

---

# **cauldron Documentation**

***Release 0.0.0***

**Azat Ibrakov**

**Nov 25, 2017**



---

## Contents:

---

|          |                             |           |
|----------|-----------------------------|-----------|
| <b>1</b> | <b>Submodules</b>           | <b>1</b>  |
| 1.1      | bases module . . . . .      | 1         |
| 1.2      | just module . . . . .       | 4         |
| 1.3      | booleans module . . . . .   | 5         |
| 1.4      | integers module . . . . .   | 5         |
| 1.5      | characters module . . . . . | 6         |
| 1.6      | strings module . . . . .    | 6         |
| 1.7      | vectors module . . . . .    | 7         |
| 1.8      | sets module . . . . .       | 8         |
| 1.9      | builder module . . . . .    | 8         |
| 1.10     | sieve module . . . . .      | 10        |
| 1.11     | facility module . . . . .   | 11        |
| <b>2</b> | <b>Indices and tables</b>   | <b>13</b> |



## 1.1 bases module

### namespace cauldron

**template** <typename *Value*, class *Derived*>

**class** **CloneHelper**

*#include* <bases.h> Helper class for implementing *Strategy::clone* method using *curiously recurring template pattern* .

#### Template Parameters

- *Value*: type of values generated by strategy.
- *Derived*: derivative strategy class.

Inherits from *cauldron::Strategy< Value >*

#### Public Functions

*std::unique\_ptr<Strategy<Value>>* **clone** () **const**

Creates a copy of the strategy instance and returns pointer to it.

**template** <typename *Value*>

**class** **Filtered**

*#include* <bases.h> *Strategy* which filters out values that satisfies defined *strategies::Requirement* instances.

Hereafter a **requirement** is an *unary predicate* .

#### Template Parameters

- *Value*: type of values generated by strategy.

Inherits from *cauldron::CloneHelper< Value, Filtered< Value > >*

## Public Functions

**Filtered** (const *Strategy*<Value> &*strategy*, const *Sieve*<Value> &*sieve*)

**Filtered** (const *Filtered*<Value> &*strategy*)

Default copy constructor doesn't fit since we're using `std::unique_ptr` as class member which is not copyable.

*Filtered*<Value> **filter** (const Requirement<Value> &*requirement*) const

Returns a new strategy that generates values from the strategy which satisfy provided `strategies::Requirement` instance.

Note that if the `requirement` is too hard to satisfy this might result in failing with *OutOfCycles*.

Value **operator** () () const

Generates value that satisfies defined `strategies::Requirement` instances.

### Exceptions

- `strategies::OutOfCycles`:

## Protected Attributes

`std::unique_ptr<Strategy<Value>>` **strategy\_**

*Sieve*<Value> **sieve\_**

**template** <typename Value>

**class Mapped**

*#include* <bases.h> *Strategy* which modifies values with defined `strategies::Converter` instances.

Hereafter a **converter** is an *operator* .

### Template Parameters

- Value: type of values generated by strategy.

Inherits from *cauldron::CloneHelper*< Value, *Mapped*< Value > >

## Public Functions

**Mapped** (const *Strategy*<Value> &*strategy*, const *Facility*<Value> &*facility*)

**Mapped** (const *Mapped*<Value> &*strategy*)

Default copy constructor doesn't fit since we're using `std::unique_ptr` as class member which is not copyable.

*Mapped*<Value> **map** (const Converter<Value> &*converter*) const

Returns a new strategy that generates values from the strategy modified with provided `strategies::Converter` instance.

Value **operator** () () const

Generates value and modifies it with defined `strategies::Converter` instances.

## Protected Attributes

`std::unique_ptr<Strategy<Value>> strategy_`

`Facility<Value> facility_`

**template** <typename Value>

**class Strategy**

*#include <bases.h>* Strategies base class.

Hereafter **strategy** is an object which encapsulates an algorithm for generating specific type of values.

### Template Parameters

- Value: type of values generated by strategy.

