Matthias Bollwein, Thomas Endres, Andreas Würl

Stuttgart, 17.07.2014

TNG TECHNOLOGY CONSULTING

Overview

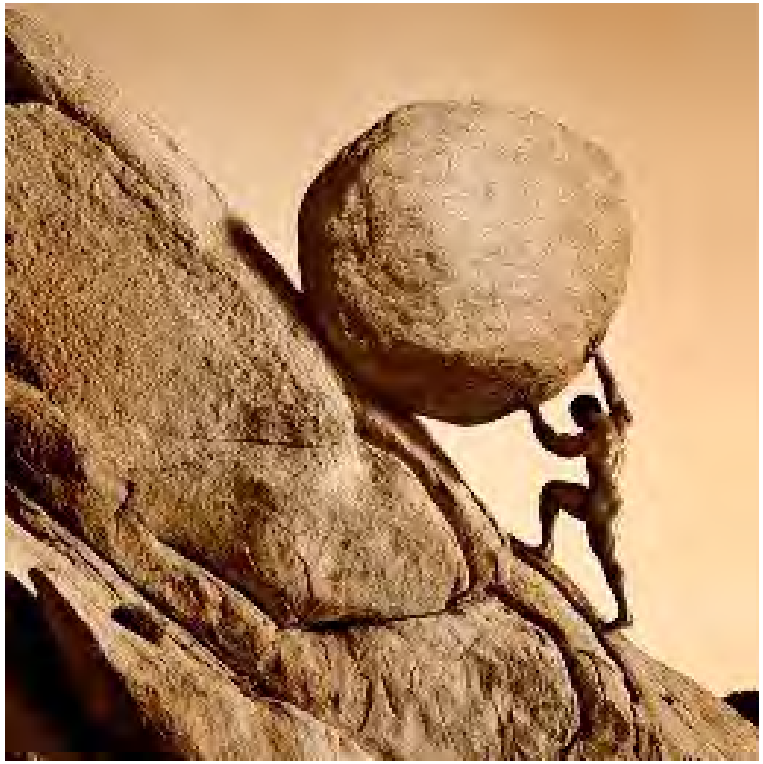
- Motivation
- Getting started
- Validation
- Property-Loader
- Extensions

Our config class

```
public class Config
{
    private String characterName;

    private int difficulty;

    private Color playerColor;

    private Set<Cheat> activeCheats;

    private String language;

    private Path currentDirectory;

    // Setters and getters
}
```

Typical architecture

```
public Config loadConfig() {
    Config config = new Config();

    loadBaseConfig(config);
    overwriteWithProperties(config);

    // do other specific stuff

    return config;
}
```



How about this?

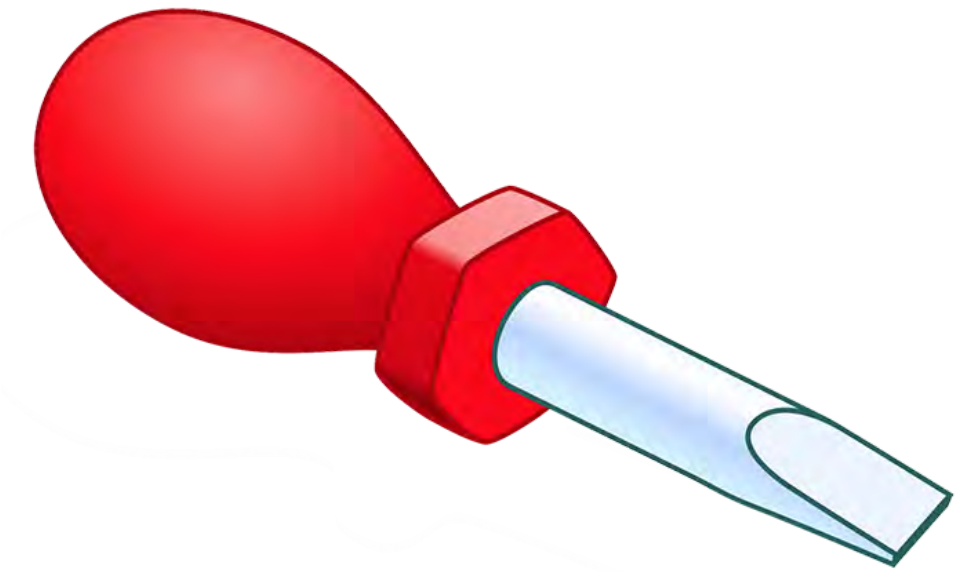
```
public Config loadConfig() {
    return ConfigBuilder.on(Config.class).build();
}
```


