(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112579094 A
(43)申请公布日 2021.03.30

(21)申请号 202011469614.6

(22)申请日 2020.12.15

(71)申请人 上海赛可出行科技服务有限公司
地址 200131 上海市浦东新区自由贸易试验区杨高北路2001号1幢4部位三层333室

(72)发明人 金小俊 蒋杰 赵化 李卫丽

(51)Int.Cl.
G06F 8/41(2018.01)
G06F 8/71(2018.01)
G06F 9/445(2018.01)

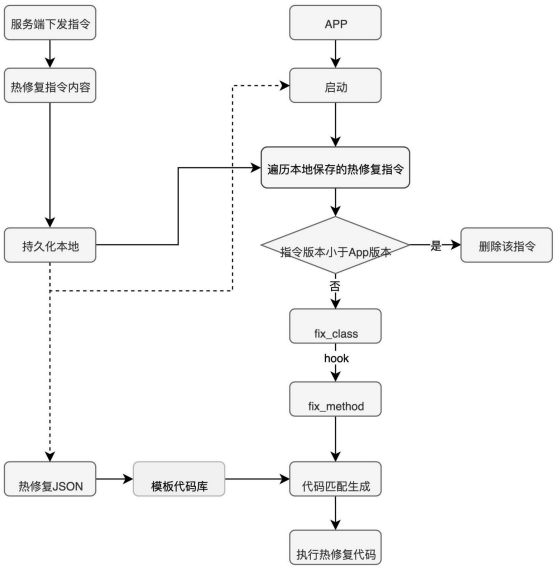
权利要求书1页 说明书9页 附图4页

(54)发明名称

一种基于模板代码匹配的轻量级热修复方法

(57)摘要

本发明公开了一种基于模板代码匹配的轻量级热修复方法，功能包括，服务端远程下发热修复指令到移动客户端，移动客户端接收到热修复指令后通过模板代码库将指令匹配成执行代码嵌入客户端附着执行，嵌入的过程通过AOP(Aspect Oriented Programming，面向切面编程)来实现。该方法可提供多种灵活的动态修复方式，包括：在指定函数或方法执行前或执行后动态嵌入热修复代码；整体替换指定函数或方法的实现；动态新增类的成员变量或属性；动态新增类的函数或方法；指定热修复代码和逻辑的执行线程。本发明符合移动平台的热修复规则并能够顺利通过严格的应用上架审核。



CN 112579094 A

1.一种基于模板代码匹配的轻量级热修复方法,其特征在于,通过后台MIS系统下发热修复指令,指令的格式为JSON文本,其内容包括:

a1、当前热修复指令所针对的客户端版本号;

a2、当前热修复指令的ID;

a3、当前热修复指令所针对的客户端的bundleId;

a4、当前热修复指令所嵌入的代码块的类名;

a5、当前热修复指令所嵌入的代码块的方法或函数;

a6、当前热修复指令嵌入方法或函数中的执行时机;

a7、当前热修复指令需要执行的修复内容;

进一步地,移动客户端接收到热修复指令后,其执行步骤为:

S1.判断热修复指令所针对的客户端bundleId和版本号,只有在这两个都匹配上的情况下才会执行热修复指令;

S2.获取热修复指令代码所需嵌入的类和方法名,并使用AOP的方式Hook住该类中的方法,当该方法执行时附着执行热修复代码;

S3.将指令中的热修复内容在客户端进行模板代码匹配,生成可执行代码后运行;

S4.热修复指令的撤销:

当发现某次下发的热修复指令有问题时,还可以通过下发一个撤销热修复的指令来删除之前下发的热修复指令;

撤销热修复的指令内容为所需撤销的热修复指令的ID,数组格式,可一次性撤销多个热修复指令。

一种基于模板代码匹配的轻量级热修复方法

技术领域

[0001] 本发明涉及移动应用热修复领域,特别涉及一种基于模板代码匹配的轻量级热修复方法。

背景技术

[0002] 现有的移动应用功能愈趋丰富,在测试阶段难以完全覆盖所有用户使用场景,应用上架后线上问题时有发生。尤其是在iOS平台上,由于苹果的审核机制,在出现线上问题时只能依赖发版,需要较长的周期来完成修复,从而导致线上问题快速扩散。

