

## स्प्रिंग के एप्लिकेशन कॉन्टेक्स्ट तक पहुंच

[illegible]

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

`ApplicationContextAware` 인터페이스를 구현하는 클래스는 `ApplicationContext` 객체를 주입받을 수 있습니다. 이 인터페이스를 구현한 클래스는 `setApplicationContext()` 메서드를 통해 `ApplicationContext` 객체를 주입받습니다. 이 메서드는 `ApplicationContext` 객체를 인자로 받으며, 이 객체는 애플리케이션의 컨테이너 역할을 합니다.

[illegible]

- [illegible]

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
import org.springframework.stereotype.Component;

@Component

public class ApplicationContextProvider implements ApplicationContextAware {

    private static ApplicationContext context;

    @Override
    public void setApplicationContext(ApplicationContext applicationContext) {
        context = applicationContext;
    }

    public static ApplicationContext getApplicationContext() {
        return context;
    }
}
```

- ```

❑ @Component 어노테이션을 통해 이 클래스를 Spring 컨테이너에 등록한다.
❑ setApplicationContext 메서드를 통해 ApplicationContext 인터페이스를 구현한 객체를 주입받는다.
❑ 이 객체를 통해 context 객체를 생성하고, 이 객체를 통해 다른 빈을 조회할 수 있다.

```

2. `processMessage()` method mein `ApplicationContext` object kaise access karenge? `ApplicationContext` object kisi `ApplicationContextProvider` class ke `getApplicationContext()` method se access kiya jayega. `ApplicationContext` object ke `getBean()` method se kisi bhi service ko access kiya jayega.

```
public class MyKafkaConsumer {
    public void processMessage() {
        ApplicationContext context = ApplicationContextProvider.getApplicationContext();
        SomeService service = context.getBean(SomeService.class);
        // Service ya dusre beans ka use karein jahan zaroori hai
    }
}
```

Ab `ApplicationContextProvider` class ko create karenge jo `ApplicationContext` object ko return karegi. `ApplicationContextProvider` class ko `ApplicationContext` object ko return karne ke liye `getApplicationContext()` method implement karna hoga.

3. `ApplicationContextProvider` class ko create karenge: `ApplicationContextProvider` class mein `getApplicationContext()` method implement karenge jo `ApplicationContext` object ko return karegi. `ApplicationContextProvider` class ko `ApplicationContext` object ko return karne ke liye `getApplicationContext()` method implement karna hoga.

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.stereotype.Component;
```

`@Component`

```
public class KafkaConsumerCreator {
    @Autowired
    private ApplicationContext context;

    public MyKafkaConsumer createConsumer() {
        return new MyKafkaConsumer(context);
    }
}
```

```
public class MyKafkaConsumer {
    private final ApplicationContext context;

    public MyKafkaConsumer(ApplicationContext context) {
        this.context = context;
    }

    public void processMessage() {
        SomeService service = context.getBean(SomeService.class);
    }
}
```

```

        // Service ka use karein
    }
}

```

[illegible]

00000 00000000-00000000 00000000

[illegible]

@KafkaListener 는 바로 이 기능을 제공하는 인터페이스로 이 인터페이스를 구현한 클래스를 Spring-Boot 애플리케이션에 등록하면, @KafkaListener 어노테이션을 붙인 메서드가 자동으로 호출됩니다. 이 메서드는 Application Context에 등록된 KafkaConsumer와 연결되어 있습니다. 이 메서드는 KafkaConsumer가 가져온 메시지를 처리하는 데 사용됩니다.

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Component;
```

@Component

```
public class MyKafkaConsumer {

    @Autowired

    private ApplicationContext context;


    @Autowired

    private SomeService someService;


    @KafkaListener(topics = "my-topic")

    public void consume(String message) {

        // Context ya someService ko directly use karein

        SomeService service = context.getBean(SomeService.class); // Agar zaroori hai

        someService.process(message);

    }

}
```

0. Өзгөчө өлчөмдөгү материалдарды сатуу үчүн, өлчөмдөгү материалдардын баасын өлчөөгө болбойт.

0. Өлчөмдөгү материалдардын баасын өлчөөгө болбойт, өлчөмдөгү материалдардын баасын өлчөөгө болбойт.

00000000 000000 0000000000 0000 000000 KafkaConsumer 00 0000000000 00000000 000000 000000 000 (0.0., 000000  
 00000000 00000000 00 0000000000 0000 000000), 00 00000 00 0000000-0000000000 0000 00 000000 00000000 00000000 000  
 00000 ApplicationContext 00 00000 00000000, 0000000 00000000000 0000000000 0000000000 00000 0000000000 00000 0000.

□ □ □ □ □ □ □ □ □ □ □ □ □ □

□ 000000 000000000000 00 00000: 000000 000 0000000000 000, 0000000 000000 00 @KafkaListener 00 000 0000000 00 0000000-00000000 0000 0000. 000 0000000 00 000000 00 000000 0000000000000 00 000000000 000000 0000 0000 ApplicationContext 000 000000 000000 00 0000000 000000 00 000000000 000000 0000.

ApplicationContextProvider 是 ApplicationContext 的父类，它负责初始化 ApplicationContext。在 Android 中，ApplicationContext 是系统提供的，而 ApplicationContextProvider 是开发者定义的。在 AndroidManifest.xml 中，我们声明了 Application 类，而在代码中，我们实现了 Application 类，并重写了 onCreate 方法，调用了 Application.onCreate() 方法。在 Application.onCreate() 方法中，我们调用了 ApplicationContextProvider 的 getApplicationContext() 方法，获取了 ApplicationContext 实例。然后，我们调用了 ApplicationContextProvider 的 registerActivityLifecycleCallbacks() 方法，注册了 ActivityLifecycleCallbacks 接口。最后，我们调用了 ApplicationContextProvider 的 registerComponentCallbacks() 方法，注册了 ComponentCallbacks 接口。

00000 00000000 00 0000000 000000, 0000 ApplicationContextAware 00 0000 000000 000000 00000 00000 000000 00000, 000000 00 000000 0000000000 0000 0000000 00 000000 0000, 00000000 00000000 00 000000000000 0000000000 0000 000000 00000, 000000 0000000000 000000 00 00000000 00.