ConfigureMe - Java OpenSource Configuration Utility

ConfigureMe - Java OpenSource Configuration Utility

ConfigureMe is a configuration utility for really simple auto configuring of the java objects. It focuses on a different configurations of the same object in a different environments (e.g. for deploying case it can be: dev, test and prod system). ConfigureMe configures POJOs, based on annotations. Currently it supports **json**, **xml** and **.properties** files.

Features

- Automatic type safe configuration via annotations
- Support for cascading environments
- Automatic re-configuration of the configurable objects on configuration (file) change
- One-to-one relation between a configuration and a configurable object
- Overview of all configuration options for a given configurable object at a glance
- Low load time, low memory footprint

Environments description

Environments is the core feature of the ConfigureMe, that gives possybility to predefine properties values for the different "using situations".

Situations (actually, anything that have a common criteria) mean something like:

- Different servers
- Different users
- Different countries
- Different groups of users
- Different moods (even 🙂)
- ...

Environments supports any level **cascading**. The main cascading principles are:

1. Property value from the lower level overrides property (same) value from the higher level.
2. If requested environment not exists, values will be taken from the higher level environment (root is the default environment).

Maven Coordinates

Dependency (update version to the latest)

pom.xml

```
<dependency>
 <groupId>net.anotheria</groupId>
 <artifactId>configureme</artifactId>
 <version>1.1.0</version>
</dependency>
```

Repository

http://nexus.anotheria.net/nexus/content/groups/public/

Direct download from nexus

http://nexus.anotheria.net/nexus/content/groups/public/net/anotheria/configureme/

Getting Started Example

Let's have a look how to use ConfigureMe on a small live example. Suppose we develop an application, witch will request some external server(s) for the users photos (image files). Also we want to configure some other properties around the photos, for example - maximum size or the uploaded photo files.

With ConfigureMe our configuration file may looks like:

photoserver-config.json

```
{
        hosts: ["default.photoserver.com"],
        port: 8080,
        maxPhotoSize: 2048,

        dev: {
                jack: {
                        hosts: ["localhost"],
                        port: 8181,
                },
        },

        testsystem : {
                hosts: ["test.photoserver.com"],
        },

        livesystem: {
                hosts: ["p1.photoserver.com", "p2.photoserver.com", "p3.photoserver.com"],
                maxPhotoSize: 4096,
        },
}
```

Configuration bean:

PhotoServerConfig.java

```
package configureme;

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

@ConfigureMe(name="photoserver-config", allfields=true)
public class PhotoServerConfig {

        private static final PhotoServerConfig INSTANCE = new PhotoServerConfig();

        private String[] hosts;
        private int port;
        private int maxPhotoSize;

        private PhotoServerConfig() {
                try {
                        ConfigurationManager.INSTANCE.configure(this);
                } catch (IllegalArgumentException configNotFound) {
                        // We can set default values here
                        hosts = new String[]{"default.photoserver.com"};
                        port = 8080;
                        maxPhotoSize = 2048;
                }
        }

        public static PhotoServerConfig getInstance() {
                return INSTANCE;
        }

        public String[] getHosts() {
                return hosts;
        }

        public void setHosts(String[] hosts) {
                this.hosts = hosts;
        }

        ...
        // Other getters and setters
}
```

Client application:

PhotoServerConfigClient.java

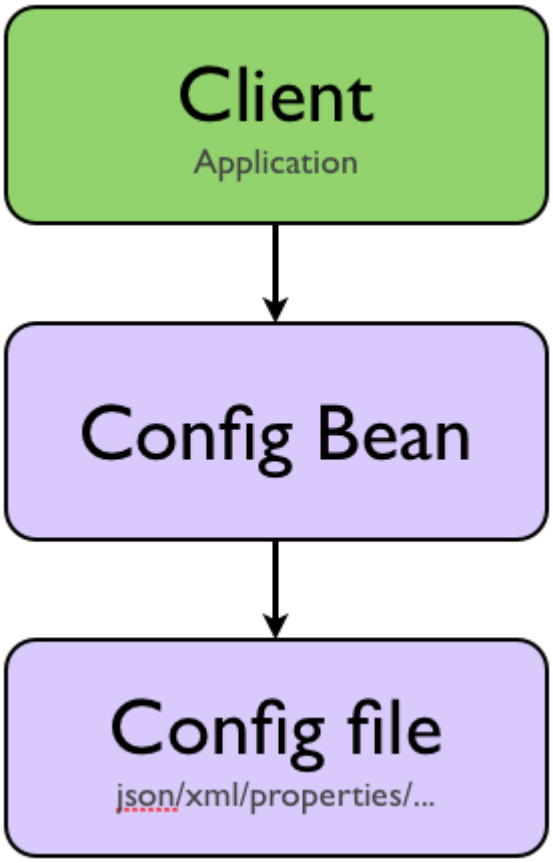
```
package configureme;

import java.util.Arrays;

public class PhotoServerConfigClient {

        public static void main(String[] args) {
                PhotoServerConfig config = PhotoServerConfig.getInstance();
                System.out.println("Hosts: " + Arrays.toString(config.getHosts()));
                System.out.println("Port: " + config.getPort());
                System.out.println("MaxPhotoSize: " + config.getMaxPhotoSize());
        }
}
```

Common schema:



All magic starts when we try to run client application with the different environments, witch is usually configurable via the system property, named “configureme.defaultEnvironment”:

java -Dconfigureme.defaultEnvironment=dev_jack ... configureme.PhotoServerConfigClient

```
Hosts: [localhost]
Port: 8181
MaxPhotoSize: 2048
```

java -Dconfigureme.defaultEnvironment=testsystem ... configureme.PhotoServerConfigClient

```
Hosts: [test.photoserver.com]
Port: 8080
MaxPhotoSize: 2048
```

java -Dconfigureme.defaultEnvironment=livesystem ... configureme.PhotoServerConfigClient

```
Hosts: [p1.photoserver.com, p2.photoserver.com, p3.photoserver.com]
Port: 8080
MaxPhotoSize: 4096
```

We see that client will have a different properties values for the different environments (like systems in our example) and so it will behave differently.

Summary

In this article you saw only a part of the ConfigureMe (OpenSource Configuration Utility) capabilities, but I hope it was enough to understand what is ConfigureMe and how it can be useful in your dev-work.

Everything else (how to get, more docs, and more examples) may be found exactly on this ConfigureMe project documentation page: http://www.configureme.org

ConfigureMe project page on java.net: http://configureme.dev.java.net

Any questions, want's to join, feedback and requests for futures are very welcome here: issues@configureme.java.net

See also:
http://opensource.anotheria.net/

Enjoy

Valeriy Kazhdan