链滴

SpringBoot 中的 spring-boot-autoconfigure 模块学习

作者：flowaters

原文链接：https://ld246.com/article/1534608851208

来源网站：链滴

背景

SpringBoot中有一个application.properties配置文件，其中的配置怎么在应用程序中使用呢？然后应的原理是？

初体验

直接通过Value读取配置

如果是一个参数，可以直接通过Value注解来取配置信息。

配置文件

application.properties

```
appconfig.dataFilePath=/tmp
```

Controller

```
package com.example.testapplication.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ValueController {

    @Value("${appconfig.dataFilePath}")
    String dataFilePath;

    /**
     * 输出结果
     *
     * @return 格式化字符串
     */
    @RequestMapping(value = "/value")
    public String sayHello() {
        return dataFilePath;
    }
}
```

效果

```
$ curl -s "localhost:8080/value"
/tmp
```

POJO的配置

如果参数多了时，可以将参数封装成一个Config Bean。

这里将配置dataFilePath，从配置文件中读取到AppConfigProperties这个Config Bean中，然后在Controller中使用。

配置类

配置类 AppConfigProperties.class

```
package com.example.testapplication.service.appconfig;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Configuration;

import lombok.Getter;
import lombok.Setter;

@ConfigurationProperties(prefix = "appconfig")
@Setter
@Getter
public class AppConfigProperties {
    private String dataFilePath;
}
```

配置类自动加载类

配置类自动加载类 AppConfigAutoConfiguration.java

```
package com.example.testapplication.service.appconfig;

import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableConfigurationProperties(AppConfigProperties.class)
public class AppConfigAutoConfiguration {
}
```

配置文件

配置文件application.properties片断

```
appconfig.dataFilePath=/tmp
```

Controller

AppConfigController

```
package com.example.testapplication.controller;

import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.testapplication.service.appconfig.AppConfigProperties;

@RestController
public class AppConfigController {

    @Autowired
    AppConfigProperties appConfigProperties;

    /**
     * 输出结果
     *
     * @return 格式化字符串
     */
    @RequestMapping(value = "/")
    public String sayHello() {
        return appConfigProperties.getDataFilePath();
    }

}
```

运行

```
$ curl -s "localhost:8080/"
/tmp
```

可见，读取了配置文件，然后加载到AppConfigProperties中，然后显示了出来。

其它

如果不想取application.properties中的配置，而是自定义配置文件名，也是可以使用PropertySource解来指定的。(测试未通过，待更正。。。)

```
package com.example.testapplication.service.appconfig;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;

import lombok.Getter;
import lombok.Setter;

@ConfigurationProperties(prefix = "appconfig")
@PropertySource(value = "appconfig.properties")
@Setter
@Getter
public class AppConfigProperties {
    private String dataFilePath;
}
```

service bean的配置

如果不仅仅是读配置，而是要初始化对应的service bean呢?

这里写一个Service Bean，如果在application.properties中配置了，则使用配置的参数；否则使用认参数。

Service

```
package com.example.testapplication.service.echo;

import lombok.Setter;

@Setter
public class EchoService {
    /**
     * 文件内容
     */
    private String text;

    /**
     * 是否显示文本
     */
    private boolean show;

    public String echo() {
        return show ? text : "******";
    }
}
```

配置类

EchoProperties.java

```
package com.example.testapplication.service.echo;

import org.springframework.boot.context.properties.ConfigurationProperties;

import lombok.Getter;
import lombok.Setter;

// 配置前缀为echo
@ConfigurationProperties(prefix = "echo")
@Setter
@Getter
public class EchoProperties {

    /**
     * text默认值, Hello, World.
```

```
     */
    private String text = "Hello, World";

    /**
     * show默认值, true
     */
    private boolean show = true;

}
```

自动配置类

EchoAutoConfiguration.java

```
package com.example.testapplication.service.echo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.condition.ConditionalOnClass;
import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
import org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableConfigurationProperties(EchoProperties.class)
@ConditionalOnClass(EchoService.class)
@ConditionalOnProperty(prefix = "echo", // 存储配置前缀echo
    value = "enabled", // 开启
    matchIfMissing = true // 缺失检查
)
public class EchoAutoConfiguration {

    @Autowired
    private EchoProperties echoProperties;

    /**
     * 缺失EchoService Bean时，刚自动创建默认的.
     *
     * @return
     */
    @Bean
    @ConditionalOnMissingBean(EchoService.class)
    public EchoService echoService() {
        System.out.println("use DEFAULT EchoService");
        EchoService echoService = new EchoService();
        echoService.setShow(echoProperties.isShow());
        echoService.setText(echoProperties.getText());
        return echoService;
    }
}
```

配置文件

application.properties

```
echo.text=Hello Note
echo.show=true
```

控制类

EchoController.java 用来测试

```
package com.example.testapplication.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.testapplication.service.echo.EchoService;

@RestController
public class EchoController {

    @Autowired
    EchoService echoService;

    /**
     * 测试访问地址/hello
     *
     * @return 格式化字符串
     */
    @RequestMapping(value = "/echo")
    public String echo() {
        return echoService.echo();
    }
}
```

效果

```
$ curl -s "localhost:8080/echo"
Hello Note
```

starter的自动配置

参考

- 第二十八章：SpringBoot使用AutoConfiguration自定义Starter
- auto-config.xml 与 antx.properties: 如何打包
- Generating Your Own Metadata by Using the Annotation Processor: idea推荐方式
- 定义一个Configuration Processor读取spring配置！
- SpringBoot非官方教程 | 第二篇：Spring Boot配置文件详解

- Spring Boot @ConfigurationProperties example: 推荐阅读，例子很多