[0003] 现有技术中,虽也有热修复功能的开源技术,比如iOS平台的JSPatch,但是其不符合iOS(iOS是由苹果公司开发的移动操作系统)应用的热修复规则,限制了热修复功能在iOS应用中的使用。因此需要一种能通过系统平台审核并符合iOS和Android热修复规则的方案。

[0004] 另外一方面,业内现有的热修复方案大都是代码或逻辑全量替换,修复方式单一。因此,亦需要一种灵活且支持多种技术手段的轻量级热修复方案。

发明内容

[0005] 本发明要解决的技术问题是克服现有技术的缺陷,提供一种基于模板代码匹配的轻量级热修复方法。

[0006] 为了解决上述技术问题,本发明提供了如下的技术方案:

[0007] 本发明一种基于模板代码匹配的轻量级热修复方法,通过后台MIS系统下发热修复指令,指令的格式为JSON文本,其内容包括:

[0008] a1、当前热修复指令所针对的客户端版本号;

[0009] a2、当前热修复指令的ID;

[0010] a3、当前热修复指令所针对的客户端的bundleId;

[0011] a4、当前热修复指令所嵌入的代码块的类名;

[0012] a5、当前热修复指令所嵌入的代码块的方法或函数;

[0013] a6、当前热修复指令嵌入方法或函数中的执行时机;

[0014] a7、当前热修复指令需要执行的修复内容;

[0015] 进一步地,移动客户端接收到热修复指令后,其执行步骤为:

[0016] S1.判断热修复指令所针对的客户端bundleId和版本号,只有在这两个都匹配上的情况下才会执行热修复指令;

[0017] S2.获取热修复指令代码所需嵌入的类和方法名,并使用AOP的方式Hook住该类中的方法,当该方法执行时附着执行热修复代码;

[0018] S3.将指令中的热修复内容在客户端进行模板代码匹配,生成可执行代码后运行;

[0019] S4.热修复指令的撤销:

[0020] 当发现某次下发的热修复指令有问题时,还可以通过下发一个撤销热修复的指令

来删除之前下发的热修复指令；

[0021] 撤销热修复的指令内容为所需撤销的热修复指令的ID，数组格式，可一次性撤销多个热修复指令。

[0022] 与现有技术相比，本发明的有益效果如下：

[0023] 1.基于模板代码匹配的思路实现线上问题的热修复，符合移动平台的规则并能通过应用审核；

[0024] 2.可提供多种灵活的动态修复方式。

[0025] 支持的修复方式包括如下五种：

[0026] 一、支持在指定函数或方法执行前或执行后动态嵌入热修复代码；

[0027] 二、支持整体替换指定函数或方法的实现；

[0028] 三、支持动态新增类的成员变量或属性；

[0029] 四、支持动态新增类的函数或方法；

[0030] 五、支持指定热修复代码和逻辑的执行线程。

附图说明

[0031] 附图用来提供对本发明的进一步理解，并且构成说明书的一部分，与本发明的实施例一起用于解释本发明，并不构成对本发明的限制。在附图中：

[0032] 图1为模板代码库匹配示意图

[0033] 图2为Android平台动态新增成员变量示意图

[0034] 图3为iOS平台动态新增函数或方法示意图

[0035] 图4为本发明实施流程示意图。

具体实施方式

[0036] 以下结合附图对本发明的优选实施例进行说明，应当理解，此处所描述的优选实施例仅用于说明和解释本发明，并不用于限定本发明。

[0037] 实施例1

[0038] 如图1-4所示，本发明提供一种基于模板代码匹配的轻量级热修复方法，通过后台MIS系统下发热修复指令，指令的格式为JSON文本，其内容包括：

[0039] a1、当前热修复指令所针对的客户端版本号；

[0040] a2、当前热修复指令的ID；

[0041] a3、当前热修复指令所针对的客户端的bundleId；

[0042] a4、当前热修复指令所嵌入的代码块的类名；

[0043] a5、当前热修复指令所嵌入的代码块的方法或函数；

[0044] a6、当前热修复指令嵌入方法或函数中的执行时机(原方法或函数执行前/后执行或替代执行)；

[0045] a7、当前热修复指令需要执行的修复内容；

[0046] 进一步地，移动客户端接收到热修复指令后，其执行步骤为：

[0047] S1.判断热修复指令所针对的客户端bundleId和版本号，只有在这两个都匹配上的情况下才会执行热修复指令；

[0048] S2.获取热修复指令代码所需嵌入的类和方法名,并使用AOP的方式Hook住该类中的方法(比如可使用开源库Aspects),当该方法执行时附着执行热修复代码;

[0049] S3.将指令中的热修复内容在客户端进行模板代码匹配,生成可执行代码后运行;

[0050] S4.热修复指令的撤销:

[0051] 当发现某次下发的热修复指令有问题时,还可以通过下发一个撤销热修复的指令来删除之前下发的热修复指令;

[0052] 撤销热修复的指令内容为所需撤销的热修复指令的ID,数组格式,可一次性撤销多个热修复指令。

[0053] 具体的,总体设计为服务端下发热修复的JSON指令,客户端接受到指令后,将JSON中的内容通过客户端预置的模板代码库进行解析后执行热修复操作。模板代码的匹配包括表达式和执行语句两个部分的内容。

[0054] 表达式包括算术运算符,逻辑运算符和关系运算符。

[0055]

编号	表达式
1	+
2	-
3	*
4	/
5	&&
6	||
7	>

编号	表达式
8	<
9	==
10	>=
11	<=

[0056]

[0057] 表达式匹配JSON格式示例如下

```
{
    "expression": {
        "value":  "+",
        "left_var": 3,
        "right_var": 5,
        "type":  "float"
    }
}。
```

[0058] 对应的模板库代码为,以iOS平台为例

```
    if (expression[@"value"] == @"+") {
        if (expression[@"value"] == @"float") {
                                    return    expression[@"left_var"].floatValue    +
expression[@"right_var"].floatValue;
        }
```

[0059]

```
        if (expression[@"value"] == @"double") {
                                   return    expression[@"left_var"].doubleValue    +
expression[@"right_var"].doubleValue;
        }
                                 return       expression[@"left_var"].intValue       +
expression[@"right_var"].intValue;
```

[0060] }。对应的执行输出代码为

[0061] (3+5)。

[0062] 执行语句包括成员变量的访问,赋值,条件判断和方法调用四类。成员变量的访问和赋值在iOS平台通过KVC的方式实现,在安卓平台通过反射的方式实现。

[0063] 成员变量访问的JSON格式示例为

```
{
    "property": {
        "name": "selectedCity"
    }
}。
```

[0064]

[0065] 对应的模板库代码,以iOS平台为例

[0066]
```
return[self valueForKey:property[@"name"]];
```

[0067] 赋值语句JSON格式示例如下

```
{
    "assignment": {
        "name": "selectedCity",
        "value": "nanjing"
    }
};
```

[0068]

[0069] 对应的模板库代码为,以iOS平台为例

[0070]
```
return[self setValue:assignment[@"value"]
```

[0071]
```
forKey:assignment[@"name"]];
```

[0072] 条件判断的JSON格式示例为

[0073]
```
{
```

[0074]

```
"condition": {
  "type": "or",
  "content": [
    {
      "condition": {
        "type": "and",
        "content": [
          {
            "expression": {
              "value": "==",
              "left_var": 1,
              "right_var": 2,
              "type": "int"
            }
          },
          {
            "expression": {
              "value": ">=",
              "left_var": 1.0,
              "right_var": 0.5,
              "type": "float"
            }
          }
        ]
      }
    },
    {
      "expression": {
        "value": "<",
        "left_var": 3,
        "right_var": 0,
        "type": "int"
      }
    }
  ],
  "true": {},
  "false": {},
```

```
    }
[0075]
}。
```