Subclassed by `cauldron::CloneHelper< Value, Derived >`, `cauldron::CloneHelper< Value, Filtered< Value >>`, `cauldron::CloneHelper< Value, Floats< Value >>`, `cauldron::CloneHelper< Value, Integers< Value >>`, `cauldron::CloneHelper< Value, Just< Value >>`, `cauldron::CloneHelper< Value, Mapped< Value >>`, `cauldron::CloneHelper< Value, Union< Value >>`

## Public Functions

**virtual Value operator () () const = 0**

Generates data.

**virtual Union<Value> operator | (const Strategy<Value> &strategy) const**

Returns a new strategy that generates values from either original strategy or provided one.

**virtual Union<Value> operator | (const Union<Value> &strategy) const**

Same result as for any other strategy. Basically added for possible overriding with support of strategies' union associativity.

**virtual Filtered<Value> filter (const Requirement<Value> &requirement) const**

Returns a new strategy that generates values from the strategy which satisfy provided `strategies::Requirement` instance.

Note that if the `requirement` is too hard to satisfy this might result in failing with `OutOfCycles`.

**virtual Mapped<Value> map (const Converter<Value> &converter) const**

Returns a new strategy that generates values from the strategy modified with provided `strategies::Converter` instance.

**virtual std::unique\_ptr<Strategy<Value>> clone () const = 0**

Creates a copy of the strategy instance and returns pointer to it.

**template** <class Value>

**class Union**

*#include <bases.h>* `Union` of strategies. Generates values from any one of given strategies.

### Template Parameters

- Value: type of values generated by strategy.

Inherits from `cauldron::CloneHelper< Value, Union< Value >>`

## Public Functions

**Union** (const *Strategy*<Value> &*strategy*, const *Strategy*<Value> &*other\_strategy*)

**Union** (const *Union*<Value> &*strategy*)

Default copy constructor doesn't fit since we're using `std::unique_ptr` in class member which is not copyable.

Value **operator** () () **const**

Generates value from one of `strategies_` elements.

*Union*<Value> **operator** | | (const *Strategy*<Value> &*strategy*) **const**

Returns a new strategy that generates values from either original strategy or provided one.

*Union*<Value> **operator** | | (const *Union*<Value> &*strategy*) **const**

Same result as for any other strategy. Basically added for possible overriding with support of strategies' union associativity.

## Private Members

`std::vector<std::unique_ptr<Strategy<Value>>>` **strategies\_**

# 1.2 just module

namespace cauldron

template <typename Value>

class **Just**

*#include <just.h>* *Strategy* which generates the same value.

## Template Parameters

- Value: type of value generated by strategy.

Inherits from *cauldron::CloneHelper*< Value, *Just*< Value > >

## Public Functions

**Just** (Value *value*)

### Parameters

- value: value to generate.

Value **operator** () () **const**

Generates the same value.

## Private Members

Value **value\_**

## 1.3 booleans module

namespace cauldron

**class Booleans**

*#include <booleans.h> Strategy* which generates pseudo-random `bool` values with defined probability.

Inherits from *cauldron::CloneHelper< bool, Booleans >*

### Public Functions

**Booleans** (double *probability* = 0.5)

#### Parameters

- *probability*: how often `true` values will be generated compared to `false` values.

`bool operator () () const`

Generates pseudo-random `bool` value.

### Private Members

double **probability\_**

## 1.4 integers module

namespace cauldron

**template <typename Value>**

**class Integers**

*#include <integers.h> Strategy* which generates pseudo-random integer values.

### Template Parameters

- *Value*: type of values generated by strategy.

Inherits from *cauldron::CloneHelper< Value, Integers< Value > >*

### Public Functions

**Integers** (Value *min\_value* = `std::numeric_limits< Value >::min()`, Value *max\_value* = `std::numeric_limits< Value >::max()`)

#### Parameters

- *min\_value*: minimum possible integer value.
- *max\_value*: maximum possible integer value.

Value **operator () () const**

Generates pseudo-random integer value.

### Private Members

Value `min_value_`

Value `max_value_`

## 1.5 characters module

namespace cauldron

### class Characters

*#include <characters.h>* *Strategy* which generates pseudo-random char values.

