目　录

套件规范

套件规范

工程规范

数量规范

请遵守一套件下只包含一个后端工程、一个前端工程、一个小程序工程

命名规范

使用套件名称[cn.flyrise.xxx]的后缀[xxx]进行相应命名，如下示例：

套件名称	后端	前端	小程序
cn.flyrise.business	pai-business	pai-business-ui	pai-business-micro
cn.flyrise.service.center	pai-service-center	pai-service-center-ui	pai-service-center-micro

代码规范

代码规范

请严格遵守本规范，提交代码前使用本规范提供的插件进行扫描，没问题再进行代码提交。本规范将纳入考核指标

中台开发规范

请严格遵守：http://api.flyrise.cn:9099/docs/open-docs/open-docs-1ctpqrs0h2ldr

API URL规范

通用规范

请严格遵守：http://api.flyrise.cn:9099/docs/open-docs//998

不允许大小写形式

不允许使用帕斯卡命名法(Pascal) 或驼峰命名法(Camel-Case)

使用 `斜杠 /` 代替上面的两种名称方式

- 反例： `/finance-api/receivableLedger/export`
- 正例： `/finance-api/receivable/ledger/export`

尽量简洁

- 反例： `/finance-api/feeCategory/acquisitionCode`
- 正例： `/finance-api/fee/category/code`

~~代码格式~~

因科技发展，大屏幕已是大势所趋，此格式限制行宽过窄，不再强调使用此格式，使用idea默认格式即可

intellij-java-google-style.xml

复制以下xml内容新建xml文件，导入idea->Preferences… -> Editor -> Code Style

```xml
<?xml version="1.0" encoding="UTF-8"?>
<code_scheme name="GoogleStyle">
  <option name="OTHER_INDENT_OPTIONS">
    <value>
      <option name="INDENT_SIZE" value="2" />
      <option name="CONTINUATION_INDENT_SIZE" value="4" />
      <option name="TAB_SIZE" value="2" />
      <option name="USE_TAB_CHARACTER" value="false" />
      <option name="SMART_TABS" value="false" />
      <option name="LABEL_INDENT_SIZE" value="0" />
      <option name="LABEL_INDENT_ABSOLUTE" value="false" />
      <option name="USE_RELATIVE_INDENTS" value="false" />
    </value>
  </option>
  <option name="INSERT_INNER_CLASS_IMPORTS" value="true" />
  <option name="CLASS_COUNT_TO_USE_IMPORT_ON_DEMAND" value="999" />
  <option name="NAMES_COUNT_TO_USE_IMPORT_ON_DEMAND" value="999" />
  <option name="PACKAGES_TO_USE_IMPORT_ON_DEMAND">
    <value />
  </option>
  <option name="IMPORT_LAYOUT_TABLE">
    <value>
      <package name="" withSubpackages="true" static="true" />
      <emptyLine />
      <package name="" withSubpackages="true" static="false" />
    </value>
  </option>
  <option name="RIGHT_MARGIN" value="100" />
  <option name="JD_ALIGN_PARAM_COMMENTS" value="false" />
  <option name="JD_ALIGN_EXCEPTION_COMMENTS" value="false" />
  <option name="JD_P_AT_EMPTY_LINES" value="false" />
  <option name="JD_KEEP_EMPTY_PARAMETER" value="false" />
  <option name="JD_KEEP_EMPTY_EXCEPTION" value="false" />
  <option name="JD_KEEP_EMPTY_RETURN" value="false" />
  <option name="KEEP_CONTROL_STATEMENT_IN_ONE_LINE" value="false" />
  <option name="KEEP_BLANK_LINES_IN_CODE" value="1" />
  <option name="BLANK_LINES_AFTER_CLASS_HEADER" value="1" />
  <option name="ALIGN_MULTILINE_PARAMETERS" value="false" />
  <option name="ALIGN_MULTILINE_FOR" value="false" />
  <option name="CALL_PARAMETERS_WRAP" value="1" />
  <option name="METHOD_PARAMETERS_WRAP" value="1" />
  <option name="EXTENDS_LIST_WRAP" value="1" />
  <option name="THROWS_KEYWORD_WRAP" value="1" />
  <option name="METHOD_CALL_CHAIN_WRAP" value="1" />
  <option name="BINARY_OPERATION_WRAP" value="1" />
  <option name="BINARY_OPERATION_SIGN_ON_NEXT_LINE" value="true" />
  <option name="TERNARY_OPERATION_WRAP" value="1" />
  <option name="TERNARY_OPERATION_SIGNS_ON_NEXT_LINE" value="true" />
```

```
    <option name="FOR_STATEMENT_WRAP" value="1" />
    <option name="ARRAY_INITIALIZER_WRAP" value="1" />
    <option name="WRAP_COMMENTS" value="true" />
    <option name="IF_BRACE_FORCE" value="3" />
    <option name="DOWHILE_BRACE_FORCE" value="3" />
    <option name="WHILE_BRACE_FORCE" value="3" />
    <option name="FOR_BRACE_FORCE" value="3" />
    <AndroidXmlCodeStyleSettings>
      <option name="USE_CUSTOM_SETTINGS" value="true" />
      <option name="LAYOUT_SETTINGS">
        <value>
          <option name="INSERT_BLANK_LINE_BEFORE_TAG" value="false" />
        </value>
      </option>
    </AndroidXmlCodeStyleSettings>
    <JSCodeStyleSettings>
      <option name="INDENT_CHAINED_CALLS" value="false" />
    </JSCodeStyleSettings>
    <Python>
      <option name="USE_CONTINUATION_INDENT_FOR_ARGUMENTS" value="true" />
    </Python>
    <TypeScriptCodeStyleSettings>
      <option name="INDENT_CHAINED_CALLS" value="false" />
    </TypeScriptCodeStyleSettings>
    <XML>
      <option name="XML_ALIGN_ATTRIBUTES" value="false" />
      <option name="XML_LEGACY_SETTINGS_IMPORTED" value="true" />
    </XML>
    <codeStyleSettings language="CSS">
      <indentOptions>
        <option name="INDENT_SIZE" value="2" />
        <option name="CONTINUATION_INDENT_SIZE" value="4" />
        <option name="TAB_SIZE" value="2" />
      </indentOptions>
    </codeStyleSettings>
    <codeStyleSettings language="ECMA Script Level 4">
      <option name="KEEP_BLANK_LINES_IN_CODE" value="1" />
      <option name="ALIGN_MULTILINE_PARAMETERS" value="false" />
      <option name="ALIGN_MULTILINE_FOR" value="false" />
      <option name="CALL_PARAMETERS_WRAP" value="1" />
      <option name="METHOD_PARAMETERS_WRAP" value="1" />
      <option name="EXTENDS_LIST_WRAP" value="1" />
      <option name="BINARY_OPERATION_WRAP" value="1" />
      <option name="BINARY_OPERATION_SIGN_ON_NEXT_LINE" value="true" />
      <option name="TERNARY_OPERATION_WRAP" value="1" />
      <option name="TERNARY_OPERATION_SIGNS_ON_NEXT_LINE" value="true" />
      <option name="FOR_STATEMENT_WRAP" value="1" />
      <option name="ARRAY_INITIALIZER_WRAP" value="1" />
```

```
    <option name="IF_BRACE_FORCE" value="3" />
    <option name="DOWHILE_BRACE_FORCE" value="3" />
    <option name="WHILE_BRACE_FORCE" value="3" />
    <option name="FOR_BRACE_FORCE" value="3" />
    <option name="PARENT_SETTINGS_INSTALLED" value="true" />
  </codeStyleSettings>
  <codeStyleSettings language="HTML">
    <indentOptions>
      <option name="INDENT_SIZE" value="2" />
      <option name="CONTINUATION_INDENT_SIZE" value="4" />
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
  </codeStyleSettings>
  <codeStyleSettings language="JAVA">
    <option name="KEEP_CONTROL_STATEMENT_IN_ONE_LINE" value="false" />
    <option name="KEEP_BLANK_LINES_IN_CODE" value="1" />
    <option name="BLANK_LINES_AFTER_CLASS_HEADER" value="1" />
    <option name="ALIGN_MULTILINE_PARAMETERS" value="false" />
    <option name="ALIGN_MULTILINE_RESOURCES" value="false" />
    <option name="ALIGN_MULTILINE_FOR" value="false" />
    <option name="CALL_PARAMETERS_WRAP" value="1" />
    <option name="METHOD_PARAMETERS_WRAP" value="1" />
    <option name="EXTENDS_LIST_WRAP" value="1" />
    <option name="THROWS_KEYWORD_WRAP" value="1" />
    <option name="METHOD_CALL_CHAIN_WRAP" value="1" />
    <option name="BINARY_OPERATION_WRAP" value="1" />
    <option name="BINARY_OPERATION_SIGN_ON_NEXT_LINE" value="true" />
    <option name="TERNARY_OPERATION_WRAP" value="1" />
    <option name="TERNARY_OPERATION_SIGNS_ON_NEXT_LINE" value="true" />
    <option name="FOR_STATEMENT_WRAP" value="1" />
    <option name="ARRAY_INITIALIZER_WRAP" value="1" />
    <option name="WRAP_COMMENTS" value="true" />
    <option name="IF_BRACE_FORCE" value="3" />
    <option name="DOWHILE_BRACE_FORCE" value="3" />
    <option name="WHILE_BRACE_FORCE" value="3" />
    <option name="FOR_BRACE_FORCE" value="3" />
    <option name="PARENT_SETTINGS_INSTALLED" value="true" />
    <indentOptions>
      <option name="INDENT_SIZE" value="2" />
      <option name="CONTINUATION_INDENT_SIZE" value="4" />
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
  </codeStyleSettings>
  <codeStyleSettings language="JSON">
    <indentOptions>
      <option name="CONTINUATION_INDENT_SIZE" value="4" />
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
```

```
    </codeStyleSettings>
    <codeStyleSettings language="JavaScript">
      <option name="RIGHT_MARGIN" value="80" />
      <option name="KEEP_BLANK_LINES_IN_CODE" value="1" />
      <option name="ALIGN_MULTILINE_PARAMETERS" value="false" />
      <option name="ALIGN_MULTILINE_FOR" value="false" />
      <option name="CALL_PARAMETERS_WRAP" value="1" />
      <option name="METHOD_PARAMETERS_WRAP" value="1" />
      <option name="BINARY_OPERATION_WRAP" value="1" />
      <option name="BINARY_OPERATION_SIGN_ON_NEXT_LINE" value="true" />
      <option name="TERNARY_OPERATION_WRAP" value="1" />
      <option name="TERNARY_OPERATION_SIGNS_ON_NEXT_LINE" value="true" />
      <option name="FOR_STATEMENT_WRAP" value="1" />
      <option name="ARRAY_INITIALIZER_WRAP" value="1" />
      <option name="IF_BRACE_FORCE" value="3" />
      <option name="DOWHILE_BRACE_FORCE" value="3" />
      <option name="WHILE_BRACE_FORCE" value="3" />
      <option name="FOR_BRACE_FORCE" value="3" />
      <option name="PARENT_SETTINGS_INSTALLED" value="true" />
      <indentOptions>
        <option name="INDENT_SIZE" value="2" />
        <option name="TAB_SIZE" value="2" />
      </indentOptions>
    </codeStyleSettings>
    <codeStyleSettings language="PROTO">
      <option name="RIGHT_MARGIN" value="80" />
      <indentOptions>
        <option name="INDENT_SIZE" value="2" />
        <option name="CONTINUATION_INDENT_SIZE" value="2" />
        <option name="TAB_SIZE" value="2" />
      </indentOptions>
    </codeStyleSettings>
    <codeStyleSettings language="protobuf">
      <option name="RIGHT_MARGIN" value="80" />
      <indentOptions>
        <option name="INDENT_SIZE" value="2" />
        <option name="CONTINUATION_INDENT_SIZE" value="2" />
        <option name="TAB_SIZE" value="2" />
      </indentOptions>
    </codeStyleSettings>
    <codeStyleSettings language="Python">
      <option name="KEEP_BLANK_LINES_IN_CODE" value="1" />
      <option name="RIGHT_MARGIN" value="80" />
      <option name="ALIGN_MULTILINE_PARAMETERS" value="false" />
      <option name="PARENT_SETTINGS_INSTALLED" value="true" />
      <indentOptions>
        <option name="INDENT_SIZE" value="2" />
        <option name="CONTINUATION_INDENT_SIZE" value="4" />
```

```
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
  </codeStyleSettings>
  <codeStyleSettings language="SASS">
    <indentOptions>
      <option name="CONTINUATION_INDENT_SIZE" value="4" />
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
  </codeStyleSettings>
  <codeStyleSettings language="SCSS">
    <indentOptions>
      <option name="CONTINUATION_INDENT_SIZE" value="4" />
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
  </codeStyleSettings>
  <codeStyleSettings language="TypeScript">
    <indentOptions>
      <option name="INDENT_SIZE" value="2" />
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
  </codeStyleSettings>
  <codeStyleSettings language="XML">
    <indentOptions>
      <option name="INDENT_SIZE" value="2" />
      <option name="CONTINUATION_INDENT_SIZE" value="2" />
      <option name="TAB_SIZE" value="2" />
    </indentOptions>
    <arrangement>
      <rules>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>xmlns:android</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>^$</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>xmlns:.*</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>^$</XML_NAMESPACE>
              </AND>
```

```
            </match>
            <order>BY_NAME</order>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:id</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>style</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>^$</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>^$</XML_NAMESPACE>
              </AND>
            </match>
            <order>BY_NAME</order>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:.*Style</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
```

```
            </match>
            <order>BY_NAME</order>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:layout_width</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:layout_height</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:layout_weight</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:layout_margin</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
```

```
            </AND>
          </match>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:layout_marginTop</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
            </AND>
          </match>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:layout_marginBottom</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
            </AND>
          </match>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:layout_marginStart</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
            </AND>
          </match>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:layout_marginEnd</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
```

```
            </AND>
          </match>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:layout_marginLeft</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
            </AND>
          </match>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:layout_marginRight</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
            </AND>
          </match>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:layout_.*</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
            </AND>
          </match>
          <order>BY_NAME</order>
        </rule>
      </section>
      <section>
        <rule>
          <match>
            <AND>
              <NAME>.*:padding</NAME>
              <XML_ATTRIBUTE />
              <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
```

```
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:paddingTop</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:paddingBottom</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:paddingStart</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:paddingEnd</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
```

```
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:paddingLeft</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*:paddingRight</NAME>
                <XML_ATTRIBUTE />
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*</NAME>
                <XML_NAMESPACE>http://schemas.android.com/apk/res/androi
d</XML_NAMESPACE>
              </AND>
            </match>
            <order>BY_NAME</order>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*</NAME>
                <XML_NAMESPACE>http://schemas.android.com/apk/res-auto</
XML_NAMESPACE>
```

```
              </AND>
            </match>
            <order>BY_NAME</order>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*</NAME>
                <XML_NAMESPACE>http://schemas.android.com/tools</XML_NAM
ESPACE>
              </AND>
            </match>
            <order>BY_NAME</order>
          </rule>
        </section>
        <section>
          <rule>
            <match>
              <AND>
                <NAME>.*</NAME>
                <XML_NAMESPACE>.*</XML_NAMESPACE>
              </AND>
            </match>
            <order>BY_NAME</order>
          </rule>
        </section>
      </rules>
    </arrangement>
  </codeStyleSettings>
  <Objective-C>
    <option name="INDENT_NAMESPACE_MEMBERS" value="0" />
    <option name="INDENT_C_STRUCT_MEMBERS" value="2" />
    <option name="INDENT_CLASS_MEMBERS" value="2" />
    <option name="INDENT_VISIBILITY_KEYWORDS" value="1" />
    <option name="INDENT_INSIDE_CODE_BLOCK" value="2" />
    <option name="KEEP_STRUCTURES_IN_ONE_LINE" value="true" />
    <option name="FUNCTION_PARAMETERS_WRAP" value="5" />
    <option name="FUNCTION_CALL_ARGUMENTS_WRAP" value="5" />
    <option name="TEMPLATE_CALL_ARGUMENTS_WRAP" value="5" />
    <option name="TEMPLATE_CALL_ARGUMENTS_ALIGN_MULTILINE" value="true"
/>
    <option name="ALIGN_INIT_LIST_IN_COLUMNS" value="false" />
    <option name="SPACE_BEFORE_SUPERCLASS_COLON" value="false" />
  </Objective-C>
  <Objective-C-extensions>
    <option name="GENERATE_INSTANCE_VARIABLES_FOR_PROPERTIES" value="ASK
```

```
" />
    <option name="RELEASE_STYLE" value="IVAR" />
    <option name="TYPE_QUALIFIERS_PLACEMENT" value="BEFORE" />
    <file>
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Import" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Macro" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Typedef" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Enum" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Constant" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Global" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Struct" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="FunctionPredecl" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Function" />
    </file>
    <class>
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Property" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="Synthesize" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="InitMethod" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="StaticMethod" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="InstanceMethod" />
      <option name="com.jetbrains.cidr.lang.util.OCDeclarationKind" valu
e="DeallocMethod" />
    </class>
    <extensions>
      <pair source="cc" header="h" />
      <pair source="c" header="h" />
    </extensions>
  </Objective-C-extensions>
  <codeStyleSettings language="ObjectiveC">
    <option name="RIGHT_MARGIN" value="80" />
    <option name="KEEP_BLANK_LINES_BEFORE_RBRACE" value="1" />
    <option name="BLANK_LINES_BEFORE_IMPORTS" value="0" />
    <option name="BLANK_LINES_AFTER_IMPORTS" value="0" />
    <option name="BLANK_LINES_AROUND_CLASS" value="0" />
```

```
      <option name="BLANK_LINES_AROUND_METHOD" value="0" />
      <option name="BLANK_LINES_AROUND_METHOD_IN_INTERFACE" value="0" />
      <option name="ALIGN_MULTILINE_BINARY_OPERATION" value="false" />
      <option name="BINARY_OPERATION_SIGN_ON_NEXT_LINE" value="true" />
      <option name="FOR_STATEMENT_WRAP" value="1" />
      <option name="ASSIGNMENT_WRAP" value="1" />
      <indentOptions>
        <option name="INDENT_SIZE" value="2" />
        <option name="CONTINUATION_INDENT_SIZE" value="4" />
      </indentOptions>
    </codeStyleSettings>
  </code_scheme>
```

插件安装

Save Actions

如IDEA的版本为较高版本，可先行到settings查看是否支持以下功能，如支持，则无须安装 Save Actions，直接使用原生功能即可。

否则，安装Save Actions插件，重启后勾选以下红框内的选项，在保存代码的时候自动格式化与多余import去除，其他选项根据需求 自行决定是否勾选。
注意，选项是针对项目的，所以需要每个项目都勾选上

SonarLint

代码规范与质量检测插件，与阿里代码规范检查器相互补充，帮助写出高质量代码。提交代码之前对项目目录进行操作：右键-> SonarLinit -> Analyze with SonarLint，也可以使用其指定快捷键进行操作。根据扫描出来的问题与建议进行修改后提交。扫描结果如下所示：

注：使用此插件进行扫描修复后，阿里的大部分问题会一并修复

Alibaba Java Coding Guidelines

阿里代码规范检查器，提交代码之前对项目目录进行操作：右键-> 编码规约扫描，也可以使用其指定快捷键进行操作。根据扫描出来的问题与建议进行修改后提交。扫描结果如下所示：

说明：
Blocker: 即系统无法执行、崩溃或严重资源不足、应用模块无法启动或异常退出、无法测试、造成系统不稳定。
严重花屏、内存泄漏、用户数据丢失或破坏、系统崩溃/死机/冻结、模块无法启动或异常退出、严重的数值计算错误、功能设计与需求严重不符、其它导致无法测试的错误, 如服务器500错误

Critical：即影响系统功能或操作，主要功能存在严重缺陷，但不会影响到系统稳定性。
功能未实现、功能错误、系统刷新错误、数据通讯错误、轻微的数值计算错误、影响功能及界面的错误字或拼写错误、安全性问题

Major：即界面、性能缺陷、兼容性。
操作界面错误（包括数据窗口内列名定义、含义是否一致）、边界条件下错误、提示信息错误（包括未给出信息、信息提示错误等）、长时间操作无进度提示、系统未优化（性能问题）、光标跳转设置不好，鼠标（光标）定位错误、兼容性问题

Minor/Trivial:即易用性及建议性问题。
界面格式等不规范、辅助说明描述不清楚、操作时未给用户提示、可输入区域和只读区域没有明显的区分标志、个别不影响产品理解的错别字、文字排列不整齐等一些小问题

Key Promoter X

IDEA快捷键提示，不强制安装，只是帮助大家更加方便快捷完成编码。

低代码平台

代码生成器（beta版）

代码生成器（beta版）

一、导入数据表

1. 登录至pai开发平台 -> 套件 -> 数据库管理 -> 生成表结构 -> 勾选需要生成的数据表（注：只需要表结构，不需要数据）
2. 访问代码生成器 -> 导入 -> 选择pai开发平台导出的数据库文件 -> 上传 -> 完成导入
3. 数据表增减字段，改动过多时，可删除原表进行重新导入；改动较小时，为保持原有配置，可自己编写alter table xxx等语句进行导入，之后便可以进行数据表同步

二、数据属性说明

2.1 基本信息

序号	属性	说明
1	表名称	po对象@ TableName注解
2	表描述	数据表描述
3	实体类名称	po、vo、service、controller、mapper等名称前缀
4	作者	此处填写自己的码名
5	备注	备注

2.2 字段信息

序号	属性	说明	备注
1	字段列名	对应的数据库字段名称	
2	字段描述	vo对象@ ApiModelProperty、I18N文件对应值	
3	物理类型	对应的数据库字段类型	vo对象会根据此属性进行长度限制操作

4	Java类型	对应的Java类型	Integer、Long、String、Date、BigDecimal
5	Java属性	vo、po、mapper.xml等生成属性名称	
6	插入	勾上则会在新增显示	
7	编辑	勾上则会在新增页面显示	增、改目的使用add.vue，查看使用view.vue。 ~~后续增加只读查看页面~~
8	列表	勾上则会在列表字段显示	
9	查询	勾上则会在列表查询处显示	
10	查询方式	查询字段对应的查询方式	目前仅生成=、Between，后续再决定是否放开其他
11	必填	勾上则在vo以及前端添加必填校验	
12	显示类型	即控件类型	文本框、文本域、下拉框、单选框、复选框、流水号、富文本、上传控件、日期控件、企业选择器、园区选择器、部门选择器 、人员选择器。目前前端并未完全标准化
13	字典类型	当显示类型为单行框、复选框及下拉框时，需填写对应的数据字典标识；同时如果字典在其他套件，需要对应修改控件及ServiceImpl.setup方法中对应的套件名称	

2.3 生成信息

序号	属性	说明
1	生成模板	支持单表、主子表（增删改查）
2	生成包路径	后端生成文件的包名前缀，如租赁意向：cn.flyrise.pai.business.lease.intention。如存在主子表关系，子表使用主表设置的包路径
3	生成模块名	对应套件后半部分，如cn.flyrise.business，则此处只需要填写business即可
4	生成业务	对应业务名称，如租赁则为：lease，租赁意向则为：lease.intention。子表

4	名	只需填写自身业务即可，如房间子表：room
5	生成功能名	Java类注释、swagger显示模块名称
6	validCode起始值	中台建议由3000起步，请根据项目已使用的code进行赋值
7	主子表关联信息	
	关联子表	选择相应的子表
	子表关联的外键名	选择子表关联主表外键

三、使用方法

1. 点击生成代码，下载一个以zip结尾的压缩包，进行解压缩
2. main、test文件夹为后端工程专用，复制到java后端工程src下
3. feign文件夹（生成条件查看特性 `4.1 name字段` 说明），将其下的文件夹复制至对应业务内部调用库的src下
4. web文件夹为前端工程专用，复制其下的api及pages文件夹至前端工程根目录
5. 启动后端
6. 前端部分控件需要安装依赖，请根据 https://pai.flyrise.cn/pai-sp-ui/#/docs/Install 指引进行安装
7. 前端修改nuxt.config.js，找到 LOCAL_PROXY_OPTIONS，进行修改，目的是让前端工程访问自身后端：

   i. 修改前

```
const LOCAL_PROXY_OPTIONS = {
  enable: false, // 是否启用
  target: 'http://127.0.0.1:8080',
  proxies: [
    // {
    //   path: 'template-api', // 替换成业务后端api
    // },
    // {
    //   path: 'test-api',
    //   target: 'http://127.0.0.1:8888'
    // }
  ]
```

```
}
```

修改后

```
const LOCAL_PROXY_OPTIONS = {
  enable: true, // 是否启用
  target: 'http://10.62.19.xx:8080',  //后端地址
  proxies: [
    {
      path: 'business-api', // 替换成业务后端api
    },
    // {
    //   path: 'test-api',
    //   target: 'http://127.0.0.1:8888'
    // }
  ]
}
```

8. 启动前端

四、特性

4.1 字段

4.1.1 name

当单表或主表包含`name`字段时，会将此模块当成字典类型处理，额外添加以下功能：

1. service添加以下方法

```
    /**
     * 获取对象名称
     *
     * @param id 主键
     * @return 名称
     */
    String getName(String id);

    /**
     * 获取对象id、名称集合
     *
     * @param ids 主键集合
     * @return 主键、名称集合
```

```
     */
    Map<String, String> getNames(Collection<String> ids);

    /**
     * 获取对象id、名称集合。用于下拉选择器
     *
     * @param query 查询对象
     * @return 主键、名称集合
     */
    Page<VO> selector(QueryVO query);
```

2. controller添加如下方法：

```
    @ApiOperation("分页查询主键、名称集合")
    ~~@PostMapping("selector")~~
    @PostMapping("dict")
    public Reply<Page<VO>> dict(@Valid @RequestBody QueryVO query) {
      return Reply.success(this.${className}Service.selector(query));
    }
```

3. 额外添加内部Controller类，用于依赖模块的保存验证等

```
@RestController
@RequestMapping("/${api.version}/xxx/inner")
public class InnerController {

  @Resource
  private IService Service;

  @ApiOperation("获取名称")
  @GetMapping("name/{id}")
  @ApiResponses({
    @ApiResponse(code = 1006, message = "id不能为空"),
  })
  @ApiImplicitParams({
    @ApiImplicitParam(name = "${table.getPkName()}" , value = "${fun
ctionName}主键", dataTypeClass = String.class, required = true)
  })
  @Inner
  public Reply<String> getName(@PathVariable("id") String id) {
    return Reply.success(this.${className}Service.getName(id));
  }

  @ApiOperation("获取主键、名称集合")
  @PostMapping("names")
  @Inner
  public Reply<Map<String, String>> getNames(@RequestBody Collection
```

```
<String> ids) {
    return Reply.success(this.${className}Service.getNames(ids));
  }

}
```

4. 额外添加feign调用类：

```
@FeignClient(
        name = "pai-xxx",
        contextId = "xxxApi",
        path = "${api.service.xxx.path:}/${api.service.xxx.version:v
1}/xxx/inner",
        fallbackFactory = RemoteFallbackFactory.class
)
public interface IService {

  @ApiOperation("获取租赁意向名称")
  @GetMapping("name/{id}")
  Reply<String> getName(@PathVariable("id") String id, @RequestHeade
r(FROM) String from);

  @ApiOperation("获取租赁意向主键、名称集合")
  @PostMapping("name")
  Reply<Map<String, String>> getNames(@RequestBody Collection<String
> ids,
          @RequestHeader(FROM) String from);

}
```

```
public class RemoteLeaseFactory implements FallbackFactory<IService>
 {

  @Override
  public IService create(Throwable throwable) {
    return new IService() {
      @Override
      public Reply<String> getName(String id, String from) {
        return fail();
      }

      @Override
      public Reply<Map<String, String>> getNames(Collection<String>
ids, String from) {
        return fail();
      }
    };
```

```
    }

    private <T> Reply<T> fail() {
      return Reply.fail("405" , "租赁意向服务异常，请稍后重试");
    }
  }
```

4.1.2 字段名称对应默认生成控件

1. enterprise_id、consumer_id ：企业选择器。规范推荐使用enterprise_id，更能体现是由企业档案而来
2. park_id：园区选择器
3. dept_id：部门选择器
4. create_by、update_by：人员选择器
5. annex、或以file结尾：上传控件
6. room_id：房源选择器
7. 以_id结尾：下拉框

五、注意事项

1. java代码import部分会引入未使用的类，请格式化所有java代码

2. vo字段验证：长度验证是根据数据字段类型进行判断，如status等表示状态的，1-10范围内基本已满足，但生成器目前尚未做此判断，所以需要自行进行调整:

```
    @ApiModelProperty(value = "状态", hidden = true)
  @Min(0)
  @Max(999999999)
  @InvalidCode(value = "3022", desc = "状态", message = "{status}")
  private Long status;
```

