DATA PROCESSING METHOD AND PROGRAM MODIFICATION METHOD FOR MIDDLE PLATFORM SYSTEM, SYSTEM, AND STORAGE MEDIUM

TECHNICAL FIELD

[0001] This specification relates to the data processing field, and in particular, to a data processing method and program modification method for a middle platform system, a system, and a storage medium.

BACKGROUND

[0002] With rapid development of information technology, a middle platform system emerges, to realize sharing and multiplexing of digital resources. The middle platform system can provide a unified technical platform for a plurality of different tenants, to avoid repeated development of technology. With economic development, a tenant scale and technical complexity gradually increase, and an architecture of a middle platform system in a related technology is unclear, and functions are crossed. Consequently, a program change of one tenant causes a program change of another tenant, thereby causing a system fault.

[0003] Content of the background part is merely information learned of by the inventor, and neither means that the information has entered a public domain before an application date of the present disclosure, nor means that the information can become a conventional technology of the present disclosure.

SUMMARY

[0004] This specification provides a more efficient data processing method and program modification method for a middle platform system, a system, and a storage medium.

[0005] According to a first aspect, this specification provides a data processing method for a middle platform system. The middle platform system is configured to provide a target service to a plurality of tenants, the target service includes a general engine layer and a personalized functional

layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer includes a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the method includes: receiving a target request of a target tenant for the target service, where the plurality of tenants include the target tenant; invoking, based on the target request, the general engine layer and a target computing module corresponding to the target tenant at the personalized functional layer to execute the target request, to obtain a target result, where the plurality of computing modules include the target computing module; and outputting the target result.

[0006] According to a second aspect, this specification provides a middle platform system. The middle platform system is configured to provide a target service to a plurality of tenants, the target service includes a general engine layer and a personalized functional layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer includes a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the middle platform system includes: at least one storage medium, storing at least one instruction set, to provide the target service for the plurality of tenants; and at least one processor, communicatively connected to the at least one storage medium, where when the middle platform system runs, the at least one processor reads the at least one instruction set, and performs the following operations based on an indication of the at least one instruction set: receiving a target request of a target tenant for the target service, where the plurality of tenants include the target tenant; invoking, based on the target request, the general engine layer and a target computing module corresponding to the target tenant at the personalized functional layer to execute the target request, to obtain a target result, where the plurality of computing modules include the target computing module; and outputting the target result.

[0007] According to a third aspect, this specification further provides a program modification method for a middle platform system. The middle platform system is configured to provide a target

service to a plurality of tenants, the target service includes a general engine layer and a personalized functional layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer includes a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the method includes: receiving a modification request of a target tenant for a program of the target service, where the plurality of tenants include the target tenant; and modifying code of a target computing module corresponding to the target tenant at the personalized functional layer based on the modification request.

[0008] According to a fourth aspect, this specification further provides a middle platform system. The middle platform system is configured to provide a target service to a plurality of tenants, the target service includes a general engine layer and a personalized functional layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer includes a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the middle platform system includes: at least one storage medium, storing at least one instruction set, to perform program modification; and at least one processor, communicatively connected to the at least one storage medium, where when the middle platform system runs, the at least one processor reads the at least one instruction set, and performs the following operations based on an indication of the at least one instruction set: receiving a modification request of a target tenant for a program of the target service, where the plurality of tenants include the target tenant; and modifying code of a target computing module corresponding to the target tenant at the personalized functional layer based on the modification request.

[0009] According to a fifth aspect, this specification further provides a computer-readable non-volatile storage medium. The computer-readable non-volatile storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the data processing method for a middle platform system according to the first aspect is implemented.

[0010] According to a sixth aspect, this specification further provides a computer-readable

non-volatile storage medium. The computer-readable non-volatile storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the program modification method for a middle platform system according to the first aspect is implemented.

[0011] It can be learned from the above-mentioned technical solutions that in the data processing method and program modification method for a middle platform system, a system, and a storage medium. A general engine layer and a personalized functional layer jointly provide a target service for a target tenant. The general engine layer provides a general function of the target service for all tenants, and a computing module at the personalized functional layer provides a personalized customization function for a corresponding tenant. Different tenants each correspond to a computing module. In the middle platform system in the method, the system, and the storage medium provided in this specification, the general engine layer and the personalized functional layer are invoked by using an interface layer, to implement code isolation between the general engine layer and the personalized functional layer. A one-to-one correspondence exists between the computing module at the personalized functional layer and the tenant. When invoking the personalized functional layer, the general engine layer invokes a target computing module corresponding to the target tenant based on a target request of the target tenant, thereby implementing isolation between computing modules of different tenants at the personalized functional layer. Therefore, the method, the system, and the storage medium provided in this specification can not only implement code isolation between the general engine layer and the personalized functional layer, but also implement code isolation between computing modules of different tenants at the personalized functional layer. Therefore, when a tenant maintains or updates code of a computing module corresponding to the tenant, code of the general engine layer does not change, and code of a computing module of another tenant does not change either, so that a fault of the middle platform system is not caused, and high availability, maintenance, and research and development efficiency of the system are improved.

[0012] Other functions of the data processing method and program modification method for a middle platform system, the system, and the storage medium provided in this specification are partially listed in the following descriptions. Creative aspects of the data processing method and program modification method for a middle platform system, the system, and the storage medium provided in this specification can be fully explained by practice or by using the methods,

apparatuses, and combinations described in the following detailed examples.

BRIEF DESCRIPTION OF DRAWINGS

[0013] To describe the technical solutions in the embodiments of this specification more clearly, the following briefly describes the accompanying drawings needed for describing the embodiments. Clearly, the accompanying drawings in the following descriptions show merely some embodiments of this specification, and a person of ordinary skill in the art can still derive other drawings from these accompanying drawings without creative efforts.

