ConfigureMe
(Configuration Utility)

January 2011

anotheria solutions

ConfigureMe Privileges

- Easy to get (OpenSource)
- Easy to understand
- Easy to start
- Easy to use

Simple config (*.json) looks like

```
{
   user: "defuser",
   mail: "defuser@anotheria.net",

   managers: {
      user: "managers",
      mail: "managers@anotheria.net",
   },

   developers: {
      mail: "devs@anotheria.net",

      tom: {
         user: "tom",
      },
      klark: {
         user: "klark",
         mail: "klark@anotheria.net",
      },
   },
}
```

The only theory you should know

- Environments
- Cascading

Environments

Environments is the different app deployment situations

- Different servers
- Different users
- Different countries
- Different groups of users
- ...

Cascading

Cascading is any level environments nesting support

- Property value from the lower level overrides property (same) value from the higher level
- If requested environment not exists, values will be taken from the higher level environment

How to set environment

- On program start:

```
java -Dconfigureme.defaultEnvironment=developers_tom ...
```

- Dynamically:

```
ConfigurationManager.INSTANCE.configure(
    EnvExampleConfig.getInstance(), "managers_tom");

String user = EnvExampleConfig.getInstance().getUser();
```

Interesting to know

- Supported config file formats:

  json, xml, properties

- Config file loading on the first config class using

- All On-The-Fly changes in the config file will be applied in the conf class in **10 sec**

Examples

- E1: Simple configuration
- E2: Using @SetAll
- E3: Using @SetIf
- E4: Using String Arrays (Lists)

E1: Simple configuration
JSON Config

```
{
   host: "mailserver.net",
   user: "defuser@mailserver.net",
   password: "defpassword",

   bob: {
      user: "bob@mailserver.net",
      password: "bobpass",
   },
   mark: {
      user: "mark@mailserver.net",
      password: "markpass",
   },
}
```

E1: Simple configuration
Config Class Implementation

```
...
@ConfigureMe(name = "mailconfig")
public class MailConfig {

    private static final MailConfig INSTANCE = new MailConfig();

    @Configure
    private String host;
    @Configure
    private String user;
    @Configure
    private String password;

    private MailConfig() {
        ConfigurationManager.INSTANCE.configure(this);
    }

    public static MailConfig getInstance() {
        return INSTANCE;
    }

    public String getHost() {
        return host;
    }

    public void setHost(String host) {
        this.host = host;
    }
    ...
}
```

E1: Simple configuration Using Code

```
String host = MailConfig.getInstance().getHost();
String user = MailConfig.getInstance().getUser();
String password = MailConfig.getInstance().getPassword();
```

E2: Using @SetAll
JSON Config

```
{
   propertyA: 123,
   propertyB: "valueC",
   propertyC: "valueC",
}
```

E2: Using @SetAll
Config Class Implementation

```
@ConfigureMe(name = "setall-example-config")
public class SetAllExampleConfig {

    private static final SetAllExampleConfig INSTANCE =
        new SetAllExampleConfig();

    private Map<String, String> propertiesMap;

    private SetAllExampleConfig() {
        propertiesMap = new HashMap<String, String>();
        ConfigurationManager.INSTANCE.configure(this);
    }

    public static SetAllExampleConfig getInstance() {
        return INSTANCE;
    }

    @SetAll
    public void allPropertiesHandler(String name, String value) {
        propertiesMap.put(name, value);
    }

    public Map<String, String> getPropertiesMap() {
        return propertiesMap;
    }
}
```

E2: Using @SetAll Using Code

```
Map<String, String> allProperties =
      SetAllExampleConfig.getInstance().getPropertiesMap();
System.out.println(allProperties);

// "{propertyC=valueC, propertyB=valueC, propertyA=123}"
```

E3: Using @SetIf
JSON Config

```
{
   url.1 = "http://buy.server.com",
   url.2 = "http://fun.server.com",
   url.3 = "http://dev.server.com",

   country.UK = "United Kingdom",
   country.US = "USA",
   country.IT = "Italy",
}
```

E3: Using @SetIf Config Class Implementation

```
@ConfigureMe(name = "setif-example-config")
public class SetIfExampleConfig {

    private static final String URLS_LIST_PREFIX = "url.";
    private static final String COUNTRIES_PREFIX = "country.";

    private static final SetIfExampleConfig INSTANCE = new SetIfExampleConfig();

    private List<String> urls;
    private Map<String, String> countryCodesToName;

    ...

    @SetIf(condition = SetIf.SetIfCondition.startsWith, value = URLS_LIST_PREFIX)
    public void putUrlsListItem(final String name, final String value) {
        urls.add(value);
    }

    @SetIf(condition = SetIf.SetIfCondition.startsWith, value = COUNTRIES_PREFIX)
    public void putCountryCodesToNameValue(final String name, final String value) {
        countryCodesToName.put(name.replaceAll(COUNTRIES_PREFIX, ""), value);
    }

    public List<String> getUrls() {
        return urls;
    }

    public Map<String, String> getCountryCodesToName() {
        return countryCodesToName;
    }
}
```

E3: Using @SetIf Using Code

```
List<String> urls = SetIfExampleConfig.getInstance().getUrls();
System.out.println(urls);
// "[http://dev.server.com, http://buy.server.com, http://fun.server.com]"

Map<String, String> countryCodeToName =
      SetIfExampleConfig.getInstance().getCountryCodesToName();
System.out.println(countryCodeToName);
// "{US=USA, UK=United Kingdom, IT=Italy}"
```

E4: Using String Arrays (Lists)
JSON Config

```
{
   actions: "CREATE,UPDATE,DELETE",
}
```

E4: Using String Arrays (Lists) Config Class Implementation

```
@ConfigureMe(name = "stringarray-example-config")
public class StringArrayExampleConfig {

    private static final StringArrayExampleConfig INSTANCE =
            new StringArrayExampleConfig();

    @Configure
    private String [] actions;

    private StringArrayExampleConfig() {
        ConfigurationManager.INSTANCE.configure(this);
    }

    public static StringArrayExampleConfig getInstance() {
        return INSTANCE;
    }

    public void setActions(String [] actions) {
        this.actions = actions;
    }

    public String [] getActions() {
        return actions;
    }
}
```

E4: Using String Arrays (Lists) Using Code

```
System.out.println(Arrays.toString(
     StringArrayExampleConfig.getInstance().getActions()));

// "[CREATE, UPDATE, DELETE]"
```