Inherits from *cauldron::CloneHelper< char, Characters >*

### Public Functions

**Characters** (`const` std::string &*domain*)

**Characters** (`const` char *domain*[])

char **operator** () () `const`  
Generates pseudo-random char value.

### Private Members

std::string **domain\_**

## 1.6 strings module

namespace cauldron

### class Strings

*#include <strings.h>* *Strategy* which generates std::string instances with lengths and characters generated from corresponding strategies.

Inherits from *cauldron::CloneHelper< std::string, Strings >*

### Public Functions

**Strings** (`const` *Strategy*<size\_t> &*lengths*, `const` *Strategy*<char> &*alphabet*)

#### Parameters

- lengths: strategy to generate strings lengths from.
- alphabet: strategy to generate strings characters from.

**Strings** (`const` *Strings* &*strings*)

Default copy constructor doesn't fit since we're using std::unique\_ptr as class members which is not copyable.

```
std::string operator() () const
    Generates pseudo-random std::string instance.
```

### Private Members

```
std::unique_ptr<Strategy<size_t>> lengths_
std::unique_ptr<Strategy<char>> alphabet_
```

## 1.7 vectors module

namespace cauldron

```
template <typename Element>
class Vectors
```

*#include <vectors.h>* *Strategy* which generates `std::vector` instances with sizes and elements generated from corresponding strategies.

### Template Parameters

- `Element`: type of vectors elements generated by strategy.

Inherits from *cauldron::CloneHelper*< *std::vector*< *Element* >, *Vectors*< *Element* > >

### Public Functions

```
Vectors (const SizesStrategy &sizes, const ElementsStrategy &elements)
```

#### Parameters

- `sizes`: strategy to generate vectors sizes from.
- `elements`: strategy to generate vectors elements from.

```
Vectors (const Vectors<Element> &vectors)
```

Default copy constructor doesn't fit since we're using `std::unique_ptr` as class members which is not copyable.

```
std::vector<Element> operator() () const
    Generates pseudo-random std::vector instance.
```

### Private Types

```
template<>
using SizesStrategy = Strategy<size_t>

template<>
using ElementsStrategy = Strategy<Element>
```

### Private Members

```
std::unique_ptr<SizesStrategy> sizes_
std::unique_ptr<ElementsStrategy> elements_
```

## 1.8 sets module

namespace cauldron

```
template <typename Element>
```

```
class Sets
```

*#include <sets.h>* *Strategy* which generates `std::set` instances with sizes and elements generated from corresponding strategies.

### Template Parameters

- `Element`: type of sets elements generated by strategy.

Inherits from *cauldron::CloneHelper*< *std::set*< *Element* >, *Sets*< *Element* > >

### Public Functions

```
Sets (const SizesStrategy &sizes, const ElementsStrategy &elements)
```

#### Parameters

- `sizes`: strategy to generate sets sizes from.
- `elements`: strategy to generate sets elements from.

```
Sets (const Sets<Element> &sets)
```

Default copy constructor doesn't fit since we're using `std::unique_ptr` as class members which is not copyable.

```
std::set<Element> operator () () const
```

Generates pseudo-random `std::set` instance.

### Private Types

```
template<>
```

```
using SizesStrategy = Strategy<size_t>
```

```
template<>
```

```
using ElementsStrategy = Strategy<Element>
```

### Private Members

```
std::unique_ptr<SizesStrategy> sizes_
```

```
std::unique_ptr<ElementsStrategy> elements_
```

## 1.9 builder module

namespace cauldron

```
template <class Object, class... Value>
```

```
class Builder
```

*#include <builder.h>* *Strategy* which generates `Object` instances with constructor arguments generated from corresponding strategies.

**Note** Object should have appropriate constructor so types and order of passed strategies agree with types and order of constructor parameters or compile-time error will arise.

## Template Parameters

- **Object**: type of objects generated by strategy.

Inherits from *cauldron::CloneHelper< Object, Builder< Object, Value... > >*

## Public Functions

**Builder** (const *cauldron::Strategy*<Value>&... *strategy*)

## Parameters

- `strategy`: strategy to generate constructor arguments from.