[0076] 对应的条件判断模板库解析后的代码为

[0077] if((1==2&&1.0>=0.5)//3<0){

[0078] }else{

[0079] }。

[0080] 方法调用的JSON格式示例为

```
{
  "method": {
    "class": "RJMainController",
[0081]
    "selector": "codeFromCity",
    "argument":["Nanjing"]
  }

}。
```

[0082] 对应的方法调用模板库解析后的代码为,以iOS平台为例

[0083] [RJMainController codeFromCity:@"Nanjing"];。

[0084] 本发明只是阐述模板库的思想,并未穷举所有模板及其实现细节。理论上只要模板库足够丰富,是可以覆盖大部分热修复场景要求的。

[0085] 另外一方面,本发明功能特性之二为动态新增成员变量。

[0086] 新增成员变量或属性在iOS平台通过runtime的关联对象来实现,Android平台由于没有相关技术,采用存储信息的方式来实现。具体方案分别如下:

[0087] iOS平台实现原理为通过runtime关联对象的方式来绑定需要新增的成员变量或属性,具体实现为:

```
    - (void)addPropertyName:(NSString *)name value:(id)value {
[0088]
        objc_setAssociatedObject(self, NSSelectorFromString(name), value, OBJC_ASSOCIATION_RETAIN_NONATOMIC);
    }
 - (id)propertyValue:(NSString *)name {
[0089]
    return objc_getAssociatedObject(self, NSSelectorFromString(name));

}。
```

[0090] Android平台实现原理为通过存储信息的方式来实现,存储的key为类名+变量名,value为变量值。

[0091] 本发明功能特性之三为动态新增函数或方法。目前此功能只在iOS平台实现,原理为利用动态决议方法,在iOS平台中调用不存在的方法,系统会自动跳转到动态决议方法中,若动态决议方法仍然未提供该方法的实现,则抛出异常。新增方法的原理即在动态决议方法中调用模板匹配代码,来实现新方法的生成和调用。其JSON指令格式如下:

[0092]
```
{
    "fix_method_define": [{
        "name": "missMethod",
        "content": {
            "method": {
                "class": "RJMainController",
                "selector": "codeFromCity",
                "argument": ["Nanjing"]
            }
        }
    }]
}。
```

[0093] 上面JSON格式新定义了一个missMethod方法，该方法的实现为调用RJMainController的codeFromCity方法。当下发JSON指令后，系统调用missMethod方法，由于方法不存在，会自动转发到动态决议方法中

[0094] -(id)forwardingTargetForSelector:(SEL)aSelector。

[0095] 在动态决议方法中，判断当前转发过来的方法是否是JSON中指定的新增方法，若是，则将content中的内容作为模板代码匹配后执行。

[0096] 本发明功能特性之四为指定热修复代码的执行线程。在JSON指令中指定代码或者新增方法的执行线程，JSON格式为：

[0097]
```
{
  "method": {
    "class": "RJMainController",
    "selector": "codeFromCity",
    "argument":["Nanjing"],
    "queue": "main" // or "thread"
  }
}。
```

[0098] 在iOS平台中，主线程和非主线程可以用下面模板代码来匹配

```
dispatch_async(dispatch_get_main_queue(), ^{ // 主线程
});

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0),
^{ // 子线程
});
```

[0099] 在Android平台中，主线程和非主线程可以用下面模板代码来匹配

```
unOnUiThread(new Runnable() { @Override public void run() { // 主线程 } });

private Thread newThread; //声明一个子线程
new Thread() {
    @Override
    public void run() {
        // 子线程
    }
}.start()。
```

[0101] 与现有技术相比，本发明的有益效果如下：

[0102] 1.基于模板代码匹配的思路实现线上问题的热修复，符合移动平台的规则并能通过应用审核；

[0103] 2.可提供多种灵活的动态修复方式。

[0104] 支持的修复方式包括如下五种：

[0105] 一、支持在指定函数或方法执行前或执行后动态嵌入热修复代码；

[0106] 二、支持整体替换指定函数或方法的实现；

[0107] 三、支持动态新增类的成员变量或属性；

[0108] 四、支持动态新增类的函数或方法；

[0109] 五、支持指定热修复代码和逻辑的执行线程。

[0110] 最后应说明的是：以上所述仅为本发明的优选实施例而已，并不用于限制本发明，尽管参照前述实施例对本发明进行了详细的说明，对于本领域的技术人员来说，其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换。凡在本发明的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