[0014] FIG. 1 is a schematic diagram of an application scenario of a middle platform system according to an embodiment of this specification;

[0015] FIG. 2 is a diagram of a hardware structure of a computing device according to an embodiment of this specification;

[0016] FIG. 3 is a diagram of a system architecture of a middle platform system according to an embodiment of this specification;

[0017] FIG. 4 is a diagram of an architecture of a personalized functional layer of a middle platform system according to an embodiment of this specification;

[0018] FIG. 5 is a schematic interface diagram of a configuration management background of a middle platform system according to an embodiment of this specification;

[0019] FIG. 6 is a flowchart of a data processing method for a middle platform system according to an embodiment of this specification;

[0020] FIG. 7 is a flowchart of a method for executing a target request in a middle platform system according to an embodiment of this specification; and

[0021] FIG. 8 is a flowchart of a program modification method for a middle platform system according to an embodiment of this specification.

DESCRIPTION OF EMBODIMENTS

[0022] Specific application scenarios and requirements of this specification are provided in the following descriptions, so that a person skilled in the art can manufacture and use content of this specification. For the person skilled in the art, various local modifications to the disclosed embodiments are clear, and the general principles defined herein can be applied to other

embodiments and applications without departing from the spirit and scope of this specification. Therefore, this specification is not limited to the shown embodiments, but has a widest scope consistent with that of the claims.

[0023] The terms used herein are only intended to describe a particular example embodiment, but do not impose a limitation. For example, unless otherwise explicitly stated in the context, the singular forms "one", "an", and "the" used herein can include plural forms. When used in this specification, the terms "include", "comprise", and/or "have" indicate/indicates existence of an associated integer, step, operation, element, and/or component, but does not exclude existence of one or more other features, integers, steps, operations, elements, components, and/or groups, or another feature, integer, step, operation, element, component, and/or group can be added to the system/method.

[0024] In consideration of the following descriptions, these and other features of this specification, operations and functions of related components of the structure, and economy of combination and manufacturing of components can be significantly improved. With reference to the accompanying drawings, all of these form a part of this specification. However, it should be clearly understood that the accompanying drawings are merely used for the purpose of illustration and description, and are not intended to limit the scope of this specification. It should be further understood that the accompanying drawings are not drawn to scale.

[0025] The flowchart used in this specification illustrates operations implemented by a system in some embodiments of this specification. It should be clearly understood that operations in the flowchart can be implemented out of sequence. On the contrary, operations can be implemented reversely or simultaneously. In addition, one or more other operations can be added to the flowchart. One or more operations can be removed from the flowchart.

[0026] For ease of description, the terms that appear below in this specification are first explained.

[0027] A middle platform system can also be referred to as a multi-tenant system, and reduces costs by sharing and multiplexing resources. The middle platform system is a product evolved between the foreground and the background, and aims to resolve contradiction between a foreground project and a background technology, and realize efficient collaboration between the foreground project and the background technology. A core of the middle platform system is to abstract a general module or service in a background system, form an independent service layer,

and provide fast response and efficient support for the foreground. In this way, when the middle platform system changes, a foreground application can be adapted quickly without a large quantity of modifications to each foreground application. The middle platform system is an organization module between a foreground combat unit and a background resource department, integrates a core capability and a resource of an enterprise to support rapid development and innovation of a foreground transaction, and maximizes use and optimization management of a background resource.

[0028] A general engine layer is a technical component or service that can be multiplexed across tenants in a middle platform system, and provides general function support for different tenants. These functions may include but are not limited to user authentication, data storage, a message transfer, payment processing, etc. Functions of the general engine layer provided by the middle platform system that implements different functions or services may also be different.

[0029] A personalized functional layer is a service or component that can be customized and developed based on requirements of different tenants in a middle platform system. The personalized functional layer is a unique function of the tenant. Personalized functional layers corresponding to different tenants are different.

[0030] A computing module is operable computing units that are customized based on requirements of different tenants at a personalized functional layer. Different tenants correspond to different computing modules, to implement personalized function customization. The computing module can be a software-level algorithm and data processing procedure.

[0031] An interface layer is interaction protocols between layers or components inside a system and between the system and an external application.

[0032] A tenant is a customer or organization in a middle platform system.

[0033] A target service is a service designed to meet specific objectives or requirements in a middle platform system.

[0034] Isolation includes physical isolation and logical isolation. Physical isolation is to isolate an internal device and network and an external public network in a physical manner. Logical isolation is an isolation mechanism implemented by software or configuration in a computer system or network, and logically separates different resources, services, or users, to ensure mutual impact thereof is smallest.

[0035] SPI extension: A full name of SPI is service provider interface. The SPI is a service

discovery mechanism, and is an API for implementation or extension by a third party, and can be used to implement framework extension or component replacement. An SPI mechanism is essentially to configure a fully qualified name of an interface implementation class in a file, read a configuration file by a service loader, and load the implementation class in the file, which can be dynamically an interface replacement implementation class during running.

[0036] An application programming interface (API) is a predefined function, and aims to provide a capability of an application and a developer to access a set of routines based on specific software or hardware without a need to access a source code or understand details of an internal working mechanism.

[0037] Tree routing is to perform conditional routing based on a tree structure.

[0038] Dynamic script is dynamic script compilation provided by groovy, to implement different functions.

[0039] The following describes an application scenario of this specification.

[0040] A data processing method and program modification method for a middle platform system, a system, and a storage medium provided in this specification can be applied to a software architecture of a middle platform system. The middle platform system can be applied to a plurality of scenarios. For example, the middle platform system can be a risk control system that provides a risk control service in a risk control scenario, a payment system that provides a payment service in a payment scenario, or a management system that provides a data management service in a data management scenario.