3. 企业、园区、部门、员工、字典等参照项显示值，如需持久化到数据表中，需要在ServiceImpl.setup方法中自行赋值：

```
private LeaseIntentionPO setup(LeaseIntentionVO vo) {

}
```

4. 子表保存验证：不同于主表验证，其是批量形式，目前还没能找齐获取集合进行验证的接口，暂时需要自行编写验证代码。在子表ServiceImpl类中搜索TODO查找相应位置，如下：

```
        List<LeaseIntentionRoomPO> items = list.stream().map(
        item -> {
          // TODO 子表验证保存
          LeaseIntentionRoomPO po = ConvertUtil.transfor(item, Lease
```

```
IntentionRoomPO.class);
            po.setIntentionId(mainId);
            return po;
          }).collect(toList());
      this.saveOrUpdateBatch(items);
      return ConvertUtil.transforList(items, LeaseIntentionRoomVO.clas
s);
```

5. 前端验证：接口访问失败处理、提交失败处理、表单验证未通过处理、子表操作等个性化处理需自行处理。搜索TODO查找相应位置

```
this.$refs.ruleForm.validate(valid => {
  if (valid) {
    this.fullLoading = true
    let apiMethod = this.isAdd ? LeaseIntentionApi.add : LeaseIntent
ionApi.update;
    apiMethod(this.form).then((res) => {
      this.fullLoading = false
      if (res.success) {
        this.back()
      }
    })
      .catch((errorRes) => {
      this.fullLoading = false
      // TODO 提交失败处理
    });
  } else {
    // TODO 表单验证未通过处理
  }
})
```

6. 主子表生成代码方式：无需勾选子表，只对主表进行生成代码操作即可，生成器会根据主表的关联关系自动生成子表对应的代码

7. i18n文件

   i. 为避免覆盖原messages.properties文件，将生成{业务名}.messages.properties文件，复制内容至messages.properties文件即可
   ii. 不同的表难免存在同名称的列，需要手动确认

8. 问题：目前前端控件标准化程度不完整，如遇到控件不存在而报错的，请在部门群内询问解决

此版本为**beta**版，并非稳定版本，请做好心理预期。使用过程中有任何问题、任何改进意见都可以进行讨论，让我们一起完善此工具，减少重复性工作，专注于业务开发。

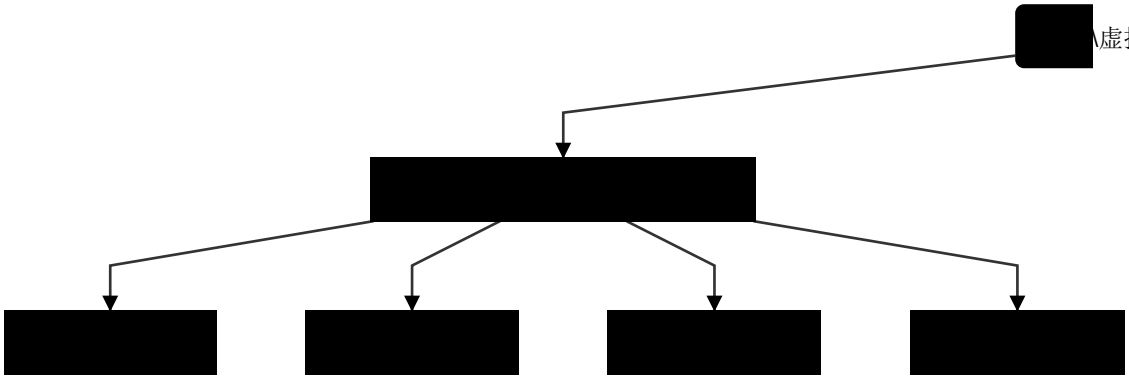
后端开发规范

必备字段及租户配置

数据表必备字段及租户配置

> 用于集团版、多园区等门户、数据报表等数据多维度过滤展现

架构



说明

1. tenant_id：租户ID, 由中台mybatis plus 插件维护。当处在MQ、多线程、XXL调试任务等无法获取当前用户的环境需要业务自己维护，以上环境需要传递此字段值，且需要进行取消租户条件进行数据库操作，否则会出现 `tenant_id=null` 条件。取消方法参考多租户说明
2. ent_id：运营公司ID，用于多运营公司数据隔离，来源IBaseController.getUser()中用户的entId，也可由前端回传
3. park_id：园区ID，用于多园区数据隔离
4. dept_id：部门ID，用于多部门数据隔离
5. create_by：创建人ID
6. create_time：创建时间
7. update_by：修改人ID
8. update_time：修改时间（用于数据变更对比）
9. is_deleted：是否已删除（用于数据变更对比）

注：tenant_id、ent_id、park_id、create_by、create_time、update_by、update_time、is_deleted为数据表必备字段，dept_id可根据业务实际需求确定

租户配置

```yaml
pai:
  tenant:
    # 全局开关，默认false 不启用
    enabled: true
    # 租户字段，默认tenant_id
    column: tenant_id
    # 全局租户策略，默认v1（当前企业），v2（运营企业）
    policy: v2
```

注：强制要求全局策略 `policy` 为 `v2`

IBaseController

IBaseController

目的

> 避免在业务service代码中使用 `SecurityUtils.getLoginUser()` 获取当前登录用户，以隔离业务与登录用户间的耦合。
>
> 好处：当业务需求的用户信息不是当前用户时，则可以通过new User()的方法自定义一个用户进行业务相关操作。

IBaseController源码：

```
public interface IBaseController {

  /**
   * 用以获取用户，以隔离业务与登录用户的耦合
   *
   * @return 当前用户
   */
  @ApiModelProperty("用于获取用户，以隔离业务与登录用户的耦合")
  default User getUser() {
    LoginUser loginUser = SecurityUtils.getLoginUser();
    return null == loginUser ? null : new User(loginUser);
  }
}
```

使用方法

1. 引入通用类库

```
    <dependency>
      <groupId>cn.flyrise</groupId>
      <artifactId>pai-fe-common</artifactId>
    </dependency>
```

2. 业务controller继承IBaseController接口，通过getUser()方法获取当前用户

```
public class XXXController implements IBaseController {
  @ApiOperation("添加单条记录")
```

```
    @PostMapping("add")
    public Reply<LeaseIntentionVO> insert(
        @Validated({Create.class}) @RequestBody LeaseIntentionVO vo) {
      return Reply.success(this.leaseIntentionService.insert(getUser()
, vo));
    }
}
```

3. 业务定义接收User对象的方法

```
    /**
     * 保存租赁意向信息
     *
     * @param user 当前操作用户
     * @param vo   租赁意向信息
     * @return 保存后的租赁意向信息
     */
    LeaseIntentionVO insert(User user, LeaseIntentionVO vo);
```

不存在当前用户时，getUser()同样会返回null对象，业务代码中需自行判断处理

业务通用类库公共POM

业务通用类库公共**POM**

> 业务中使用的所有类库，包括工具包、feign调用库的parent pom，统一版本、统一工程

示例

实现示例：

1. IDEA -> File -> New -> Module… - Maven
2. 顶级pom.xml加入新建的module，填充modules、dependencyManagement-dependency,如：pai-fe-common

```
<project>
  <modules>
    <module>pai-fe-common</module>
  </modules>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>cn.flyrise</groupId>
        <artifactId>pai-fe-common</artifactId>
        <version>${pai.fe.version}</version>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

3. 注意各子模块pom中的parent.version统一与父pom的保持版本一致，不允许出现版本不一致的情况
4. 如果只是测试某个子模块，则只需要针对此模块进行deploy即可，不要在pai-fe-bom中deploy，这是把所有模块都重新发布
5. SNAPSHOT版本无须频繁更新版本号，勾上IDEA->settings-> Build, Execution, Deployment -> Build Tools -> Maven ->
   Always update snapshots即可. 所有工程都需勾选

使用示例：

1. 业务pom中引入公共pom文件

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>cn.flyrise</groupId>
        <artifactId>pai-fe-bom</artifactId>
        <version>1.0.0-SNAPSHOT</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <!--业务dependencies-->
  <dependencies>
    ...
  </dependencies>
</project>
```

2. 引用对应的依赖，无须写版本号，会自动引用dependencyManagement中对应的版本

```
<!--业务dependencies-->
<dependencies>
  <dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-fe-common</artifactId>
  </dependency>
</dependencies>
```

类库说明

一、通用类库

1. pai-fe-common：通用类库
2. pai-fe-common-excel：excel工具类库
3. pai-fe-common-routine：常规通用类库

二、**feign**调用库

1. pai-fe-common-enterprise：企业档案
2. pai-fe-common-business：智慧招商
3. pai-fe-common-space：空间资源

4. pai-fe-common-contract：合同中心
5. pai-fe-common-finance：财务中心
6. pai-fe-common-property：智慧物业
7. pai-fe-common-voucher：凭证管理
8. pai-fe-common-service-center：服务中心
9. pai-fe-common-monthcard：智慧车行
10. pai-fe-common-walking：智慧人行
11. pai-fe-common-work-engine 工单中心

Mockito单元测试

Mockito单元测试

一、文档参考

官方文档
中文翻译

或自行查找

通用类库

通用类库

> 抽取产品开发过程中具有代表性、以及重复性的功能，集成到通用类库中供大家使用，提高开发效率
>
> pai平台已经集成了hutool工具类库，可先了解其包含的功能，避免重复造轮子

```
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-fe-common</artifactId>
</dependency>
```

常量类

常量类

1. CommonConstants 通用

- ENABLE : 启用状态（1: int）
- DISABLE：禁用状态（0: int）
- YES：是（1 : String）
- NO：否（0:String）
- YES_INT：是（1:int）
- NO_INT：否（0:int）
- PARK：园区权限（P：String）
- DEPARTMENT：部门权限（D：String）
- USER：用户权限（U：String）

2. FieldConstants 数据库字段

- DEPT_ID：部门ID（dept_id ：String）
- DEPT_NAME：部门名称（”dept_name” ：String）
- PARK_ID：园区ID（”park_id”：String）
- PARK_NAME：园区名称（ “park_name” ：String）
- ENTERPRISE_ID：企业档案ID（”enterprise_id” ：String）
- ENTERPRISE_NAME：企业档案名称 （ “enterprise_name” ：String）
- TENANT_ID：租户ID（ “tenant_id” ：String）
- ID：主键 （ “id” ：String）
- NAME：名称（”name”
- SORTING：排序 （”sorting”：String）
- STATUS：状态 （”status”：String）
- IS_DELETED：是否已删除 （”is_deleted” ：String）
- CREATE_BY： 创建人 （”create_by” ：String）
- CREATE_TIME：创建时间 （”create_time” ：String）
- UPDATE_BY：修改人 （”update_by” ：String）
- UPDATE_TIME：修改时间 （”update_time” ：String）
- PHONE 联系电话 （”phone” ：String）
- ID_LENGTH：ID长度 （ 32， int） ：String）

3. DateConstants 日期常量类

- YEAR：年（year：String）
- MONTH：月（month：String）
- SECTION：区间（section：String）
- YEAR_START：年份有效起始值（1900：int）

- YEAR_END：年份有效终止值（9999: int）
- MONTH_START：月份有效起始值（1：int）
- MONTH_END：月份有效终止值（12：int）

4. SuitConstants 套件常量类

- ENTERPRISE：企业档案套件标识
- BUSINESS：智慧招商套件标识
- BUSINESS_BASE：招商基础套件标识
- BUSINESS_LEASE：招商租赁业务套件标识
- BUSINESS_SALE：招商销售业务套件标识
- BUSINESS_LAND：招商土地业务套件标识
- BUSINESS_ENTERPRISE：招商客户管理套件标识
- BUSINESS_PLAN：招商计划管理套件标识
- PERFORMANCE_AGREEMENT：履约管理套件标识
- BUSINESS_SPACE：招商业务空间

5. MqConstants 消息订阅常量类

- LEASE_INTENTION：招商租赁意向TAG
- LEASE_SUBSCRIBE：招商租赁协议TAG
- LEASE_CONTRACT：招商租赁合同TAG
- PURCHASE_INTENTION：招商销售意向TAG
- PURCHASE_SUBSCRIBE：招商认购协议TAG
- PURCHASE_CONTRACT：招商销售合同TAG
- LAND_INTENTION：招商购地意向TAG
- LAND_CONTRACT：招商购地合同TAG
- SETTLE：招商入驻业务TAG
- NEW：通用业务细分-新增
- UPDATE：通用业务细分：修改
- DELETE：通用业务细分：删除
- FLOW：通用业务细分：流程

异常类

异常类

> 1. 大部分验证可以通过VO中添加@NotBlank、@Length、@Size等注解进行框架级的自动验证。但难免会存在根据业务特定条件进行数据属性判断的情况，这时就需要我们在代码中自行进行判断
> 2. 因I18N的原因，如使用通用异常类，需复制此jar包/resource/i18n.properties文件内容至使用工程对应文件

1. CommonErrors 通用错误枚举

```
  DUPLICATE_KEY("1001", "数据已存在:{}", "exist"),
  NO_EXISTS_KEY("1002", "数据不存在:{}", "no_exist"),
  DUPLICATE_NAME("1003", "名称已存在:{}", "name_exist"),
  RANGE("1004", "{}输入范围限定在{}到{}之间", "range_limit"),
  EMPTY("1006", "{}不能为空", "not_empty"),
  LENGTH("1007", "{}长度必须是{min}-{max}个字符", "length_limit_param"),
  NOT_EXIST_QUARTER("3001", "不存在的季度：{}", "not_exist_quarter"),
  NOT_EXIST_MONTH("3002", "不存在的月份：{}", "not_exist_month"),
  NOT_EXIST_DATE_RANGE("3003", "不存在的时间区间：{}", "not_exist_date_rang
e"),
  FEIGN_INVOKE("4001", "服务器繁忙，请稍候再试！{}", "feign_invoke"),
  SYS_BUSY("4002", "系统繁忙，请稍候再试", "sys_busy")
```

2. CommonException 通用exception类

```
  public CommonException(CommonErrors commonErrors, Object... params) {
    super(commonErrors.getCode(), commonErrors.format(params));
  }

  public CommonException(String code, String msg) {
    super(code, msg);
  }
```

3. DuplicateException 数据重复异常类

```
public DuplicateException(Object... params) {
  super(DUPLICATE_KEY.getCode(), DUPLICATE_KEY.format(params));
}
```

4. DuplicateNameException 名称重复异常类

```
public DuplicateNameException(Object... params) {
  super(DUPLICATE_NAME.getCode(), DUPLICATE_NAME.format(params));
}
```

5. FeignInvokeException feign内部调用异常类

```
public FeignInvokeException(Object... params) {
  super(FEIGN_INVOKE.getCode(), FEIGN_INVOKE.format(params));
}
```

6.NotExistException 数据不存在异常类

```
public NotExistException(Object... params) {
  super(NO_EXISTS_KEY.getCode(), NO_EXISTS_KEY.format(params));
}
```

7. OutOfLengthException 超出长度异常类

```
public OutOfLengthException(Object... params) {
  super(LENGTH.getCode(), LENGTH.format(params));
}
```

8. OutOfRangeException 超出范围异常类

```
public OutOfRangeException(Object... params) {
  super(RANGE.getCode(), RANGE.format(params));
}
```

9. RequiredException 必填属性异常类

```
public RequiredException(Object... params) {
  super(EMPTY.getCode(), EMPTY.format(params));
}
```

使用示例

```
if (CharSequenceUtil.isBlank(id)) {
  throw new RequiredException(ID);
```

```
}
```

实体类

通用实体类

1. 通用列表查询类

> 内置了园区ID（park_id）、页码（page）、行数（size），业务实际查询类可继承它来进行简化

```
public class CommonQueryVO implements Serializable {

  private static final long serialVersionUID = 6990794642352554429L;

  @ApiModelProperty("园区ID")
  @Length(max = ID_LENGTH)
  @InvalidCode(value = "1203", message = "{park}")
  private String parkId;

  @ApiModelProperty(value = "页码", required = true)
  @NotNull
  @Range(min = 1, max = 999)
  @InvalidCode(value = "1201", message = "{page}")
  private Integer page;

  @ApiModelProperty(value = "行数", required = true)
  @NotNull
  @Range(min = 1, max = 999)
  @InvalidCode(value = "1202", message = "{size}")
  private Integer size;
}
```

2. 通用业务用户实体类

> 配合IBaseController，用以获取用户，以隔离业务与登录用户的耦合：IBaseController

```
@Api("业务通用用户类")
public class User implements Serializable {

  private static final long serialVersionUID = 5664504431278276 54L;

  public User() {
  }
```

实体类

```
    /**
     * 使用当前用户初始化业务用户
     *
     * @param loginUser 登录用户
     */
    public User(LoginUser loginUser) {
      BeanUtil.copyProperties(loginUser, this);
    }

    @ApiModelProperty("用户ID")
    private String userId;
    @ApiModelProperty("用户名称")
    private String nickName;
    @ApiModelProperty("部门ID")
    private String deptId;
    @ApiModelProperty("运营企业ID/租户ID")
    private String entId;
    private String platform;
    private String openId;
    @ApiModelProperty("园区ID")
    private String parkId;
    @ApiModelProperty("园区编号")
    private String parkCode;
    @ApiModelProperty("园区名称")
    private String parkName;
    @ApiModelProperty("部门名称")
    private String deptName;
    @ApiModelProperty("运营企业名称")
    private String entName;
    @ApiModelProperty("员工ID")
    private String staffId;
    @ApiModelProperty("员工名称")
    private String staffName;
    @ApiModelProperty("手机号码")
    private String phoneNumber;
    @ApiModelProperty("版本")
    private String version;
    @ApiModelProperty("头像")
    private String avatar;
}
```

###

数据权限过滤

通用数据权限过滤工具

1. 通用数据权限实体类

> 配合IPermissionCommonService通用权限接口进行使用，目的是获取传入用户数据权限进行过滤数据

```
public class PermissionVO implements Serializable {

  @ApiModelProperty("园区权限")
  private Set<String> parks;

  @ApiModelProperty("部门权限")
  private Set<String> departments;

  @ApiModelProperty("用户权限")
  private Set<String> users;
}
```

2. 通用数据权限接口类

```
@Api("通用权限接口")
public interface IPermissionCommonService {

  /**
   * 获取指定用户所具备的数据权限集合
   *
   * @param user       进行权限过滤的用户
   * @param suiteCode 套件标识
   * @return 数据权限集合
   */
  @ApiModelProperty("获取指定用户所具备的数据权限集合")
  Map<String, Set<String>> getPermissions(User user, String suiteCode);

  /**
   * 获取指定用户所具备的数据权限集合
   *
   * @param user       进行权限过滤的用户
   * @param suiteCode 套件标识
   * @return 数据权限集合
   */
```

```
    @ApiModelProperty("获取指定用户所具备的数据权限集合")
    PermissionVO getPermission(User user, String suiteCode);
}
```

3. 通用数据权限组合工具类

```
public class QueryUtil {
    /**
     * 通用权限查询对象获取，根据权限集合设置数据查询权限，部门 or 园区 or  个人
     *
     * @param permissions 权限集合
     * @param wrapper     wrapper
     */
    public static <T> void setPermission(Map<String, Set<String>> permissi
ons,
        QueryWrapper<T> wrapper) {
      setPermission(permissions, wrapper, PARK_ID, DEPT_ID, CREATE_BY);
    }

    /**
     * 通用权限查询对象获取，根据权限集合设置数据查询权限，部门 or 园区 or  个人
     *
     * @param permissions 权限集合
     * @param wrapper     查询wrapper
     * @param parkField   园区字段
     * @param deptField   部门字段
     * @param userField   用户字段
     */
    public static <T> void setPermission(Map<String, Set<String>> permissi
ons,
        QueryWrapper<T> wrapper, String parkField, String deptField, Strin
g userField) {
      wrapper.and(w -> {
        if (permissions.containsKey(PARK)) {
          w.or().in(parkField, permissions.get(PARK));
        }
        if (permissions.containsKey(DEPARTMENT)) {
          w.or().in(deptField, permissions.get(DEPARTMENT));
        }
        if (permissions.containsKey(USER)) {
          w.or().in(userField, permissions.get(USER));
        }
      });
      return wrapper;
    }
```

使用示例

1. 开发者-指定套件-权限管理-配置数据权限

   > 招商业务使用招商基础设置（cn.flyrise.business.base）套件中的配置，业务可自行决定是否遵从

2. 开发沙箱配置相应人员、角色

3. 代码示例

   业务类使用

   > 以传入user为例：
   >
   > i. 当user存在园区数据权限时，将获取园区所有数据：park_id in (?,?)
   > ii. 当user存在部门数据权限时，将获取部门所有数据：dept_id in (?,?)
   > iii. 以及获取个人添加数据：create_by in (?,?)

```
@Resource
private IPermissionCommonService permissionCommonService;

public Page<LeaseIntentionVO> page(User user, QueryVO vo) {
  QueryWrapper<LeaseIntentionPO> wrapper = getWrapper(vo);
  Map<String, Set<String>> permissions = permissionCommonService.get
Permissions(user, BUSINESS_BASE);
  QueryUtil.setPermission(permissions, wrapper);
  wrapper.orderByDesc(ID);
  Page<LeaseIntentionPO> page = baseMapper
    .selectPage(new Page<>(vo.getPage(), vo.getSize()), wrapper);
  Page<LeaseIntentionVO> result = CastUtils.cast(page);
  result.setRecords(convertRecords(user, page.getRecords()));
  return result;
}
```

工具类

通用工具类

1. 时间工具类

1. 月份枚举

```
package cn.flyrise.common.util.date;

public enum MonthEnum {
  JAN(1, "1月"),
  FEB(2, "2月"),
  MAR(3, "3月"),
  APR(4, "4月"),
  MAY(5, "5月"),
  JUN(6, "6月"),
  JUL(7, "7月"),
  AUG(8, "8月"),
  SEP(9, "9月"),
  OCT(10, "10月"),
  NOV(11, "11月"),
  DEC(12, "12月");
}
```

2. 季度枚举

```
package cn.flyrise.common.util.date;

public enum QuarterEnum {
  Q1(1, "第一季度", "01-01"),
  Q2(2, "第二季度", "04-01"),
  Q3(3, "第三季度", "07-01"),
  Q4(4, "第四季度", "10-01");
}
```

3. 时间区间实体类

```
package cn.flyrise.common.util.date;

@Api("时间区间实体类")
public class DateRange {
```

```
    @ApiModelProperty("起始时间")
    private Date begin;

    @ApiModelProperty("结束时间")
    private Date end;
}
```

4. 时间区间区域获取接口

```
package cn.flyrise.common.util.date;

@Api("时间区间区域接口")
public interface IDateRange {

    /**
     * 获取当前时间对应的时间区间
     *
     * @return 时间区间
     */
    DateRange get();

    /**
     * 获取指定时间的区间
     *
     * @param date 指定时间
     * @return 时间区间
     */
    DateRange get(Date date);
}
```

实现类：

```
package cn.flyrise.common.util.date.impl;
```

i. 年度区间（YearRange）：获取当前时间或指定时间的年度起始时间、终止时间

ii. 季度区间（QuarterRange）：获取当前时间或指定时间的季度起始时间、终止时间

iii. 月度区间（MonthRange）：获取当前时间或指定时间的月份起始时间、终止时间

iv. 周区间（WeekRange）：获取当前时间或指定时间的星期起始时间、终止时间

v. 上半年区间（FirstHalfOfYearRange）：获取当前时间或指定时间的上半年起始时间、终止时间

vi. 下半年区间（SecondHalfOfYearRange）：获取当前时间或指定时间的下半年起始时间、终止时间

5. 时间区间枚举

```
public enum DateRangeTypeEnum {
  /**
   * 年度
   */
  YEAR(1),
  /**
   * 季度
   */
  QUARTER(2),
  /**
   * 月
   */
  MONTH(3),
  /**
   * 周
   */
  WEEK(4),
  /**
   * 上半年
   */
  FIRST_HALF_OF_YEAR(5),
  /**
   * 下半年
   */
  SECOND_HALF_OF_YEAR(6);
}
```

6. 时间区间工具类

```
package cn.flyrise.common.util.date;

public class DateRangeUtil {
      /**
   * 获取当前时间内对应类型的时间区间
   *
   * @param type 时间区间枚举
   * @return 时间区间
   */
  public static DateRange get(DateRangeTypeEnum type) {
    return get(type, DateUtil.date());
  }

  /**
   * 获取指定时间内对应类型的时间区间
   *
```

```
     * @param type 时间区间枚举
     * @param date 年份
     * @return 时间区间
     */
    public static DateRange get(DateRangeTypeEnum type, Date date) {
      DateRange range = DATE_RANGE_MAP.get(type).get(date);
      //据中台的说法数据库会自动将 23:59:59 四舍五入 到第二天 00:00:00，此处-999进行修正
      range.setEnd(DateUtil.date(range.getEnd().getTime() - 999));
      return range;
    }
  }
```

使用示例，查询某时间区间段的数量：

```
  public int count(int dateType) {
    DateRange range = DateRangeUtil.get(DateRangeTypeEnum.of(dateType));
    int cnt;
    if (null != range) {
      cnt = baseMapper.count(range.getBegin(), range.getEnd());
    } else {
      throw new OutOfRangeException("dateType", "0", "6");
    }
    return cnt;
  }
```

2. 正则表达式常量

> cn.hutool.core.lang.PatternPool中已存在部分正则表达式，无谓重复造轮子

```
package cn.flyrise.common.util.regular;

public class RegExpression {

  private RegExpression() {
  }

  /**
   * PatternPool.MOBILE 手机号正则表达式，用于VO验证
   */
  public static final String MOBILE = PatternPool.MOBILE.pattern();
}
```

3. 通用线程池

```
package cn.flyrise.common.util.thread;

public class CommonThreadPool {
    // 网上推荐线程池相关，可根据实际情况调优
  private static final int CORE_SIZE = Runtime.getRuntime().availablePro
cessors() + 1;
  private static final int MAX_SIZE = Runtime.getRuntime().availableProc
essors() << 1;
  private static final int QUEUE_SIZE = Runtime.getRuntime().availablePr
ocessors() << 5;
  private static final int KEEP_ALIVE_TIME = 1;

  /**
   * 创建线程池
   *
   * @param prefix 线程名称前缀
   * @return 线程池
   */
  public static ThreadPoolExecutor newThreadPoolExecutor(String prefix)
{
    return new ThreadPoolExecutor(CORE_SIZE, MAX_SIZE, KEEP_ALIVE_TIME,
TimeUnit.MINUTES,
        new ArrayBlockingQueue<>(QUEUE_SIZE), new NamedThreadFactory(pre
fix, false));
  }
}
```

使用示例：

1. 创建 `ThreadPoolUtil` 类，自定义线程名称前缀

```
public class ThreadPoolUtil {

  private static final String THREAD_NAME_PREFIX = "BUSINESS-LEASE-"
;
  private static final ThreadPoolExecutor EXECUTOR = CommonThreadPoo
l
      .newThreadPoolExecutor(THREAD_NAME_PREFIX);

  /**
   * 执行线程任务
   *
   * @param command 任务
   */
  public static void execute(Runnable command) {
    EXECUTOR.execute(command);
  }
```

```
    /**
     * 获取线程池
     *
     * @return 线程池
     */
    public static ThreadPoolExecutor getExecutor() {
      return EXECUTOR;
    }
}
```

2. 使用

```
// 1、 直接使用execte方法
ThreadPoolUtil.execute(() -> {
  //转换租赁意向为意向客户业务
  leaseConverter.convertIntention(vo);
});

//2、 使用线程池
// 批量验证数据合法性
CompletableFuture<StaffEntity> staffFuture = CompletableFuture.suppl
yAsync(() ->
    staffService.findByIdNoAuth(vo.getSigner(), FROM_IN).getData(),
ThreadPoolUtil.getExecutor());
CompletableFuture<DeptEntity> deptFuture = CompletableFuture.supplyA
sync(() ->
    deptService.findSimpleById(vo.getDeptId(), FROM_IN).getData(), T
hreadPoolUtil.getExecutor());
CompletableFuture<ParkModel> parkFuture = CompletableFuture.supplyAs
ync(() ->
    parkService.findParkByIdNoAuth(vo.getParkId(), FROM_IN).getData(
), ThreadPoolUtil.getExecutor());
CompletableFuture<String> entFuture = CompletableFuture.supplyAsync(
() ->
enterpriseService.queryName(vo.getEnterpriseId(), FROM_IN).getData()
, ThreadPoolUtil.getExecutor());
```

4. CastUtils

```
/**
 * copy from org.springframework.data.util.CastUtils，在没有使用data包的情
况可使用此类
 */
public interface CastUtils {
```

```
    /**
     * 抑制ide提示warnings
     *
     * @param object 转换对象
     * @param <T>    任意类型
     * @return 转换后的对象
     */
    @SuppressWarnings("unchecked")
    static <T> T cast(Object object) {
        return (T) object;
    }
}
```

