Spring @Profile

Spring Profiles are configured by:

- Specifying which beans are part of which profile
- Specifying which profiles are active

You can specify beans being part of profile in following ways:

- Use `@Profile` annotation at `@Component` class level - bean will be part of profile/profiles specified in annotation
- Use `@Profile` annotation at `@Configuration` class level - all beans from this configuration will be part of profile/profiles specified in annotation
- Use `@Profile` annotation at `@Bean` method of `@Configuration` class - instance of bean returned by this method will be part of profile/profiles specified in annotation
- Use `@Profile` annotation to define custom annotation - `@Component` / `@Configuration` / `@Bean` method annotated with custom annotation will be part of profile/profiles specified in annotation

If Bean does not have profile specified in any way, it will be created in every profile.
You can use '!' to specify in which profile bean should not be created.

You can activate profiles in following way:

- Programmatically with usage of `ConfigurableEnvironment`
- By using `spring.profiles.active` property
- On JUnit Test level by using `@ActiveProfiles` annotation
- In Spring Boot Programmatically by usage of `SpringApplicationBuilder`
- In Spring Boot by `application.properties` or on yml level

Spring Profiles are useful in following cases:

- Changing Behavior of the system in Different Environments by changing set of Beans that are part of specific environments, for example prod, cert, dev
- Changing Behavior of the system for different customers
- Changing set of Beans used in Development Environment and also during Testing Execution
- Changing set of Beans in the system when monitoring or additional debugging capabilities should be turned on