[0041] That the middle platform system is a risk control system is used as an example. The middle platform system can provide the risk control service for different tenants. The risk control service provided by the middle platform system for the tenant can include both a general risk control computing model and a personalized customization model. The general risk control computing model can be used by all tenants, to implement resource sharing and multiplexing and reduce costs. The personalized customization model can perform personalized customization based on requirements of different tenants, to satisfy use by different tenants. For example, as a tenant, a bank or financial institution can assess and manage a credit risk of a financial product such as a loan or a credit card by using the risk control service provided by the middle platform system. Each institution may have a specific risk assessment model and compliance requirement, and the middle platform system can customize the service for each tenant. As a tenant, an e-

commerce platform needs to perform a risk control on a user transaction behavior, to prevent fraud and abuse behaviors. In this case, the middle platform system can provide real-time monitoring and analysis services, to help a platform identify an abnormal transaction mode and take a corresponding measure. As a tenant, an insurance company needs to perform insurance fraud detection by using the middle platform system. By integrating insurance claim data and market data, the middle platform system can help the insurance company assess authenticity of claims and reduce fraud losses.

[0042] That the middle platform system is a payment system is used as an example. The middle platform system can provide the payment service for different tenants. The payment service provided by the middle platform system for the tenant can include both a general payment computing model and a personalized customization model. The general payment computing model can be used by all tenants, to implement resource sharing and multiplexing and reduce costs. The personalized customization model can perform personalized customization based on requirements of different tenants, to satisfy use by different tenants. For example, as a tenant, an online retailer needs a stable and secure payment system to process a payment transaction of a customer. The middle platform system can provide a customized payment gateway service, support a plurality of payment manners, and ensure transaction security. As a tenant, a mobile application developer may need to integrate a payment function of in-app purchase or subscription services. The middle platform system can provide a flexible payment solution to the developer, including payment preferences and compliance with laws and regulations in different regions. As a tenant, an online education or medical consulting company needs to process advances, periodical payments, etc. of users. A middle platform payment system can provide customized billing management and fund flow services for these enterprises.

[0043] It should be noted that the above-mentioned scenarios are only several of a plurality of use scenarios provided in this specification. The data processing method and program modification method for a middle platform system, the system, and the storage medium provided in this specification not only can be applied to a risk control and payment scenario, but also can be applied to all scenarios in which the middle platform system is needed such as a merchant management scenario and a marketing management scenario. A person skilled in the art should understand that, when the data processing method and program modification method for a middle platform system, the system, and the storage medium provided in this specification are applied to another use

scenario, implementations and technical effects thereof are similar.

[0044] FIG. 1 is a schematic diagram of an application scenario of a middle platform system according to an embodiment of this specification. As shown in FIG. 1, the application scenario 100 can include a tenant 110, a client 120, a network 130, and a middle platform system 140.

[0045] The middle platform system 140 can be a device or server that executes a target service. When the middle platform system 140 is a risk control system, the target service can be a risk control service. When the middle platform system 140 is a payment system, the target service can be a payment service. The middle platform system 140 can store data or instructions for executing the data processing method described in this specification, and can execute or be configured to execute the data and/or instructions. In some embodiments, the middle platform system 140 can store data or instructions that perform the program modification method described in this specification, and can execute or be configured to execute the data and/or instructions. In some embodiments, the middle platform system 140 can include a hardware device with a data information processing function and a necessary program required for driving the hardware device to work.

[0046] The tenant 110 can be a customer of the middle platform system 140, namely, a user who needs to use the target service provided by the middle platform system 140. The middle platform system 140 can provide the target service to a plurality of tenants 110. The tenant 110 can be an enterprise, or can be a natural person. In some embodiments, the tenant 110 can be an e-commerce platform, a mobile payment service provider, a bank, another financial service provider, etc.

[0047] The client 120 can be a middle device in response to the tenant 110 and the middle platform system 140. The client 120 can be a device for data exchange between the tenant 110 and the middle platform system 140. The client 120 can be communicatively connected to the middle platform system 140. The client 120 are not only a first interface in which the user contacts and operates a system, but also a direct platform for displaying a middle platform data processing result and providing a user operation entry. In some embodiments, the client can include a mobile device, a tablet computer, a laptop computer, a device built in a motor vehicle or similar content, or any combination thereof. In some embodiments, the mobile device can include a smart home device, a smart mobile device, a similar device, or any combination thereof. In some embodiments, the smart home apparatus can include a smart television, a desktop computer, or any combination. In

some embodiments, the smart mobile device can include a smartphone, a personal digital assistant, etc., or any combination thereof. In some embodiments, the client 120 can include a hardware device with a data information processing function and a program needed to drive the hardware device to work. In some embodiments, the client 120 can be a device with a location technology, configured to locate the client 120. The middle platform system 140 can be communicatively connected to a plurality of clients 120, and receive data sent by the plurality of clients 120.

[0048] The network 130 can facilitate exchange of information and/or data. As shown in FIG. 1, the client 120 and the middle platform system 140 can be connected to the network 130, and transmit information and/or data to each other through the network 130. For example, the middle platform system 140 can collect desired data from the client 120 over the network 130. In some embodiments, the network 130 can be any type of wired or wireless network, or a combination thereof. For example, the network 130 can include a cable network, a wired network, an optical fiber network, a telecommunications network, an intranet, the Internet, a local area network (LAN), a wide area network (WAN), a wireless local area network (WLAN), a metropolitan area network (MAN), a wide area network (WAN), a public switched telephone network (PSTN), a Bluetooth network, a ZigBee network, a near field communication (NFC) network, or a similar network. In some embodiments, the network 130 can include one or more network access points. For example, the network 130 can include a wired or wireless network access point, for example, a base station and/or an Internet switching point. Through the access point, one or more components of the client 120 and the middle platform system 140 can be connected to the network 130 to exchange data and/or information.

[0049] It should be understood that a quantity of clients 120, a quantity of networks 130, and a quantity of middle platform systems 140 in FIG. 1 are merely an example. There can be any quantity of clients 120, networks 130, and middle platform systems 140 based on implementation needs.

[0050] It should be noted that user data obtained in this specification is authorized by the user, and does not relate to user privacy.