USED THE CONFIG BUILDER
GOT HOME EARLIER

Overview

Basic tool box

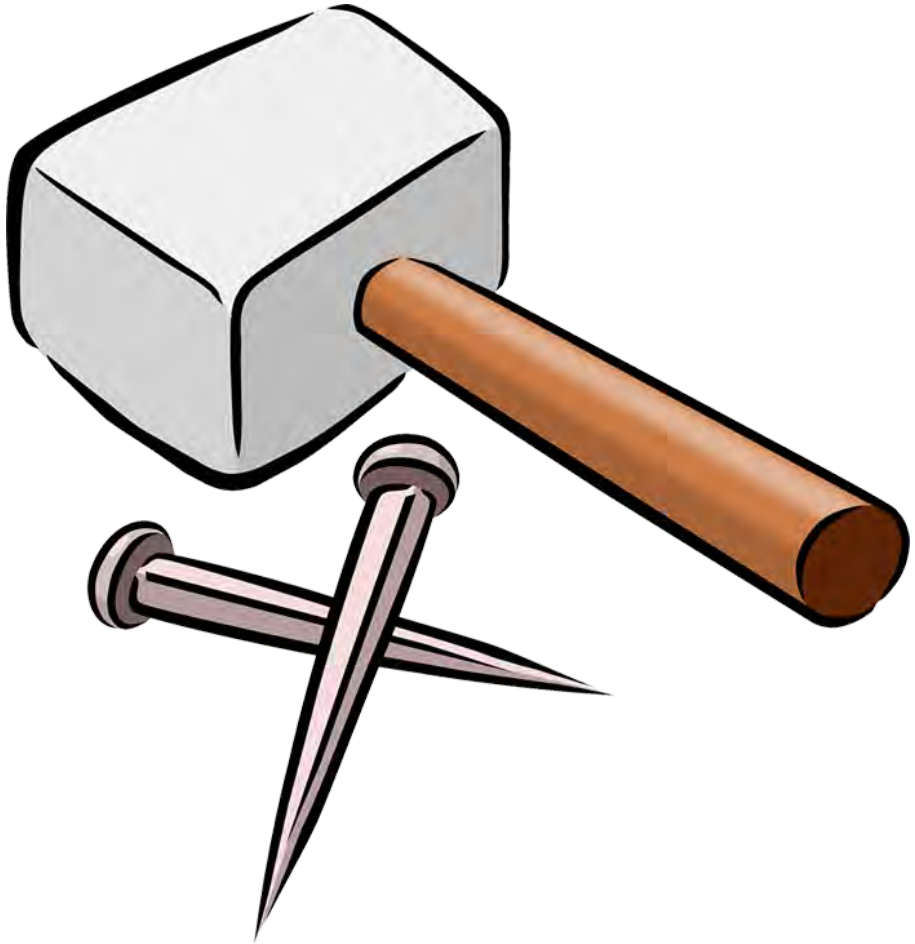
Default values



Default values

```
public class Config
{
    @DefaultValue("Player")
    private String characterName;

    @DefaultValue("1")
    private int difficulty;

    // ...
}
```

Command line values



Command line values

```
public class Config
{
    @DefaultValue("Player")
    @CommandLineValue(shortOpt="c", longOpt="character-name", hasArg=true)
    private String characterName;

    @DefaultValue("1")
    @CommandLineValue(shortOpt="d", longOpt="difficulty", hasArg=true)
    private int difficulty;

    // ...
}
```

Property values

Property values

```
@PropertiesFiles({"config", "test-config"})
public class Config
{
    // ...

    @DefaultValue("1")
    @CommandLineValue(shortOpt="d", longOpt="difficulty", hasArg=true)
    @PropertyValue("difficulty")
    private int difficulty;

    @PropertyValue("player.color")
    private int playerColor;

    // ...
}
```

Env variables and system properties

Env vars/system properties

```
public class Config
{
    // ...

    @SystemPropertyValue("user.language")
    private String language;

    @EnvironmentVariableValue("PWD")
    private Path currentDirectory;

    // ...
}
```