**Builder** (const *Builder*<Object, Value...> &*builder*)

Default copy constructor doesn't fit since we're using `std::unique_ptr` as class members which is not copyable.

## Object operator () () const

Generates pseudo-random Object instance.

## Private Functions

**template** <std::size\_t... *Indices*>

```
Object produce (const std::tuple<std::unique_ptr<cauldron::Strategy<Value>>...> &strategies,
std::index_sequence<Indices...>) const
```

Helper function for unpacking `Builder::strategies_` tuple into variadic *Strategy* instances.

Object **produce** (const std::unique\_ptr<*cauldron::Strategy*<Value>>&... strategy) const

Helper function for producing values from variadic *Strategy* instances.

```
std::tuple<std::unique_ptr<cauldron::Strategy<Value>>...> clone_strategies () const
```

**template** <std::size\_t... *Indices*>

```
std::tuple<std::unique_ptr<cauldron::Strategy<Value>>...> clone_strategies (const
                                                                    std::tuple<std::unique_ptr<cauldron::S
                                                                    &strategies,
                                                                    std::index_sequence<Indices...>)
const
```

Helper function for unpacking `Builder::strategies_` tuple into variadic `Strategy` instances.

[illegible]

Helper function for cloning variadic *Strategy* instances.

## Private Members

```
std::tuple<std::unique_ptr<cauldron::Strategy<Value>>...> strategies_
```

## 1.10 sieve module

namespace cauldron

### Typedefs

```
using cauldron::Requirement = typedef std::function<bool(Product)>
```

### Variables

```
const unsigned MAX_CYCLES = 1'000
```

```
class OutOfCycles  
    #include <sieve.h> Inherits from exception
```

### Public Functions

```
OutOfCycles (unsigned max_cycles)
```

```
const char *what () const
```

### Private Members

```
std::string msg_
```

```
template <typename Product>
```

```
class Sieve  
    #include <sieve.h>
```

### Public Functions

```
Sieve (unsigned max_cycles = MAX_CYCLES)
```

```
Sieve (std::initializer_list<Requirement<Product>> requirements, unsigned max_cycles =  
    MAX_CYCLES)
```

```
Sieve (const std::vector<Requirement<Product>> &requirements, unsigned max_cycles =  
    MAX_CYCLES)
```

```
bool satisfactory (Product product) const
```

```
Sieve<Product> expand (const Requirement<Product> &requirement) const
```

```
Product sift (std::function<Product>  
    > producer const
```

### Protected Attributes

```
std::vector<Requirement<Product>> requirements_
```

```
unsigned max_cycles_
```

## 1.11 facility module

namespace cauldron

### Typedefs

```
using cauldron::Converter = typedef std::function<Product (Product)>  
template <typename Product>  
class Facility  
    #include <facility.h>
```

### Public Functions

```
Facility()  
Facility (std::initializer_list<Converter<Product>> converters)  
Facility (const std::vector<Converter<Product>> &converters)  
Facility<Product> expand (const Converter<Product> &converter) const  
Product convert (Product product) const
```