[0051] FIG. 2 is a diagram of a hardware structure of a computing device 200 according to an embodiment of this specification. The computing device 200 can serve as an execution device of the middle platform system 140 in FIG. 1, and perform the data processing method and/or program modification method for a middle platform system described in this specification.

[0052] As shown in FIG. 2, the computing device 200 can include at least one storage medium 230 and at least one processor 220. In some embodiments, the computing device 200 can further include a communication port 250 and an internal communication bus 210. The computing device 200 can further include an I/O component 260.

[0053] The internal communication bus 210 can be connected to different system components. For example, the internal communication bus 210 can be connected to the storage medium 230, the processor 220, the communication port 250, and the I/O component 260.

[0054] The I/O component 260 supports input/output between the computing device 200 and other components.

[0055] The communication port 250 is used for data communication between the computing device 200 and the outside. For example, the communication port 250 can be used for data communication between the computing device 200 and a network 140. The communication port 250 can be a wired communication port, or can be a wireless communication port.

[0056] The storage medium 230 can include a data storage apparatus. The data storage apparatus can be a non-transitory storage medium, or can be a transitory storage medium. For example, the data storage apparatus can include one or more of a disk 232, a read-only storage medium (ROM) 234, or a random access storage medium (RAM) 236. The storage medium 230 further includes at least one instruction set stored in the data storage apparatus. The instruction set can include computer program code. The computer program code can include a program, a routine, an object, a component, a data structure, a process, a module, etc.

[0057] The at least one processor 220 can be communicatively connected to the at least one storage medium 230. When the computing device 200 runs, the at least one processor 220 reads the at least one instruction set, and performs, based on an indication of the at least one instruction set, the data processing method and/or program modification method for a middle platform system provided in this specification. The processor 220 can perform the steps included in the data processing method and/or program modification method for a middle platform system. The processor 220 can be in a form of one or more processors. In some embodiments, the processor 220 can include one or more hardware processors, for example, a microcontroller, a microprocessor, a reduced instruction set computer (RISC), an application-specific integrated circuit (ASIC), an application-specific instruction set processor (ASIP), a central processing unit (CPU), a graphics processing unit (GPU), a physical processing unit (PPU), a microcontroller unit,

a digital signal processor (DSP), a field programmable gate array (FPGA), an advanced RISC machine (ARM), a programmable logic device (PLD), any circuit or processor that can perform one or more functions, or any combination thereof.

[0058] To merely describe a problem, only one processor 220 is shown in the computing device 200 in the accompanying drawings. However, it should be noted that the computing device 200 in this specification can further include a plurality of processors. Therefore, the operations and/or method steps disclosed in this specification can be performed by one processor, or can be jointly performed by a plurality of processors. For example, if the processor 220 in the computing device 200 described in this specification performs step A and step B, it should be understood as that step A and step B can be jointly or separately performed by two different processors 220 (for example, a first processor performs step A, and a second processor performs step B, or a first processor and a second processor jointly perform step A and step B).

[0059] FIG. 3 is a diagram of a system architecture of a middle platform system 140 according to an embodiment of this specification. As shown in FIG. 3, the middle platform system 140 is configured to provide a target service to a plurality of tenants 110. The target service includes a general engine layer 141 and a personalized functional layer 143. The general engine layer 141 is configured to provide the target service to a plurality of tenants 110. The personalized functional layer 143 includes a plurality of computing modules corresponding to the plurality of tenants 110. A computing module is configured to provide a personalized function of the target service to a corresponding tenant 110. The computing module is configured by a corresponding tenant 110. Invoking between the general engine layer 141 and the personalized functional layer 143 is implemented by using an interface layer 145.

[0060] For all tenants 110, the general engine layer 141 can provide a general function of the target service to all tenants 110. The general function can be understood as that different tenants 110 use the same technology, model, or data in a process of implementing the target service, etc. The middle platform system 140 is provided with the general engine layer 141 to implement technology sharing and multiplexing in a process of providing the target service to different tenants 110, thereby resolving computing resources while improving computing efficiency. When the middle platform system 140 is a risk control system, the general function provided by the general engine layer 141 can be attribute conversion, engine analysis (a plurality of risk domains), decision merging, follow-up tasks, etc. An attribute conversion module can convert original data into a

format suitable for risk assessment. Engine analysis is responsible for using a rule engine, a model, and another analysis tool to assess a risk level of a transaction or user behavior. A decision merging module is responsible for integrating output of a plurality of risk control policies or models to form a unified risk control decision, and can determine a final risk score or decision, for example, approval, rejection or further review. A follow-up task module is responsible for processing an activity after a risk control decision, for example, recording a risk control event, updating a user risk file, and notifying a related system to take actions. The general engine layer 141 is responsible for series connection and orchestration of the general functions. The general function provided by the general engine layer 141 in FIG. 3 is merely described as examples. Different middle platform systems 140 can provide different general functions. This is not limited in this specification.

[0061] The personalized functional layer 143 is configured to provide personalized customization functions to different tenants 110. When using the target service, different tenants 110 may have different requirements. For example, when the middle platform system 140 is a risk control system, different tenants 110 may face different compliance requirements, risk types, risk levels, etc. Therefore, to improve accuracy of risk control computing, for different requirements of different tenant 110, the middle platform system 140 is provided with the personalized functional layer 140 to meet the requirements of different tenant 110. To implement configuration of personalized functions of different tenants 110, the personalized functional layer 143 is configured by a corresponding tenant 110. A personalized functional layer that is configured by the tenant 110 and that belongs to the tenant 110 is a computing module. For ease of description, FIG. 3 shows two computing modules, which are marked as a computing module 143a and a computing module 143b. The computing module 143a corresponds to a tenant 110a, provides a personalized function of the target service for the tenant 110a, and is maintained and configured by the tenant 110a. The computing module 143b corresponds to a tenant 110b, provides a personalized function of the target service for the tenant 110b, and is maintained and configured by the tenant 110b. A computing module can have different computing submodules based on different personalized customization requirements. The computing submodule can perform maintenance, configuration, change, or even deletion according to a requirement of the tenant 110. For ease of description, the computing submodule is defined as an extension point.