Type Conversion

Our config class

```
public class Config
{
    // ...

    @PropertyValue("player.color")
    private Color playerColor;

    @PropertyValue("cheats.active")
    private Set<Cheat> activeCheats;

    @EnvironmentVariableValue("PWD")
    private Path currentDirectory;

    // Setters and getters
}
```

Transformer for Color

```
public class Config
{
    // ...

    @PropertyValue("player.color")
    @TypeTransformers(StringToColorTransformer.class)
    private Color playerColor;

    // ...

    public static class StringToColorTransformer
        extends TypeTransformer<String, Color> {

        @Override
        public Color transform(String colorText) {
            StyleSheet styleSheet = new StyleSheet();
            return styleSheet.stringToColor(colorText);
        }
    }
}
```

Transformer for Set<Cheat>

```
public class Config
{

    @PropertyValue("cheats.active")
    @TypeTransformers(StringToCheatTransformer.class)
    private Set<Cheat> cheats;

    // ...

    public static class StringToCheatTransformer
        extends TypeTransformer<String, Cheat> {

        @Override
        public Cheat transform(String cheatText) {
            return Cheat.valueOf(cheatText);
        }
    }
}
```

Tools so far

- Value sources
  - Default values
  - Properties files
  - System properties
  - Environment variables
- Type transformers
  - Primitive/Collection (integrated)
  - Custom transformers
  - with transitive combination

Final config class

```
@PropertiesFiles({"config", "test-config"})
public class Config
{
    @DefaultValue("Player")
    @CommandLineValue(shortOpt="c", longOpt="character-name", hasArg=true)
    private String characterName;

    @DefaultValue("1")
    @CommandLineValue(shortOpt="d", longOpt="difficulty", hasArg=true)
    @PropertyValue("difficulty")
    private int difficulty;

    @PropertyValue("player.color")
    @TypeTransformers(StringToColorTransformer.class)
    private Color playerColor;

    @PropertyValue("cheats.active")
    @TypeTransformers(StringToCheatTransformer.class)
    private Set<Cheat> activeCheats;

    @SystemPropertyValue("user.language")
    private String language;

    @EnvironmentVariableValue("PWD")
    private Path currentDirectory;

    public static class StringToColorTransformer extends TypeTransformer<String, Color> {
        @Override
        public Color transform(String colorText) {
            StyleSheet styleSheet = new StyleSheet();
            return styleSheet.stringToColor(colorText);
        }
    }

    public class StringToCheatTransformer extends TypeTransformer<String, Cheat> {
        @Override
        public Cheat transform(String cheatText) {
            return Cheat.valueOf(cheatText);
        }
    }

    // Setters and getters
}
```

Example main class

```
public class GameEntry {

    private Config config;

    public static void main(final String[] args) {

        config = ConfigBuilder.on(Config.class)
                              .withCommandLineArgs(args)
                              .build();

        // run game with config ...

    }
}
```

Results so far

Contents of config.properties file:

```
player.color = red
cheats.active = WALLHACK, GODMODE
```

Command line input:

```
> cd /home/player/new_game
> ./new-game -d 10 -c winner
```

Resulting Config object:

```
config = {
    characterName = "winner",
    difficulty = 10,
    playerColor = Color(255, 0, 0),
    activeCheats = {Cheat.WALLHACK, Cheat.GODMODE},
    language = "de",
    currentDirectory = Path("/home/player/new_game")
}
```

Overview

- Motivation
- Getting started
- Validation
- Property-Loader
- Extensions

Validation

JSR303

```
public class Config
{
    // ...

    @Size(min = 2, max = 14)
    @DefaultValue("Player")
    @CommandLineValue(shortOpt="c", longOpt="character-name", hasArg=true)
    private String characterName;

    @NotNull
    @EnvironmentVariableValue("PWD")
    private Path currentDirectory;

    // Setters and getters
}
```

