Using Profiles

To enable multiple profiles in Spring XML configuration (`config.xml`), you can use the `spring.profiles.active` property to specify a comma-separated list of profiles that you want to activate. Here's how you can do it:

1. **Activate Multiple Profiles**: Set the `spring.profiles.active` property in your `config.xml` with a comma-separated list of profiles that you want to activate.

   ```xml
   <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd"
       xmlns:util="http://www.springframework.org/schema/util"
       profile="dev,prod">

     <!-- Beans for the "dev" and "prod" profiles -->

   </beans>
   ```

   In this example, both the "dev" and "prod" profiles are activated.

2. **Define Beans for Each Profile**: Define beans within the `<beans>` element for each profile you want to support. You can use the `profile` attribute to specify which profile the beans belong to.

   ```xml
   <beans profile="dev">
     <!-- Beans for the "dev" profile -->
   </beans>

   <beans profile="prod">
     <!-- Beans for the "prod" profile -->
   </beans>
   ```

   Define beans specific to each profile within the respective `<beans>` elements.

By setting the `spring.profiles.active` property with a comma-separated list of profiles, you can activate multiple profiles in your Spring XML configuration (`config.xml`). This allows you to configure your application differently based on the combination of profiles that are active.

In Spring, you can use profiles to selectively assign beans to the context based on the active profile. Here's how you can do it:

1. **Define Beans for Each Profile**: Define beans for each profile you want to support. You can do this by annotating your bean methods with `@Profile("profileName")`.

2. **Activate Profiles**: Activate profiles in your application using one of the following methods:
   - Set the `spring.profiles.active` property in your `application.properties` or `application.yml`.
   - Use the `SpringApplicationBuilder` API to set profiles programmatically.
   - Set profiles using JVM system properties or environment variables.

3. **Example**:

   ```java
   import org.springframework.context.annotation.Bean;
   import org.springframework.context.annotation.Configuration;
   import org.springframework.context.annotation.Profile;

   @Configuration
   public class MyConfiguration {

       @Bean
       @Profile("dev")
       public MyBean devBean() {
           return new MyDevBean();
       }

       @Bean
       @Profile("prod")
       public MyBean prodBean() {
           return new MyProdBean();
       }
   }
   ```

In this example, `MyDevBean` will be registered as a bean only when the "dev" profile is active, and `MyProdBean` will be registered only when the "prod" profile is active.

4. **Activate Profiles**:

   - **Using `application.properties`**:
     ```
     spring.profiles.active=dev
     ```

   - **Using `application.yml`**:
     ```yaml
     spring:
       profiles:
         active: dev
     ```

   - **Programmatically**:
     ```java
     import org.springframework.boot.builder.SpringApplicationBuilder;
     import org.springframework.context.ConfigurableApplicationContext;

     public class MyApp {

         public static void main(String[] args) {
             ConfigurableApplicationContext context = new
SpringApplicationBuilder(MyApp.class)
                 .profiles("dev")
                 .run(args);
         }
     }
     ```

   - **Using JVM System Property**:
     ```
     -Dspring.profiles.active=dev
     ```

5. Now, when you run your application with the specified active profile, only beans associated with that profile will be loaded into the application context.

By leveraging profiles in this way, you can easily manage different configurations for different environments or use cases within your Spring application.

To activate multiple profiles in Spring Boot, you can use any of the following methods:

1. **Using Command Line Arguments**: You can specify multiple profiles using the `spring.profiles.active` property when running your Spring Boot application. For example:

   ```
   java -jar my-application.jar --spring.profiles.active=profile1,profile2
   ```

2. **Using application.properties or application.yml**: You can define the active profiles in your `application.properties` or `application.yml` file:

   ```properties
   spring.profiles.active=profile1,profile2
   ```

   ```yaml
   spring:
     profiles:
       active: profile1, profile2
   ```

3. **Using Environment Variables**: You can set the `SPRING_PROFILES_ACTIVE` environment variable to specify the active profiles:

   ```
   export SPRING_PROFILES_ACTIVE=profile1,profile2
   ```

4. **Programmatically in Java Code**: You can set the active profiles programmatically in your Spring Boot application's Java code using the `SpringApplicationBuilder`:

   ```java
   new SpringApplicationBuilder(MyApplication.class)
       .profiles("profile1", "profile2")
       .run(args);
   ```

Choose the method that best fits your use case and environment. Using multiple profiles allows you to configure your application differently based on different environments or conditions.