01 Getting Started

Getting started with ConfigureMe

To get started with ConfigureMe you need three things. First you need something *to configure* which we call **configurable**. Than you need something to configure your configurable with, that'll be the **configuration**. And finally you need to ask the **ConfigurationManager** to configure your **configurable** with your **configuration** somewhere in your code.

Selling hamburgers

Lets get concrete. Imagine you are opening a fast food restaurant. To make it real fast you offer just one product, the BigPack. Since you are a great developer you are writing the software for your cash box yourself, and you want the price to come from a configuration file to be able to react to the financial crises fast.

Creating a configurable

First of all we need an object to store the prices, our configurable object:

Pricing.java

```
package pricing;

import org.configureme.annotations.Configure;
import org.configureme.annotations.ConfigureMe;

@ConfigureMe
public class Pricing {
        /**
         * The local currency.
         */
        @Configure
        private String currency;
        /**
         * The price of the BigPack in local currency.
         */
        @Configure
        private float price;

        public void setCurrency(String aCurrency){
                currency = aCurrency;
        }

        public void setPrice(float aPrice){
                price = aPrice;
        }

        /**
         * Returns the price for the customer.
         */
        public String getProductPrice(){
                return price+" "+currency;
        }
}
```

Our configurable is nothing more than a JavaBean which an additional method: *getProductPrice_* which return the price we have to show the customer. Since we are planing to get international pretty soon, we introduced a special field for the currency, to be able to operate in different countries.

Creating a configuration file.

Now that we have a configurable we need a configuration.

pricing.json

```
{
        price: 3.57,
        currency: "$",
}
```

Note that the default name resolution is deduced from the class name, but you can change it by parameterizing the annotation.

Showing the price

Now that we have prepared everything, the first customer can come in. All we need is to add 3 more lines of code:

ShowPrice.java

```
package pricing;

import org.configureme.ConfigurationManager;

public class ShowPrice {
        public static void main(String a[]){
                Pricing pricing = new Pricing();
                ConfigurationManager.INSTANCE.configure(pricing);
                System.out.println("Please pay "+pricing.getProductPrice());
        }
}
```

Once run the output of the program is:

> *Please pay 3.57 $*

—

Changing environment

Now since your business is running well, it's time to become a global player and run international branches. To start with you launch 3 international branches, one in germany, one in austria and one in united kingdom. The only thing you need to change is the config file, you just have to specify the different environments:

pricing.json

```
{
        price: 3.57,
        currency: "$",

        europe: {
                //many european countries have euro as currency
                currency: "€",
                uk:{
                        price: 2.29,
                        currency: "£",
                },

                de: {
                        price: 3.00,
                },

                at: {
                        price: 3.50,
                },

        },
}
```

Done? Almost. Now you simply have to tell the program in which environment its executed so it can select the proper configuration. In your branch in united kingdom you will have to run the program with following parameter:
`java -Dconfigureme.defaultEnvironment=europe_uk pricing.ShowPrice`

The output will be as expected:

> *Please pay 2.29 £*

Explicitly specifying the environment

Now that your BigPack business is flourishing you don't need to work anymore and spent time with things that matter. You even gave away the pricing policy to a high ranked manager in your conglomerate. However, for the sake of good old times you want to check the sales and prices from time to time. But doing it with the method from last paragraph is rather uncomfortable, since you have to travel to the target country to check the price in it. Fortunately for you ConfigureMe offers a possibility to specify execution environment explicitly:

ShowPrice.java

```
public class ShowPrice {
        public static void main(String a[]){
                showPriceIn("USA", GlobalEnvironment.INSTANCE);
                showPriceIn("United Kingdom", new DynamicEnvironment("europe", "uk"));
                showPriceIn("Germany", new DynamicEnvironment("europe", "de"));
                showPriceIn("Austria", new DynamicEnvironment("europe", "at"));
        }

        private static void showPriceIn(String description, Environment environment){
                Pricing pricing = new Pricing();
                ConfigurationManager.INSTANCE.configure(pricing, environment);
                System.out.println("Price in "+description+" is "+pricing.getProductPrice());
        }
}
```

When you are starting the program now it prints the prices in all branches, leaving you enough time for your other projects:

> *Price in USA is 3.57 $*
> *Price in United Kingdom is 2.29 £*
> *Price in Germany is 3.0 €*
> *Price in Austria is 3.5 €*

Configuring the whole class.

Finally you may ask (and actually people asked 🙂), if all fields in the class, as in the above example, are configureable, why they all must be annotated? They don't. In fact you can add a parameter to the ConfigureMe annotation telling it, that all fields are configureable:

Pricing.java

```
...
@ConfigureMe (allfields=true)
public class Pricing {
        private String currency;
        private float price;
...
```

If you run this program now, you will see that the results are exactly the same as in previous version.

Using ConfigureMe within a Spring Projekt

This is quite easy. ConfigureMe has Spring support allready included, just use the org.configureme.spring.ConfigureSpringBeans class and add all SpringBeans that should be configure as constructor argument.
For example when using the default environment:

applicationContext.xml

```
<bean id="pricing" class="Pricing" />

<bean class="org.configureme.spring.ConfigureSpringBeans">
    <constructor-arg ref="pricing" />
</bean>
```

you can also specify the a environment which should be used when adding a DynamicEnvironment as first constructor argument.

applicationContext.xml

```
<bean id="pricing" class="Pricing" />

<bean class="org.configureme.spring.ConfigureSpringBeans">
    <constructor-arg>
        <bean class="org.configureme.environments.DynamicEnvironment">
            <constructor-arg value="europe" />
            <constructor-arg value="de" />
        </bean>
    </constructor-arg>
    <constructor-arg ref="pricing" />
</bean>
```

Download and try ConfigureMe now: https://configureme.dev.java.net