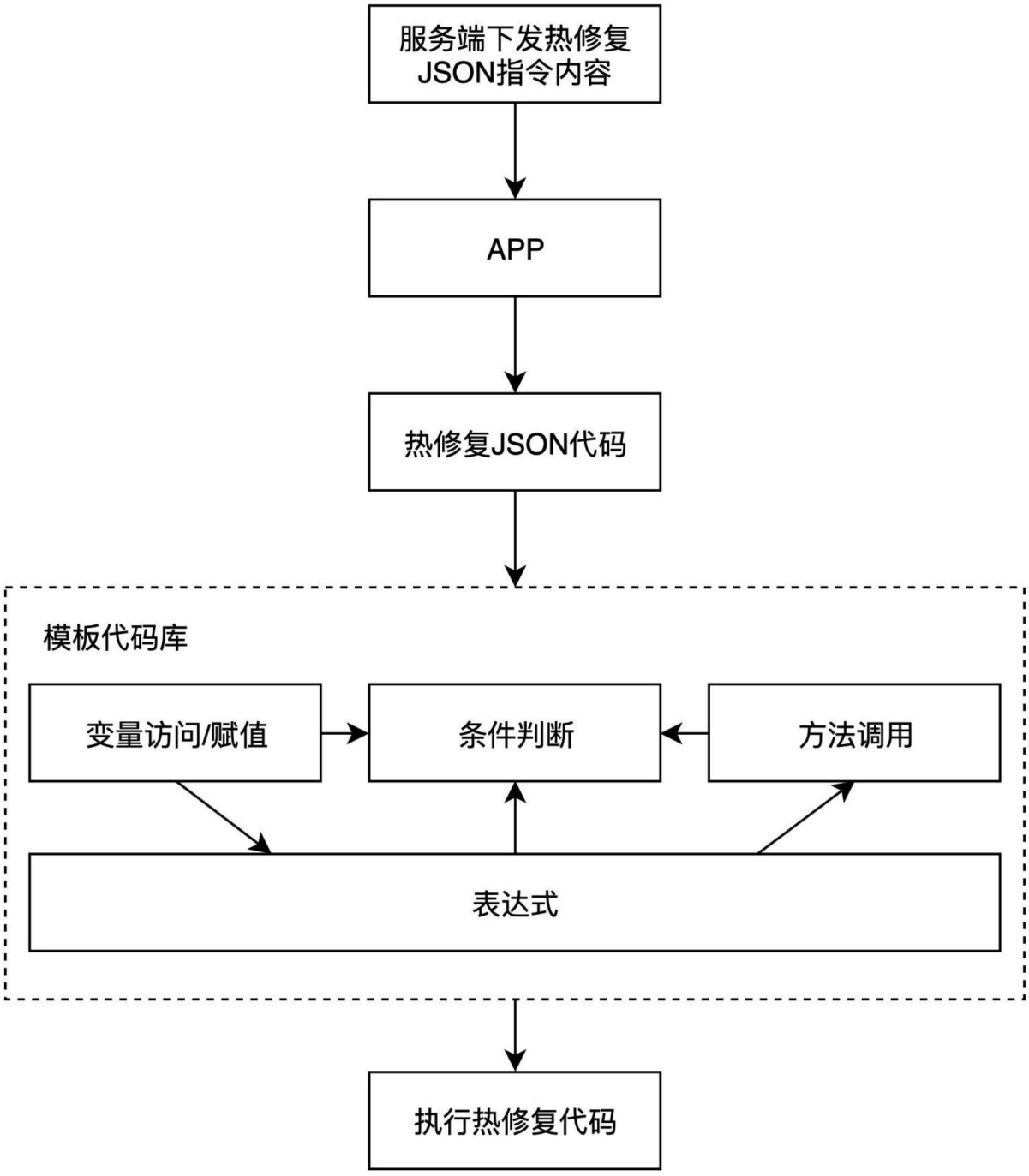

服务端下发热修复JSON指令内容
APP
热修复JSON代码
模板代码库
变量访问/赋值
条件判断
方法调用
表达式
执行热修复代码

图1