[0062] The general engine layer 141 and the personalized functional layer 143 cooperate with each other to implement a complete target service. The general engine layer 141 can provide a

basic implementation framework to implement the target service, and the personalized functional layer 143 provides personalized data and implementations for computing of the general engine layer 141. That is, when a tenant 110 uses the target service, a complete function of the target service needs to be completed by jointly participating in computing by computing modules corresponding to the tenant 110 at the general engine layer 141 and the personalized functional layer 143. In a process of completing the target service, the general engine layer 141 needs to invoke the personalized functional layer 143, and an invoking process is implemented by using the interface layer 145. The interface layer 145 can be an application programming interface (API interface). The API is a set of rules that need to be followed when an application interacts with another application in a programming manner. The API represents the application programming interface. In the context of the API, the term "application" is any software that has a unique function. The interface can be considered as a service contract between two applications. The contract defines how the two applications communicate with each other by using requests and responses. In some embodiments, the interface layer 145 can be an SPI extension interface. The SPI extension interface can configure a fully qualified name of an interface implementation class in a file, read a configuration file by a service loader, and load the implementation class in the file, which can be dynamically an interface replacement implementation class during running. That is, the SPI extension interface can configure a fully qualified name of a computing module in the file. When the general engine layer 141 invokes the computing module, the corresponding computing module can be invoked based on a fully qualified name of a computing module corresponding to a target tenant that initiates the target service, to implement the target service. That is, when invoking the computing module, the SPI extension interface invokes the computing module based on the fully qualified name corresponding to the computing module. Different computing modules have different fully qualified names. As described above, the tenant 110 corresponds to a computing module. That is, the tenant 110 corresponds to a fully qualified name of the computing module. The middle platform system 140 can prestore a mapping relationship between the tenant 110 and the fully qualified name of the computing module. The middle platform system 140 can determine, based on the target tenant, the fully qualified name of the computing module corresponding to the target tenant, and then invoke code of the corresponding computing module to implement application. When the code of the computing module changes, no invoking error occurs provided that the mapping relationship between the fully qualified name of the computing

module and the tenant 110 does not change. In addition, the SPI extension interface can implement dynamic replacement and extension of the computing module. When a new computing module is added, extension of the computing module can be quickly implemented provided that a mapping relationship between a fully qualified name of a newly added extended module and a corresponding tenant is stored.

[0063] The interface layer 145 can be connected to the general engine layer 141 and the personalized functional layer 143, thereby implementing isolation between the general function of the general engine layer 141 and the personalized function of the personalized functional layer 143. Usually, the general function changes at a low frequency, and the personalized function changes at a high frequency. Therefore, after the middle platform system 140 isolates the general engine layer 141 and the personalized functional layer 143, the tenant 110 does not modify code of the general engine layer 141 when modifying code of a computing module of the personalized functional layer 143 configured by the tenant, and therefore does not incorrectly modify a system fault caused by the general engine layer 141.

[0064] In addition, there is a one-to-one correspondence between the computing module at the personalized functional layer 143 and the tenant 110, so that when computing modules of different tenants 110 are isolated, personalized functions of different tenants are also isolated. A specific isolation manner can be that the fully qualified name of the computing module is in a one-to-one correspondence with a tenant identifier of the tenant 110, and tenant identifiers of different tenants 110 are unique. Different tenants 110 have different tenant identifiers. The tenant identifier can be used to identify an identity of the tenant 110. The tenant identifier can be a string generated based on a specific coding rule. In some embodiments, the tenant identifier is related to a device identifier of a client 120 used or bound to the tenant 110. For example, the tenant identifier can be generated based on the device identifier of the client 120. For another example, the tenant identifier can be associated with the device identifier of the client 120. When invoking the personalized functional layer 143, the general engine layer 141 can invoke a computing module corresponding to the personalized functional layer 143 based on the tenant identifier of the tenant 110. Such a setting can make computing modules of tenants 110 isolated from each other. A change of a computing module of one tenant 110 does not cause a change of a computing module of another tenant 110, and therefore does not cause a fault of the target service.

[0065] FIG. 4 is a diagram of an architecture of a personalized functional layer 143 of a middle

platform system 140 according to an embodiment of this specification. As described above, an interface layer 145 is an SPI extension interface. Code of each extension point in a computing module at the personalized functional layer 143 can be managed in a form of a software package. When a tenant configures a corresponding computing module, a software package of code of an extension point configured by the tenant can be loaded into a module extension package store 147 for storage. When the middle platform system 140 is started, a software package of code of an extension point corresponding to each of a plurality of tenants 110 can be loaded into the module extension package store 147. As mentioned above, the plurality of tenants 110 correspond to a plurality of computing modules. Specifically, the plurality of tenants 110 are in a one-to-one correspondence with the plurality of computing modules. A correspondence between the plurality of tenants 110 and the plurality of computing modules can be implemented based on a mapping relationship between a tenant identifier of the tenant 110 and a fully qualified name of a computing module. The middle platform system 140 can prestore a mapping relationship between a tenant identifier of a tenant 110 and a fully qualified name of a computing module. The mapping relationship between a tenant identifier of a tenant 110 and a fully qualified name of a computing module can be stored in a plurality of manners such as a list form or a key-value pair form. In some embodiments, the computing module and a corresponding tenant identifier are stored in a key-value pair. That is, the mapping relationship between a fully qualified name of a computing module and a tenant identifier of a tenant 110 is stored in a key-value pair, for example, a tenant identifier of a tenant 110a => a fully qualified name of a computing module 143a, and a tenant identifier of a tenant 110b => a fully qualified name of a computing module 143b. The key-value pair is easy to use and has advantages of high performance, scalability, and flexibility. When a data amount is large, the key-value pair is found and retrieved more quickly and efficiently.

