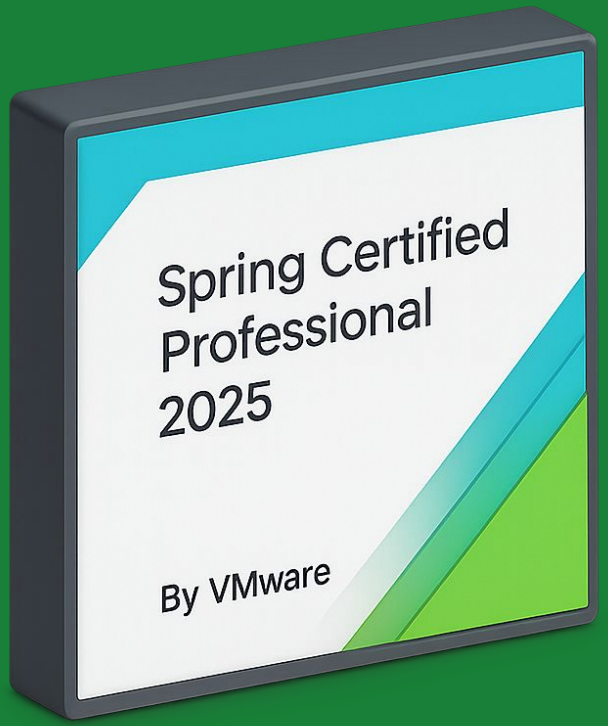
# Spring Certified #5




A question lead guide to prepare Spring certification

## Security

_____ is an interface in Spring Security that represents a user's information, such as username, password, and authorities.

- ➔ **UserDetails**
- ➔ **UserDetailsService**
- ➔ **Authentication**
- ➔ **PasswordEncoder**

## UserDetails

UserDetails

`UserDetails` is a Spring Security interface that represents a user's security data: username, password, roles/authorities, and account status (expired, locked, enabled). Classes like `User` implement it. Spring Security uses it during authentication and authorization.

UserDetailsService

`UserDetailsService` loads user data at login. Its method `loadUserByUsername()` takes a username and returns a `UserDetails`.

Authentication

`Authentication` represents an authentication request or an authenticated user. It holds the principal (the user) and the credentials (like the password).

PasswordEncoder

`PasswordEncoder` encodes raw passwords before storing them. Implementations include `BCryptPasswordEncoder`, `SCryptPasswordEncoder`, etc.

## Spring Actuator

What is a Spring MVC-related metric that is automatically collected by Spring Boot Actuator? (select 2)

- ➔ **Number of requests served**
- ➔ **Request duration**
- ➔ **Number of database queries executed**
- ➔ **Amount of memory used by the application**

## ➔ Number of requests served

## ➔ Request duration

When Spring Boot Actuator is added to a Spring MVC application, it automatically collects and exposes several HTTP server and container metrics. Some of the key metrics that are automatically collected include:

**`http.server.requests`**
This timer tracks all incoming HTTP requests handled by Spring MVC.

These tags let you break down latency and error rates per endpoint.

**`/actuator/mappings` (endpoint, not a metric)**
Spring Boot also exposes the list of request mappings (controllers and handler methods) via the `mappings` actuator endpoint.
This is useful for debugging routing, but it is not a metric and it's not part of `http.server.requests`.

**Tomcat metrics (`tomcat.*`)**
If you're running with embedded Tomcat and you enable Tomcat's MBean registry (`server.tomcat.mbeanregistry.enabled=true`), Actuator will also expose Tomcat internals under the `tomcat.` prefix.

https://docs.spring.io/spring-boot/reference/actuator/metrics.html#actuator.metrics.supported.spring-mvc



At what point does the JdbcTemplate class obtain a database connection?

- ➔ **The JdbcTemplate class obtains a connection from the DataSource at runtime when needed.**
- ➔ **The JdbcTemplate class acquires a connection during the application startup process.**
- ➔ **The JdbcTemplate class does not require a database connection to execute SQL statements.**

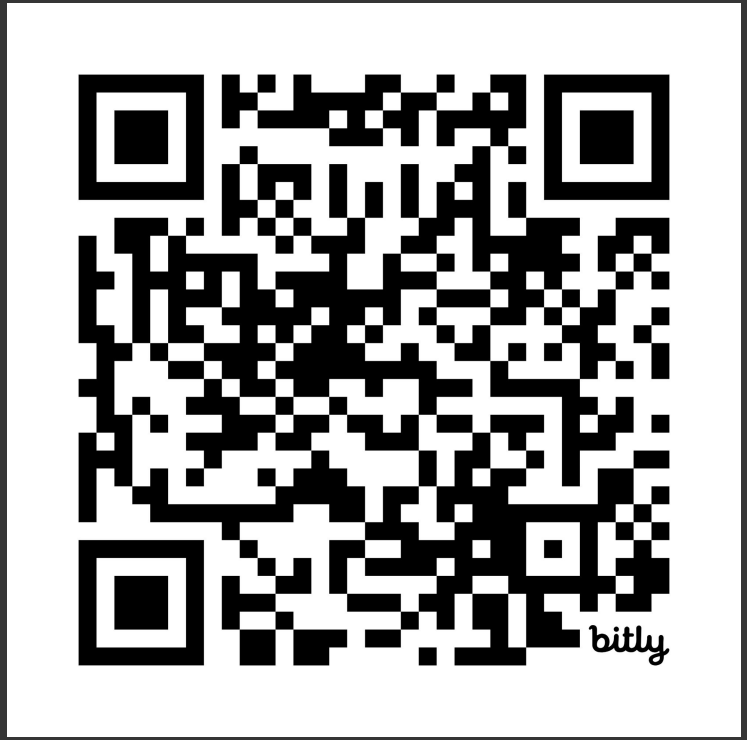
## The JdbcTemplate class obtains a connection from the DataSource at runtime when needed.

The JdbcTemplate class obtains a database connection when executing a method that req

https://docs.spring.io/spring-framework/reference/data-access/jdbc.html

Because creating each new physical connection is time-consuming, the server maintains a pool of available connections to increase performance. When an application requests a connection, it obtains one from the pool. When an application closes a connection, the connection is returned to the pool.

https://docs.oracle.com/cd/E19830-01/819-4712/ablii/index.html




https://bit.ly/2v7222