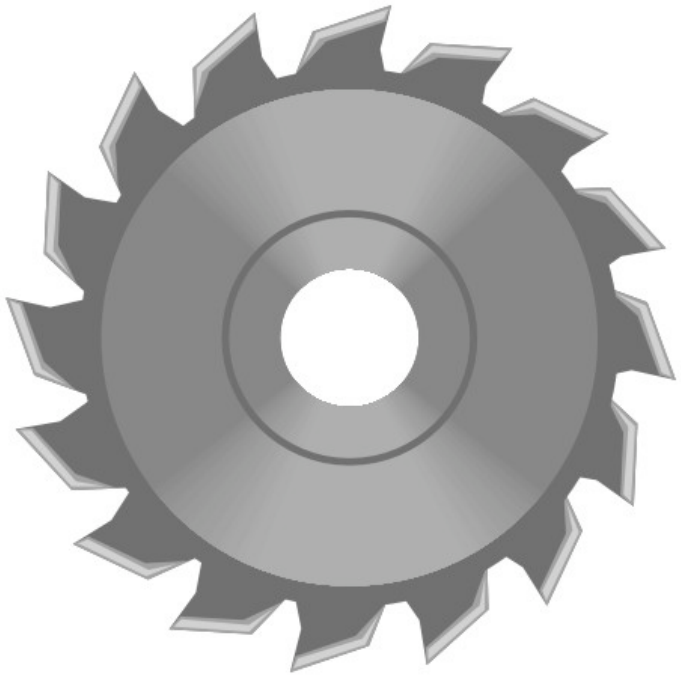
Validate method

```
public class Config
{
    // ...

    @Size(min = 2, max = 14)
    @DefaultValue("Player")
    @CommandLineValue(shortOpt="c", longOpt="character-name", hasArg=true)
    private String characterName;

    @NotNull
    @EnvironmentVariableValue("PWD")
    private Path currentDirectory;

    @Validation
    private void validate() {
        //Do whatever we want...
    }

    // Setters and getters
}
```

Overview

- Motivation
- Getting started
- Validation
- Property-Loader
- Extensions

Property-Loader



File Names, Suffixes & Extensions

```
@PropertiesFiles({"config", "test-config"})
@PropertySuffixes(extraSuffixes = {"tngtech","myname"}, hostNames = true)
@PropertyExtension("fileextension")
public class Config {

    @PropertyValue("cheats.active")
    @TypeTransformers(StringToCheatTransformer.class)
    private Set<Cheat> cheats;

    // ...
}
```

Locations

```
@PropertyLocations(directories = {"/home/user"}, contextClassLoader = true)
@PropertiesFiles({"config", "test-config"})
public class Config {

    @PropertyValue("cheats.active")
    @TypeTransformers(StringToCheatTransformer.class)
    private Set<Cheat> cheats;

    // ...
}
```

Property Loader Overview

- List of file basenames
- File extension
- Suffixes
  - User name
  - Local host name
  - "override"
  - Custom extra suffixes
- Search locations
  - Classpath
  - Home directory
  - Custom directories

Overview

- Motivation
- Getting started
- Validation
- Property-Loader
- Extensions

Custom tools

Value source extension – File content value

- Custom Annotation

```
@ValueExtractorAnnotation(DefaultValueProcessor.class)
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface FileContentValue {
    String value();
    String encoding() default "UTF-8";
}
```

- Annotation Processor

```
public class DefaultValueProcessor implements ValueExtractorProcessor {
    public String getValue(Annotation ann,
                           ConfigBuilderFactory configBuilderFactory) {
        try {
            FileContentValue fileContentValue = (FileContentValue) ann;
            byte[] content = Files.readAllBytes(
                 Paths.get(fileContentValue.value()));
            return new String(content, fileContentValue.encoding());
        } catch(IOException e) { e.printStackTrace(); }
    }
}
```

Custom Annotation Example

```
public class Config {

    @FileContentValue("/home/user/help.txt", encoding="ISO-8859-1")
    private String helptext;

    // ...
}
```

How to use ConfigBuilder?

Just add the Dependency!

For example with Maven:

```
<dependency>
    <groupId>com.tngtech.java</groupId>
    <artifactId>config-builder</artifactId>
    <version>1.3</version>
</dependency>
```

Requirements

- Java 7
- Java 6 support is planned
- Libraries used:
  - Property Loader
  - Google Reflections
  - Logging Abstraction via SLF4J
  - Commons CLI

Can I add features?

ConfigBuilder and PropertyLoader are open source licensed under Apache 2.0

Feel free to fork them on GitHub

- https://github.com/TNG/config-builder
- https://github.com/TNG/property-loader

https://github.com/TNG/config-builder
http://www.tngtech.com

TNG TECHNOLOGY CONSULTING

Special thanks

- Various TNG employees for creating the property loader
- http://openclipart.org for providing Public Domain pictures
- Jeremy Chapman for the multimeter image