[0066] In some embodiments, the middle platform system 140 can further store a mapping relationship between a fully qualified name of a computing module and a storage path. Specifically, the mapping relationship between a fully qualified name of a computing module and a storage path can be stored in an extension point router 148. The mapping relationship between a fully qualified name of a computing module and a storage path can be stored in a plurality of manners such as a list form or a key-value pair form. The middle platform system 140 can determine a fully qualified name of a corresponding computing module based on the tenant identifier of the tenant 110, and further determine a storage path of the corresponding computing module, so that a software

package of code of the computing module is found in a corresponding storage path of the module extension package store 147, to implement a function of an application.

[0067] As described above, one computing module can include a plurality of submodules (also referred to as extension points). Different submodules are corresponding to different personalized scenarios. That is, one computing module can include submodules in a plurality of scenarios, to meet personalized requirements of different scenarios. For different scenarios, requirements are different, and function implementations are also different. To cope with personalized functions of different scenarios, a scenario can be further divided into deeper sub-scenarios. The middle platform system 140 can implement division of a scenario by using an implementation of tree routing+dynamic script. The submodules in the plurality of scenarios are classified based on a classification rule preset by a target tenant. Standard classification rules can be preset in the middle platform system 140, and different tenants can also customize classification rules. Different tenants can set different classification rules. The classification rule is pre-configured by the tenant.

[0068] Tree routing is performing routing based on a condition customized by the tenant. When a branch path of a tree is satisfied, routing finally arrives at a leaf node. In consideration that changes of a target service of the same tenant in different scenarios are isolated from each other, tree routing includes standard routing and customized routing.

[0069] A top-level node of standard routing is a general function (for example, attribute conversion, engine analysis, decision merging, and follow-up tasks) of the general engine layer 141 in the target service, and sub-nodes are different scenarios in the general function.

[0070] Customized routing is to set a routing condition according to a requirement of the tenant 110.

[0071] An administrator or a user of the tenant can manage tree routing in a configuration management background.

[0072] Code of the submodule is implemented by using script, and the script can be implemented in programming languages including but not limited to Python, JavaScript, Java, etc. For the script of the submodule, the administrator or the user of the tenant can change and modify the script in the configuration management background. The script of the submodule can be dynamically added, deleted, modified, or published in the configuration management background. The configuration management background allows the administrator to manage the script of submodule in a visual interface. Once the script is updated in the background, the middle platform

system 140 can load new script in real time, to implement an immediate change to a function. In addition, the system can record detailed log information, and perform monitoring, to facilitate troubleshooting and performance optimization.

[0073] FIG. 5 is an interface diagram of a configuration management background of a middle platform system according to an embodiment of this specification. FIG. 5 is an example interface diagram of a configuration management background. A person skilled in the art should understand that an interface diagram of another configuration management background also falls within the protection scope of this specification. As shown in FIG. 5, for example, for an attribute conversion computing module in the tenant 110a, risk control identification can include risk identification at registration, risk identification at login, risk identification at payment, and risk identification at refund. In this case, registration, login, payment, and refund can be used as a plurality of scenarios for risk identification, and risk identification at registration, risk identification at login, risk identification at payment, and risk identification at refund can be used as a plurality of submodules of the attribute conversion computing module. Submodules in each scenario can be further divided. For example, in a payment scenario, further division can be performed based on a country, for example, a country A, a country B, and a country C. After a country is determined, division can be further performed based on a user status, for example, a new user and an old user. After the user status is determined, corresponding computing code in a final scenario can be determined. Certainly, the tenant 110 can also configure another scenario division rule. For example, the tenant 110 can further perform sub-scenario division based on an account receivable age, a card age, and/or a payment manner.

[0074] In conclusion, it can be learned that the tenant 110 can divide a scenario into different layers through tree routing, so that code of a computing module is divided into code in a plurality of scenarios or a plurality of sub-scenarios based on different scenarios. A same scenario can be divided simultaneously based on different classification rules. However, a quantity of layers into which the scenario is divided can be configured according to requirements of different tenants 110.

[0075] It should be noted that, through tree routing of the scenario, code of a submodule on a leaf node (namely, a last layer obtained through division) of tree routing is implemented by using script. The leaf node of tree routing is associated with groovy script of the code of the submodule. The script associated with the leaf node can be changed and modified in the configuration management background, and a function can be dynamically and flexibly personalized through

addition, deletion, and modification in the background.

[0076] In conclusion, it can be learned that, in the middle platform system 140, different scenarios of the same computing module are isolated, and changes to personalized functions of different scenarios of the same computing module are isolated through tree routing. In addition, personalized functions are dynamically extended based on a groovy script dynamic update technology. Different scenarios are isolated, so that code changes in different scenarios are independent of each other. A change of one scenario does not affect a change of another scenario, so that an implementation of the code is simpler and more efficient. In addition, dynamic script compilation and dynamic loading technologies are implemented by using dynamic script, to implement dynamic modification of the personalized function, and greatly improve research and development efficiency and modification efficiency of code. In addition, the personalized function is migrated to a page through tree routing and by using dynamic script, to visually present the personalized function. In subsequent code arrangement, arrangement costs can be greatly reduced.

[0077] FIG. 6 is a flowchart of a data processing method P300 for a middle platform system according to an embodiment of this specification. As described above, a computing device 200 can perform the data processing method P300 for a middle platform system in this specification. Specifically, a processor 220 in the computing device 200 can read an instruction set stored in a local storage medium of the computing device 200, and then perform the data processing method P300 for a middle platform system in this specification based on a stipulation of the instruction set. As shown in FIG. 6, a risk control system is used as an example, and the method P300 can include S310 to S330.