5. 日期验证类

```
package cn.flyrise.common.valid;

public class DateValidator {


    /**
     * 校验年份是否在合理范围
     *
     * @param year 年份
     */
    public static void year(Integer year) {
        //...
    }

    /**
     * 校验区间是否在合理范围
     *
     * @param section 区间
     */
    public static void section(Integer section) {
        //...
    }

    /**
     * 校验月份是否在合理范围
     *
     * @param month 月份
     */
    public static void month(Integer month) {
        //...
    }
}
```

工具类

6. bean对象属性复制类

```
public class BeanUtils {

    /**
     * 复制bean属性，忽略tenant_id字段
     *
     * @param source bean对象
     * @param clazz  bean类型
     * @return 填充后的对象
     */
    public static <T> T copyProperties(Object source, Class<T> clazz) {
        return BeanUtil.copyProperties(source, clazz, TENANT_ID);
    }

    /**
     * 复制bean属性，忽略id、tenant_id字段
     *
     * @param source bean对象
     * @param clazz  bean类型
     * @return 填充后的对象
     */
    public static <T> T copyInsertProperties(Object source, Class<T> clazz
) {
        return BeanUtil.copyProperties(source, clazz, TENANT_ID, ID);
    }
}
```

通用控件

通用控件

通用工具

数据库文档导出工具

数据库文档导出工具

引入工具包

针对进行导出的工程引入

```
  <dependency>
      <groupId>com.zaxxer</groupId>
      <artifactId>HikariCP</artifactId>
      <version>3.4.5</version>
    </dependency>
    <dependency>
      <groupId>cn.smallbun.screw</groupId>
      <artifactId>screw-core</artifactId>
      <version>1.0.5</version>
    </dependency>
```

新建测试类

请仔细查看以下配置，都有注释说明，请按需进行配置！！！

```
package cn.flyrise.pai.parkproperty.controller;

import cn.smallbun.screw.core.Configuration;
import cn.smallbun.screw.core.engine.EngineConfig;
import cn.smallbun.screw.core.engine.EngineFileType;
import cn.smallbun.screw.core.engine.EngineTemplateType;
import cn.smallbun.screw.core.execute.DocumentationExecute;
import cn.smallbun.screw.core.process.ProcessConfig;
import com.alibaba.druid.pool.DruidDataSource;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import java.util.ArrayList;
import javax.annotation.Resource;
import javax.sql.DataSource;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
```

```java
/**
 * @author lhr
 * @date 2022/4/25 10:21
 */
@SpringBootTest
@RunWith(SpringJUnit4ClassRunner.class)
public class DbScrewTest {

  @Resource
  private DruidDataSource druidDataSource;

  @Test
  public void documentGeneration() {
    //数据源
    HikariConfig hikariConfig = new HikariConfig();
    hikariConfig.setDriverClassName(druidDataSource.getDriverClassName()
);
    hikariConfig.setJdbcUrl(druidDataSource.getUrl());
    hikariConfig.setUsername(druidDataSource.getUsername());
    hikariConfig.setPassword(druidDataSource.getPassword());
    //设置可以获取tables remarks信息
    hikariConfig.addDataSourceProperty("useInformationSchema", "true");
    hikariConfig.setMinimumIdle(2);
    hikariConfig.setMaximumPoolSize(5);
    DataSource dataSource = new HikariDataSource(hikariConfig);
    //生成配置
    EngineConfig engineConfig = EngineConfig.builder()
        //生成文件路径
        .fileOutputDir("/Users/lucky./Documents/fe-work/2022/db-document
")
        //打开目录
        .openOutputDir(true)
        //文件类型
        .fileType(EngineFileType.WORD)
        //生成模板实现
        .produceType(EngineTemplateType.freemarker)
        //自定义文件名称
        .fileName(druidDataSource.getUsername().replace("-", "_")).build
();

    //忽略表
    ArrayList<String> ignoreTableName = new ArrayList<>();
    ignoreTableName.add("test_user");
    ignoreTableName.add("test_group");
    //忽略表前缀
    ArrayList<String> ignorePrefix = new ArrayList<>();
    ignorePrefix.add("test_");
    //忽略表后缀
```

```
        ArrayList<String> ignoreSuffix = new ArrayList<>();
        ignoreSuffix.add("_test");
        ProcessConfig processConfig = ProcessConfig.builder()
                //指定生成逻辑、当存在指定表、指定表前缀、指定表后缀时，将生成指定表，其余表
    不生成、并跳过忽略表配置
                //根据名称指定表生成
                .designatedTableName(new ArrayList<>())
                //根据表前缀生成
                .designatedTablePrefix(new ArrayList<>())
                //根据表后缀生成
                .designatedTableSuffix(new ArrayList<>())
                //忽略表名
                .ignoreTableName(ignoreTableName)
                //忽略表前缀
                .ignoreTablePrefix(ignorePrefix)
                //忽略表后缀
                .ignoreTableSuffix(ignoreSuffix).build();
        //配置
        Configuration config = Configuration.builder()
                //版本
                .version("1.0.0")
                //描述
                .description("数据库设计文档生成")
                //数据源
                .dataSource(dataSource)
                //生成配置
                .engineConfig(engineConfig)
                //生成配置
                .produceConfig(processConfig)
                .build();
        //执行生成
        new DocumentationExecute(config).execute();
    }
}
```

部分说明

1. 工具gitee地址
2. 可导出word后进行复制粘贴至excel中

ER图工具

ER图通用工具

下载地址

使用方式

连接方式：

```
driver-class:com.mysql.cj.jdbc.Driver
url:jdbc:mysql://10.62.17.188:3306/pai_enterprise?characterEncoding=UTF-8&useSSL=false&useUnicode=true&serverTimezone=UTC
username:fe
password:@fe123
```

先从pai开发者数据库中获取建表语句进行建库建表操作，然后再到ER图工具中进行连接：

导入表结构：

新建关系图：

拖拽数据表到关系图，并进行关系连线

过多字段时可隐藏部分，双击图中具体的表对象，进行相应的操作即可：

导出操作：

导出结果：

工程批量镜像脚本

工程批量镜像脚本

本脚本仅支持mac、linux下执行，分支管理参考

```
# 指定分支，确定需要的项目分支
from_version=develop
# 指定镜像版本号
image_version=latest-$(date +%Y%m%d%H%M%S)

echo $image_version

function image() {
    for service in ${services[@]}
    do
        echo $service
        # /home/pai/pai-cli/pai：  pai工具位置
        # xxx -u：开发者账号 -p：开发者密码
        /home/pai/pai-cli/pai login -uxxx -pxxx
        /home/pai/pai-cli/pai image publish -n $service -b $from_version
 -t $image_version
    done
}

# 后端工程
services=(
    # 企业档案
    pai-enterprise
    # 招商
    pai-business
    # pai-business-plan pai-performance-agreement
    # 物业
    pai-park-space pai-contract pai-finance pai-park-property pai-vouche
r
    # 物联
    pai-parking pai-walking pai-security
    # 公用
    pai-fe-config
    # 园区门户
    pai-park-portal
    # 服务中心
    pai-service-center
    # 工单中心
```

```
    pai-work-engine
    # 电子合同
    # pai-electronic-contract-cloud
    pai-electronic-contract
    # 门户(首页、招商、物业等)
    pai-portal
 )

# 打镜像
image

# 前端工程
services=(
    # 企业档案
    pai-enterprise-ui
    # 招商
    pai-business-ui
    # 物业
    pai-park-space-ui pai-park-property-ui pai-contract-ui pai-finance-u
i pai-voucher-ui
    # 物联
    pai-parking-car-ui pai-walking-ui pai-security-ui pai-security-patro
l-ui pai-walking-app pai-parking-month-uniapp pai-parking-temporary-unia
pp
    # 服务中心
    pai-service-center-ui pai-work-place-ui pai-property-service-ui
    # 企业工作台
    pai-enterprise-workbench-ui
    # 工单中心
    pai-work-order-ui
    # 电子合同
    # pai-electronic-contract-cloud-ui
    pai-electronic-contract-ui
    # 门户(首页、招商、物业等)
    pai-portal-ui
)

# 打前端镜像
image
```

业务工单开发实践

工单开发-WEB

工单开发-WEB

一、工单注册

1.填写信息/地址

2.配置节点-【功能按钮/人员/岗位/消息提醒】

二、引入 pai-sp-ui

安装 插件库

```
npm install @flyriselink/pai-sp-ui -S
```

```
import Vue from 'vue'
import paispui from '@flyriselink/pai-sp-ui'
import Cookies from 'js-cookie'

// 引入样式文件
import '@flyriselink/pai-sp-ui/dist/paispui.css'

export default ({ store }) => {
    // 初始化
    Vue.use(paispui, {
        // 接口地址
        baseURL: process.env.REQUEST_BASE_URL,
        // 获取token方法
        token: () => {
            return Cookies.get(process.env.TOKEN_KEY)
        },
    })
}
```

三、工单上报

1、注意：上报的前提：

- ①、先注册工单，复制 orderKey
- ②、业务执行保存后，返回id
- ③、中台 entId != 档案 enterprise，需先转换

2、调用档案匹配组件

第**3**点的 **enterprise** 可以使用组件 `<sp-enterprise-match />` 获取中台入驻企业的**entId** 在企业档案里的**id**

```
<sp-enterprise-match v-model="enterprise" :entId="entId" :parkId="parkId"/>
```

3、执行上报

```
// 上报
this.$paispui.workOrder.report({
  "bizId": "1234566666666666",
  "orderKey": "14942039210345144 32",
  "parkId": "1433617920071446528",
  "title": "SP-UI上报工单标题",
  "enterprise": "14336179200714465 29",
  "priority": 1
}) .then((res) => {
    console.log('上报成功')
})
```

参数	说明	类型	必填
bizId	业务ID	String	是
orderKey	工单编号	String	是
parkId	园区ID	String	是
title	工单上报标题	String	是
serialId	流水号	String	是
enterprise	企业ID，企业档案的ID	String	是
priority	优先级，1一般，2紧急，3非常紧急	Number / String	否

四、工单处理

1.一般在【受理中心】、【我的工单】应用里处理

- 受理中心：

- 我的工单：

2.单独执行处理时：

> 使用【工单引擎组件】 `<sp-work-order />`

```
<template>
    <sp-work-order
    :dialog.sync="handleDialog"
    :readOnly="false"
    :entId="'1433617920071446529'"
    @update="updateList"
    ></sp-work-order>
</template>

<script>
export default {
  data() {
      return {
        handleDialog: {
            visible: false,
            id: '1507195874022633474',
        },
    }
  },
  methods: {
    updateList() {
      console.log('更新列表')
    },
  },
}
</script>
```

> 办理时需传入：
>
> - id: 工单流程实例ID
> - visible: true, 打开显示弹窗，办理/处理页面

pai-sp-ui 文档链接： `https://pai.flyrise.cn/pai-sp-ui/#/docs/work-order`

五、工单评价

使用【工单引擎组件】`<sp-work-order-evaluation />`

```
<template>
    <div>
        <sp-work-order-evaluation
        :dialog.sync="evaDialog"
        :bizUrl="evaDialog.bizUrl"
        @update="updateList"
        ></sp-work-order-evaluation>

        <el-button @click="clickHandle">评价</el-button>
        <el-button @click="clickViewEva">已评价</el-button>
    </div>
</template>

<script>
export default {
  data() {
      return {
        evaDialog: {
          visible: false,
          parkId: '1433617920071446528',
          bizUrl: '/service-property-repairs-api/repairs/v1/customer/eva
luate'
        },
    }
  },
  methods: {
      clickHandle() {
          this.evaDialog.type = 'add'
          this.evaDialog.title = '物业报修工单 xxx'
          this.evaDialog.id = '1497402882945126402'
          this.evaDialog.visible = true
      },
      clickViewEva() {
          this.evaDialog.type = 'view'
          this.evaDialog.title = '物业报修工单 xxx'
          this.evaDialog.id = '1497402882945126402'
          this.evaDialog.visible = true
      },
  },
}
</script>
```

一般情况：需要判断状态为已验收后，进行评价

> parkId: 需传入当前园区id

> bizUrl: 为业务评价的接口后端 api
> type: add 为新增评价，需传入业务id
> type: view 为查看评价，需传入业务id

工单开发-APP

工单开发-APP

> 可跳转【小程序】、【H5页面】

> 需先注册工单

一、引入 pai-sp-mobile

1.安装【中台】**pai-mp-ui** 插件库

```
npm install @flyriselink/pai-mp-ui -S
```

2.安装 业务组件库

```
npm install @flyriselink/pai-sp-mobile -S
```

```
// main.js
import flyrise from "@flyriselink/pai-mp-ui";
import SPUI from "@flyriselink/pai-sp-mobile";

//pai工具$p
Vue.use(flyrise)

let { $p } = Vue.prototype

// SPUI框架
Vue.use(SPUI, {
    baseUrl: $p.tool.getBasePath(),
    token: () => {
        return $p.tool.getToken()
    },
});
```

3.引入组件

```
"easycom": {
    "autoscan": true,
    "custom": {
        "^sp-(.*)": "@flyriselink/pai-sp-mobile/components/sp-$1/sp-$1.vue"
    }
},
```

4.vue.config.js

```
module.exports = {
    transpileDependencies: ['@flyriselink/pai-mp-ui', '@flyriselink/pai-sp-mobile'],
    ...
}
```

二、工单上报

1、注意：上报的前提：

- ①、先注册工单，复制 orderKey
- ②、业务执行保存后，返回id
- ③、中台 entId != 档案 enterprise，需先转换

2、调用档案匹配组件

第**3**点的 **enterpriseId** 可以使用组件 `<sp-enterprise-match />` 获取中台入驻企业的**entId** 在企业档案里的**id**

```
<sp-enterprise-match v-model="enterpriseId" :entId="entId" :parkId="parkId"/>
```

3、执行上报

```
this.$spui.workOrder.report({
  "bizId": "1234566666666666",
  "orderKey": "1494203921034514432",
  "parkId": "1433617920071446528",
  "title": "SP-UI上报工单标题",
  "enterpriseId": "1433617920071446529",
  "priority": 1
}) .then((res) => {
    uni.showToast({
        title: '上报成功',
    })
})
```

参数	说明	类型	必填
bizId	业务ID	String	是
orderKey	工单编号	String	是
parkId	园区ID	String	是

title	工单上报标题	String	是
serialId	流水号	String	是
enterpriseId	企业ID，企业档案的ID	String	是
priority	优先级，1一般，2紧急，3非常紧急	Number / String	否

三、工单处理（流程记录、处理按钮）

> 与 **WEB** 不同：
>
> - ①、是小程序之间进行跳转
> - ②、分开【进度】与【办理】，需分别调用组件
> - ③、办理时，一般同时调 `<sp-work-order />` 与 `<sp-work-order-record />`

1.一般在【我的工单】小程序里，进行跳转到【业务小程序】，携带的参数：**instanceId** 与

2.APP处理时：

在表单里，使用【工单引擎组件】 `<sp-work-order />`

```
<template>
    <sp-work-order
        ref="WorkOrder"
        v-if="formType != 'add' && instanceId"
        :parkId="form.parkId"
        :instance-id="instanceId"
        @update="updateData"
    ></sp-work-order>
</template>

<script>
export default {
  data() {
      return {
        formId: '',
        instanceId: '',
        formType: 'view'
    }
  },
  onLoad(option) {
    this.isRedirect = this.$p.tool.redirect(this, option)
    this.$nextTick(() => {
        let bizData = option.biz || '{}'
        try {
```

```
                bizData = JSON.parse(bizData)
            } catch (e) {
                bizData = {}
            }
            let formId = option.id || bizData.id
            this.formId = formId
            this.instanceId = option.instanceId || bizData.instanceId

            if (this.formId) {
                this.formType = 'view'
            }

            this.loadInit()
        })
    },
    methods: {
        updateData() {
            console.log('更新数据')
        },
    },
}
</script>
```

pai-sp-ui 文档链接： https://pai.flyrise.cn/pai-sp-ui/#/docs/work-order

---

3.处理时查看记录：

在表单里，使用【工单进度组件】`<sp-work-order-record />`

```
<sp-work-order-record
ref="WorkOrderRecord"
v-if="formType != 'add' && instanceId"
:instance-id="instanceId"
></sp-work-order-record>
```

4.办理时效果如下：

四、工单评价

使用【工单评价组件】`<sp-work-order-evaluation />`

```
<template>
    <div>
        <sp-work-order-evaluation
```

```
          :dialog.sync="evaDialog"
          :bizUrl="evaDialog.bizUrl"
          @update="updateList"
          ></sp-work-order-evaluation>

          <button @click="clickHandle">评价</button>
          <button @click="clickViewEva">已评价</button>
      </div>
</template>

<script>
export default {
  data() {
      return {
        evaDialog: {
          visible: false,
          parkId: '1433617920071446528',
          bizUrl: '/service-property-repairs-api/repairs/v1/customer/eva
luate'
        },
      }
  },
  methods: {
      clickHandle() {
          this.evaDialog.type = 'add'
          this.evaDialog.title = '物业报修工单 xxx'
          this.evaDialog.id = '1497402882945126402'
          this.evaDialog.visible = true
      },
      clickViewEva() {
          this.evaDialog.type = 'view'
          this.evaDialog.title = '物业报修工单 xxx'
          this.evaDialog.id = '1497402882945126402'
          this.evaDialog.visible = true
      },
  },
}
</script>
```

一般情况：需要判断状态为已验收后，进行评价

> parkId: 需传入当前园区id
> bizUrl: 为业务评价的接口后端 api
> type: add 为新增评价，需传入业务id
> type: view 为查看评价，需传入业务id

工单开发-后端

工单后端开发实践

标准业务工单流程图

工单引擎公共业务包

```
<!--业务通用依赖-->
<dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>cn.flyrise</groupId>
        <artifactId>pai-fe-bom</artifactId>
        <version>1.0.1-SNAPSHOT</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
```

```
<!--消息监听依赖-->
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-common-mq</artifactId>
</dependency>
<!--工单引擎-->
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-fe-common-work-engine</artifactId>
</dependency>
```

工单引擎状态解释

> status == 0 –> 待受理
> status == 1 –> 待确认
> status == 2 –> 待派单
> status == 3 –> 待接单
> status == 4 –> 待处理
> status == 5 –> 待提交

> status == 6 –> 待验收
> status == 7 –> 结束
> status == 7 && isClose == 1 –> 工单被关闭
> status == 7 && isReject == 1 –> 工单被驳回
> status == 7 && isRevocation == 1 –> 工单被撤回（发起方撤回）
> isReject == 2 –> 工单被退回（仅退回至上一节点时所展示的状态）

MQ消费者编写

```
import cn.flyrise.common.work.engine.order.model.NotifyMqVO;
import cn.flyrise.mq.core.annotation.PaiMqListener;
import cn.flyrise.mq.core.model.PaiMqMessageBean;
import cn.flyrise.pai.servicepropertyrepairs.factory.OrderMessageFactory
;
import cn.flyrise.pai.servicepropertyrepairs.service.IMessageConsumerSer
vice;
import cn.hutool.json.JSONUtil;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.TypeReference;
import org.springframework.stereotype.Component;

//${pai.mq.tag} --> cn.flyrise.work.order
//${pai.mq.topic} --> work_order_notify
//建议配置在nacos的配置文件中
@Component
public class WorkOrderConsumer {
  @PaiMqListener(suiteCode = "${pai.mq.tag}", topic = "${pai.mq.topic}")
  public void workEngineMsgHandler(String msg) {
    PaiMqMessageBean<String> bean = JSON
        .parseObject(msg, new TypeReference<PaiMqMessageBean<String>>()
{
        });
    NotifyMqVO notifyMqVo = JSONUtil.toBean(bean.getData(), NotifyMqVO.c
lass);
    try {
        // 业务处理逻辑
    } catch (Exception e) {
       // 异常处理逻辑
    }
  }
}
```

拓展接口描述

查看客户评价

内部接口调用方式: `IRemoteWorkOrderService#selectCustomerEvaluate(String id)`

接口地址: `/work-engine-api/work/order/v1/inner/customer/evaluate`

请求方式: `GET`

请求数据类型: `application/x-www-form-urlencoded`

响应数据类型: `*/*`

接口描述:

请求参数:

参数名称	参数说明	in	是否必须	数据类型	schema
bizId	bizId	query	false	string	

响应状态:

状态码	说明	schema
200	OK	Reply«CustomerEvaluateVO»
401	Unauthorized	
403	Forbidden	
404	Not Found	

响应参数:

参数名称	参数说明	类型	schema
annex		object	
code		string	
data		CustomerEvaluateVO	CustomerEvaluateVO
bizId	业务ID	string	
createTime	评价时间	string	
evaluate	评价	string	
parkId	园区ID	string	

score	评分	integer	
msg		string	
success		boolean	
time		integer(int64)	integer(int64)

响应示例:

```
{
    "annex": {},
    "code": "",
    "data": {
        "bizId": "",
        "createTime": "",
        "evaluate": "",
        "parkId": "",
        "score": 0
    },
    "msg": "",
    "success": true,
    "time": 0
}
```

客户评价

内部接口调用方式: `IRemoteWorkOrderService#customerEvaluate(CustomerEvaluateVO vo)`

接口地址: `/work-engine-api/work/order/v1/inner/customer/evaluate`

请求方式: `POST`

请求数据类型: `application/json`

响应数据类型: `*/*`

接口描述:

请求示例:

```
{
  "bizId": "",
  "createTime": "",
  "evaluate": "",
  "parkId": "",
```

```
  "score": 0
}
```

请求参数:

参数名称	参数说明	in	是否必须	数据类型	schema
vo	vo	body	true	CustomerEvaluateVO	CustomerEvaluateVO
bizId	业务ID		false	string	
createTime	评价时间		false	string	
evaluate	评价		false	string	
parkId	园区ID		false	string	
score	评分		false	integer	

响应状态:

状态码	说明	schema
200	OK	Reply«boolean»
201	Created	
401	Unauthorized	
403	Forbidden	
404	Not Found	

响应参数:

参数名称	参数说明	类型	schema
annex		object	
code		string	
data		boolean	

msg		string	
success		boolean	
time		integer(int64)	integer(int64)

响应示例:

```
{
    "annex": {},
    "code": "",
    "data": true,
    "msg": "",
    "success": true,
    "time": 0
}
```

撤回工单

内部接口调用方式: `IRemoteWorkOrderService#revocation(RevocationVO vo)`

接口地址: `/work-engine-api/work/order/v1/inner/revocation`

请求方式: `POST`

请求数据类型: `application/json`

响应数据类型: `*/*`

接口描述:

请求示例:

```
{
  "bizId": "",
  "reason": ""
}
```

请求参数:

参数名称	参数说明	in	是否必须	数据类型	schema
vo	vo	body	true	撤回请求实体	撤回请求实体
bizId	业务ID		false	string	

reason	撤回原因		false	string	

响应状态:

状态码	说明	schema
200	OK	Reply«boolean»
201	Created	
401	Unauthorized	
403	Forbidden	
404	Not Found	

响应参数:

参数名称	参数说明	类型	schema
annex		object	
code		string	
data		boolean	
msg		string	
success		boolean	
time		integer(int64)	integer(int64)

响应示例:

```
{
    "annex": {},
    "code": "",
    "data": true,
    "msg": "",
    "success": true,
    "time": 0
}
```

NotifyMqVO实体（MQ消息数据实体）

```
public class NotifyMqVO implements Serializable {
  @ApiModelProperty("工单业务主键")
  private String bizId;
  @ApiModelProperty("PC表单链接")
  private String formLink;
  @ApiModelProperty("APP表单链接")
  private String appFormLink;
  @ApiModelProperty("工单实例ID")
  private String instanceId;
  @ApiModelProperty("工单状态")
  private Integer status;
  @ApiModelProperty("园区ID")
  private String parkId;
  @ApiModelProperty("工单KEY")
  private String orderKey;
  @ApiModelProperty("工单标题")
  private String title;
  @ApiModelProperty("工单紧急程度")
  private Integer priority;
  @ApiModelProperty("工单类型")
  private String orderType;
  @ApiModelProperty("企业ID")
  private String enterprise;
  @ApiModelProperty("操作类型")
  private String operation;
  @ApiModelProperty("是否关闭")
  private Integer isClose;
  @ApiModelProperty("是否驳回")
  private Integer isReject;
  @ApiModelProperty("关闭原因")
  private String closeReason;
  @ApiModelProperty("驳回原因")
  private String rejectReason;
  @ApiModelProperty("是否撤回")
  private Integer isRevocation;
  @ApiModelProperty("撤回原因")
  private String revocationReason;
  @ApiModelProperty("租户ID")
  private String tenantId;
  }
```

边端部署/同步

边端 私有化部署同步

—宝控—

套件：访问网址：**https://testpai.cndi.com/data-sync-edge-agent-api/tools/sync**

镜像：使用 **Postman**

PUT

http://testpai.cndi.com/data-sync-edge-agent-api/set-image?name=pai-blogis-biz-device&namespace=pai-cloud&image=dev.flyrise.cn:8082/pi-dev/pai-blogis-biz-device:latest-20220808095423&k=202208121740

name : 服务名
namespace: 命名空间(不用改，pai-cloud就好)
image：镜像地址(pai image publish 得到)
k: 年月日时分

—翠亨—

套件：访问网址：**http://10.21.6.204:32666/?appCode=cn.flyrise.service.center&k=202208231453**

镜像：使用 **Postman**

PUT

http://testpai.zschzhcs.com/data-sync-edge-agent-api/set-image?name=pai-service-center&namespace=pai-cloud&image=dev.flyrise.cn:8082/pi-dev/pai-service-center:latest-20220818162152&k=202208181657

华为软件开发云使用

华为软件开发服务：https://devcloud.huaweicloud.com/home

企业账号名：accpt27chen
默认账号: 通常为姓名全拼

用户名及密码由管理员（艳芳）提供的为准
首次登录后要求修改密码，特别提醒要设置昵称! 特别提醒要设置昵称! 特别提醒要设置昵称! 特别提醒要设置昵称!