### Protected Attributes

```
std::vector<Converter<Product>> converters_
```



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## C

- cauldron (C++ type), 1, 4–8, 10, 11
- cauldron::Booleans (C++ class), 5
- cauldron::Booleans::Booleans (C++ function), 5
- cauldron::Booleans::operator() (C++ function), 5
- cauldron::Booleans::probability\_ (C++ member), 5
- cauldron::Builder (C++ class), 8
- cauldron::Builder::Builder (C++ function), 9
- cauldron::Builder::clone\_strategies (C++ function), 9
- cauldron::Builder::operator() (C++ function), 9
- cauldron::Builder::produce (C++ function), 9
- cauldron::Builder::strategies\_ (C++ member), 9
- cauldron::Characters (C++ class), 6
- cauldron::Characters::Characters (C++ function), 6
- cauldron::Characters::domain\_ (C++ member), 6
- cauldron::Characters::operator() (C++ function), 6
- cauldron::CloneHelper (C++ class), 1
- cauldron::CloneHelper::clone (C++ function), 1
- cauldron::Facility (C++ class), 11
- cauldron::Facility::convert (C++ function), 11
- cauldron::Facility::converters\_ (C++ member), 11
- cauldron::Facility::expand (C++ function), 11
- cauldron::Facility::Facility (C++ function), 11
- cauldron::Filtered (C++ class), 1
- cauldron::Filtered::filter (C++ function), 2
- cauldron::Filtered::Filtered (C++ function), 2
- cauldron::Filtered::operator() (C++ function), 2
- cauldron::Filtered::sieve\_ (C++ member), 2
- cauldron::Filtered::strategy\_ (C++ member), 2
- cauldron::Integers (C++ class), 5
- cauldron::Integers::Integers (C++ function), 5
- cauldron::Integers::max\_value\_ (C++ member), 6
- cauldron::Integers::min\_value\_ (C++ member), 6
- cauldron::Integers::operator() (C++ function), 5
- cauldron::Just (C++ class), 4
- cauldron::Just::Just (C++ function), 4
- cauldron::Just::operator() (C++ function), 4
- cauldron::Just::value\_ (C++ member), 4
- cauldron::Mapped (C++ class), 2
- cauldron::Mapped::facility\_ (C++ member), 3
- cauldron::Mapped::map (C++ function), 2
- cauldron::Mapped::Mapped (C++ function), 2
- cauldron::Mapped::operator() (C++ function), 2
- cauldron::Mapped::strategy\_ (C++ member), 3
- cauldron::MAX\_CYCLES (C++ member), 10
- cauldron::OutOfCycles (C++ class), 10
- cauldron::OutOfCycles::msg\_ (C++ member), 10
- cauldron::OutOfCycles::OutOfCycles (C++ function), 10
- cauldron::OutOfCycles::what (C++ function), 10
- cauldron::Sets (C++ class), 8
- cauldron::Sets::elements\_ (C++ member), 8
- cauldron::Sets::operator() (C++ function), 8
- cauldron::Sets::Sets (C++ function), 8
- cauldron::Sets::sizes\_ (C++ member), 8
- cauldron::Sets<Element>::ElementsStrategy (C++ type), 8
- cauldron::Sets<Element>::SizesStrategy (C++ type), 8
- cauldron::Sieve (C++ class), 10
- cauldron::Sieve::expand (C++ function), 10
- cauldron::Sieve::max\_cycles\_ (C++ member), 10
- cauldron::Sieve::requirements\_ (C++ member), 10
- cauldron::Sieve::satisfactory (C++ function), 10
- cauldron::Sieve::Sieve (C++ function), 10
- cauldron::Sieve::sift (C++ function), 10
- cauldron::Strategy (C++ class), 3
- cauldron::Strategy::clone (C++ function), 3
- cauldron::Strategy::filter (C++ function), 3
- cauldron::Strategy::map (C++ function), 3
- cauldron::Strategy::operator() (C++ function), 3
- cauldron::Strategy::operator|| (C++ function), 3
- cauldron::Strings (C++ class), 6
- cauldron::Strings::alphabet\_ (C++ member), 7
- cauldron::Strings::lengths\_ (C++ member), 7
- cauldron::Strings::operator() (C++ function), 7
- cauldron::Strings::Strings (C++ function), 6
- cauldron::Union (C++ class), 3
- cauldron::Union::operator() (C++ function), 4
- cauldron::Union::operator|| (C++ function), 4
- cauldron::Union::strategies\_ (C++ member), 4

cauldron::Union::Union (C++ function), [4](#)  
cauldron::Vectors (C++ class), [7](#)  
cauldron::Vectors::elements\_ (C++ member), [7](#)  
cauldron::Vectors::operator() (C++ function), [7](#)  
cauldron::Vectors::sizes\_ (C++ member), [7](#)  
cauldron::Vectors::Vectors (C++ function), [7](#)  
cauldron::Vectors<Element>::ElementsStrategy (C++  
type), [7](#)  
cauldron::Vectors<Element>::SizesStrategy (C++ type),  
[7](#)