[0078] S310: A target tenant sends a target request for a target service to a middle platform system 140 through a client 120, and the middle platform system 140 receives the target request of the target tenant for the target service.

[0079] The plurality of tenants 110 include the target tenant. That is, the target tenant can be one of the plurality of tenants 110 of the middle platform system 140. For example, the target tenant is a tenant that is performing risk control identification in the middle platform system 140. The client 120 can be a terminal device used by the target tenant to interact with the middle platform system 140. The target request can be a request of the target tenant for requesting the middle platform system 140 to perform risk control computing. In some embodiments, the target request carries a target tenant identifier of the target tenant.

[0080] S330: The middle platform system 140 invokes a target computing module corresponding to the target tenant at a general engine layer 141 and a personalized functional layer 143 to execute the target request, to obtain a target result.

[0081] The target computing module can be a computing module corresponding to the target tenant in a plurality of computing modules corresponding to the plurality of tenants 110. The plurality of computing modules include the target computing module. In step S330, the middle platform system 140 can determine a fully qualified name of the target computing module based on the target tenant identifier of the target tenant, thereby invoking the general engine layer 141 and the target computing module to execute the target request.

[0082] S350: The middle platform system 140 outputs the target result.

[0083] The target result can be a result obtained by the middle platform system 140 by executing the target service based on the target request. For example, when the middle platform system 140 is a risk control system, the target result can be a result of risk control identification. The middle platform system 140 can output the target result to a client 120 of the target tenant. The client 120 can present the target result to the target tenant.

[0084] FIG. 7 is a flowchart of a method for executing a target request in a middle platform system 140 according to an embodiment of this specification. FIG. 7 corresponds to step S330. As shown in FIG. 7, step S330 can include the following steps.

[0085] S331: Identify the target computing module based on the target tenant identifier.

[0086] S333: Invoke, based on the target request, the general engine layer 141 to execute a general function of the target service.

[0087] S335: Invoke, by using an interface layer 145, the target computing module to execute a function corresponding to the target tenant in the target service.

[0088] S337: Obtain the target result.

[0089] In some embodiments, the middle platform system 140 can further include an identification layer 149. The identification layer 149 is configured to: receive the target request, and identify the corresponding target computing module based on the target tenant identifier in the target request. In step S331, the identification layer 149 in the middle platform system 140 can identify the fully qualified name of the target computing module corresponding to the target tenant based on a mapping relationship between a tenant identifier of a tenant 110 and a fully qualified name of a computing module and the target tenant identifier carried in the target request. The

middle platform system 140 can add the fully qualified name of the target computing module to an invoking link of the target service. In a process of executing the target service, each invoking of the personalized functional layer 145 by the general engine layer 141 carries the fully qualified name of the target computing module. In such a setting, the fully qualified name of the target computing module accompanies an entire execution process of the target service, to ensure that invoking can be performed accurately each time.

[0090] In some embodiments, the middle platform system 140 can further identify a storage path of the target computing module based on a mapping relationship between a fully qualified name of a computing module and a storage path, to accurately invoke the target computing module, and ensure that implementations of personalized customization functions of different tenants do not cross, thereby ensuring accurate execution of the target service.

[0091] As described above, the computing module can be divided into a plurality of submodules based on scenarios. In addition, the scenario can be further divided into sub-scenarios, so that the submodules are further divided. In this case, the target computing module can include submodules in a plurality of scenarios. The target request can further include a target scenario identifier corresponding to a target scenario. The target scenario can be one of the plurality of scenarios, can be one of sub-scenarios included in the plurality of scenarios, or can be one of scenarios of leaf nodes in tree routing. In some embodiments, the target scenario identifier can include a scenario identifier of each node on an entire link in tree routing.

[0092] In this case, step S335 can include: invoking, by using the interface layer, a submodule corresponding to the target scenario identifier in the target computing module to execute a function that is of the target tenant and that corresponds to the target scenario in the target service. In this case, the submodule corresponding to the target scenario identifier in the target computing module can be a submodule corresponding to a leaf node that meets a condition in tree routing. Specifically, the middle platform system 140 can match the target scenario identifier with an identifier of a node in tree routing, to determine a leaf node corresponding to the target scenario identifier, and invoke a submodule corresponding to the leaf node to execute a corresponding function.

[0093] In conclusion, the general engine layer 141 and the personalized functional layer 143 in the middle platform system 140 provided in this specification are isolated by using the interface layer 145, so that code of the general engine layer 141 and code of the personalized functional layer 143 are isolated. In such a setting, when the code of the personalized functional layer 143

changes, the code of the general engine layer 141 does not change, and a fault of the middle platform system 140 is avoided. Therefore, while high reusability of the general engine layer 141 of the middle platform system 140 is ensured, personalized customization of different tenants 110 is implemented, and stability of the middle platform system 140 is ensured. In addition, for different tenants 110 at the personalized functional layer 143, code of the different tenants 110 is isolated based on a mapping relationship between a tenant identifier and a fully qualified name of code of a computing module. When invoking the computing module, the interface layer 145 invokes the code of the corresponding computing module based on the tenant identifier, to avoid a code invoking error and ensure accuracy of the middle platform system 140. In addition, because code of different tenants 110 is isolated, a code change of one tenant does not lead to a code change of another tenant, to further ensure stability of the middle platform system 140 and reduce a fault rate. In addition, fine scenario division can be performed for a computing module of a tenant 110 in a tree routing manner according to different requirements, and code is compiled on a leaf node in tree routing by using dynamic script. In such a setting, a computing module is divided into submodules in a plurality of scenarios. When a personalized function changes, a changed leaf node can be quickly located based on tree routing, and a code change can be implemented in real time in a dynamic script compilation manner, thereby greatly improving research and development efficiency.

