Environment&Properties

1. 接口&类

1.1 PropertyResolver

简单纯粹的`properties`操作接口；

1.2 Environment

1.2.1 父接口

PropertyResolver

1.2.2 作用

- 接口，表示应用环境变量，主要包含2个方面`profiles`和`properties`；
- 通过PropertyResolver中的方法暴露`properties`；

```
Methods related to property access are exposed via the PropertyResolver superinterface.
```

- `Profile`不区分`xml`创建，还是`annotations`创建的bean，统一合并到对应的`profile`中；
- `Properties`的来源有多种形式

```
properties files, JVM system properties, system environment variables, JNDI, servlet context
parameters, ad-hoc Properties objects, Maps, and so on.
```

- 配置`environment`必须通过ConfigurableEnvironment(`AbstractApplicationContext$getEnvironment()`方法获取)：

```
Configuration of the environment object must be done through the ConfigurableEnvironment interface,
returned from all AbstractApplicationContext subclass getEnvironment() methods.
```

- 激活`Profiles`的方法：

```
Profiles can be activated by setting "spring.profiles.active" as a system property or by calling
ConfigurableEnvironment.setActiveProfiles(String...).
```

1.3 EnvironmentAware

只是一个`Aware`子接口，作用参考`Aware`；

1.4 PropertySourcesPlaceholderConfigurer

since3.1

1.4.1 父

EnvironmentAware、PlaceholderConfigurerSupport

1.4.2 作用

- 当配置了`<context:property-placeholder/>`，Spring会默认引入此类；

- 代替`PropertyPlaceholderConfigurer`（3.1之前使用此类）； 继续。。。2019-12-12

1.5 ConfigurableEnvironment

1.5.2 作用

- 对`property source`进行替换、重排序、删除等操作，使用MutablePropertySources（returned from `getPropertySources()`）；
- 当`ApplicationContext`中引用了`Environment`，很重要的一点：保证`PropertySource`之类的全部在`refresh()`之前加载完毕；

```
When an Environment is being used by an ApplicationContext, it is important that any such
PropertySource manipulations be performed before the context's refresh() method is called. This ensures
that all property sources are available during the container bootstrap process, including use by
property placeholder configurers.
```

1.6 ConfigurablePropertyResolver

1.6.1 父

PropertyResolver

作用

和ConversionService有关

1.7 ConversionService

1.7.1 作用

- 类型转换，如：string → LocalDatetime

1.8 MutablePropertySources

1.8.2 作用

- 增加、修改、模拟`environment`中的`properties`；

```
/* ConfigurableEnvironment environment */
/* 例：测试环境-模拟properties */
MutablePropertySources propertySources = environment.getPropertySources();
MockPropertySource mockEnvVars = new MockPropertySource().withProperty("xyz", "myValue");
propertySources.replace(StandardEnvironment.SYSTEM_ENVIRONMENT_PROPERTY_SOURCE_NAME, mockEnvVars);

/* 例：adding a new property source with highest search priority */
ConfigurableEnvironment environment = new StandardEnvironment();
MutablePropertySources propertySources = environment.getPropertySources();
Map<String, String> myMap = new HashMap<>();
myMap.put("xyz", "myValue");
propertySources.addFirst(new MapPropertySource("MY_MAP", myMap));
```

1.9 PropertySource

1.9.1 作用

- 保存`property`source信息，里面提供了`getProperty()`方法获取属性值；
- 参考`PropertySources`

1.10 AbstractEnvironment

1.10.1 父

1.10.2 作用

Environment的实现模板类，有很多默认实现：

- `spring.profiles.active`、`spring.profiles.default`等属性的设置；更多默认包括：`spring.getenv.ignore`
- 子类主要实现`PropertySource`的区分就可以了，子类通过`customizePropertySources(MutablePropertySources)`（MutablePropertySources）来提供`PropertySource`； 查看`customizePropertySources(MutablePropertySources)`方法的doc文档，对理解更有帮助；
- 如果调用了`setActiveProfiles()`，那么，`spring.properties`中的`spring.profiles.active`配置会失效；（*参考doGetActiveProfiles()方法*）

1.11 StandardEnvironment

没啥讲的，是Spring的默认实现；

1.13 @Configuration

1.14 PropertiesLoaderSupport

1.14.1 作用

- 属性操作相关过程类；

1.14 PropertiesPersister

1.14.1 作用

支持读写Property的文件（操作文件流`IOStream`或`Reader`）；

1.15 PlaceholderConfigurerSupport

1.15.1 父

PropertyResourceConfigurer

1.15.2 作用

- `Abstract base class for property resource configurers that resolve placeholders in bean definition property values`
  （解析bean中的property：拉取properties，填充到`bean definition`中），比如解析`${****:defaultValue}`形式的属性；
- 解析属性的方法，请关注此类的`doProcessProperties()`；

1.16 PropertyResourceConfigurer

1.16.1 父

类：PropertiesLoaderSupport 接口：PriorityOrdered

1.17 PriorityOrdered

1.17.1 父

接口：`Ordered`

1.17.2 作用

- 表示优先级顺序；
- 具有相同`order`值的`PriorityOrdered`总是排在（plain）`Ordered`之前；
- 实现了`PriorityOrdered`的`post-processor`比其他`post-processor`先执行；

```
PriorityOrdered post-processor beans are initialized in a special phase, ahead of other post-processor
beans.
```

1.18 Properties

1.18.1 父

类：Hashtable<Object,Object>

1.18.2 作用

- 保存`key-value`property；
- 因为继承了`Hashtable<Object,Object>`，所以，`Hashtable<Object,Object>`的`put/putAll`方法在此可以被调用，但是建议不要使用，应为无法保证`key/value`为`String`类型，建议使用`setProperty(string, string)`方法；类似需要注意的还有`Hashtable<Object,Object>`的其他方法。如果不能保证`key/value`是`String`类型，那么此类的许多方法调用将会失败；

2. Plankton

1. `SystemProperty`：通过java -jar test.jar -Denv=123启动时指定的值;`SystemEnviroment`：在Linux下使用export $ENV=123指定的值;