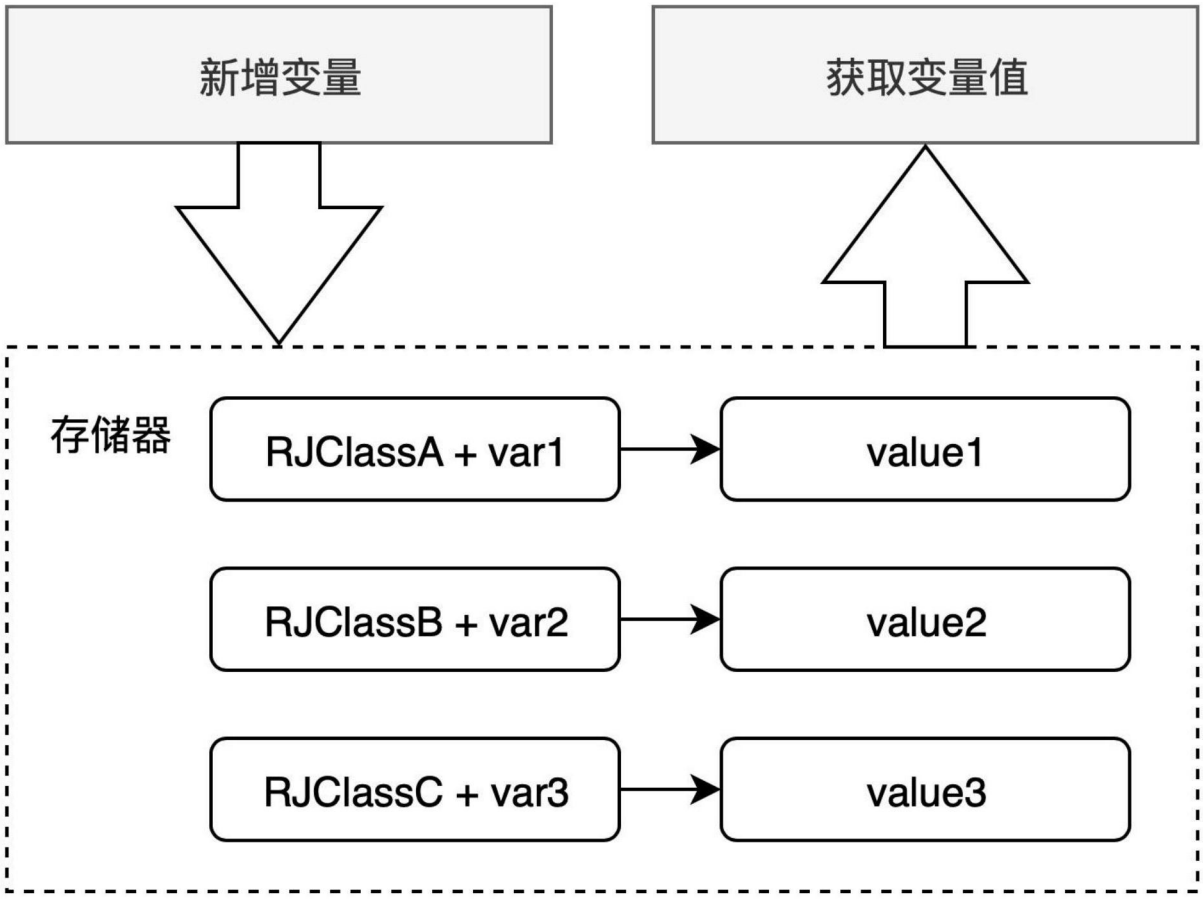

新增变量
获取变量值
存储器
RJClassA + var1
value1
RJClassB + var2
value2
RJClassC + var3
value3