事前准备下载软件：
TortoiseGit工具 ： https://tortoisegit.org/download/ （可选）

Git 客户端： https://git-scm.com/download/win

一、登录云平台

二、 获取Maven配置文件setting.xml

将下载后的setting.xml配置到IDEA中（此文件包含了用户的账号及密码）

三、设置代码托管之HTTPS密码

设置 完成后再次点开[设置我的HTTPS密码]获取git 对应的账号用于TortoiseGit下载代码用

四、配置TortoiseGit

配置全局账号及密码

开始下载代码

获取仓库代码地址

开始下载（首次会提示输入账号及密码，注意是第三步中的用户名）

五、配置IDEA开发环境
直接用IDEA打开已下载好的代码目录，直到加载完成（此前注意配置setting.xml否则会下载不了Maven配置的相关Jar）

以上取代码的过程亦可直接用IDEA进行，但仍需要git.exe客户端

六、本地干净项目转为Git并推送到远程仓库

（由于IDEA中git插件功能不足，所以还要回到命令行进行操作）
创建项目并准备好相关源代码文件

```
1.本地初始化git目录
git init

2.添加到暂存区
git add .
git commit -m "初始化项目"

3.添加远程仓库
git remote add origin https://xxxx/fe_trunk.git

4.推送到远程仓库
git push -u origin master
或
git pull origin master
git push origin master
```

异常时考虑做以下两步

```
1.本地仓库也远程仓库关联
git branch --set-upstream-to=origin/master master

2.拉取远程仓库内容到本地
这时候用git pull会提示(毕竟本地和远程仓库没啥关系指针连接不起来的缘故吧):
fatal: refusing to merge unrelated histories
因此命令应该改为:
git pull origin master --allow-unrelated-histories
```

七、提交日志规范
为了与任务自动关联，请在提交日志填写时按如下规范：

点击版本号可以显示修改过的代码差异内容

华发

C端无租户隔离接口

C端功能，无租户隔离接口

公告
/service-center-api/announcement/v1/customer/page?parkId=1451029848867622912
公告类型（无parkid)/service-center-api/announcement/v1/type/list
公告详情 service-center-api/announcement/v1/personal/info

活动
/service-center-api/activity/v1/park/new?page=1&size=5&publishType=C
/service-center-api/activity/v1/park/open
/service-center-api/activity/v1/park/old

活动类型（无parkId）/service-center-api/activity-type/v1/activity/park/type
活动详情 /service-center-api/activity/v1

广告位 service-center-api/advertise-content/v1/contents

政策
/service-center-api/policy/v1/customer/page
政策类型 /service-center-api/policy/v1/type/list
政策详情 /service-center-api/policy/v1/personal/info

新闻
/service-center-api/news/v1/customer/page
新闻类型 /service-center-api/news/v1/type/list
新闻详情 /service-center-api/news/v1/personal/info

场馆
/service-center-api/stadiumReservation/customer/reservation/list
场馆详情
/service-center-api/stadiumReservation/customer/reservation/query
场馆 我的预定
/service-center-api/stadiumReservation/customer/list

凌云入门教程（文聪）

快速学习

前言

参考《凌云中台开放文档》与个人实操得出来的简易教程，方便交付开发同事短时间进场开发，目标是减少前期学习成本，减少踩坑次数。

本教程以一个完整的业务开发案例，从项目后台、前端工程搭建，到中台开放能力都有涉及。所以建议阅读学习本教程，从入门跟着操作，把所有功能都敲遍，以达到边学边熟识中台开发。

教程中大部分来源于《凌云中台开放文档》，通过简化原有的描述、去除不需要了解的技术知识、加强业务开发连贯性。如您对中台开发有经验请忽略本教程。

适合对象

后端：有JAVA开发基础
前台：
APP:

平台简介

介绍

业务中台

业务中台是以业务领域划分边界，形成高内聚、低耦合的面向业务领域的能力中心。打造持续演进的业务能力共享服务平台，同时也是配置、编排和扩展业务对象、业务能力、业务规则及业务流程，完成园区资源运营管理的平台。

业务中台是一个充满生命力的个体，它承载业务逻辑、沉淀业务数据、产生业务价值，并随着业务不断发展进化。

基于业务中台，我们在凌云中台构建了以场景化为建设体系的智慧园区运营平台。

从使用者角度来划分为平台运营管理方、平台系统应用开发方及平台系统应用使用方三类，其中平台系统应用使用方又划分为管理者和普通用户两类。

平台系统应用开发方

通过开放平台快速组建开发团队；学习基于技术中台、业务中台、数据中台的支撑能力，快速开放上架并管理应用市场中的产品。
https://pai.flyrise.cn/developer

平台运营管理方

基于运营后台可以进行用户管理、开发者管理、应用市场管理、园区管理、内容运维及营运分析。协

助开发团队将新上线的应用产品向园区、企业等使用方推广，并及时收集、分析、反馈相关使用情况，帮助应用产品进化提高服务质量。
https://pai.flyrise.cn/console

平台系统应用使用方

可以是园区级也可以是企业级用户，通过业务中台可快速建立组织架构，通过应用市场安装配置管理应用产品，基于数据统计结果快速定位决策以提高生效效率。
https://pai.flyrise.cn/workbench

> 数据中台、技术中台、物联中台详细介绍请看：http://api.flyrise.cn:9099/docs/open-docs/open-docs-1ctpr89es5log

平台术语

为了日后更好的沟通，请熟识以下术语

名词	说明
开发者	开发者是平台的重要组成部分，所有应用都由开发者完成，并上架到应用市场中。初期直接通过运营后台的开发者管理，创建开发者的账号，并交付给对应的开发人员进行开发
套件	套件是基于业务模块的项目，项目中根据多种应用场景拆分出多个微应用。开发者可以通过开放平台创建套件，创建好后，需要下载pai工具进行接下来的多终端工程创建和后续的开发
应用	套件对应的应用，有不同的版本。每个套件应用在控制台都有一个独立应用后台管理页面，管理员在该后台可以对应用做基础信息、角色管理、参数配置等操作。
微应用	微应用是由应用开发者开发的，提供给园区运营方、入驻企业、个人的微服务应用。一个套件应用可以有多个微应用
工程	pai工具创建的多终端工程，微应用需要前端工程、后端工程等
小组件	分为APP端及Web端两种
小程	基于UniApp技术方案实现的可运行于APP的业务应用小程序

序	
消息配置	套件消息的配置，管理业务消息模板，每种发送方式只能有一个。配置不同的发送方式，就会发送该渠道的消息。
消息模板	消息内容的模板，有短信、邮件、应用消息、app推送四种发送方式

开发环境准备

> 欲善其事，必先利其器

必装工具

JDK 安装

PAI(下章有介绍)支持在windows、linux、MacOS操作系统中使用，使用pai依赖JDK。

- JDK1.8.0 Widnows版 、MacOS版、 Linux版

pai工具

- windows
  点击此处下载 保存后解压到指定位置

- Linux安装

```
# 安装解压工具 unzip
# CentOS
sudo yum install -y unzip

# Ubuntu
sudo apt-get install unzip

# 下载并解压
cd /usr/local && curl -o pai-cli.zip https://api.flyrise.cn/pai/wrapper/latest/pai-cli-release.zip && unzip pai-cli.zip

# 进行目录授权测试运行
cd pai-cli && chmod 755 pai && sh pai
```

- MacOS安装

```
# 下载并解压
curl -o pai-cli.zip https://api.flyrise.cn/pai/wrapper/latest/pai-cli-release.zip && unzip pai-cli.zip

# 进行目录授权测试运行
cd pai-cli && chmod 755 pai && sh pai
```

环境配置

- Windows

- Linux、MacOS
  编辑配置文件

```
vi ~/.bash_profile

#添加以下配置（注意Jdk、pai以实现安装路径配置）
export JAVA_HOME="/usr/local/jdk1.8.0_211"
export PATH="/usr/local/pai-cli:$JAVA_HOME/bin:$PATH"
```

配置文件生效

```
source ~/.bash_profile
```

后端开发

- IntelliJ IDEA 下载

- Maven 下载
  解压后替换settingx.xml
  有两个配置，

1. 一般开发配置

一般开发的定义为：仅使用标准骨架进行业务开发，无需上传公共组件包。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
-->

<!--
 | This is the configuration file for Maven. It can be specified at two
levels:
 |
 |  1. User Level. This settings.xml file provides configuration for a s
ingle user,
 |                 and is normally provided in ${user.home}/.m2/settings
.xml.
 |
 |                 NOTE: This location can be overridden with the CLI op
tion:
 |
 |                 -s /path/to/user/settings.xml
 |
 |  2. Global Level. This settings.xml file provides configuration for a
ll Maven
 |                 users on a machine (assuming they're all using the sa
me Maven
 |                 installation). It's normally provided in
 |                 ${maven.conf}/settings.xml.
 |
 |                 NOTE: This location can be overridden with the CLI op
tion:
 |
 |                 -gs /path/to/global/settings.xml
 |
 | The sections in this sample file are intended to give you a running s
tart at
```

```
 | getting the most out of your Maven installation. Where appropriate, t
he default
 | values (values used when the setting is not specified) are provided.
 |
 |-->

<settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0 htt
ps://maven.apache.org/xsd/settings-1.2.0.xsd">
    <!-- localRepository
   | The path to the local repository maven will use to store artifacts.
   |
   | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->
    <!-- interactiveMode
     | This will determine whether maven prompts you when it needs input
. If set to false,
     | maven will use a sensible default value, perhaps based on some ot
her setting, for
     | the parameter in question.
     |
     | Default: true
    <interactiveMode>true</interactiveMode>
    -->

    <!-- offline
     | Determines whether maven should attempt to connect to the network
 when executing a build.
     | This will have an effect on artifact downloads, artifact deployme
nt, and others.
     |
     | Default: false
    <offline>false</offline>
    -->

    <!-- pluginGroups
     | This is a list of additional group identifiers that will be searc
hed when resolving plugins by their prefix, i.e.
     | when invoking a command line like "mvn prefix:goal". Maven will a
utomatically add the group identifiers
     | "org.apache.maven.plugins" and "org.codehaus.mojo" if these are n
ot already contained in the list.
     |-->
    <pluginGroups>
        <!-- pluginGroup
         | Specifies a further group identifier to use for plugin lookup
```

```
.
        <pluginGroup>com.your.plugins</pluginGroup>
        -->
    </pluginGroups>

    <!-- proxies
     | This is a list of proxies which can be used on this machine to co
nnect to the network.
     | Unless otherwise specified (by system property or command-line sw
itch), the first proxy
     | specification in this list marked as active will be used.
     |-->
    <proxies>
        <!-- proxy
         | Specification for one proxy, to be used in connecting to the
network.
         |
        <proxy>
          <id>optional</id>
          <active>true</active>
          <protocol>http</protocol>
          <username>proxyuser</username>
          <password>proxypass</password>
          <host>proxy.host.net</host>
          <port>80</port>
          <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
        </proxy>
        -->
    </proxies>

    <!-- servers
     | This is a list of authentication profiles, keyed by the server-id
 used within the system.
     | Authentication profiles can be used whenever maven must make a co
nnection to a remote server.
     |-->
    <servers>
        <!-- server
         | Specifies the authentication information to use when connecti
ng to a particular server, identified by
         | a unique name within the system (referred to by the 'id' attr
ibute below).
         |
         | NOTE: You should either specify username/password OR privateK
ey/passphrase, since these pairings are
         |       used together.
         |
        <server>
```

```
      <id>deploymentRepo</id>
      <username>repouser</username>
      <password>repopwd</password>
    </server>
    -->

    <!-- Another sample, using keys to authenticate.
    <server>
      <id>siteServer</id>
      <privateKey>/path/to/private/key</privateKey>
      <passphrase>optional; leave empty if not used.</passphrase>
    </server>
    -->
  </servers>

  <!-- mirrors
   | This is a list of mirrors to be used in downloading artifacts fro
m remote repositories.
   |
   | It works like this: a POM may declare a repository to use in reso
lving certain artifacts.
   | However, this repository may have problems with heavy traffic at
times, so people have mirrored
   | it to several places.
   |
   | That repository definition will have a unique id, so we can creat
e a mirror reference for that
   | repository, to be used as an alternate download site. The mirror
site will be the preferred
   | server for that repository.
   |-->
  <mirrors>
    <!-- mirror
     | Specifies a repository mirror site to use instead of a given
repository. The repository that
     | this mirror serves has an ID that matches the mirrorOf elemen
t of this mirror. IDs are used
     | for inheritance and direct lookup purposes, and must be uniqu
e across the set of mirrors.
     |
    <mirror>
      <id>mirrorId</id>
      <mirrorOf>repositoryId</mirrorOf>
      <name>Human Readable Name for this Mirror.</name>
      <url>http://my.repository.com/repo/path</url>
    </mirror>
     -->
```

```
        <mirror>
            <id>insecure-repo</id>
            <mirrorOf>external:http:*</mirrorOf>
            <url>http://dev.flyrise.cn:8081/repository/maven-public/</ur
l>
            <blocked>false</blocked>
        </mirror>
    </mirrors>

    <!-- profiles
     | This is a list of profiles which can be activated in a variety of
 ways, and which can modify
     | the build process. Profiles provided in the settings.xml are inte
nded to provide local machine-
     | specific paths and repository locations which allow the build to
work in the local environment.
     |
     | For example, if you have an integration testing plugin - like cac
tus - that needs to know where
     | your Tomcat instance is installed, you can provide a variable her
e such that the variable is
     | dereferenced during the build process to configure the cactus plu
gin.
     |
     | As noted above, profiles can be activated in a variety of ways. O
ne way - the activeProfiles
     | section of this document (settings.xml) - will be discussed later
. Another way essentially
     | relies on the detection of a system property, either matching a p
articular value for the property,
     | or merely testing its existence. Profiles can also be activated b
y JDK version prefix, where a
     | value of '1.4' might activate a profile when the build is execute
d on a JDK version of '1.4.2_07'.
     | Finally, the list of active profiles can be specified directly fr
om the command line.
     |
     | NOTE: For profiles defined in the settings.xml, you are restricte
d to specifying only artifact
     |       repositories, plugin repositories, and free-form properties
 to be used as configuration
     |       variables for plugins in the POM.
     |
     |-->
    <profiles>
        <!-- profile
         | Specifies a set of introductions to the build process, to be
activated using one or more of the
```

```
          | mechanisms described above. For inheritance purposes, and to
activate profiles via <activatedProfiles/>
          | or the command line, profiles have to have an ID that is uniq
ue.
          |
          | An encouraged best practice for profile identification is to
use a consistent naming convention
          | for profiles, such as 'env-dev', 'env-test', 'env-production'
, 'user-jdcasey', 'user-brett', etc.
          | This will make it more intuitive to understand what the set o
f introduced profiles is attempting
          | to accomplish, particularly when you only have a list of prof
ile id's for debug.
          |
          | This profile example uses the JDK version to trigger activati
on, and provides a JDK-specific repo.
        <profile>
          <id>jdk-1.4</id>

          <activation>
            <jdk>1.4</jdk>
          </activation>

          <repositories>
            <repository>
              <id>jdk14</id>
              <name>Repository for JDK 1.4 builds</name>
              <url>http://www.myhost.com/maven/jdk14</url>
              <layout>default</layout>
              <snapshotPolicy>always</snapshotPolicy>
            </repository>
          </repositories>
        </profile>
        -->

        <!--
     | Here is another profile, activated by the system property 'target
-env' with a value of 'dev',
     | which provides a specific path to the Tomcat instance. To use thi
s, your plugin configuration
     | might hypothetically look like:
     |
     | ...
     | <plugin>
     |   <groupId>org.myco.myplugins</groupId>
     |   <artifactId>myplugin</artifactId>
     |
     |   <configuration>
```

```
      |     <tomcatLocation>${tomcatPath}</tomcatLocation>
      |   </configuration>
      | </plugin>
      | ...
      |
      | NOTE: If you just wanted to inject this configuration whenever so
meone set 'target-env' to
      |       anything, you could just leave off the <value/> inside the
activation-property.
      |
    <profile>
      <id>env-dev</id>

      <activation>
        <property>
          <name>target-env</name>
          <value>dev</value>
        </property>
      </activation>

      <properties>
        <tomcatPath>/path/to/tomcat/instance</tomcatPath>
      </properties>
    </profile>
    -->
        <profile>
            <id>pai</id>
            <repositories>
                <repository>
                    <id>pai</id>
                    <name>Pai Maven</name>
                    <url>http://dev.flyrise.cn:8081/repository/maven-pub
lic/</url>
                </repository>
            </repositories>
        </profile>
    </profiles>

    <!-- activeProfiles
     | List of profiles that are active for all builds.
     |
    <activeProfiles>
      <activeProfile>alwaysActiveProfile</activeProfile>
      <activeProfile>anotherAlwaysActiveProfile</activeProfile>
    </activeProfiles>
    -->
    <activeProfiles>
        <activeProfile>pai</activeProfile>
```

```
    </activeProfiles>
</settings>
```

2. 组件开发配置

组件开发的定义为：开发公共Jar包，需要上传到Maven仓库，因此需要上传仓库的账号密码及解密密钥文件。

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
-->

<!--
 | This is the configuration file for Maven. It can be specified at two
levels:
 |
 |  1. User Level. This settings.xml file provides configuration for a s
ingle user,
 |                 and is normally provided in ${user.home}/.m2/settings
.xml.
 |
 |                 NOTE: This location can be overridden with the CLI op
tion:
 |
 |                 -s /path/to/user/settings.xml
 |
 |  2. Global Level. This settings.xml file provides configuration for a
ll Maven
 |                 users on a machine (assuming they're all using the sa
me Maven
```

```
  |                 installation). It's normally provided in
  |                 ${maven.conf}/settings.xml.
  |
  |                 NOTE: This location can be overridden with the CLI op
tion:
  |
  |                 -gs /path/to/global/settings.xml
  |
  | The sections in this sample file are intended to give you a running s
tart at
  | getting the most out of your Maven installation. Where appropriate, t
he default
  | values (values used when the setting is not specified) are provided.
  |
  |-->
<settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0 htt
ps://maven.apache.org/xsd/settings-1.2.0.xsd">
    <!-- localRepository
   | The path to the local repository maven will use to store artifacts.
   |
   | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
   -->
    <!-- interactiveMode
     | This will determine whether maven prompts you when it needs input
. If set to false,
     | maven will use a sensible default value, perhaps based on some ot
her setting, for
     | the parameter in question.
     |
     | Default: true
    <interactiveMode>true</interactiveMode>
    -->

    <!-- offline
     | Determines whether maven should attempt to connect to the network
 when executing a build.
     | This will have an effect on artifact downloads, artifact deployme
nt, and others.
     |
     | Default: false
    <offline>false</offline>
    -->

    <!-- pluginGroups
     | This is a list of additional group identifiers that will be searc
```

```
hed when resolving plugins by their prefix, i.e.
     | when invoking a command line like "mvn prefix:goal". Maven will a
utomatically add the group identifiers
     | "org.apache.maven.plugins" and "org.codehaus.mojo" if these are n
ot already contained in the list.
     |-->
    <pluginGroups>
        <!-- pluginGroup
         | Specifies a further group identifier to use for plugin lookup
.
        <pluginGroup>com.your.plugins</pluginGroup>
        -->
    </pluginGroups>

    <!-- proxies
     | This is a list of proxies which can be used on this machine to co
nnect to the network.
     | Unless otherwise specified (by system property or command-line sw
itch), the first proxy
     | specification in this list marked as active will be used.
     |-->
    <proxies>
        <!-- proxy
         | Specification for one proxy, to be used in connecting to the
network.
         |
        <proxy>
          <id>optional</id>
          <active>true</active>
          <protocol>http</protocol>
          <username>proxyuser</username>
          <password>proxypass</password>
          <host>proxy.host.net</host>
          <port>80</port>
          <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
        </proxy>
        -->
    </proxies>

    <!-- servers
     | This is a list of authentication profiles, keyed by the server-id
 used within the system.
     | Authentication profiles can be used whenever maven must make a co
nnection to a remote server.
     |-->
    <servers>
        <!-- server
         | Specifies the authentication information to use when connecti
```

```
ng to a particular server, identified by
         | a unique name within the system (referred to by the 'id' attr
ibute below).
         |
         | NOTE: You should either specify username/password OR privateK
ey/passphrase, since these pairings are
         |       used together.
         |
        <server>
          <id>deploymentRepo</id>
          <username>repouser</username>
          <password>repopwd</password>
        </server>
        -->

        <!-- Another sample, using keys to authenticate.
        <server>
          <id>siteServer</id>
          <privateKey>/path/to/private/key</privateKey>
          <passphrase>optional; leave empty if not used.</passphrase>
        </server>
        -->
        <server>
            <id>releases</id>
            <username>uploader</username>
            <password>{gzgHcvA10/AHd/7gZ3HyEbtEnCC0u1j0gCJ4b5K6SS0=}</pa
ssword>
        </server>
        <server>
            <id>snapshots</id>
            <username>uploader</username>
            <password>{gzgHcvA10/AHd/7gZ3HyEbtEnCC0u1j0gCJ4b5K6SS0=}</pa
ssword>
        </server>
        <server>
            <id>z_mirrors</id>
        </server>
    </servers>

    <!-- mirrors
     | This is a list of mirrors to be used in downloading artifacts fro
m remote repositories.
     |
     | It works like this: a POM may declare a repository to use in reso
lving certain artifacts.
     | However, this repository may have problems with heavy traffic at
times, so people have mirrored
     | it to several places.
```

```
      |
      | That repository definition will have a unique id, so we can creat
e a mirror reference for that
      | repository, to be used as an alternate download site. The mirror
site will be the preferred
      | server for that repository.
      |-->
    <mirrors>
        <!-- mirror
         | Specifies a repository mirror site to use instead of a given
repository. The repository that
         | this mirror serves has an ID that matches the mirrorOf elemen
t of this mirror. IDs are used
         | for inheritance and direct lookup purposes, and must be uniqu
e across the set of mirrors.
         |
        <mirror>
          <id>mirrorId</id>
          <mirrorOf>repositoryId</mirrorOf>
          <name>Human Readable Name for this Mirror.</name>
          <url>http://my.repository.com/repo/path</url>
        </mirror>
         -->
        <mirror>
            <id>z_mirrors</id>
            <mirrorOf>*,!releases,!snapshots</mirrorOf>
            <url>http://dev.flyrise.cn:8081/repository/maven-public/</ur
l>
        </mirror>
        <mirror>
            <id>insecure-repo</id>
            <mirrorOf>external:http:*</mirrorOf>
            <url>http://dev.flyrise.cn:8081/repository/maven-public/</ur
l>
            <blocked>false</blocked>
        </mirror>
    </mirrors>

    <!-- profiles
     | This is a list of profiles which can be activated in a variety of
 ways, and which can modify
     | the build process. Profiles provided in the settings.xml are inte
nded to provide local machine-
     | specific paths and repository locations which allow the build to
work in the local environment.
     |
     | For example, if you have an integration testing plugin - like cac
tus - that needs to know where
```

```
      | your Tomcat instance is installed, you can provide a variable her
e such that the variable is
      | dereferenced during the build process to configure the cactus plu
gin.
      |
      | As noted above, profiles can be activated in a variety of ways. O
ne way - the activeProfiles
      | section of this document (settings.xml) - will be discussed later
. Another way essentially
      | relies on the detection of a system property, either matching a p
articular value for the property,
      | or merely testing its existence. Profiles can also be activated b
y JDK version prefix, where a
      | value of '1.4' might activate a profile when the build is execute
d on a JDK version of '1.4.2_07'.
      | Finally, the list of active profiles can be specified directly fr
om the command line.
      |
      | NOTE: For profiles defined in the settings.xml, you are restricte
d to specifying only artifact
      |       repositories, plugin repositories, and free-form properties
 to be used as configuration
      |       variables for plugins in the POM.
      |
      |-->
    <profiles>
        <!-- profile
         | Specifies a set of introductions to the build process, to be
activated using one or more of the
         | mechanisms described above. For inheritance purposes, and to
activate profiles via <activatedProfiles/>
         | or the command line, profiles have to have an ID that is uniq
ue.
         |
         | An encouraged best practice for profile identification is to
use a consistent naming convention
         | for profiles, such as 'env-dev', 'env-test', 'env-production'
, 'user-jdcasey', 'user-brett', etc.
         | This will make it more intuitive to understand what the set o
f introduced profiles is attempting
         | to accomplish, particularly when you only have a list of prof
ile id's for debug.
         |
         | This profile example uses the JDK version to trigger activati
on, and provides a JDK-specific repo.
        <profile>
          <id>jdk-1.4</id>
```

```
      <activation>
        <jdk>1.4</jdk>
      </activation>

      <repositories>
        <repository>
          <id>jdk14</id>
          <name>Repository for JDK 1.4 builds</name>
          <url>http://www.myhost.com/maven/jdk14</url>
          <layout>default</layout>
          <snapshotPolicy>always</snapshotPolicy>
        </repository>
      </repositories>
    </profile>
    -->

    <!--
  | Here is another profile, activated by the system property 'target
-env' with a value of 'dev',
  | which provides a specific path to the Tomcat instance. To use thi
s, your plugin configuration
  | might hypothetically look like:
  |
  | ...
  | <plugin>
  |   <groupId>org.myco.myplugins</groupId>
  |   <artifactId>myplugin</artifactId>
  |
  |   <configuration>
  |     <tomcatLocation>${tomcatPath}</tomcatLocation>
  |   </configuration>
  | </plugin>
  | ...
  |
  | NOTE: If you just wanted to inject this configuration whenever so
meone set 'target-env' to
  |       anything, you could just leave off the <value/> inside the
activation-property.
  |
  <profile>
    <id>env-dev</id>

    <activation>
      <property>
        <name>target-env</name>
        <value>dev</value>
      </property>
    </activation>
```

```
      <properties>
        <tomcatPath>/path/to/tomcat/instance</tomcatPath>
      </properties>
    </profile>
    -->
        <profile>
            <id>pai</id>
            <properties>
                <altSnapshotDeploymentRepository>snapshots::default::htt
p://dev.flyrise.cn:8081/repository/maven-snapshots/</altSnapshotDeployme
ntRepository>
                <altReleaseDeploymentRepository>releases::default::http:
//dev.flyrise.cn:8081/repository/maven-releases/</altReleaseDeploymentRe
pository>
            </properties>
            <repositories>
                <repository>
                    <id>releases</id>
                    <url>http://dev.flyrise.cn:8081/repository/maven-rel
eases/</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>false</enabled>
                    </snapshots>
                </repository>
                <repository>
                    <id>snapshots</id>
                    <url>http://dev.flyrise.cn:8081/repository/maven-sna
pshots/</url>
                    <releases>
                        <enabled>false</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</enabled>
                    </snapshots>
                </repository>
            </repositories>
        </profile>
    </profiles>

    <!-- activeProfiles
     | List of profiles that are active for all builds.
     |
    <activeProfiles>
      <activeProfile>alwaysActiveProfile</activeProfile>
```

```
        <activeProfile>anotherAlwaysActiveProfile</activeProfile>
      </activeProfiles>
      -->
      <activeProfiles>
          <activeProfile>pai</activeProfile>
      </activeProfiles>
  </settings>
```

3. IDE配置Maven:

前端开发

版本要求

- docker 镜像
  pai 在构建时使用的docker源镜像版本为 `node:14-alpine`
- node

```
$ node -v
v14.15.1
```

- npm

```
$ npm -v
6.14.8
```

移动端开发

后续补充

PAI工具

介绍

PAI 是一款针对开发团队设计的简单好用的开发辅助工具系统。主要体现在以下几个特点：

- 可以通过登录开发者账号，进行快速创建项目骨架，并根据表结构快速生存模板式代码，避免开发者花费大量重复的时间在工程搭建下。
- 封装了运行项目和执行单元测试的能力。
  运行项目即相当于mvn clean compile package 及 java -jar XXXX.jar。
  执行单元测试则是相当于 mvn clean compile test。
- 提供的Git功能，可根据项目名称clone、pull、push 对远程Git仓库进行操作；可直接在项目根目录下进行 init、add、commit对本进项目进行有效的版本管理。
- 对远端微服务的测试环境及正式环境的部署能力。

PAI开发的主要流程：创建工程 -> 代码生成 -> 单元测试 -> 工程运行 -> 代码托管 -> 环境部署 -> 信息查看。

> pai安装请看上一章节

操作示例

1. 打开管理员命令提示符窗口

2.输入pai

在输入PAI后，我们可以看到PAI工具的一些相关的介绍。

PAI工具的命令实际上可以分为两大类：

1. 针对PAI工具自身命令

- 基础命令

2. 针对实际开发的命令

- 开发调式
- 代码管理
- 部署管理

```
命令：
基础命令
  version               查看版本
  upgrade               更新升级
  changelog             更新日志
  login                 开发者登录
  password              修改密码

开发调试
  create                创建工程骨架
  code                  根据表结构生成代码
```

```
  widget                小组件管理
  form                  表单管理
  param                 项目骨架参数
  test                  工程单元测试
  run                   运行工程
  stop                  停止工程

代码管理（Git）
  init                  将当前目录初始化为本地Git仓库
  add                   将代码添加到本地Git仓库
  commit                提交代码到本地Git仓库
  clone                 从Git仓库克隆工程代码到本地
  push                  将本地工程代码推送到远端Git仓库
  pull                  从远端Git仓库拉取工程代码到本地
  releases              发布工程版本

部署管理（Kubernetes）
  services              查询Kubernetes服务列表
  pods                  查询Kubernetes应用列表
  logs                  读取Kubernetes应用日志
  deploy                将工程部署到Kubernetes
  delete                将工程从Kubernetes删除

镜像管理（Image）
  image                 Docker镜像管理
```

> 输入命令后都可以加 `-h` 使用帮助

例如：
输入：

```
pai create -h
```

返回:

```
创建工程骨架

语法：
pai create <命令> [参数]

命令：
  service               创建后端工程
  front                 创建前端工程
  micro                 创建APP小程序工程
  data                  创建数据中台工程
  docker                创建Docker工程
```

```
 protocol            创建物联协议工程

参数:
 -h,--help    使用帮助

示例:
创建后端工程
 pai create service -n pai-demo
 pai create service -n pai-demo -s cn.flyrise.demo
 pai create service -n pai-demo -p demo

创建前端工程
 pai create front -n pai-demo-ui
 pai create front -n pai-demo-ui -m node
 pai create front -n pai-demo -s cn.flyrise.demo

创建APP小程序工程
 pai create micro -n pai-demo
 pai create micro -n pai-demo -s cn.flyrise.demo

创建数据中台工程
 pai create data -n pai-demo
 pai create data -n pai-demo -s cn.flyrise.demo

创建Docker工程
 pai create docker -n pai-demo
 pai create docker -n pai-demo -s cn.flyrise.demo
```

> 命令的应用在后面章节有介绍，在此不详细说明

搭建工程

介绍

搭建工程开发环境有两种方式：

1. 创建新工程进行搭建
2. 拉取已有工程进行搭建

按实际情况选择

创建工程

通过pai工具生成我们后端工程的基本骨架，将一些基本的增删查代码通过表结构进行生成，节省了我们因搭建项目所花费的时间。

操作示例

> 前置条件：已经配置开发环境准备，请看以往章节

1. 先进行登录

```
pai login
```

2. 查看**pai**创建工程帮助

```
pai create -h
```

```
### 创建工程骨架

语法：
pai create <命令> [参数]

命令：
 service                 创建后端工程
 front                   创建前端工程
 micro                   创建APP小程序工程
 data                    创建数据中台工程
 docker                  创建Docker工程
```

```
 protocol              创建物联协议工程

参数：
 -h,--help    使用帮助

示例：
创建后端工程
 pai create service -n pai-demo
 pai create service -n pai-demo -s cn.flyrise.demo
 pai create service -n pai-demo -p demo

创建前端工程
 pai create front -n pai-demo-ui
 pai create front -n pai-demo-ui -m node
 pai create front -n pai-demo -s cn.flyrise.demo

创建APP小程序工程
 pai create micro -n pai-demo
 pai create micro -n pai-demo -s cn.flyrise.demo

创建数据中台工程
 pai create data -n pai-demo
 pai create data -n pai-demo -s cn.flyrise.demo

创建Docker工程
 pai create docker -n pai-demo
 pai create docker -n pai-demo -s cn.flyrise.demo
```

