code-blocks

Release 0.1.2

Nov 09, 2022

this is code-blocks

#Code block

Provide various component demonstrations, written in the form of code blocks; Use Multi Project Schema

###Project Structure

```
|--Code blocks: root project, no code is written, only all subprojects are integrated

|--Code blocks common:: general module

|--Code blocks spring:: based on spring general module

|--Code blocks spring boot:: based on spring boot general module
```

###New features

CHAPTER 1

code-blocks

Version: 0.1.4

-Add gradle.properties file to handle general version

CHAPTER 2

code-blocks-common

Version: 0.1.2

-Definition of exception class

-Use of Juint

-Definition of Enumeration

- rpc
- cglib. Jdk: jdk dynamic proxy, implementing the interface InvocationHandler
- cglib. Cglib: cglib dynamic proxy, implementing the interface MethodInterceptor
- cglib. Demo: static proxy and dynamic proxy demo

-JsonUtils: Use of Jason's json

-TreeUtils: tree construction and parsing

-ListUtils: Sort

-CommonUtils: string length verification, BigDecimal conversion String length verification

-Base64Utils: Jdk Base64 encoding and decoding

-RsaUtils: RSA asymmetric encryption and decryption

-AesUtils: AES symmetric encryption and decryption

-DataTypeUtils: string to byte [32], byte [32] to string

Version: 0.1.2

-PropertyUtils: Read configuration file

-JsonUtils: Update the configuration used by jackson and initialize variables

Version: 0.1.3

-ThreadPoolUtils: Use of thread pool

-JsonUtils: Add generic conversion

#####Differences between jdk dynamic proxy and cglib dynamic proxy

> The dynamic proxy mechanism of JDK can only proxy interface classes, and non interface classes cannot implement the dynamic proxy of JDK. Its principle is to use Proxy The proxy object created by newProxyInstance is an object dynamically generated at the jvm runtime. It is not our InvocationHandler type or the type of the set of interfaces we define, but an object dynamically generated at runtime. The naming method is the same, starting with $, "proxy" in the middle, and the last number represents the label of the object;
>
> Cglib implements proxies for classes. Its principle is to generate a subclass of the specified target class and override its method implementation enhancements. However, because it uses inheritance, it cannot proxy final modified classes;

CHAPTER 3

code-blocks-spring

Version: 0.1.1

-Spring Framework Read Configuration File

-Spring cglib generates POJOs

CHAPTER 4

code-blocks-springboot

Version: 0.1.1

-Aspect: log interception

-Implement the CommandLineRunner interface: perform a class of operations after startup

-Implement ApplicationContextAware interface: get beans by getting context

-Implement scheduled tasks

-Spring boot uses the following two methods to load spring xml files

- @ImportResource(locations={”classpath:codeBolcksSpringbootApplicationContext.xml”})

-ClassPathXmlApplicationContext method, see BaseServiceImpl.java of code blocks spring

Version: 0.1.3

-HttpAspect: optimize aop logs

-CommonExceptionHandler: add aop exception handling

-@ Valid @ Validated: increase the verification usage

-Mybatis: add mybatis database field to transfer to hump

Version: 0.1.4

-HttpFilter: request interception filter, handling cross domain in gateway mode

-CrosConfig: universal cross domain processing

CHAPTER 5

code-blocks-springboot-jdbc

Version: 0.1.1

-Add, delete, modify and query Mybatis

CHAPTER 6

code-blocks-designpattern

Version: 0.1.1

Creation mode

These design patterns provide a way to hide the creation logic while creating objects,

Instead of using the new operator to directly instantiate the object< br>

This makes the program more flexible in determining which objects need to be created for a given instance< br>

Factory Pattern

Abstract Factory Pattern

Singleton Pattern

Builder Pattern

Prototype Pattern

Structural mode

These design patterns focus on the combination of classes and objects< br>

The concept of inheritance is used to combine interfaces and define how composite objects can obtain new functions< br>

Adapter Pattern

Bridge Pattern

Filter, Criteria Pattern

Composite Pattern

Decorator Pattern

Facade Pattern

Flyweight Pattern

Proxy Pattern

Behavioral mode

These design patterns pay particular attention to communication between objects< br>

Chain of Responsibility Pattern

Command Pattern

Interpreter Pattern

Iterator Pattern

Mediator Pattern

Memo Pattern

Observer Pattern

State Pattern

Null Object Pattern

Strategy Pattern

Template Pattern

Visitor Pattern

J2EE pattern

These design patterns pay particular attention to the presentation layer. These patterns are qualified by Sun Java Center< br>

MVC Pattern

Business Delegate Pattern

Composite Entity Pattern

Data Access Object Pattern

Front Controller Pattern

Intercepting Filter Pattern

Service Locator Pattern

Transfer Object Pattern

CHAPTER 7

code-blocks-springboot-security

Version: 0.1.1

-New

CHAPTER 8

code-blocks-springboot-eureka

Version: 0.1.1

-New

CHAPTER 9

code-blocks-springboot-eureka-article

Version: 0.1.1

-New

CHAPTER 10

code-blocks-springboot-eureka-discovery

Version: 0.1.1

-New

CHAPTER 11

Indices and tables

- genindex
- modindex
- search