图2

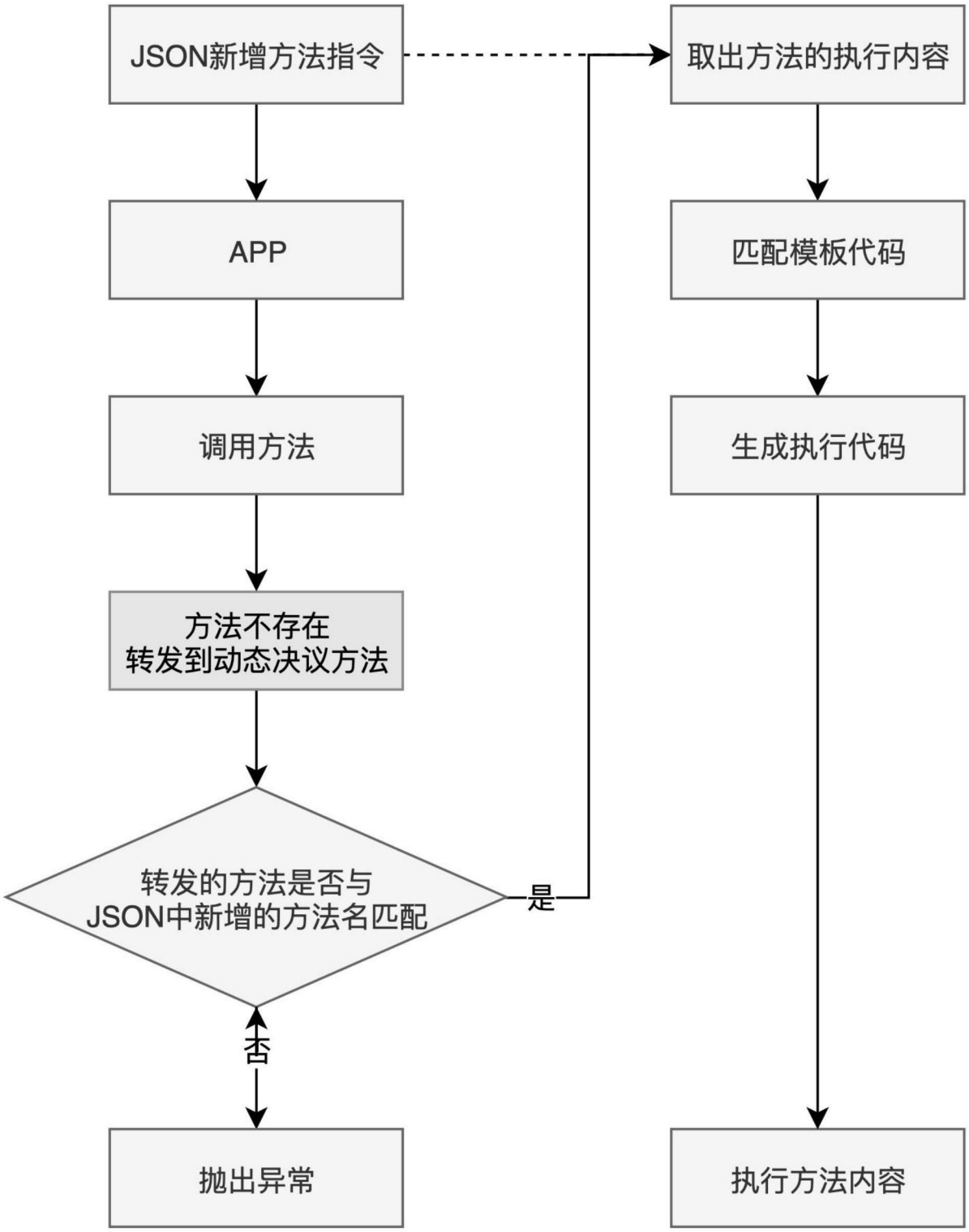

JSON新增方法指令
APP
调用方法
方法不存在
转发到动态决议方法
转发的方法是否与
JSON中新增的方法名匹配
是
否
抛出异常
取出方法的执行内容
匹配模板代码
生成执行代码
执行方法内容