3. 创建后端工程

> 命名规范
> 后端：以 pai 开头， 如：pai-xxx
> 前端：以 pai 开头，前端以 ui 结尾， 如：pai-xxx-ui
> Docker：以 pai 开头，前端以 dk 结尾， 如：pai-xxxx-dk
> 数据端：以 pai-dp 开头， 如：pai-dp-xxxx

```
#注意工程将创建在当前目录下，目录不对请先更改位置再执行命令

#创建后端工程
pai create service
#创建后端工程 指定工程名称 pai-demo
pai create service -n pai-demo
#创建后端工程 指定工程名称 pai-demo 并绑定应用套件
pai create service -n pai-demo -s 套件标识
```

绑定应用套件并非强制性，也可通过开发者后台>应用套件>绑定工程

项目基本骨架结构

```
pai-xxxx
├─src
│  └─main
│     ├─java
│     │  └─com
│     │     └─example
│     │          └─demo
│     │              │ Application.java                    // 程序启动入口
│     │              ├─config                              // 全局配置
│     │              ├─controller                          // 接口控制器
│     │              ├─dao                                 // 数据持久层
│     │              ├─exception                           // 异常处理
│     │              │   ├─BusinessErrors.java            // 错误常量
│     │              │   └─DemoBizException.java           // 业务异常类
│     │              ├─model                               // 模型层
│     │              │  ├─po                               // 持久层数据模
型
│     │              │  └─vo                               // 业务层数据模
型
│     │              ├─service                             // 服务逻辑层接
口
│     │              │  └─impl                             // 服务逻辑层实
现
│     │              └─util                                // 通用工具
│     └─resources                                          // 应用配置
│         │ ├─application.yml                             // 应用参数
│         │ ├─bootstrap-dev.yml                           // 开发域参数
│         │ ├─bootstrap-test.yml                          // 测试域参数
│         │ ├─bootstrap.yml                               // 程序引导
│         │ └─logback-config.xml                          // 日志配置
│         ├─i18n                                          // 国际化配置
│         │  └─messages.properties                        // 默认配置
│         └─mapper                                        // Mapper XML
配置
├─.gitignore                                              // Git版本管理
配置
├─Deployment.yaml                                         // K8s部署Depl
oyment描述
├─Service.yaml                                            // K8s部署Serv
ice描述
├─Dockerfile                                              // Docker镜像打
包描述
└─pom.xml                                                 // Maven管理配
置
```

resourse目录下的配置文件说明

- logback-config.xml
  关于日志的一个配置文件。
- bootstrap-dev.yml和bootstrap-test.yml
  声明关于SpringCloud Config Server的一些信息。
- bootstrap.yml
  是应用程序的父上下文，加载优先于 application。
  主要声明了端口号和环境配置的一些信息。
- application.yml
  容器启动默认扫描的配置文件，同目录下优先级低于bootstrap配置文件。
  主要配置mybatis，swagger，security等。
- messages.properties
  关于异常错误信息的配置文件。

启动类

```
@EnablePaiResourceServer
@SpringCloudApplication
@EnablePaiFeignClients("cn.flyrise.*")
@EnableAspectJAutoProxy(exposeProxy = true)
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

注解说明

- @EnablePaiResourceServer
  意味着服务(就OAuth 2.0而言 – 资源服务器)需要访问令牌才能进行请求。
  在调用资源服务器之前，应通过OAuth 2.0客户端从授权服务器获取访问令牌。
  如果注释掉此注解，会走原生SpringSecurity，从而需要登陆，就算登陆成功，在调用资源服务器也可能面临被拒绝访问。

- @EnablePaiFeignClients
  启用feign客户端。
  将远端服务映射成为本地方法进行调用。
  扫描和注册feign客户端bean定义。

- @SpringCloudApplication
  启动类，用于启动容器

- @EnableAspectJAutoProxy
  启用AOP动态代理

拉取工程

操作示例

> 前提:用户已加入应用套件，应用套件>成员管理

1. 先进行登录

```
pai login
```

3. 拉取工程

```
pai clone -n 工程名
```

IDE导入工程

前提

IntelliJ IDEA已安装
已配置JDK环境变量

操作示例

1、创建空白项目

操作：打开IntelliJ IDEA -> New Project

创建一个空白项目

设置项目名称

2、配置

配置SDK
操作：菜单栏 File -> Project Structure

配置项目的Maven配置
操作：点击菜单栏File-> Preferences

> 这样设置的好处是不影响你其他项目的Maven配置

3、导入工程

操作：菜单栏 File -> Project Structure -> Modules

选择新创建或拉取的工程目录，一定要选择以Maven目录导入

项目导入成功

运行Application启动工程

> 分享
> 一项目都是有多个工程，如果有新工程只需要再次操作第三步引入工程

创建数据表

目标

通过本章节介绍可以让开发者通过数据库工具便捷管理目标工程对应的数据库表及数据。

> 注意：基于数据资源图表对应的数据表系标准规范生成，为了避免误操作导致数据异常暂不提供在线维护管理。

> 前置条件
>
> - 拥有开发者账号
> - 已完成工程创建或已加入工程成员

操作示例

1. 打开数据库工具

具体的操作步骤如下：
使用工程成员的用户（创建工程的人或工程成员）登录开发者后台。
路径：我的套件 -> 工程管理 -> 数据库管理。

> 注意：一个后端工程对应一个数据库

2. 创建表对象

具体的操作步骤如下：

路径：表对象 -> 新建。

定义表对象。

> 设计表对象，包含：字段、索引、关联的子表（用于生成主子表代码逻辑）。

3. 编辑数据

具体的操作步骤如下：

路径：表对象 -> 数据。

点击数据单元格编辑内容

> 说明：
> 在此页面可对数据进行新增、修改、删除及导出操作
> 数据为单条保存机制，如需批量修改请通过 查询 功能写SQL语句进行操作
> 导出数据会分两步进行：导出数据（先准备数据，选择导出格式、字段等） -> 下载文件

4. 查询

具体的操作步骤如下：
路径：表对象 -> 查询。

编辑SQL语句

> 说明：
> 在此页面可自行编写SQL语句进行查询，维护数据等操作，但 `不能执行建库建表等DDL操作`
> 批量执行SQL时需要在SQL语句后加上分号做为结束符，每段语句会单显示一个结果页
> 可选择部分SQL内容后通过 运行选中 来执行一段自选SQL
> 消息页可查看执行异常或信息

代码生成

目标

通过本章节介绍可以让开发者通过数据表生成CRUD代码

操作示例

1. 首先切换到工程目录下

先进行登录

```
pai login
```

2. 根据提示帮助，选择语句。

查看生成代码帮助

```
pai code -h

根据表结构生成代码
语法:
pai code [参数]
参数:
 -h,--help              使用帮助
 -t,--table <arg>       表名
 -n,--name <arg>        工程名称(默认当前目录名称)
 -o,--output <arg>      文件保存绝对路径(默认当前路径)

示例:
 pai code -t user
 pai code -t user -n pai-demo
```

生成代码

```
pai code -n pai-demo-one -t cars
```

3. 到这里代码已经自动生成了

我们使用idea打开该项目，打开src目录，相比刚创建的工程，部分增加的代码都是根据你的表结构所自动生成的。

以下是项目创建好后大致的src目录结构：

```
├─main
│  ├─java
│  │  └─cn
│  │      └─flyrise
│  │          └─pai
│  │              └─demoone
│  │                  │ Application.java
│  │                  ├─config
│  │                  ├─controller
│  │                  │        CarsController.java
│  │                  ├─dao
│  │                  │        CarsMapper.java
│  │                  ├─exception
│  │                  │        BusinessErrors.java
│  │                  │        DemoOneBizException.java
│  │                  ├─model
│  │                  │  ├─po
│  │                  │  │     CarsPO.java
│  │                  │  └─vo
│  │                  │          CarsVO.java
│  │                  │
│  │                  └─service
│  │                      │ ICarsService.java
│  │                      └─impl
│  │                           CarsServiceImpl.java
│  │
│  └─resources
│      └─mapper
│              CarsDao.xml
```

代码结构解析

model 领域模型层

`model` 包分为 `po` 、 `vo` 两个包

- `po` 目录来对应数据库的表的对象
- `vo` 目录是用于对外暴露的对象

例如请求的对象放vo而不要放po对象，返回值同样。

下面用一个简单的例子说明，例如一个请假表中有个审核字段 `state` {0 发起 1同意 2拒绝}，但是我们在写接口的时候并不会传递该值，因为发起的时候该状态就是0，而调用了同意的接口该状态才会变为1，拒绝依此类推。这就意味着我们写发起接口的时候不会有 `state` 该字段，所以我们用VO做为参数的时候可以吧 `state` 字段给去掉。

编写 model 实体

PO Model 规范

- 一个 `PO` 对应一张数据表
- 使用 `@TableId('id')` 注解标注主键
- 实体继承 BaseEntity，BaseEntity 包含（create_by:创建人,create_time：创建时间，update_by:更新人,update_time:更新时间）
- 使用 `@Table (javax.persistence.Table)` 映射表名
- 使用 `@TableField (javax.persistence.TableField)` 映射字段名
- 属性上写注释/** 注释 */
- 非数据库字段使用 `@Transient` 注解标注，如果页面用到的非数据库字段比较多，建议使用 DTO 封装数据。
- 所有属性均为private属性，每一个属性需要生成对应的 getter、setter 方法。
- 不使用基本类型，全部使用基本类型的包装类，如 `Long` 对应数据库中的 `INTEGER`，而不是使用 `long`。
- 数字类型主键统一采用 `Long`。金额、数量 等精度严格浮点类型采用 `BigDecimal` (注意：BigDecimal 在计算、比较方面的特殊性)

VO Model 规范

- 使用 `@ApiModel` 注解说明实体含义，在 Swagger 文档上就可以看到实体说明。
- 实体字段使用 `@ApiModelProperty` 说明字段含义，在 Swagger 文档上可以看到字段说明。
- 所有属性均为private属性，每一个属性需要生成对应的 getter、setter 方法。
- 不使用基本类型，全部使用基本类型的包装类，如 `Long` 对应数据库中的 `INTEGER`，而不是使用 `long`。
- 数字类型主键统一采用 `Long`。金额、数量 等精度严格浮点类型采用 `BigDecimal` (注意：BigDecimal 在计算、比较方面的特殊性)

基础设施层（dao,mapper）

Mapper 接口类

- mapper 接口类即为传统意义上的 DAO，但与 interface 不同，mapper 本身就是对数据访问的具体实现，所以属于供应方的服务实现层。创建在 项目模块 的 `xxx.dao` 包下。
- 每一个 mapper 接口类封装了对数据库表的操作，每一个 mapper 对应一个 实体 类，命名为 实体类名尾缀替换为 Mapper 。
- 基础的 CRUD 操作不需要再次实现，通过继承 `BaseMapper< PO >` 类实现。其中 PO 为 对应 实体 的泛型。

- 复杂的数据库操作需要定义具体的接口方法

mapper.xml

- Mapper的xml文件 是数据库的的具体映射，与 Mapper 接口同级，创建在 项目模块 resources 目录下的 mapper 目录下。
- Mapper的xml文件，与 Mapper 接口对应。所以命名与 Mapper 接口类相同。
- Mapper的xml文件非必须，由于继承BaseMapper类后基本的 CRUD 不需要进行配置，所以只有CRUD操作时不需要创建对应的 xml 文件。
- 对于自定义的数据库方法，需要创建对应的 mapper.xml 文件。
- Mapper的xml 中的操作 id 对应 Mapper 接口类的方法名。

应用层（service）

service介绍

`service` 调用领域对象或服务来解决问题，应用层Service主要有以下特性：

- 负责事务处理，所以事务的注解可以在这一层的service中使用。
- 只处理非业务逻辑，重点是调度业务处理流程。业务逻辑处理一定要放在领域层处理。
- 不做单元测试，只做验收测试。

service 实现类介绍

`ServiceImpl` 实现类

- Service 接口的具体实现通过服务实现层提供，所以属于供应方的服务实现层。创建在项目模块的 `xxx.service.impl` 包下。
- `@Service` 标识为Service实现类
- `@Transactional` 声明式事务管理

API 展现层

Controller介绍

- `Controller` 负责对 `Model` 的处理，创建在项目模块的 `xxx.controller` 包下。
- 需要通过 `@Controller` 指定该类为一个 `Controller` 类。
- 使用Restful风格，使用 `Post` 、`Delete` 、`Put` 、`Get` ，使用不同的方法对资源进行操作，分别对应 添加、删除、修改、查询

Controller 类相关标签

- `@RestController` 是一个组合注解，是 `@ResponseBody` 和 `@Controller` 的组合。
- `@ApiOperation` 显示在swagger ui上的接口注释，同时与该接口对应的权限表中的描述字段对应

- `@Api(tags = "打卡API")`，在类上对类进行说明，显示在 Swagger 文档上
- `@ApiImplicitParams`，在方法上对方法参数进行说明，显示在 Swagger 文档上
- `@ApiParam` 在方法参数上对参数进行说明，显示在 Swagger 文档上
- `@GetMapping` 是 `@RequestMapping(mathod = RequestMethod.GET)` 的缩写，处理get请求
- `@PostMapping` 处理post请求
- `@PutMapping` 处理put请求
- `@DeleteMapping` 处理delete请求

建议URI规范

每个URI代表一种资源或者资源集合，因此，建议只包含名词，不包含动词。

- 反例：http://api.example.com/get-all-employees
- 正例：http://api.example.com/employees

那么，如何告诉服务器端我们需要进行什么样的操作？CRUD？
答案是由HTTP动词表示。

- GET（SELECT）：从服务器取出资源（一项或多项）。
- POST（CREATE）：在服务器新建一个资源。
- PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）。
- PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）。
- DELETE（DELETE）：从服务器删除资源。

举个例子

- GET /zoos：列出所有动物园
- POST /zoos：新建一个动物园
- GET /zoos/ID：获取某个指定动物园的信息
- PUT /zoos/ID：更新某个指定动物园的信息（提供该动物园的全部信息）
- PATCH /zoos/ID：更新某个指定动物园的信息（提供该动物园的部分信息）
- DELETE /zoos/ID：删除某个动物园
- GET /zoos/ID/animals：列出某个指定动物园的所有动物
- DELETE /zoos/ID/animals/ID：删除某个指定动物园的指定动物

Swagger功能

目标

通过本章节介绍可以让开发者通过Swagger对接口在线调试

> 前置条件
>
> - 在本地通过Application启动工程

操作示例

进入**Swagger**功能

工程默认8080端口
工程名字：pai-工程名

- 本地默认入口：http://127.0.0.1:8080/doc.html
- 远程部署入口：https://pai.flyrise.cn/工程名-api/doc.html

界面：

通过工程 resources/application.yml 配置swagger

```
swagger:
  apiTitle: 凌云中台 - 文章管理系统
  description: 凌云中台 - 文章管理系统
```

```
  contactName: 吴先生
  contactEmail: wwc@flyrise.cn
  version: 1.0
  pathMapping: /demo-article-api  #在本地环境需要去掉，方便调试
```

`pathMapping: /demo-article-api` 在本地环境运行需要注释掉，否则会影响到接口调试

Authorize

进行授权

默认生成代码是需要登录授权才能访问的，否则调用接口提示：

```
{
  "msg": "401 UNAUTHORIZED 认证失败： 14b69dc5-48e3-4737-8a51-b325c5228d22",
  "code": "401",
  "time": 1647852284953
}
```

测试账号：

- username： 13012345678
- password： @LinkCloud123
- clientId： web
- clientSecret： 123456

接口调用

1. 确保已授权

如Authorize成功这里还是空的，请关闭接口选项卡，重新打开调试

2. 填写参数

3. 接口返回结果

开发学习案例

需求

新闻管理模块

管理端：

- 新闻管理
- 分类管理
- 标签管理
- 评论管理

前端：

- 新闻展示
- 评论

表设计

news

代码开发

权限管理

目标

通过本章节介绍可以对接口等业务做权限限制，对数据做数据权限限制

业务权限

配置

操作：应用套件 -> 权限管理

1. 新增权限
2. 新增用户组
3. 权限与用户组进行绑定

代码使用

在接口上可以使用@PreAuthorize(“@pms.hasSuiteRoles(‘’)”) 来进行接口鉴权。
参数描述：
套件**code**：cn.flyrise.demoone
角色标识：cars
权限标识：add
@PreAuthorize(“@pms.hasSuiteRoles(‘cn.flyrise.demo:in_work:add’)”)参数是由套件 + 角色 + 权限组成，也允许只控制到角色，即@pms.hasSuiteRoles(‘cn.flyrise.demo:in_work’)

```
@ApiOperation("添加单条记录")
@PostMapping
@PreAuthorize("@pms.hasSuiteRoles('cn.flyrise.demoone:cars:add')")
public Reply<CarsVO> insert(@RequestBody @Valid CarsVO vo) {
        return Reply.success(this.carsService.add(vo));
    }
```

无权限返回信息

```
{
  "code": "999",
  "data": null,
  "time": 1648002813068,
  "msg": "未知错误",
```

```
  "annex": {
    "applicationName": "pai-demo-one",
    "ExceptionClassName": "org.springframework.security.access.AccessDen
iedException",
    "errorMsg": "不允许访问"
  },
  "success": false
}
```


数据权限（后续更新）

配置

操作：应用套件 -> 权限管理 -> 数据权限 -> 新增权限

数据字典

目标

通过本章节介绍可以对接口等业务做权限限制，对数据做数据权限限制

系统字典

系统数据字典在运营后台维护，配置中心存储，根据开发需要不定期添加、修改。
系统数据字典都是普通数据字典，只读，租户不可编辑

套件自定义字典
普通数据、只读 数据字典
由开发者自己定义，租户安装套件之后不可编辑字典数据，
套件更新时会更新字典数据

普通数据、编辑 数据字典
由开发者先行定义，租户安装套件之后可以编辑字典数据，
套件更新时不会覆盖租户数据(若租户未修改字典数据，还是保留第一次安装时的租户数据)

后端配置
入口：开发中心-应用套件-套件详情-能力-数据字典

配置完成之后请于沙箱入口安装或更新套件，否则前端使用会找不到资料

前端使用
示例入口：http://dev.k8s.flyrise.cn/pai-ui/#/docs/dict

参数管理

目标

通过本章节介绍可以预定义套件中工程要用到的参数声明，利于在安装套件时进行配置实际的参数

说明

为了支持项目独立性，该功能会实现一个通用的适配器来适配云与本地的数据。
用户在访问微应用的时候，app、web 都有需要获取到应用的企业配置的基本属性配置。

开发者：
开发者创建项目 -> 配置参数

企业管理员：
安装套件 -> 配置套件参数

> 开发者配置的参数是默认配置，最终使用是以企业管理员配置为准

开发者配置

操作: 应用套件 -> 参数管理

新增

企业管理员配置

操作：企业控制台 -> 应用 -> 配置 -> 参数配置

使用

前端工程使用

需要在 `Application.java` 启动类添加 `@EnablePaiConfig` 注解
在Swagger会出现应用参数的接口

这里有两个关键接口

1、通过套件Code获取应用的参数配置

请求参数

- appCode 套件标识
- entId 企业id

返回：


```
{
  "code": "200",
  "data": {
    "items": [
      {
        "itemName": "微信appid",
        "itemType": "text",
        "itemValue": "wx825751f92e719951",
        "itemKey": "wx-appid",
        "itemDefaultValue": "wx825751f92e719951",
        "itemFormat": null
      },
      {
        "itemName": "微信商户mchId",
        "itemType": "text",
        "itemValue": "123",
        "itemKey": "wx-mchId",
        "itemDefaultValue": null,
        "itemFormat": null
      }
```

```
      ],
      "roles": [
        "cars"
      ],
      "perms": [
        "cars:delete",
        "cars:update",
        "cars:add"
      ]
    },
    "time": 1648016326774,
    "msg": "操作成功",
    "annex": null,
    "success": true
}
```

2、通过套件Code获取角色权限

请求参数

- appCode 套件标识
- entId 企业id
- uid 用户id

返回：

```
{
    "code": "200",
    "data": {
      "roles": [
        "cars"
      ],
      "perms": [
        "cars:add",
        "cars:update",
        "cars:delete"
      ]
    },
    "time": 1648018506323,
    "msg": "操作成功",
    "annex": null,
    "success": true
}
```

> 通过这个接口前端可以对用户实现权限控制

后端代码调用

IConfigAppService.java 提供的接口方法去获取参数配置，使用以下

```
    @Resource
    private IConfigAppService configAppService;
```

包含的方法

```
  @ApiOperation("获取用户在这个套件所有的权限")
  @GetMapping({"/sysFormPermission/v1/findAppPerms"})
  Reply<UserRolePermission> findAppPerms(@RequestParam("suiteId") String
 var1, @RequestParam("uid") String var2, @RequestParam("entId") String v
ar3, @RequestHeader("from") String var4);

  @ApiOperation("获取用户在这个套件所有的权限")
  @GetMapping({"/sysFormPermission/v1/findAppPermsByCode"})
  Reply<UserRolePermission> findAppPermsByCode(@RequestParam("suiteCode"
) String var1, @RequestParam("uid") String var2, @RequestParam("entId")
String var3, @RequestHeader("from") String var4);

  @ApiOperation("获取用户在这个企业所有的套件权限")
  @GetMapping({"/sysFormPermission/v1/findAppPermsByEnt"})
  Reply<UserRolePermission> findAppPermsByEnt(@RequestParam("entId") Str
ing var1, @RequestParam("uid") String var2, @RequestHeader("from") Strin
g var3);

  @ApiOperation("获取员工在这个套件所有的权限")
  @GetMapping({"/sysFormPermission/v1/findAppPermsByStaffAndCode"})
  Reply<UserRolePermission> findAppPermsByStaffAndCode(@RequestParam("su
iteCode") String var1, @RequestParam("entId") String var2, @RequestParam
("staffId") String var3, @RequestHeader("from") String var4);

  @ApiOperation("获取员工在这个企业所有的套件权限")
  @GetMapping({"/sysFormPermission/v1/findAppPermsByStaff"})
  Reply<UserRolePermission> findAppPermsByStaff(@RequestParam("suiteId")
 String var1, @RequestParam("entId") String var2, @RequestParam("staffId
") String var3, @RequestHeader("from") String var4);

  @ApiOperation("获取应用的配置-权限与配置-appCode")
  @GetMapping({"/appItemConfig/v1/getConfigByCode"})
  Reply<SysItemConfig> getConfigByCode(@RequestParam("appCode") String v
ar1, @RequestParam("entId") String var2, @RequestHeader("from") String v
ar3);

  @ApiOperation("获取应用的配置-权限与配置-appId")
  @GetMapping({"/appItemConfig/v1/getConfigById"})
```

```
  Reply<SysItemConfig> getConfigById(@RequestParam("appId") String var1,
 @RequestParam("entId") String var2, @RequestHeader("from") String var3)
;

  @ApiOperation("卸载应用的配置")
  @PostMapping({"/suite/v1/uninstall"})
  Reply uninstall(@RequestBody UninstallSuiteRequest var1, @RequestHeade
r("from") String var2);
```

调用例子：

```
 Reply<SysItemConfig> configByCode = configAppService
            .getConfigByCode("cn.flyrise.demoone", "1501473186812678145"
, "Y");
```

返回 Reply对象：

```
{"code":"200","data":{"items":[{"itemDefaultValue":"wx825751f92e719951",
"itemKey":"wx-appid","itemName":"微信appid","itemType":"text","itemValue
":"wx825751f92e719951"},{"itemKey":"wx-mchId","itemName":"微信商户mchId",
"itemType":"text","itemValue":"123"}],"perms":["cars:delete","cars:updat
e","cars:add"],"roles":["cars"]},"msg":"操作成功","success":true,"time":1
648022106577}
```

自定义流水号

目标

通过本章节介绍可以配置流水号并在业务中使用。

说明

各个行业的各种业务都有一个系统自动编码的号码,用来记录处理的顺序的号,就是流水号
每个业务流水号生成规则不相同，但有一定的相似性，同时要保证流水号的唯一性，连续性，还有高并发下的性能问题。
为了将来各业务更好的整合以及用户对流水号使用要求的提高，所以提供流水号能力，统一管理流水号的生成和使用。

使用

配置流水号

1. 操作：应用套件 -> 流水号服务

2. 新增模板

3. 配置规则

> 新增的流水号需要在水箱环境执行一下重新安装应用套件，否则调用不了

应用流水号

前端工程获取

代码引用获取

1. 引入依赖pom.xml

```
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-common-sn</artifactId>
</dependency>
```

2. 调用接口

```
@Resource
private SnService snService;

//参数1 流水号标识
//参数2  企业id
String car_no = snService.obtainSn("car_no", "1501473186812678145").getD
ata();

//参数1 流水号标识
//参数2  企业id
//参数3  园区id
String car_no1 = snService.obtainSn("car_no", "1501473186812678145", "15
01473186812678144").getData();
```

后端工程使用

后续更新

任务调度

> 感谢于以恒提供本节教程

目标

通过本章节介绍可以编写定时任务方法，并运行

说明

XXL-JOB是一个中心化的任务调度引擎，由各个执行器启动时主动向调度中心进行任务注册。

操作

1. pom.xml引入依赖

```
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-common-job</artifactId>
</dependency>
```

2. 启用执行器配置

@EnablePaiJobExecutor 用于启用执行器配置。

```
@SpringBootApplication
@EnablePaiJobExecutor
public class SpringBootDemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootDemoApplication.class, args);
    }

}
```

3. 编写任务逻辑

```
@Component
public class TestJob {

    @XxlJob("test1")
    @XxlJobTask(desc = "测试任务1，1分钟执行一次", cron = "0 0/1 * * * ?",
author = "Joe")
    public ReturnT<String> test1(String param) {
        System.out.println("test1 >>> no param");
        return ReturnT.SUCCESS;
    }

    @XxlJob("test2")
    @XxlJobTask(desc = "测试任务2，5分钟执行一次", cron = "0 0/5 * * * ?",
author = "Joe", param = "{\"name\":\"David\"}", childs = "test1")
    public ReturnT<String> test2(String param) {
        System.out.println("test2 >>>" + param);
        return ReturnT.SUCCESS;
    }

    @XxlJob("test3")
    @XxlJobTask(desc = "测试任务3，5分钟执行一次", cron = "0 0/5 * * * ?",
author = "Joe", param = "{\"name\":\"David\"}", childs = {"test1", "test
2"})
    public ReturnT<String> test3(String param) {
        System.out.println("test3 >>>" + param);
        return ReturnT.SUCCESS;
    }
}
```

4. 增加配置

放到Nacos配置

xxl.job.executor.appname 和 xxl.job.executor.title 必填 且去掉注释

```
xxl:
  job:
    admin:
      #本地建议把addresses清空，不注册到调度中心，云端调度中心也不能调度到本地
      #此地址是云端服务注册地址，本地使用https://pai.flyrise.cn/xxl-job-admin
注册执行器，云端不要使用域名，可能出现在同一节点访问不通
      addresses: http://xxl-job-admin:8080/xxl-job-admin
    accessToken:
    executor:
      appname: #  一般为工程代码
      title: # 执行器名称 可以是工程中文名
      address:
```

```
        ip:
        port: 9999
        logpath: /data/applogs/xxl-job/jobhandler
        logretentiondays: 30
```

5. 修改Dockerfile

EXPOSE 增加9999端口

```
EXPOSE 8080 9999
```

6. 本地调试

本地仅支持注册执行器和任务(addresses需配成域名)
注册的注解使用只要按规范使用就一定会注册上去，可以看日志有没有提示注册成功

c.f.job.config.JobHandlerRegistrar : >>>>>>>>>>> 执行器pai-XXXX注册成功，共发现n个任务

- 本地测试方式1：使用junit
- 本地测试方式2：自定义一个Controller去调用, 并配置免鉴权白名单，因为调度器是不具备获取用户信息的

7.进入XXLJOB任务调试中心

进入系统可查看任务执行情况，执行日志等
账号需要研发提供
地址：https://pai.flyrise.cn/xxl-job-admin

备注

1、本地开发环境不建议把任务注册到任务调度中心，可以通过配置设定。
2、如果同一工程在不同地方运行，则按第一个注册到任务调度中心进行执行。

3、工程没停止会一直调度。工程停止过一会儿任务调度中心会在执行器列表删除任务

审批流程

> 感谢于以恒提供本节教程

一、准备工作

- 创建园区套件
- 创建后端工程
- 创建前端工程

二、 操作流程

1. 进入应用套件

- 进入开发者后台->应用套件->创建的园区套件

2. 进入数据库管理

- 进入工程管理->创建的后端工程->数据库管理

3. 添加请假表

4. 新建请假表单

- 进入表单建模->新建请假表单

5. 编辑表单

6. 生成表单

- 通过数据库表字段生成表单页面

7. 保存并导出vue文件

8. 修改表单代码

- 打开前端工程修改表单代码
- 查找目录：**static/form**
- copy **form-test**文件夹并修改名称为**form-leave**
- 复制表单id 至**meta.json** 文件中的 **formId**
- 打开导出的vue文件，复制内容覆盖进 **form-leave/web/index.vue** 文件里面
- 修改**static/data-form.json**文件，将name改为**form-leave**

9. 编译及发布表单

- 新建终端
- 进入到我的前端项目路径
- 执行命令 **npm run build:form** 编译表单
- 执行命令 **pai form deploy -n form-leave** 部署表单

```
yyh@192 pai-yyh-park-ui % npm run build:form-web
> pai-yyh-park-ui@1.0.0 build:form-web /usr/local/web-projects/pai/pai-y
```

```
yh-park-ui
> cross-env BUILD_TYPE=form CLIENT_TYPE=web node build-widget/index.js
#------------- PAI-FORM ------------#
 Building  for pai-form-web...
yyh@yuyihengdeMacBook-Pro pai-yyh-park-ui % pai form deploy -n form-leav
e
Found /usr/local/pai-cli/.pai/wrapper/pai-wrapper.jar
表单form-leave发布成功
yyh@yuyihengdeMacBook-Pro pai-yyh-park-ui %
```

10. 添加请假流程

- 点击流程服务并添加请假流程

11. 编辑流程

12. 设计流程并保存

- 设计流程节点并选择对应表单

13. 安装套件

- 在开发沙箱中安装套件

14. 登陆控制台搜索套件

15. 配置流程节点

- 进入套件配置流程节点执行信息

16. 发布流程

- 保存后发布流程（需要刷新页面）

17. 进入流程中心

- 进入工作台打开我的事项->智慧审批->流程中心->创建的套件

18. 发起请假流程

三、错误解决

- 执行npm run build:form报错，提示如下需降低npm版本至6，执行命令npm install npm@6.14 -g 降低版本

```
#------------- PAI-FORM -------------#

undefined:1
undefined
^

SyntaxError: Unexpected token u in JSON at position 0
    at JSON.parse (<anonymous>)
    at Object.<anonymous> (/usr/local/web-projects/pai/pai-yyh-park-ui/node_modules/@flyriselink/pai-build-widget/build-widget/webpack-pai-form-conf.js:22:28)
```

四、开放接口

- 预计规划 通过流程标识获取当前任务的状态。
- 预计规划 办理后通过消息传递任务进度
- 预计规划 一个节点一个表单
- 预计规划 子流程

  …

五、流程消息订阅

- 消息订阅参考订阅配置
  消息订阅功能配置

- 引入依赖pom.xml

```
        <dependency>
            <groupId>cn.flyrise</groupId>
            <artifactId>pai-common-mq</artifactId>
            <version>1.6.0</version>
        </dependency>
```

- 代码

```
@Service
public class PulsarConsumer {

    @PaiMqListener (suiteCode = "zhsp",topic = "flow_change")
    public void consumer3(PulsarMessage<String> string) {
        System.out.println("PulsarConsumer Listener3: !!! msg:"+stri
ng.getValue());
    }
}
```

六、参考资源

- 凌云中台文档->开放能力->流程能力
- 凌云中台文档->开发指南->流程开发->web表单
- 凌云中台文档->培训讲解->表单建模及应用
- 凌云中台文档->培训讲解->流程开发及应用

规则引擎

> 感谢于以恒提供本节教程

一、注册规则

1. 在pom.xml文件中添加规则依赖

```
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-common-rules</artifactId>
    <version>1.7.3</version>
</dependency>
```

2. 通过实现Rule接口注册规则

```
/**
 * 注册规则
 */
public class PointRule implements Rule {
    @Override
    public String getProjectName() {
        //返回工程名称
        return "pai-yyh-park";
    }

    @Override
    public String getRuleName() {
        //返回规则名称
        return "积分规则";
    }
}
```

3. 创建数据源

```
package cn.flyrise.pai.yyhpark.model.vo;

import cn.flyrise.common.rules.annotation.RuleModel;
import cn.flyrise.pai.yyhpark.rule.PointRule;
```

```
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.io.Serializable;

/**
 * 积分规则决策数据源
 */
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@RuleModel(rules = {PointRule.class}, desc = "积分明细信息")
public class PointRuleVO implements Serializable {
    @ApiModelProperty("当天是否第一次签到 0:不是 1: 是")
    private Integer dayFirstSignScope;
    @ApiModelProperty("连续签到天数")
    private Integer continuousSignDays;
    @ApiModelProperty("连续签到得分")
    private Integer continuousSignPoints;
    @ApiModelProperty("每天签到得分")
    private Integer dayFirstSignPoint;
    @ApiModelProperty("目前签到总分")
    private Integer totalPoint;
}
```

4. 启动项目

启动项目控制台提示以下内容表示规则注册成功

```
15:10:38.342 [main] INFO  c.f.c.r.c.RuleRegister -
 [afterSingletonsInstantiated,51] - >>>>>>>>>>> 规则注册成功，共发现1个规则
```

5. 规则管理

登陆开发者平台进入应用套件->>我的套件->>规则管理

6. 查看规则和数据源

二、配置规则

1. 规则记录列表

2. 添加规则

3. 添加条件

- 添加条件

- 条件配置

- 添加事件-赋值

- 赋值1

- 赋值2

- 配置好的条件

- 规则调试

- 保存条件

三、代码调用

代码执行规则

```
package cn.flyrise.pai.yyhpark.rule;

import cn.flyrise.common.core.domain.Reply;
import cn.flyrise.common.rules.service.RuleService;
import cn.flyrise.common.rules.vo.business.BusinessRuleVO;
import cn.flyrise.pai.yyhpark.model.vo.PointRuleVO;
import com.alibaba.fastjson.JSON;
```

```
import com.alibaba.fastjson.JSONObject;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;


@Api(tags = "规则测试controller")
@RestController
@RequestMapping("ruleTest")
public class PointRuleTestController {

    @Resource
    private RuleService ruleService;


    @ApiOperation("规则测试")
    @PostMapping("/test")
    public Reply test() {
        PointRuleVO ruleVO = new PointRuleVO();
        ruleVO.setTotalPoint(1);
        ruleVO.setContinuousSignDays(7);
        ruleVO.setDayFirstSignPoint(0);
        ruleVO.setContinuousSignPoints(0);
        ruleVO.setDayFirstSignScope(1);

        //执行规则得到返回值
        //套件是入驻企业安装时使用这个接口
        //Reply<BusinessRuleVO> reply = ruleService.execute(new TestRule
(), userInfoVO);
        //执行规则得到返回值
        //套件是运营企业安装时使用这个接口，parkId前端切换园区产生
        Reply<BusinessRuleVO> reply =
                ruleService.execute(new PointRule(), null, ruleVO);
        if (reply.isSuccess()) {
            //继续业务逻辑
            BusinessRuleVO businessRuleVO = reply.getData();
            JSONObject jsonObject = businessRuleVO.getObjectMap().get(ru
leVO.getClass().getSimpleName());
            ruleVO = JSONObject.toJavaObject(jsonObject, PointRuleVO.cla
ss);
            System.out.println(JSON.toJSONString(ruleVO));
        } else {
            //异常处理
        }
```

```
            return reply;
        }


    }
```

四、支持函数

1. 字符串函数

```
(1)charAt函数：返回字符串中指定的某个字符。
(2)indexOf函数：返回字符串中第一个查找到的下标index，从左边开始查找。
(3)lastIndexOf函数：返回字符串中第一个查找到的下标index，从右边开始查找。
(4)length函数：返回字符串的长度。(不用带括号)
(5)substr函数：返回字符串从指定位置开始，截取几个字符。
(6)substring函数：返回字符串中指定的几个字符。
(7)trim函数：清除字符串中的空格。
(8)toLowerCase函数：将字符串转换为小写。
(9)toUpperCase函数：将字符串转换为大写。
(10)endsWith函数：返回字符串是否以某段结尾。
(11)contains函数：返回字符串是否包含某段。
```

实现方式：利用java脚本引擎ScriptEngineManager，执行js内置的字符串函数
使用方式：编辑器输入字符串类型的值.会弹出提示

2. 日期函数

```
(1)getDate函数：返回日期的“日”部分，值为1～31
(2)getDay函数：返回星期几，值为0～6，其中0表示星期日，1表示星期一，…，6表示星期六
(3)getHours函数：返回日期的“小时”部分，值为0～23。
(4)getMinutes函数：返回日期的“分钟”部分，值为0～59。
(5)getMonth函数：返回日期的“月”部分，值为0～11。其中0表示1月，2表示3月，…，11表示12月。
(6)getSeconds函数：返回日期的“秒”部分，值为0～59。
(7)getTime函数：返回系统时间。
(8)getTimezoneOffset函数：返回此地区的时差(当地时间与GMT格林威治标准时间的地区时差)，单位为分钟。
(9)getYear函数：返回日期的“年”部分。返回值以1900年为基数，例如1999年为99。
(10)parse函数：返回从1970年1月1日零时整算起的毫秒数(当地时间)。
(11)setDate函数：设定日期的“日”部分，值为0～31。
(12)setHours函数：设定日期的“小时”部分，值为0～23。
(13)setMinutes函数：设定日期的“分钟”部分，值为0～59。
(14)setMonth函数：设定日期的“月”部分，值为0～11。其中0表示1月，…，11表示12月。
(15)setSeconds函数：设定日期的“秒”部分，值为0～59。
(16)setTime函数：设定时间。时间数值为1970年1月1日零时整算起的毫秒数。
(17)setYear函数：设定日期的“年”部分。
(18)toGMTString函数：转换日期成为字符串，为GMT格林威治标准时间。
```

实现方式：利用java脚本引擎ScriptEngineManager，执行js内置的日期函数
使用方式：编辑器输入日期类型的值.会弹出提示

3. 数学函数

(1)abs函数：即Math.abs(以下同)，返回一个数字的绝对值。
(2)acos函数：返回一个数字的反余弦值，结果为0～π弧度(radians)。
(3)asin函数：返回一个数字的反正弦值，结果为-π/2～π/2弧度。
(4)atan函数：返回一个数字的反正切值，结果为-π/2～π/2弧度。
(5)atan2函数：返回一个坐标的极坐标角度值。
(6)ceil函数：返回一个数字的最小整数值(大于或等于)。
(7)cos函数：返回一个数字的余弦值，结果为-1～1。
(8)exp函数：返回e(自然对数)的乘方值。
(9)floor函数：返回一个数字的最大整数值(小于或等于)。
(10)log函数：自然对数函数，返回一个数字的自然对数(e)值。
(11)max函数：返回两个数的最大值。
(12)min函数：返回两个数的最小值。
(13)pow函数：返回一个数字的乘方值。
(14)random函数：返回一个0～1的随机数值。
(15)round函数：返回一个数字的四舍五入值，类型是整数。
(16)sin函数：返回一个数字的正弦值，结果为-1～1。
(17)sqrt函数：返回一个数字的平方根值。
(18)tan函数：返回一个数字的正切值。

实现方式：利用java脚本引擎ScriptEngineManager，执行js内置的数学函数
使用方式：编辑器输入Math.会弹出提示

4. 数组或集合函数

(1)**join**函数：转换并连接数组中的所有元素为一个字符串。
(2)**length**函数：数组的长度(不用带括号)。
(3)[**index**]：根据数组小标**index**获取元素。
(4)**sort**函数：将数组元素从小到大排序。
(5)includes函数：判断数组是否包含某元素

实现方式：利用java脚本引擎ScriptEngineManager，执行js内置的数组函数
使用方式：编辑器数组或集合类型的值.会弹出提示

5. 自定义统计函数

(1) max函数

返回给定数组或数组元素某属性的最大值。

请求参数

名称	类型	描述	是否必需	备注
array	Array<>	统计最小值的数组	是	Array元素类型的可以是整型、浮点、时间、字符串、布尔和对象类型，当元素是对象类型时需指定key
key	字符串	对象类型元素的属性	否	Array元素是对象类型时必须，且该属性的类型是整型、浮点、时间、字符串、布尔。

(2) min函数

返回给定数组或数组元素某属性的最小值。

请求参数

名称	类型	描述	是否必需	备注
array	Array<>	统计最小值的数组	是	Array元素类型的可以是整型、浮点、时间、字符串、布尔和对象类型，当元素是对象类型时需指定key
key	字符串	对象类型元素的属性	否	Array元素是对象类型时必须，且该属性的类型是整型、浮点、时间、字符串、布尔。

(3) sum函数

返回给定数组或数组元素某属性的和值。

请求参数

名称	类型	描述	是否必需	备注
array	Array<>	统计和值的数组	是	Array元素类型的可以是整型、浮点和对象类型，当元素是对象类型时需指定key
key	字符串	对象类型元素的属性	否	Array元素是对象类型时必须，且该属性的类型是整型、浮点。

(4) average函数

返回给定数组或数组元素某属性的平均值（保留两位小数浮点型）。

请求参数

名称	类型	描述	是否必需	备注
		统计平均值		Array元素类型的可以是整型、浮点和对象类型，

		的数组		当元素是对象类型时需指定key
key	字符串	对象类型元素的属性	否	Array元素是对象类型时必须，且该属性的类型是整型、浮点。

(5) median函数

返回给定数组的中值（保留两位小数浮点型）。

请求参数

名称	类型	描述	是否必需	备注
array	Array<>	统计中值的数组	是	Array元素类型的可以是整型、浮点类型

(6) rank函数

返回一个数值在给数组中的排位(小的排在前面)。

请求参数

名称	类型	描述	是否必需	备注
array	Array<>	统计排名的数组	是	Array元素类型的可以是整型、浮点、时间、字符串、布尔类型
value	同Array元素类型	统计排名的值	是	

(7) avedev函数

返回给定数组的平均绝对误差值（保留两位小数浮点型）。

请求参数

名称	类型	描述	是否必需	备注
array	Array<>	统计平均绝对误差值的数组	是	Array元素类型的可以是整型、浮点类型

(8) frequency函数

返回给定数组中出现次数最多的值。

请求参数

名称	类型	描述	是否必需	备注
array	Array<>	统计出现次数最多值的数组	是	Array元素类型的可以是整型、浮点、时间、字符串、布尔类型

实现方式：编写自定义的函数，利用java脚本引擎ScriptEngineManager执行
使用方式：编辑器输入Statistic.会弹出提示

消息开发

一、简介

消息中心承担应用系统与消息终端之间的桥接工作，有以下能力：

- 提供统一的消息接口服务给应用系统，接受应用系统发送过来的消息；
- 按照设定的规则将消息分发给消息终端；
- 将用户通过消息终端反馈的信息推送给应用系统或者其他消息终端

消息中心使用统一格式标准接收消息数据，集中分发各类消息到各类设备和即时消息系统，起到消息总线的作用，解决应用系统直接与各类设备或即时消息系统通讯带来的复杂性。

二、消息模板配置

操作示例

1. 添加消息

- 进入开发者后台进入首页——>我的套件——>开发管理——>消息配置
- 地址：http://pai.flyrise.cn/developer/auth

- 添加消息

2. 添加消息模版

- 模板管理，先添加消息，再管理消息模板，可添加多种推送方式消息的模板，每种推送方式只能有一条。包含几种推送方式，则发送时会以这种方式发送。例如下图包含短信，邮件，应用消息，app推送4种模板，消息发送时会同时发送到短信，邮件，应用消息，app推送。

- 创建模版

- 从模板库引用

3. 获取编号

- 获取后端工程发送消息会使用到的标识信息
- 消息编号

- 套件标识

三、套件后端工程处理

准备工作

1. 引入依赖pom.xml

```
<!-- 消息发送依赖包 -->
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-common-mc</artifactId>
    <version>1.6.0</version>
```

```
</dependency>
```

2. 配置短信

application.yml or nacos->pai-xxx-xx.yaml
pai.sms.enable=false，默认值是true，false的时候关闭短信的发送
pai.sms.fixed-code，默认值是1234，使用verificationGeneratorHelper工具类获取随机的验证码
当enable为false的时候，验证码是当前设置的fixedCode

```
pai:
  sms:
    enable: true
    fixed-code: 1234
```

verificationGeneratorHelper获取验证码示例

```
    //注入工具类
    @Resource
    private VerificationGeneratorHelper verificationGeneratorHelper;

    //获取数字的验证码
    String code = verificationGeneratorHelper.getNumberCode();
    String code = verificationGeneratorHelper.getNumberCode(Integer);
    //获取字符的随机验证码
    String code = verificationGeneratorHelper.getCode();
    String code = verificationGeneratorHelper.getCode(Integer);
```

3. 定义消息编号常量

```
@Resource
private McTemplateService mcTemplateService;

//发送消息给外部用户-短信
public static final String MP_OUT_SMS = "MP_10000302";
//发送消息给外部用户-邮件
public static final String MP_OUT_EMAIL = "MP_10000303";
//发送消息给外部用户-短信和邮件
public static final String MP_OUT_SMS_AND_EMAIL = "MP_10000304";
//发送消息给内部用户-短信
public static final String MP_INNER_SMS = "MP_10000305";
//发送消息给内部用户-邮件
public static final String MP_INNER_EMAIL = "MP_10000306";
//发送消息给内部用户-应用消息
public static final String MP_INNER_COMPUTER_APP = "MP_10000307";
//发送消息给内部用户-app推送
```

```
public static final String MP_INNER_MOBILE_APP = "MP_10000308";
//发送消息给内部用户-短信和邮件
public static final String MP_INNER_SMS_AND_EMAIL = "MP_10000305";
//发送消息给内部用户-应用消息和app推送
public static final String MP_INNER_COMPUTER_AND_MOBILE_APP = "MP_100003
07";
//发送消息给内部用户-短信、邮件、应用消息和app推送
public static final String MP_INNER_ALL = "MP_10000311";
```

4. 业务消息模板

- 短信模板
  自定义的短信模板需要人工到阿里云申请，因此从模板库引入一个系统内置的短信模板

- 邮件模板

- 应用消息模板

- app推送模板

系统发送消息给外部用户

1. 消息配置下只添加了推送方式为短信的模板

```
@ApiOperation("发送消息给外部用户-短信")
@PostMapping("/sendOutSms")
public Reply sendOutSms(){
    Map<String, String> map = new HashMap<>();
    //短信模板使用了{sms_code}占位符，value由业务系统生成
    map.put("sms_code", "02468");
    OuterUserBody outerUserBody = new OuterUserBody();
    //接收人手机号
    outerUserBody.setReceiveMobiles(new String[]{"17715810635"});
    //来源业务系统套件标识，用于将消息发送结果发布到topic，为空时不发布结果
    outerUserBody.setBusinessSystemId("cn.flyrise.yyh.park");
    //企业id，可以为空，为空时使用开发者配置的模板
    outerUserBody.setEnterpriseId("entId");
    //运营企业运营的园区id 园区id可以为空，为空时使用企业配置的模板
    //若企业id也为空，使用开发者配置的模板
    outerUserBody.setParkId("parkId");
    //需要延时发送时，设置延时发送时间
    //outerUserBody.setDelayedSendTime(DateUtils.parseDate("2020-08-15 1
6:35:00"));
    //设置模版参数，替换模板中的占位符
    outerUserBody.setParam(map);
    //新建模板时得到的消息编号
    outerUserBody.setCategoryCode(MP_OUT_SMS);
    //调用发送方法
    Reply reply = mcTemplateService.sendOuterUser(outerUserBody);
    return reply;
}
```

2. 消息配置下只添加了推送方式为邮件的模板

```
@ApiOperation("发送消息给外部用户-邮件")
@PostMapping("/sendOutEmail")
public Reply sendOutEmail(){
    Map<String, String> map = new HashMap<>();
    //邮件模板使用了{title},{content}占位符,map中需要赋值
    map.put("title", "测试邮件");
    map.put("content", "这是一个测试邮件的内容");
    OuterUserBody outerUserBody = new OuterUserBody();
    //接收人手机号
    outerUserBody.setReceiveMails(new String[]{"457857183@qq.com"});
    //来源业务系统套件标识，用于将消息发送结果发布到topic，为空时不发布结果
    outerUserBody.setBusinessSystemId("cn.flyrise.yyh.park");
    //企业id，可以为空，为空时使用开发者配置的模板
    outerUserBody.setEnterpriseId("entId");
    //运营企业运营的园区id 园区id可以为空，为空时使用企业配置的模板
    //若企业id也为空，使用开发者配置的模板
    outerUserBody.setParkId("parkId");
    //需要延时发送时，设置延时发送时间
    //outerUserBody.setDelayedSendTime(DateUtils.parseDate("2020-08-15 1
6:35:00"));
    //设置模版参数，替换模板中的占位符
    outerUserBody.setParam(map);
    //新建模板时得到的消息编号
    outerUserBody.setCategoryCode(MP_OUT_EMAIL);
    //调用发送方法
    Reply reply = mcTemplateService.sendOuterUser(outerUserBody);
    return reply;
}
```

3. 消息配置下添加了推送方式为短信和邮件的模板

```
@ApiOperation("发送消息给外部用户-短信和邮件")
@PostMapping("/sendOutSmsAndEmail")
public Reply sendOutSmsAndEmail(){
    Map<String, String> map = new HashMap<>();
    //短信模版使用了{sms_code}占位符，value是业务系统生成，map中需要赋值
    map.put("sms_code", "02468");
    //邮件模板使用了{title},{content}占位符,map中需要赋值
    map.put("title", "测试邮件");
    map.put("content", "这是一个测试邮件的内容");
    OuterUserBody outerUserBody = new OuterUserBody();
    //接收人手机号
    outerUserBody.setReceiveMobiles(new String[]{"17715810635"});
    //接收人邮箱地址
    outerUserBody.setReceiveMails(new String[]{"457857183@qq.com"});
    //来源业务系统套件标识，用于将消息发送结果发布到topic，为空时不发布结果
    outerUserBody.setBusinessSystemId("cn.flyrise.yyh.park");
    //企业id，可以为空，为空时使用开发者配置的模板
    outerUserBody.setEnterpriseId("entId");
    //运营企业运营的园区id 园区id可以为空，为空时使用企业配置的模板
    //若企业id也为空，使用开发者配置的模板
    outerUserBody.setParkId("parkId");
    //需要延时发送时，设置延时发送时间
    //outerUserBody.setDelayedSendTime(DateUtils.parseDate("2020-08-15 1
6:35:00"));
    //设置模版参数，替换模板中的占位符
    outerUserBody.setParam(map);
    //新建模板时得到的消息编号
    outerUserBody.setCategoryCode(MP_OUT_SMS_AND_EMAIL);
    //调用发送方法
    Reply reply = mcTemplateService.sendOuterUser(outerUserBody);
    return reply;
```

```
}
```

通过用户id发送消息给内部用户

1. 消息配置下只添加了推送方式为短信的模板

与发送短信给外部用户一样，使用了{sms_code},map中需要赋值

```
@ApiOperation("发送消息给内部用户-短信")
@PostMapping("/sendInnerSms")
public Reply sendInnerSms(){
    InnerUserBody innerUser = InnerUserBody.builder()
            .businessSystemId("cn.flyrise.yyh.park")
            .categoryCode(MP_INNER_SMS)
            .parkId("parkId")
            .enterpriseId("entId")
            .sendUserId("1460189858932723712")
            .priority(PriorityEnum.EXTRA_URGENT)
            .sendUserName("yuyh")
            .sendUserOrgId("1433623657082400768")
            .sendUserOrgName("常规应用组")
            .receiveUserType("user")
            .receiveUserIds(new String[]{"1460189858932723712"})
            .imageUrl("https://avatars.githubusercontent.com/u/317776?s=
200&v=4")
            .param("sms_code", "02468")
            .build();
    //调用发送方法
    Reply reply = mcTemplateService.sendInnerUser(innerUser);
```

```
        return reply;
    }
```

2. 消息配置下只添加了推送方式为邮件的模板

与发送邮件给外部用户一样，使用了{title},{content},map中需要赋值

```
@ApiOperation("发送消息给内部用户-邮件")
@PostMapping("/sendInnerEmail")
public Reply sendInnerEmail(){
    InnerUserBody innerUser = InnerUserBody.builder()
            .businessSystemId("cn.flyrise.yyh.park")
            .categoryCode(MP_INNER_EMAIL)
            .parkId("parkId")
            .enterpriseId("entId")
            .sendUserId("1460189858932723712")
            .priority(PriorityEnum.EXTRA_URGENT)
            .sendUserName("yuyh")
            .sendUserOrgId("1433623657082400768")
            .sendUserOrgName("常规应用组")
            .receiveUserType("staff")
            .receiveUserIds(new String[]{"1504812619359326208"})
            .imageUrl("https://avatars.githubusercontent.com/u/317776?s=
200&v=4")
            .param("title", "测试邮件")
            .param("content", "这是一个测试邮件的内容")
            .build();
    //调用发送方法
    Reply reply = mcTemplateService.sendInnerUser(innerUser);
    return reply;
```

```
}
```

3. 消息配置下只添加了推送方式为应用消息的模板

pc端需要与消息中心建立websocket连接才能接收到消息，以下代码在浏览器中打开，点击连接按钮，即可建立连接。其中userId修改为接收人的id，后端发送消息，就会在页面显示出来
应用消息的模板使用了{title},{content},map中需要赋值

```
@ApiOperation("发送消息给内部用户-应用消息")
@PostMapping("/sendInnerComputerApp")
public Reply sendInnerComputerApp(){
    InnerUserBody innerUser = InnerUserBody.builder()
            .businessSystemId("cn.flyrise.yyh.park")
            .categoryCode(MP_INNER_COMPUTER_APP)
            .parkId("parkId")
            .enterpriseId("entId")
            .sendUserId("1460189858932723712")
            .priority(PriorityEnum.EXTRA_URGENT)
            .sendUserName("yuyh")
            .sendUserOrgId("1433623657082400768")
            .sendUserOrgName("常规应用组")
            .receiveUserType("staff")
            .receiveUserIds(new String[]{"1504812619359326208"})
            .pageWebId("test")
            .imageUrl("https://avatars.githubusercontent.com/u/317776?s=
200&v=4")
            .param("title", "测试应用消息")
            .param("content", "这是一个测试应用消息的内容")
            .build();
    //调用发送方法
```

```
    Reply reply = mcTemplateService.sendInnerUser(innerUser);
    return reply;
}
```

4. 消息配置下只添加了推送方式为app推送的模板

app推送的模板使用了{title},{content},map中需要赋值

```
@ApiOperation("发送消息给内部用户-app推送")
@PostMapping("/sendInnerMobileApp")
public Reply sendInnerMobileApp(){
    InnerUserBody innerUser = InnerUserBody.builder()
            .businessSystemId("cn.flyrise.yyh.park")
            .categoryCode(MP_INNER_MOBILE_APP)
            .parkId("parkId")
            .enterpriseId("entId")
            .sendUserId("1460189858932723712")
            .priority(PriorityEnum.EXTRA_URGENT)
            .sendUserName("yuyh")
            .sendUserOrgId("1433623657082400768")
            .sendUserOrgName("常规应用组")
            .receiveUserType("staff")
            .receiveUserIds(new String[]{"1504812619359326208"})
            .pageAppId("test")
            .imageUrl("https://avatars.githubusercontent.com/u/317776?s=
200&v=4")
            .param("title", "app推送消息")
            .param("content", "这是一个app推送消息的内容")
            .build();
    Reply reply = mcTemplateService.sendInnerUser(innerUser);
```

```
    return reply;
}
```

5. 消息配置下添加了推送方式为短信和邮件的模板

```
@ApiOperation("发送消息给内部用户-短信和邮件")
@PostMapping("/sendInnerSmsAndEmail")
public Reply sendInnerSmsAndEmail(){
    InnerUserBody innerUser = InnerUserBody.builder()
            .businessSystemId("cn.flyrise.yyh.park")
            .categoryCode(MP_INNER_SMS_AND_EMAIL)
            .parkId("parkId")
            .enterpriseId("entId")
            .sendUserId("1460189858932723712")
            .priority(PriorityEnum.EXTRA_URGENT)
            .sendUserName("yuyh")
            .sendUserOrgId("1433623657082400768")
            .sendUserOrgName("常规应用组")
            .receiveUserType("staff")
            .receiveUserIds(new String[]{"1504812619359326208"})
            .imageUrl("https://avatars.githubusercontent.com/u/317776?s=
200&v=4")
            .param("sms_code", "02468")
            .param("title", "短信和邮件")
            .param("content", "这是一个短信和邮件的内容")
            .build();
    //调用发送方法
    Reply reply = mcTemplateService.sendInnerUser(innerUser);
    return reply;
}
```

6. 消息配置下添加了推送方式为应用消息和app推送的模板