图3

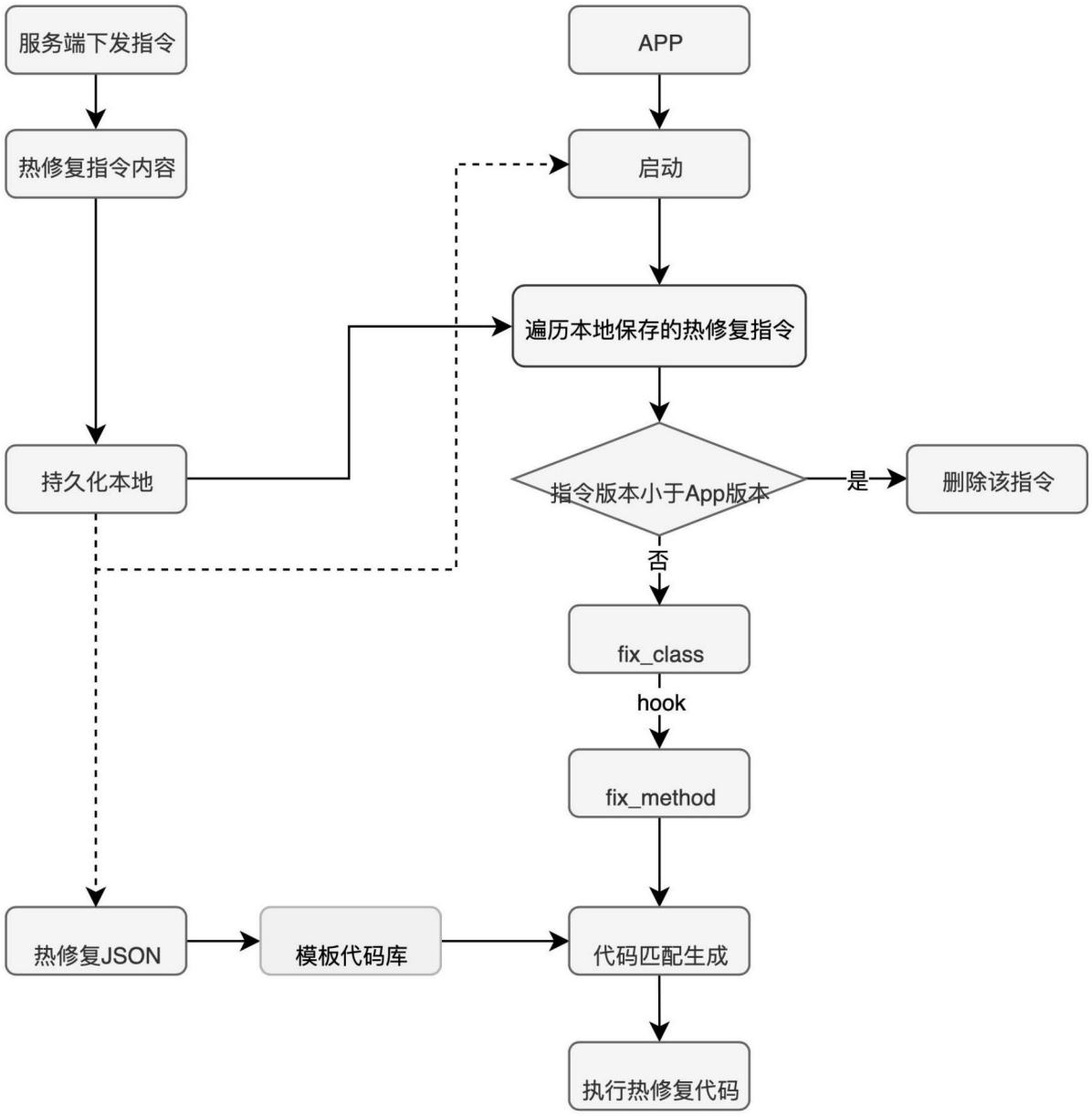

服务端下发指令
热修复指令内容
持久化本地
热修复JSON
模板代码库
APP
启动
遍历本地保存的热修复指令
指令版本小于App版本
是
删除该指令
否
fix_class
hook
fix_method
代码匹配生成
执行热修复代码

图4