```
@ApiOperation("发送消息给内部用户-应用消息和app推送")
@PostMapping("/sendInnerComputerAndMobileApp")
public Reply sendInnerComputerAndMobileApp(){
    InnerUserBody innerUser = InnerUserBody.builder()
            .businessSystemId("cn.flyrise.yyh.park")
            .categoryCode(MP_INNER_COMPUTER_AND_MOBILE_APP)
            .parkId("parkId")
            .enterpriseId("entId")
            .sendUserId("1460189858932723712")
            .priority(PriorityEnum.EXTRA_URGENT)
            .sendUserName("yuyh")
            .sendUserOrgId("1433623657082400768")
            .sendUserOrgName("常规应用组")
            .receiveUserType("staff")
            .receiveUserIds(new String[]{"1504812619359326208"})
            .pageAppId("test")
            .pageWebId("test")
            .pageH5Id("test")
            .imageUrl("https://avatars.githubusercontent.com/u/317776?s=
200&v=4")
            .param("title", "应用消息和app推送消息")
            .param("content", "这是一个应用消息和app推送消息的内容")
            .build();
    //调用发送方法
    Reply reply = mcTemplateService.sendInnerUser(innerUser);
    return reply;
}
```

7. 消息配置下添加了短信、邮件、应用消息和app推送4种方式的模板

```
@ApiOperation("发送消息给内部用户-短信、邮件、应用消息和app推送")
@PostMapping("/sendInnerAll")
public Reply sendInnerAll(){
    InnerUserBody innerUser = InnerUserBody.builder()
            .businessSystemId("cn.flyrise.yyh.park")
            .categoryCode(MP_INNER_ALL)
            .parkId("parkId")
            .enterpriseId("entId")
            .sendUserId("1460189858932723712")
            .priority(PriorityEnum.EXTRA_URGENT)
            .sendUserName("yuyh")
            .sendUserOrgId("1433623657082400768")
            .sendUserOrgName("常规应用组")
            .receiveUserType("staff")
            .receiveUserIds(new String[]{"1504812619359326208"})
            .pageAppId("test")
            .pageWebId("test")
            .pageH5Id("test")
            .imageUrl("https://avatars.githubusercontent.com/u/317776?s=
200&v=4")
            .param("sms_code", "02468")
            .param("title", "短信、邮件、应用消息和app推送")
            .param("content", "这是一个短信、邮件、应用消息和app推送的内容")
            .build();
    //调用发送方法
    Reply reply = mcTemplateService.sendInnerUser(innerUser);
    return reply;
```

```
}
```

通过组织id发送消息给内部用户

通过组织id发送消息给内部用户，只需要将

```
    //设置接收人
    innerUser.setReceiveUserIds(new String[]{"1300385167474561024"});
```

替换为

```
    //设置接收组织
    innerUser.setReceiveOrgIds(new String[]{"1308335669181026304"});
```

也可同时保留接收人、和接收组织，发送时先找到接收人，再找到接收组织下所有人，去重之后发送消息

四、版本调整

配置页面调整

从1.1.0版本开始，我们将采用配置页面标识字段和业务标识字段来实现消息跳转。
参考地址：消息调整对接

1. 如果涉及消息页面跳转，则必须要配置页面标识即页面管理中的页面标识(开发者后台->应用套件->页面管理)

- pageWebId用于PC端消息跳转
- pageAppId用于小程序消息跳转
- pageH5Id用于小程序消息H5页面跳转

2. 业务标识字段则根据业务需求自行决定，可填可不填。

- actionCode用于页面根据不同的业务消息作出不同的显示或操作比如说：流程的待办消息、已办消息，它所对应的业务标识就是todo、done

```
    HashMap<String, String> map = new HashMap<>(7);
    // 设置业务标识
    map.put("actionCode", actionCode);
    //设置页面标识
    map.put("pageAppId", properties.getPageAppId());
    map.put("pageWebId", properties.getPageWebId());
```

```
        map.put("pageH5Id", properties.getPageH5Id());
        //消息实体类
        innerUserBody.setParam(map);
```

innerUser对象调整

参数说明：

- businessSystemId ： 填写套件的标识
- categoryCode ： 模版编号
- priority ： 紧急程度 PriorityEnum
- sendUserName ： 发送人的名称
- sendUserOrgId ： 发送人的组织
- receiveUserType ： ”user“ “staff”,关系到ReceiveUser 接收人的id
- receiveUserIds ： 接受人的id
- pageAppId ： app的页面id
- pageH5Id ： h5的页面id
- pageWebId：pc的页面id
- imageUrl ： 富文本的可以通过传递图片url来覆盖模版的图片
- param ： 模版中的参数内容 {title}

事例：

```
InnerUserBody innerUser = InnerUserBody.builder()
        .businessSystemId("cn.flyrise.yyh.park")
        .categoryCode(MP_INNER_ALL)
        .parkId("parkId")
        .enterpriseId("entId")
        .sendUserId("1460189858932723712")
        .priority(PriorityEnum.EXTRA_URGENT)
        .sendUserName("yuyh")
        .sendUserOrgId("1433623657082400768")
        .sendUserOrgName("常规应用组")
        .receiveUserType("staff")
        .receiveUserIds(new String[]{"1504812619359326208"})
        .pageAppId("test")
        .pageWebId("test")
        .pageH5Id("test")
        .imageUrl("https://avatars.githubusercontent.com/u/317776?s=200&
v=4")
        .param("sms_code", "02468")
        .param("title", "短信、邮件、应用消息和app推送")
        .param("content", "这是一个短信、邮件、应用消息和app推送的内容")
        .build();
```

五、短信模板审核

等待运营人员审核模板成功后，方可使用短信模板发送

六、发送结果订阅

需要通过套件去订阅mc_result 主题，并且businessSystemId 填写套件的标识，其他配置参考消息订阅能力
目前**mc_result**主题已经关闭，有需求使用时请联系中台负责人员

1. 开启套件订阅服务

2. 引入依赖**pom.xml**

```
<!-- 消息订阅依赖包 -->
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-common-mq</artifactId>
    <version>1.6.0</version>
</dependency>
```

3. 开启mq配置

```
pai:
  resource:
    appId: '1506162495875088385'
    appSecret: 'pswxrslh9hjgyy83llswp0juth27crcr'
  mq:
    enable: true
```

4. 代码实现

```
/**
 * 订阅套件标识为"cn.flyrise.yyh.park"的套件（消息中心的套件），
 * 发布的主题"mc_result"（消息发送结果对应的主题）
 */
@PaiMqListener(suiteCode = "cn.flyrise.yyh.park",topic = "mc_result")
public void consumerMcResult(String msg){
    JSONObject jsonObject = JSON.parseObject(msg);
    //发送消息时，消息中心返回的id
    String msgId = jsonObject.getString("msgId");
    //消息的发送结果：全部成功、部分成功、全部失败，根据结果做出相应的处理
    String result = jsonObject.getString("result");
    System.out.println("MessageTestConsumer Listener:" + result);
}
```

服务间调用

创建接口

```
package cn.flyrise.pai.demoone.feign;

import cn.flyrise.common.core.domain.Reply;
import io.swagger.annotations.ApiOperation;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

/**
 * @author : wwcong
 * @date : 5:24 下午 2022/3/24
 */
@FeignClient(name = "pai-demo-article")
public interface ArticleFeignClient {

  @ApiOperation("获取文章列表")
  @GetMapping("${api.service.console.path:}/article/page")
  Reply<?> pageArticlePage(@RequestParam("page") Integer page, @RequestP
aram("size") Integer size);

}
```

调用

```
    @Resource
    private ArticleFeignClient articleFeignClient;

    @ApiOperation("分页查询")
    @GetMapping(value = "pageArticle")
    public Reply<?> pageArticle(Integer page, Integer size) {
        return  articleFeignClient.pageArticlePage(page, size);
    }
```

基础API

调用方法

操作示例

maven依赖

在工程pom.xml文件加入依赖

```
<dependency>
    <groupId>cn.flyrise</groupId>
    <artifactId>pai-common-user-center</artifactId>
    <version>${pai.version}</version>
</dependency>
```

调用方法示例

```
    //用户信息接口
    @Resource
    private IUserService userService;

    //数据权限接口
    @Resource
    private IPermissionService permissionService;

    //部门信息接口
    @Resource
    private IDeptService deptService;

    //企业信息接口
    @Resource
    private IEnterpriseService enterpriseService;

    //园区信息接口
    @Resource
    private IParkService parkService;

    //岗位信息接口
    @Resource
    private IPostService postService;

    //员工信息接口
    @Resource
    private IStaffService staffService;
```

调用方法

用户信息接口(IUserService)

数据权限接口(IPermissionService)

调用接口IPermissionService

```
    //数据权限接口
    @Resource
    private IPermissionService permissionService;
```

内置方法

> 目前拥有5个接口

1. getCurrentUserDataAccess
获取当前用户的数据权限

```
    /**
     * 获取当前用户的数据权限
     *
     * @param suiteCode 套件标识
     * @param tenantId  租户id
     * @return
     */

Reply<List<DataAccessPermissions>> getCurrentUserDataAccess(@RequestParam("suiteCode") String suiteCode, @RequestParam("tenantId") String tenantId)
```

2. getStaffDataAccess
获取用户的数据权限-内部接口

```
    /**
     * 内部接口
     * 获取用户的数据权限
     *
     * @param suiteCode 套件标识
     * @param tenantId  租户id
     * @param staffId   员工id
     * @param from      内部调用凭证
```

```
     * @return
     */
Reply<List<DataAccessPermissions>> getStaffDataAccess(@RequestParam("sui
teCode") String suiteCode,@RequestParam("tenantId") String tenantId, @Re
questParam("staffId") String staffId, @RequestHeader("from") String from
)
```

3. findAppPermsByCode

查看用户拥有的套件的权限-内部接口

```
    /**
     * 内部接口
     * 查看用户拥有的套件的权限
     *
     * @param suiteCode 套件标识
     * @param uid       用户id
     * @param entId     企业id
     * @param from      内部调用凭证
     * @return
     */
Reply<RoleAndPermEntity> findAppPermsByCode(@RequestParam("suiteCode") S
tring suiteCode, @RequestParam("uid") String uid,@RequestParam("entId")
String entId, @RequestHeader("from") String from)
```

4. hasAppPermission

判断是否拥有应用权限-内部接口

```
    /**
     * 判断是否拥有应用权限
     *
     * @param entId      企业id
     * @param uid        用户id
     * @param permission 权限 {suiteCode}:{role}:{perm}
     * @param from       内部调用凭证
     * @return
     */
Reply<Boolean> hasAppPermission(@RequestParam(value = "entId") String en
tId, @RequestParam(value = "uid") String uid, @RequestParam("permission"
) String permission, @RequestHeader(SecurityConstants.FROM) String from)
```

部门信息接口(IDeptService)

调用接口IDeptService

```
        //部门信息接口
    @Resource
    private IDeptService deptService;
```

内置方法

> 目前拥有9个接口

1. findById
获取部门信息

```
    /**
     * 获取部门信息
     *
     * @param deptId 部门id
     * @return
     */
Reply<DeptEntity> findById(@PathVariable("deptId") String deptId)
```

2. findSimpleById
获取简单的部门信息(名称)

```
    /**
     * 获取简单的部门信息
     *
     * @param id 部门id
     * @return
     */
Reply<String> findSimpleById(@PathVariable("deptId") String id)
```

3. findSimpleByIds
获取多个部门信息

```
    /**
     * 获取多个部门信息 key(部门id)-value(部门名称)
     *
     * @param ids 部门id集合
     * @return
     */
Reply<Map<String, String>> findSimpleByIds(@RequestBody List<String> ids
)
```

4. deptTree
获取组织机构树-内部接口

```
    /**
     * 获取组织机构树
     *
     * @param entId 企业id
     * @return
     */
Reply<List<TreeSelect>> deptTree(@PathVariable("entId") String entId,@Re
questHeader("from") String from)
```

5. findLeaderByDeptId
获取部门负责人

```
 /**
     * 获取部门负责人
     *
     * @param deptId 部门id
     * @return
     */
Reply<ContactEntity> findLeaderByDeptId(@PathVariable("deptId") String d
eptId)
```

6. findMinisterByDeptId
获取部门分管领导

```
    /**
     * 获取部门分管领导
     *
     * @param deptId 部门id
     * @return
```

```
     */
Reply<ContactEntity> findMinisterByDeptId(@PathVariable("deptId") String
 deptId)
```

7. findById
获取下属的组织机构

```
    /**
     * 获取下属的组织机构
     *
     * @param entId  企业id
     * @param deptId 部门id 为0获取最顶级的部门
     * @return
     */
Reply<DeptEntity> findById(@PathVariable("deptId") String deptId)
```

8. findById
获取下属的人员及组织机构

```
    /**
     * 获取下属的人员及组织机构
     *
     * @param entId      企业id
     * @param deptId     部门id
     * @param showLeave 是否显示离职
     * @return
     */
Reply<DeptEntity> findById(@PathVariable("deptId") String deptId)
```

企业信息接口(IEnterpriseService)

调用接口IEnterpriseService

```
    //企业信息接口
    @Resource
    private IEnterpriseService enterpriseService;
```

内置方法

> 目前拥有8个接口

1.findById
获取企业信息

```
    /**
     * 获取企业信息
     *
     * @param entId 企业id
     * @return
     */
Reply<EnterpriseEntity> findById(@PathVariable("entId") String entId)
```

2. findByIds
获取多个企业信息

```
   /**
     * 获取多个企业信息
     *
     * @param ids 企业id集合
     * @return
     */
Reply<List<EnterpriseEntity>> findByIds(@RequestBody List<String> ids)
```

3. findSimpleById
获取简单企业信息(企业名称)

```
    /**
     * 获取简单企业信息
     *
     * @param entId 企业id
     * @return
     */
Reply<String> findSimpleById(@PathVariable("entId") String entId)
```

4. findSimpleByIds
获取多个简单企业信息

```
    /**
     * 获取多个简单企业信息 key(企业id)-value(企业名称)
     *
     * @param ids 企业id集合
     * @return
     */
Reply<Map<String,String>> findSimpleByIds(@RequestBody List<String> ids)
```

5.findMyEnterprises
获取我的企业列表

```
    /**
     * 获取我的企业列表
     *
     * @return
     */
Reply<List<EnterpriseEntity>> findMyEnterprises()
```

6. findMyEnterprise
获取企业信息-园区信息 (获取当前我的企业信息)

```
    /**
     * 获取企业信息-园区信息 (获取当前我的企业信息)
     *
     * @return
     */
Reply<EnterpriseEntity> findMyEnterprise();
```

7. findTenantEnterpriseByEntId

获取所有的入驻企业

```
    /**
     * 获取所有的入驻企业
     *
     * @param entId 企业id
     * @return
     */
Reply<List<ParkTenant>> findTenantEnterpriseByEntId(@RequestParam("entId
") String entId)
```

8. findTenantEnterpriseByParkId

获取所有的入驻企业

```
    /**
     * 取所有的入驻企业
     *
     * @param parkId 园区id
     * @return
     */
Reply<ParkTenant> findTenantEnterpriseByParkId(@RequestParam("parkId") S
tring parkId)
```

园区信息接口(IParkService)

调用接口IParkService

```
    //园区信息接口
    @Resource
    private IParkService parkService;
```

内置方法

> 目前拥有6个接口

1. findParkById
根据园区id获取园区信息

```
    /**
     * 根据园区id获取园区信息
     *
     * @param parkId 园区id
     * @return
     */
Reply<ParkModel> findParkById(@PathVariable("parkId") String parkId)
```

响应示例：

```
{
  "code": "200",
  "data": {
    "parkId": "1309331342667943937",
    "parkCode": "0mtyithe",
    "parkName": "开发测试园区",
    "parkEntId": "1309331342667943936",
    "parkDomain": "http://0mtyithe.fedemo.cn",
    "parkType": "5",
    "parkScale": "2",
    "parkArea": "300",
    "parkFloorArea": "3000",
    "parkPhone": "0756-8263251",
    "parkAddress": "广东省珠海市香洲区南方软件园B6",
```

```
    "parkStatus": "1",
    "remark": null,
    "parkLogo": "f058d5fec6114f7bba8b184e5ab33eb4",
    "entCount": 0,
    "userCount": 0
  },
  "time": 1617245405485,
  "msg": "操作成功",
  "annex": null,
  "success": true
}
```

2. findParkByIds
根据多个园区**id**获取多个园区信息

```
    /**
     * 根据多个园区id获取多个园区信息
     *
     * @param parkIds 园区id集合
     * @return
     */
Reply<List<ParkModel>> findParkByIds(@RequestBody List<String> parkIds)
```

3. findParkByIdNoAuth
根据园区**id**获取园区信息**-**内部接口

```
    /**
     * 根据园区id获取园区信息-无鉴权
     *
     * @param parkId 园区id
     * @param from 内部调用凭证
     * @return
     */
Reply<ParkModel> findParkByIdNoAuth(@PathVariable String parkId, @Reques
tHeader("from") String from)
```

4. findParkByIdsNoAuth
根据多个园区**id**获取多个园区信息**-**内部接口

```
    /**
     * 根据多个园区id获取多个园区信息-无鉴权
     *
```

```
     * @param parkIds 园区id集合
     * @param from 内部调用凭证
     * @return
     */
Reply<List<ParkModel>> findParkByIdsNoAuth(@RequestBody List<String> par
kIds, @RequestHeader("from") String from)
```

5. findSimpleParkById
获取简单的园区信息(园区名称)

```
    /**
     * 根据园区id获取园区信息
     *
     * @param parkId 园区id
     * @return
     */
Reply<String> findSimpleParkById(@PathVariable("parkId") String parkId)
```

6. findSimpleParkByIds
获取多个简单的园区信息

```
    /**
     * 根据多个园区id获取多个园区信息 key(园区id)-value(园区名称)
     *
     * @param parkIds 园区id集合
     * @return
     */
Reply<Map<String, String>> findSimpleParkByIds(@RequestBody List<String>
 parkIds)
```

岗位信息接口(IPostService)

调用接口IPostService

```
    //岗位信息接口
    @Resource
    private IPostService postService;
```

内置方法

> 目前拥有7个接口

1. findPostByEntId
获取公司下的所有岗位>获取公司下的所有岗位

```
    /**
     * 获取公司下的所有岗位
     *
     * @param entId  企业id
     * @param enable 是否启用 true|false
     * @param page   页码
     * @param size   页数
     * @param search 搜索条件 （岗位名称）
     * @return
     */
Reply<?> findPostByEntId(@RequestParam("entId") String entId,@RequestParam(value = "enable", required = false) String enable,@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size", required = false) Integer size,@RequestParam(value = "search", required = false) String search)
```

2. findPostById
根据岗位id获取岗位信息

```
    /**
     * 根据岗位id获取岗位信息
     *
     * @param postId 岗位id
```

```
     * @return
     */
Reply<PostEntity> findPostById(@PathVariable("postId") String postId)
```

响应示例：

```
{
  "code": "200",
  "data": {
    "createTime": "2020-11-19T01:37:32.000+0000",
    "updateTime": "2020-11-19T01:37:32.000+0000",
    "createBy": null,
    "updateBy": null,
    "remark": null,
    "postId": "1329237336441098240",
    "postName": "前端开发",
    "postLevel": 5,
    "entId": "0",
    "enable": 0
  },
  "time": 1617246324986,
  "msg": "操作成功",
  "annex": null,
  "success": true
}
```

3. findPostByIds
根据多个岗位**id**获取多个岗位信息

```
  /**
     * 根据多个岗位id获取多个岗位信息
     *
     * @param postIds 岗位id集合
     * @return
     */
Reply<List<PostEntity>> findPostByIds(@RequestBody List<String> postIds)
;
```

响应示例：

```
{
  "code": "200",
  "data": [
    {
      "createTime": "2020-11-19T01:37:32.000+0000",
```

```
        "updateTime": "2020-11-19T01:37:32.000+0000",
        "createBy": null,
        "updateBy": null,
        "remark": null,
        "postId": "1329237336441098240",
        "postName": "前端开发",
        "postLevel": 5,
        "entId": "0",
        "enable": 0
      },
      {
        "createTime": "2020-11-19T01:37:44.000+0000",
        "updateTime": "2020-11-19T01:37:44.000+0000",
        "createBy": null,
        "updateBy": null,
        "remark": null,
        "postId": "1329237387645161472",
        "postName": "后端开发",
        "postLevel": 5,
        "entId": "0",
        "enable": 0
      }
    ],
    "time": 1617246421725,
    "msg": "操作成功",
    "annex": null,
    "success": true
}
```


4. findPostByIdNoAuth

根据岗位**id**获取岗位信息-内部接口

```
    /**
     * 根据岗位id获取岗位信息-无鉴权
     *
     * @param postId 岗位id
     * @param from   内部调用凭证
     * @return
     */
Reply<PostEntity> findPostByIdNoAuth(@PathVariable("postId") String post
Id, @RequestHeader("from") String from)
```

5. findPostByIdsNoAuth

根据多个岗位**id**获取多个岗位信息-内部接口

```
    /**
     * 根据多个岗位id获取多个岗位信息-无鉴权
     *
     * @param postIds 岗位id集合
     * @param from    内部调用凭证
     * @return
     */
Reply<List<PostEntity>> findPostByIdsNoAuth(@RequestBody List<String> po
stIds, @RequestHeader("from") String from)
```

6. findSimplePostById
获取岗位名称

```
    /**
     * 获取岗位名称
     *
     * @param postId
     * @return
     */
Reply<String> findSimplePostById(@PathVariable("postId") String postId)
```

7. findSimplePostByIds
获取简单的岗位信息

```
    /**
     * 获取岗位信息 key(岗位id)-value(岗位名称)
     *
     * @param postIds
     * @return
     */
Reply<Map<String, String>> findSimplePostByIds(@RequestBody List<String>
 postIds)
```

员工信息接口(IStaffService)

调用接口IStaffService

```
    //员工信息接口
    @Resource
    private IStaffService staffService;
```

内置方法

> 目前拥有14个接口

1. findById
获取员工详情

```
    /**
     * 获取员工详情
     *
     * @param id 员工id
     * @return
     */
Reply<StaffEntity> findById(@PathVariable("id") String id)
```

响应示例：

```
{
  "code": "200",
  "data": {
    "staffId": "1309331342793773056",
    "staffUsId": "1309331342693109760",
    "staffName": "测试管理员1",
    "staffDeptId": "1333317403593019392",
    "staffPostId": "1344199581033644034",
    "staffPostName": "测试岗位",
    "staffNo": "FE005",
    "staffHiredate": "2020-08-31T16:00:00.000+0000",
    "staffType": "2",
    "staffPhone": "13012345678",
    "staffEmail": "",
```

```
    "staffSex": "2",
    "staffHead": "a254c3de15e04e6d814f66862a761232",
    "staffStatus": "1",
    "entId": "1309331342667943936",
    "entName": null,
    "deptName": "测试小组",
    "staffDepartureDate": null,
    "partTimes": [
      {
        "deptName": "测试小组",
        "postName": "财务岗位",
        "deptId": "1333317403593019392",
        "postId": "1344199631373680642"
      }
    ]
  },
  "time": 1617243200951,
  "msg": "操作成功",
  "annex": null,
  "success": true
}
```

2. findByIdNoAuth

获取员工详情-内部接口

```
    /**
     * 获取员工详情-无鉴权
     *
     * @param id   员工id
     * @param from 内部调用凭证
     * @return
     */
Reply<StaffEntity> findByIdNoAuth(@PathVariable("id") String id, @Reques
tHeader("from") String from)
```

3. findStaffByIds

通过多个员工id获取多个员工信息

```
    /**
     * 通过多个员工id获取多个员工信息
     *
     * @param ids 员工id集合
     * @return
     */
```

```
Reply<List<StaffEntity>> findStaffByIds(@RequestBody List<String> ids)
```

响应示例:


```
{
  "code": "200",
  "data": [
    {
      "staffId": "1309331342793773056",
      "staffUsId": "1309331342693109760",
      "staffName": "测试管理员1",
      "staffDeptId": "1333317403593019392",
      "staffPostId": "1344199581033644034",
      "staffPostName": "测试岗位",
      "staffNo": "FE005",
      "staffHiredate": "2020-08-31T16:00:00.000+0000",
      "staffType": "2",
      "staffPhone": "13012345678",
      "staffEmail": "",
      "staffSex": "2",
      "staffHead": "a254c3de15e04e6d814f66862a761232",
      "staffStatus": "1",
      "entId": "1309331342667943936",
      "entName": null,
      "deptName": "测试小组",
      "staffDepartureDate": null,
      "partTimes": [
        {
          "deptName": "测试小组",
          "postName": "财务岗位",
          "deptId": "1333317403593019392",
          "postId": "1344199631373680642"
        }
      ]
    },
    {
      "staffId": "1309334869112918016",
      "staffUsId": "1309334869008060416",
      "staffName": "张三",
      "staffDeptId": "1309334869087752192",
      "staffPostId": null,
      "staffPostName": null,
      "staffNo": null,
      "staffHiredate": null,
      "staffType": "2",
      "staffPhone": "13825601204",
      "staffEmail": null,
```

```
        "staffSex": null,
        "staffHead": null,
        "staffStatus": "1",
        "entId": "1309334868978700288",
        "entName": null,
        "deptName": "东岸本部",
        "staffDepartureDate": null,
        "partTimes": []
      }
    ],
    "time": 1617243803825,
    "msg": "操作成功",
    "annex": null,
    "success": true
}
```

4. findStaffByIdsNoAuth

通过多个员工**id**获取多个员工信息-内部接口

```
    /**
     * 通过多个员工id获取多个员工信息-无鉴权
     *
     * @param ids  员工id集合
     * @param from 内部调用凭证
     * @return
     */
Reply<List<StaffEntity>> findStaffByIdsNoAuth(@RequestBody List<String>
ids, @RequestHeader("from") String from)
```

5. findPartStaffById

获取员工部分信息

```
    /**
     * 获取员工部分信息
     *
     * @param id 员工id集合
     * @return
     */
Reply<SimpleStaffModel> findPartStaffById(@PathVariable("id") String id)
```

响应示例：

```
{
```

```
  "code": "200",
  "data": {
    "staffId": "1309331342793773056",
    "staffName": "测试管理员1"
  },
  "time": 1617244099319,
  "msg": "操作成功",
  "annex": null,
  "success": true
}
```


6. findPartStaffByIds

通过多个员工**id**获取多个员工部分信息

```
    /**
     * 通过多个员工id获取多个员工部分信息
     *
     * @param ids 员工id集合
     * @return
     */
Reply<List<SimpleStaffModel>> findPartStaffByIds(@RequestBody List<Strin
g> ids)
```

响应示例：


```
{
  "code": "200",
  "data": [
    {
      "staffId": "1309331342793773056",
      "staffName": "测试管理员1"
    },
    {
      "staffId": "1309334869112918016",
      "staffName": "张三"
    }
  ],
  "time": 1617244264338,
  "msg": "操作成功",
  "annex": null,
  "success": true
}
```


7. findStaffByDeptId

通过部门id获取部门的员工信息

```
    /**
     * 通过部门id获取部门的员工信息
     *
     * @param page    页码
     * @param size    页数
     * @param search 搜索条件 （邮箱|员工名称|员工编号|手机号）
     * @param deptId 部门id
     * @param status 状态(0未激活 1已激活 2停用 3离职)
     * @return
     */
Reply<PageInfo<StaffEntity>> findStaffByDeptId(@RequestParam(name = "pag
e", required = false) Integer page,
                                               @RequestParam(name =
"size", required = false) Integer size,
                                               @RequestParam(name =
"search", required = false) String search,
                                               @RequestParam(name =
"deptId") String deptId,
                                               @RequestParam(name =
"status", required = false) String status
    )
```

8. findStaffByUser
通过用户id、企业id查找对应的员工数据

```
    /**
     * 通过用户id、企业id查找对应的员工数据
     *
     * @param userId 用户id
     * @param entId  企业id
     * @return
     */
Reply<StaffEntity> findStaffByUser(@RequestParam("userId") String userId
,
                                   @RequestParam("entId") String ent
Id)
```

响应示例：

```
{
  "code": "200",
  "data": {
```

```
    "staffId": "1309331342793773056",
    "staffUsId": "1309331342693109760",
    "staffName": "测试管理员1",
    "staffDeptId": "1333317403593019392",
    "staffPostId": "1344199581033644034",
    "staffPostName": null,
    "staffNo": "FE005",
    "staffHiredate": "2020-08-31T16:00:00.000+0000",
    "staffType": "2",
    "staffPhone": "13012345678",
    "staffEmail": "",
    "staffSex": "2",
    "staffHead": "a254c3de15e04e6d814f66862a761232",
    "staffStatus": "1",
    "entId": "1309331342667943936",
    "entName": "开发测试企业",
    "deptName": "测试小组",
    "staffDepartureDate": null,
    "partTimes": [
      {
        "deptName": "测试小组",
        "postName": "测试岗位",
        "deptId": "1333317403593019392",
        "postId": "1344199581033644034"
      },
      {
        "deptName": "测试小组",
        "postName": "财务岗位",
        "deptId": "1333317403593019392",
        "postId": "1344199631373680642"
      }
    ]
  },
  "time": 1617244372197,
  "msg": "操作成功",
  "annex": null,
  "success": true
}
```


9. findStaffByUserNoAuth

通过用户**id**、企业**id**查找对应的员工数据**-**内部接口

```
    /**
     * 通过用户id、企业id查找对应的员工数据
     *
     * @param entId 企业id
```

```
     * @param uid   用户id
     * @param from  内部调用凭证
     * @return
     */
Reply<StaffEntity> findStaffByUserNoAuth(@RequestParam("entId") String e
ntId,
                                         @RequestParam("uid") String
 uid,
                                         @RequestHeader("from") Stri
ng from)
```

10. findSimpleStaffById
获取员工名称

```
    /**
     * 获取员工名称
     *
     * @param id 用户id
     * @return
     */
Reply<String> findSimpleStaffById(@PathVariable String id)
```

11. findSimpleStaffByIds
获取简单的员工信息(Map<String, String>)

```
    /**
     * 获取简单的员工信息 key(员工id)-value(员工名称)
     *
     * @param ids 用户id集合
     * @return
     */
Reply<Map<String, String>> findSimpleStaffByIds(@RequestBody List<String
> ids)
```

12. findStaffByEntId
通过企业id获取员工名称信息-支持模糊查询

```
    /**
     * 根据租户id模糊搜索员工名称 key员工id-value员工名称
     *
     * @param entId  企业id/租户id
     * @param search 搜索条件(员工名称)
```

```
     * @param status 员工状态(0未激活 1已激活 2停用 3离职),null为全部。
     * @return
     */
    Reply<Map<String, String>> findStaffByEntId(@RequestParam("entId") S
tring entId,
                                                @RequestParam(value = "s
earch", required = false) String search,
                                                @RequestParam(value = "s
tatus", required = false) String status)
```

13. findStaffByParkCode

通过园区标识获取已激活入驻企业的员工名称信息-支持模糊查询

```
    /**
     * 根据园区标识获取已激活入驻企业员工名称 key员工id-value员工名称
     *
     * @param parkCode 园区标识
     * @param search   搜索条件(员工名称)
     * @param status   员工状态(0未激活 1已激活 2停用 3离职),null为全部。
     * @return
     */
Reply<Map<String, String>> findStaffByParkCode(@RequestParam("parkCode")
 String parkCode,
                                                @RequestParam(value =
 "search", required = false) String search,
                                                @RequestParam(value =
 "status", required = false) String status)
```

14. findStaffByParkId

通过园区id获取已激活入驻企业的员工名称信息-支持模糊查询

```
    /**
     * 根据园区id获取已激活入驻企业员工名称 key员工id-value员工名称
     *
     * @param parkId 园区id
     * @param search 搜索条件(员工名称) 模糊搜索
     * @param status 员工状态(0未激活 1已激活 2停用 3离职),null为全部。
     * @return
     */
Reply<Map<String, String>> findStaffByParkId(@RequestParam("parkId") Str
ing parkId,
                                             @RequestParam(value = "
```

```
search", required = false) String search,
                                                    @RequestParam(value = "
status", required = false) String status)
```

模块化

后端工程产品、项目all in one 规范

后端工程产品、项目**all in one** 规范

工程改造

1. 新建std模块（命名规范：[原工程名称]-std，例如pai-park-space : space-std），将原标准产品src（除启动类 `Application.java` 外）迁移至此模块。

   (std: standard简写，即代表标准产品)

2. 新建application模块（命名规范：[原工程名称]-application，例如pai-park-space : space-application），迁移原标准产品启动类 `Application.java` 至此模块，保持原包路径

3. 为避免后期项目过多从而导致首层级过于凌乱，新建客制化pom模块（命名规范：[原工程名称]-project，例如[pai-park-space] : space-project），用于存放客制化工程，修改pom文件，添加以下节点

   ```
   <packaging>pom</packaging>
   ```

4. 在客制化pom模块下新建客制化模块（命名规范：[原工程名称]-[项目英文/简拼]，例如[pai-park-space] + [海南港航]: space-hngh），用于存放客制化代码

风格**1**（标准单模块）

```
pai-park-space                          工程根目录(注意所有模块要带工程缩写做前缀，避免Maven .m2目录下Jar同名冲突)
├─space-application                     应用启动模块
│  └─src
│      └─main
│          └─java                      应用启动模块主类
├─space-project
│  └─contract-hngh                      客制化项目模块
│      └─src
│          └─main
│              ├─java                  客制化项目模块代码类
│              └─resources             客制化项目模块配置资源
├─space-std                             标准模块
│  └─src
│      └─main
│          ├─java                      标准模块代码类
│          └─resources                 标准模块配置资源
└─doc                                   文档资源目录
```

风格**2**（标准多模块）

```
pai-contract                                工程根目录(注意所有模块要带工程缩写做前
缀，避免Maven .m2目录下Jar同名冲突)
├─contract-application                      应用启动模块
│  └─src
│     └─main
│         └─java                           应用启动模块主类
├─contract-project
│  └─contract-hngh                         客制化项目模块
│     └─src
│         └─main
│             ├─java                       客制化项目模块代码类
│             └─resources                  客制化项目模块配置资源
├─contract-std                             标准模块
│  ├─contract-app                          标准子模块
│  │  └─src
│  │     └─main
│  │         ├─java                       标准子模块代码类
│  │         └─resources                  标准子模块配置资源
│  ├─contract-base                         标准子模块
│  │  └─src
│  │     └─main
│  │         ├─java                       标准子模块代码类
│  │         └─resources                  标准子模块配置资源
│  ├─contract-land                         标准子模块
│  │  └─src
│  │     └─main
│  │         ├─java                       标准子模块代码类
│  │         └─resources                  标准子模块配置资源
│  ├─contract-lease                        标准子模块
│  │  └─src
│  │     └─main
│  │         ├─java                       标准子模块代码类
│  │         └─resources                  标准子模块配置资源
│  ├─contract-property                     标准子模块
│  │  └─src
│  │     └─main
│  │         ├─java                       标准子模块代码类
│  │         └─resources                  标准子模块配置资源
│  ├─contract-sale                          标准子模块
│  │  └─src
│  │     └─main
│  │         ├─java                       标准子模块代码类
│  │         └─resources                  标准子模块配置资源
│  ├─contract-template                     标准子模块
│  │  └─src
```

```
|   |       └─main
|   |           ├─java                  标准子模块代码类
|   |           └─resources             标准子模块配置资源
|   └─contract-work-space               标准子模块
|       └─src
|           └─main
|               ├─java                  标准子模块代码类
|               └─resources             标准子模块配置资源
└─doc                                   文档资源目录
```

5. application模块pom文件改造

   i. 添加 `profiles` 节点

```xml
<profiles>
        <profile>
            <id>std</id>
            <activation><!--默认使用标准产品进行开发、打包-->
                <activeByDefault>true</activeByDefault>
            </activation>
            <properties>
                <product.path>space-std</product.path>
            </properties>
            <dependencies>
                <dependency>
                    <groupId>cn.flyrise</groupId>
                    <artifactId>space-std</artifactId>
                    <version>${space.version}</version>
                </dependency>
            </dependencies>
        </profile>
        <profile>
            <id>hngh</id>
            <properties>
                <product.path>space-hngh</product.path>
            </properties>
            <dependencies>
                <dependency>
                    <groupId>cn.flyrise</groupId>
                    <artifactId>space-hngh</artifactId>
                    <version>${space.version}</version>
                </dependency>
            </dependencies>
        </profile>
    </profiles>
```

   默认std标准产品开发

当进行非标准产品开发时，可在idea maven面板中进行勾选切换

ii. 添加 `build` 节点

```
        <build>
        <finalName>pai-park-space</finalName>
        <resources>
            <!-- 执行顺序为3，即使用客制化工程（space-xxx）的resources文件覆盖 【执行顺序2】的标准产品工程resources文件 -->
            <resource>
                <directory>../space-project/${product.path}/src/main/resources</directory>
                <includes>
                    <include>**/*</include>
                </includes>
            </resource>
            <!-- 执行顺序为2，即使用标准产品工程（space-std）的resources文件覆盖【执行顺序1】的resources文件 -->
            <resource>
                <directory>../space-std/src/main/resources</directory>
                <includes>
                    <include>**/*</include>
```

```
                </includes>
            </resource>
            <!-- 执行顺序为1，即先使用本工程（space-application）的resource
        s文件 -->
            <resource>
                <directory>src/main/resources</directory>
                <includes>
                    <include>**/*</include>
                </includes>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

6. 工程根目录pom文件改造
   i. 将 `dependencyManagement` ，`dependencies` 迁移到std模块pom文件中
   ii. 去除 `build` 节点
7. std模块pom改造，添加build节点

```
    <build>
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <excludes>
                    <exclude>mapper/**/*</exclude>
                </excludes>
            </resource>
        </resources>
    </build>
```

8. 客制化模块pom改造,
   i. 添加build节点

```
<build>
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <excludes>
                    <exclude>mapper/**/*</exclude>
                </excludes>
            </resource>
        </resources>
    </build>
```

ii. 引入std标准产品模块的依赖

```
        <dependency>
            <groupId>cn.flyrise</groupId>
            <artifactId>space-std</artifactId>
            <version>${space.version}</version>
        </dependency>
```

9. Dockerfile改造

```
ADD ./target/pai-park-space.jar ./
```

替换为

```
ADD ./space-application/target/pai-park-space.jar ./
```

all in one

实现**Service**类覆盖

1. std模块添加BeanConfiguration.java 装载类，对于需要客制化的类进行兼容处理（多项目时会存在A需要客制、B不需要的情况）。

   (`@ConditionalOnMissingBean` 直接在类声明中使用无效，具体参考说明)

```
package cn.flyrise.pai.park.space.config;

/**
 * bean自动装载配置类（取消配置bean的Service注解，用于客户化定制时更替换实现类
）
 *
 * @author lhr
 * @date 2023/8/30 14:38
```

```
 */
@Configuration
public class BeanConfiguration {

    @Bean
    @ConditionalOnMissingBean(IRoomService.class)
    public IRoomService roomService() {
        return new RoomServiceImpl();
    }
}
```

2. 取消需要客制化Service类声明中的@Service注解
3. 改动不大，如：只修改其中的某个方法(如方法为 `private` ，可改为 `protected`)，则可创建与父类同包名的继承类

```
@Service("roomService")
@Transactional(rollbackFor = RuntimeException.class)
public class HnghRoomServiceImpl extends RoomServiceImpl {
    @Override
    public List<RoomVO> imports(MultipartFile file, String parkId) {
        // do somethings
    }
}
```

4. 如改动过大，也可直接整个类（同包、同类名）覆盖即可

VO、PO、DTO ...等类覆盖

针对VO、PO、DTO ...等类的增减字段、必填变更等，直接复制std模块vo类对象到客制化模块中相同包位置处进行调整即可

注意：

1. 先将对应的类复制到客制化工程，提交到git后再进行改动，方便后期回溯改动
2. service、mapper、dao.xml尽可能避免完整覆盖，完整覆盖会导致后期回溯变得异常困难，尽量使用上述【实现Service类覆盖】的方式进行覆盖

数据库相关

1. 在客制化模块src/main/resources/meta/sql下新建文件，文件名在原有命名规范上，以_{客制化模块名称}.sql结尾
2. 复制标准产品vo、po类到客制化相同位置，进行类覆盖字段增加
3. 增加字段有特殊处理时应使用上述【Service类覆盖】的方式进行处理

```
@Service("roomService")
@Transactional(rollbackFor = RuntimeException.class)
public class HnghRoomServiceImpl extends RoomServiceImpl {
    @Override
    public RoomVO add(RoomVO room) {
      if (CharSequenceUtil.isBlank(room.getNewField())) {
          throw new RequiredException(I18nUtil.get("new_field"));
      }
      return super.add(room);
    }
}
```

mybatis mapper

1. 新建对应mapper，命名规范为：客制化模块名称+原mapper名称，例：RoomMapper.java-> HngnRoomMapper.java。(用于区分namespace，如有完全新建功能也可置于此处)

2. 新建*.xml，命名规范为：客制化模块名称+原xml名称，例：RoomDao.xml -> HngnRoomDao.xml

```
<!--namespace指定为上述新建mapper-->
<mapper namespace="cn.flyrise.pai.park.space.dao.HnghRoomMapper">
  ...
</mapper>
```

3. 继承

   `<resultMap>` 提供了 `extends` 属性，可直接指向原 `resultMap` 实现继承

```
<resultMap type="cn.flyrise.pai.park.space.model.po.RoomPO" id="Room
Map" extends="cn.flyrise.pai.park.space.dao.RoomMapper.RoomMap">
  <result property="isUse" column="is_use"/> <!--新增字段-->
  <result property="canArea" column="can_area"/><!--新增字段-->
  <result property="sharedArea" column="shared_area"/><!--新增字段-->
  <result property="useArea" column="use_area"/><!--新增字段-->
  <result property="isBreak" column="is_break"/><!--新增字段-->
</resultMap>
```

4. 覆盖

   `<sql>`、`<select>`、`<insert>`、`<update>` 使用覆盖机制，将需要客制化的标签复制到新建的xml中进行调整即可

   注： `<select>`、`<insert>`、`<update>` 中使用到的 `<resultMap>`、`<sql>` 需要一并复制，否则mybatis启动报错

5. nacos.yml（复制标准工程至客制化工程后）添加mybatis配置（开发过程在客制化工程中复制标准工程bootstrap-dev.yml，并添加以下配置）

```
mybatis:
  overrides:
     # key为标准产品xml，value为需覆盖的xml
    RoomDao.xml: HnghRoomDao.xml
```

注意：如xml文件在mapper目录的下级目录，需要一并配置，例如：

> key出现特殊字符需要用"[]"包住，否则会丢失

```
mybatis:
  overrides:
     "[enterprise/ClueEnterpriseMapper.xml]": enterprise/HnghClueEnterpriseMapper.xml
```

i18n

当部分字段、提示信息与标准产品不同时，可复制std模块下的i18n文件至客制化模块相同位置，按需求替换相应文字即可

common包

1. 在工程【pai-fe】内新建【pai-fe-(项目名称)】，此处以hngh为例，即【pai-fe-hngh】模块，与 `VO类覆盖` 操作一致，复制需要更改的common包中对应的类到【pai-fe-hngh】模块中相同包位置，根据需求调整后进行覆盖替换。

2. 引用包时，根据JVM 的类加载规则【先找到，先加载，后续忽略】的原则，在客制化模块（space-hngh）的pom文件中先引用【pai-fe-hngh】模块，再引用std标准模块，即可达到类覆盖的需求

```
    <dependencies>
        <dependency>
            <groupId>cn.flyrise</groupId>
            <artifactId>pai-fe-hngh</artifactId>
            <version>2.0.0.0-SNAPSHOT</version>
        </dependency>
        <dependency>
            <groupId>cn.flyrise</groupId>
            <artifactId>space-std</artifactId>
            <version>${space.version}</version>
        </dependency>
    </dependencies>
```

发布

```
pai image publish -n pai-park-space -r http://dev.flyrise.cn:8090/pai-park-space/pai-park-space.git -b dev-2.0.0.0 -P std
```

增加 `-P` 参数，用于指定打包的profile，如需发布客制版本，以hngh为例，则将 `-P std` 替换为 `-P hngh` 进行发布即可

注意事项

idea切换profile的时候，有可能会导致不同工程模块文件冲突、或者会出现代码覆盖未达预期的情况，此时可考虑执行以下maven命令进行修复：

```
maven clean compile
```

后续

此方案进入实施阶段，可能会存在未考虑部分客制化实现的情况，需要大家集思广益、共同努力，希望大家多多思考，共同完善，使我们从繁乱的分支解放出来，最大程度达到即可在项目中完善标准产品，也能完美交付项目的理想状态。

前端工程产品、项目all in one 规范

前端工程产品、项目**all in one** 规范

工程改造

一、新建**std**模块

将原标准产品工程代码，移到该文件夹

```
pai-finance-ui                  工程根目录
├─pai-project                   应用启动配置
├─project                       客制化根目录
|   ├─hf                       客制化项目模块
|   └─hngh                     客制化项目模块
└─std                           标准模块
```

二、新建**project**模块

里面按项目，区分文件夹，将代码移到对应项目

> 项目客制化功能：

> - 项目代码，新建 [project] 文件夹下，新建 [项目A] 文件夹，根据 [std] 目录层级：新增、修改文件
> - 与 [std] 内代码一样的，不需建在项目下

三、引入 `@flyriselink/pai-project` 插件

具体安装说明及使用：

> https://www.npmjs.com/package/@flyriselink/pai-project

配置初始化

根目录下新建文件 pai-project/index.js

```
const PaiProject = require('@flyriselink/pai-project')

PaiProject({
```

```
  process: process,
  // log: false,
})
```

更改 package.json

```
{
  "scripts": {
    "p-i": "cross-env MODE=install node pai-project/index",
    "p-install": "cross-env MODE=install node pai-project/index",
    "dev": "cross-env MODE=dev node pai-project/index",
    "build": "cross-env MODE=build node pai-project/index",
    "prod": "cross-env MODE=prod node pai-project/index",
    "compare": "cross-env MODE=prod node pai-project/compare"
  },
  "dependencies": {
    "cross-env": "7.0.3",
    "@flyriselink/pai-project": "^1.2.0"
  },
}
// 以上为：安装项目依赖、启动项目、编译项目
```

四、忽略文件提交

编辑 .gitignore 文件，忽略 [application] 文件夹内容提交

五、本地运行

```
// 1.安装依赖
$ npm i

// 2.安装项目依赖（标准版）
$ npm run p-i
或者（安装某项目依赖）
$ npm run p-i hngh(项目文件夹名称)

// 3.运行项目（标准版）
$ npm run dev
或者（运行某项目）
$ npm run dev hngh(项目文件夹名称)
```

> - 项目启动：最终以 [application] 文件夹下代码进行：运行/编译

六、更改 **Dockerfile** 部署配置

原配置

~~ARG NODE_ENV=test~~
~~RUN npm install~~
~~RUN npm run build:$NODE_ENV~~

~~COPY --from=builder /build/pai-hello-ui/dist/ /usr/share/nginx/html/hello/~~

调整配置

```
ARG NODE_ENV=test
RUN npm install
RUN npm run p-install -P $NODE_ENV
RUN npm run build -P $NODE_ENV

COPY --from=builder /build/pai-hello-ui/application/dist/ /usr/share/ngi
nx/html/hello/
```

> - 增加了 p-install
> - 增加了 -P $NODE_ENV
> - COPY 增加了 /application 路径

七、发布

1.部署标准版

```
pai image publish -n pai-hello-ui -b dev-2.0.0.0
```

2.部署项目

```
pai image publish -n pai-hello-ui -b dev-2.0.0.0 -P hngh(项目文件夹名称)
```

---

pai-project 插件

介绍

> https://www.npmjs.com/package/@flyriselink/pai-project

扩展功能 文件夹差异对比

- 快速删除项目中相同的文件

1. 初始配置

1. 根目录下新建文件 pai-project/compare.js

```
const PaiProject = require('@flyriselink/pai-project')

PaiProject({
  process: process,
  // log: false,
  isCompare: true,
})
```

2. 更改 package.json

```
{
  "scripts": {
    "compare": "cross-env MODE=build node pai-project/compare",
    ...
  }
}
```

2. 进行对比、删除文件

```
$ npm run compare hngh(项目文件夹名称)
```

Spring-Boot3.2镜像打包

在pom.xml中添加

```
<plugin>
            <groupId>com.google.cloud.tools</groupId>
            <artifactId>jib-maven-plugin</artifactId>
            <version>3.4.0</version>
            <configuration>
              <from>
                <image>dev.flyrise.cn:8082/library/openjdk:21-alpine</im
age>
              </from>
              <!--构建后的镜像名称以及私服地址、鉴权信息-->
              <to>
                <image>dev.flyrise.cn:8082/pi-dev/pai-auth:v2024.0.0.0-R
C1</image>
                <auth>
                  <username>***</username>
                  <password>***</password>
                </auth>
                <!--允许非https pai-->
              </to>
              <container>
                <!--要暴露的端口-->
                <ports>
                  <port>8080</port>
                </ports>
                <filesModificationTime>EPOCH_PLUS_SECOND</filesModificat
ionTime>
              </container>
              <!--允许非https-->
              <allowInsecureRegistries>true</allowInsecureRegistries>
            </configuration>

          </plugin>
```

金蝶AAMS

参考文献

> 附件中的 xxx.rar 改成 .jar 然后用mvn install

AAMS中间件集成

> AAMS根据Spring boot 版本引入需要的依赖就好了
> 以下集成以中台2.1.0.0的版本作为基础集成

中台2.1.0.0 (spring boot 2.3)

1、注册starter到本地Maven仓库（或者注册到私有仓库）

> 本次集成只注册到本地仓库
> aams-spring-boot-starter-all-2.1.7.RELEASE.jar 附件里面有

```
# 使用命令把aams-spring-boot-starter-all-2.1.7.RELEASE.jar注册到本地仓库
mvn install:install-file -Dfile=aams-spring-boot-starter-all-2.1.7.RELEASE.jar -DgroupId=com.apusic -DartifactId=aams-spring-boot-starter-all -Dversion=2.1.7.RELEASE -Dpackaging=jar
```

2、引入aams-spring-boot-starter-all-2.1.7.RELEASE.jar

```
        <dependency>
            <groupId>com.apusic</groupId>
            <artifactId>aams-spring-boot-starter-all</artifactId>
            <version>2.1.7.RELEASE</version>
        </dependency>
```

3、排除**tomcat**的依赖

```
        <dependency>
            <groupId>cn.flyrise</groupId>
            <artifactId>pai-spring-boot-starter</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-tomcat</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
```

4、添加授权文件 **license.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<license
  product="Apusic Application Server"
  edition="Agile Edition"
  version="V10.0"
  licensee="金蝶天燕季度测试许可"
  datefrom="2024-06-06"
  expiration="2024-09-30"
  CPUs="1"
  license_ver="43"
  signature="Bzfd0hLb0eIJObvlnniM5xOcQCKGlWsrRbgRDfI3S+JiIUq8BnIac2lk5aX
E94zrRuFFVf7mA7lZk/r0BOUa6V3PhHilAr/ShXcBErENFKGduMRIgXorzJPnGXwAa4j3vgQ
Tt8H/nCqCjVd2vlIGc1HPf5p8LNBrDyHjTU3ayB0mr9RicQYwmt/UonyLmbQysbuQmCiG6zi
3ayGzxGdFQI4wyNUn2b91kCU0qMJ5JRFmWD19daE76NE8qUkBxZM3cPTN3KFttHkgF8NE36D
9/bNazNzMBZhDzrjkS3t/YzN0+kuasLRIBElNiN68MjikxN8ByWzlD/00gxZe/fgaHDPE/72
y7PVjF5dc2KuxIMTHdZeb3UkWFqO4hkyqSJnjduPfO39uDEerEKQbJjv0gt1S4wZDCqt1M83
jPnv/CLIYqd8pmM8Onjn88oFqa+ih9A3em1/r+D4LcJVmQ5kWRWhgMIOfyExb13FOJshgsUP
KHgeYbdH6W802CJ1usPJXCqGABqqROVgizvGGYncOtcbU5ruDe9nWo27LdtSMBo7RBox5cLV
8xx8Yg53tFNedU/GHTHFKxb31w/NL2fVcHg=="
/>
```

5、成功的日志

```
Apusic Application Server Agile Edition V10.0
Licensed to 金蝶天燕季度测试许可
Licensed from 2024-06-06 to 2024-09-30
CPUs limited to 1
... ...
Starting Servlet engine: [Apusic AAS/10.1.0SP2]
```

6、常见的错误

> 1、spring boot 版本与AAMS版本不对应
> 2、授权文件过期或者缺少，日志出现 licentse out due 等关键字

中台2024 (spring boot 3.2)

> 需使用 3.2.4的AAMS版本

配置文件修改（其实和上面是一样的啦，就是版本改一下）

```
<dependency>
            <groupId>cn.flyrise</groupId>
            <artifactId>pai-spring-boot-starter</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-tomcat</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <dependency>
            <groupId>com.apusic</groupId>
            <artifactId>aams-spring-boot-starter-all</artifactId>
            <version>3.2.4</version>
        </dependency>
```

spring boot 3.2 常见错误

> 1、scan jar ,可以通过添加环境变量 -Daas.util.scan.StandardJarScanFilter.jarsToSkip=*.jar 忽略错误

附件

- aams-spring-boot-starter-all-3.2.0.rar
- aams-spring-boot-starter-all-2.1.7.RELEASE.rar

流水号测试

测试人员测试

环境

https://lingyun.flyrise.cn/

账户

13711687995

考勤接口

```
/lingyun-demo-api/v1/sign/in
```

请求参数示例

```
{
  "address": "南方软件园",
  "lat": "113.58",
  "lng": "22.38",
  "parkId": "",
  "source": 0,
  "sourceName": "手动打卡"
}
```

查询单条

```
/lingyun-demo-api/v1/sign/in/{id}
```

请求示例

分页查询

```
/lingyun-demo-api/v1/sign/in/page
```

请求参数示例

```
{
  "page": 1,
  "size": 10,
  "source": 0
}
```

场景一：

模板各项配置互相切换，比如：
1.允许漏号和不允许漏号切换
2.流水位长度发生变化
3.模板规则发生变化
4.重置规则发生变化（需要后端配合）

场景二：

在场景一的基础上进行并发压测。

后端开发验证测试

场景一： 记录丢失

1. 缓存无记录，数据表(sn_runtime_record)无记录
    - 如果为旧版本切换到新版本，则会将旧的转移到新的key，并移除旧的。
    - 如果为故障丢失，会重新以起始位返回，业务保存触发唯一索引异常，并刷新。
2. 缓存无记录，数据表(sn_runtime_record)有记录
    - 如果数据表为最新最大流水号，则会刷回缓存，如果不是可能触发唯一索引异常
3. 缓存有记录，数据表(sn_runtime_record)无记录
    - 人工误删了，当前版本需要人工介入修复，把缓存snRunKey删除即可恢复

场景二：业务库发生变化

1. 业务库发生数据导入，业务库最大流水号发生变化
    - 可能触发唯一索引，刷新最大流水号，以业务数据表最大流水号作为起始
2. 业务库发生数据删除，业务库最大流水号发生变化
    - 会跳号，以当前缓存最大流水号作为起始

场景三：重置流水位

1. 按天、按月、按年重置
2. 未设置则不重置（不兼容旧数据）

场景四：漏号和不允许漏号相互切换

无影响

场景五：缓存和数据库不一致

> 当前版本为优先更新数据库，再更新缓存

场景流：模板流水位发送变化

东方通适配

Springboot工程

注意：非所有工程都需要，只有用到tomcat才需要，适配前检查是否含tomcat依赖

pom文件

```
    <profiles>
        <profile>
            <id>tongweb</id>
            <activation>
                <activeByDefault>false</activeByDefault>
            </activation>
            <dependencies>
                <!-- 特定依赖，只有当指定 profile 时才引入 -->
                <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-web</artifactId>
                    <exclusions>
                        <exclusion>
                            <groupId>org.springframework.boot</groupId>
                            <artifactId>spring-boot-starter-tomcat</arti
factId>
                        </exclusion>
                    </exclusions>
                </dependency>
                <dependency>
                   <groupId>com.tongweb.springboot</groupId>
                    <artifactId>tongweb-spring-boot-websocket-2.x</artif
actId>
                     <version>8.0.E.2</version>
                </dependency>
                <dependency>
                    <groupId>com.tongweb.springboot</groupId>
                    <artifactId>tongweb-spring-boot-starter-2.x</artifac
tId>
                    <version>8.0.E.2</version>
                </dependency>
            </dependencies>
        </profile>
    </profiles>
```

本地测试

引入**license.dat**

> 不要提交到代码中

勾选**profiles**

验证

部署

非all in one项目可不指定-t (推荐不指定)

```
pai image publish -n 工程名称  -b 分支 -t 镜像版本 -P tongweb
```

all in one项目， 多profile用逗号隔开，且必须指定-t！！！

```
pai image publish -n 工程名称  -b 分支 -t 镜像版本 -P tongweb,项目profile
```

配置字典增加证书配置

工程动态挂载

设置镜像

启动验证

适配问题

端口号

tongweb的默认端口号为8088，如需调整为8080，请在共享配置中设置

```
server:
  port: 8080
```