[0094] FIG. 8 is a flowchart of a program modification method P400 for a middle platform system according to an embodiment of this specification. As described above, a computing device 200 can perform the program modification method P400 for a middle platform system in this specification. Specifically, a processor 220 in the computing device 200 can read an instruction set stored in a local storage medium of the computing device 200, and then perform the program modification method P400 for a middle platform system in this specification based on a stipulation of the instruction set. As shown in FIG. 8, the method P400 includes the following steps.

[0095] S410: Receive a modification request of a target tenant for a program of a target service.

[0096] S430: Modify code of a target computing module corresponding to the target tenant at a personalized functional layer 146 based on the modification request.

[0097] Step S410 is similar to the data processing method P300 for a middle platform system. A difference is that, in the method P300, the middle platform system 140 receives a target request for the target service, and in the method P400, the middle platform system 140 receives the

modification request for the program of the target service. Similar to the target request, the modification request carries a target tenant identifier of the target tenant. The modification request further carries modified script code.

[0098] In step S430, the middle platform system 140 modifies code of a target computing module corresponding to the target tenant at a personalized functional layer 143 based on the modification request. Specifically, step S430 can include the following steps.

[0099] S431: Identify the target computing module based on the target tenant identifier. Herein, it is similar to S331 in the method P300. An identification layer 149 of the middle platform system 140 is configured to: receive the modification request, and identify the corresponding target computing module based on the target tenant identifier in the modification request. In step S431, the identification layer 149 in the middle platform system 140 can identify the fully qualified name of the target computing module corresponding to the target tenant based on a mapping relationship between a tenant identifier of a tenant 110 and a fully qualified name of a computing module and the target tenant identifier carried in the modification request. In some embodiments, the middle platform system 140 can further identify a storage path of the target computing module based on a mapping relationship between a fully qualified name of a computing module and a storage path, to accurately locate the target computing module, ensure that code of the target computing module is accurately modified, and code of a computing module of another tenant is not modified.

[0100] S433: Modify the code of the target computing module. As described above, the target computing module includes submodules in a plurality of scenarios. Code of the submodule is implemented by using script, and the script can be implemented in programming languages including but not limited to Python, JavaScript, Java, etc. For the script of the submodule, the administrator or the user can change and modify the script in the configuration management background. The script of the submodule can be dynamically added, deleted, or modified in the configuration management background. The configuration management background allows the administrator to manage the script of submodule in a visual interface. Once the script is updated in the background, the middle platform system 140 can load new script in real time, to implement an immediate change to a function. In addition, the system can record detailed log information, and perform monitoring, to facilitate troubleshooting and performance optimization.

[0101] In some embodiments, the modification request can further carry a to-be-modified scenario identifier. The to-be-modified scenario identifier can be an identifier corresponding to a

sub-scenario of a leaf node that needs to be modified and that is selected by the administrator in a visual interface, or can be a set of scenario identifiers of all nodes on a link on which the sub-scenario of the leaf node is located. The middle platform system 140 can determine, based on the to-be-modified scenario identifier, a submodule that needs to be modified in the target computing module. For example, the middle platform system 140 can locate the leaf node corresponding to the to-be-modified scenario identifier based on tree routing, to determine script on the leaf node, and modify the script.

[0102] According to the method P300 and the method P400, the middle platform system 140 can implement sharing and multiplexing of digital resources. In addition, when a tenant maintains or updates code of a computing module of the tenant, code of the general engine layer is not affected, and a computing module of another tenant does not change, thereby avoiding a fault of the middle platform system 140, and improving high availability, maintainability, and research and development efficiency of the system.

[0103] Another aspect of this specification provides a computer-readable non-transitory storage medium, storing at least one instruction set, to perform data processing and/or program modification for a middle platform system. When the at least one instruction set is executed by a processor, the at least one instruction set instructs the processor to implement the steps of the data processing method P300 and/or program modification method P400 for a middle platform system in this specification. In some possible implementations, aspects of this specification can further be implemented in a form of a program product, including program code. When the program product runs on the computing device 200, the program code is used to enable the computing device 200 to perform the steps of the data processing method P300 and/or program modification method P400 for a middle platform system in this specification. The program product configured to implement the foregoing method can use a portable compact disc read-only memory (CD-ROM) that includes program code, and can run on the computing device 200. However, the program product in this specification is not limited thereto. In this specification, a readable storage medium can be any tangible medium that includes or stores a program, and the program can be used by or in combination with an instruction execution system. The program product can be any combination of one or more readable media. The readable medium can be a readable signal medium or a readable storage medium. The readable storage medium can be, for example, an electrical system, apparatus, or device, a magnetic system, apparatus, or device, an optical system, apparatus, or

device, an electromagnetic system, apparatus, or device, an infrared system, apparatus, or device, or a semiconductor system, apparatus, or device, or any combination thereof. More specific examples of the readable storage medium include: a portable disk, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage component, a magnetic storage component any proper combination thereof. The computer-readable storage medium can include a data signal propagated in a baseband as part of a carrier, and carries readable program code. The propagated data signal can be in a plurality of forms, and includes but is not limited to an electromagnetic signal, an optical signal, or any proper combination thereof. The readable storage medium can further be any readable medium other than the readable storage medium, and the readable medium can be used to send, propagate, or transmit a program used by or in combination with an instruction execution system, apparatus, or component. The program code included in the readable storage medium can be transmitted by using any proper medium, including but not limited to a wireless medium, a wired medium, an optical cable, an RF medium, or any proper combination thereof. Program code for performing an operation of this specification can be written in any combination of one or more program design languages. The program design language includes an object-oriented program design language, for example, Java or C++, and further includes a conventional procedural program design language, for example, "C" language, or a similar program design language. The program code can be completely executed on the computing device 200, partially executed on the computing device 200, executed as an independent software package, partially executed on the computing device 200 and partially executed on a remote computing device, or completely executed on a remote computing device.

[0104] Specific embodiments of this specification are described above. Other embodiments fall within the scope of the appended claims. In some cases, the actions or steps described in the claims can be performed in an order different from that in the embodiments, and the desired results can still be achieved. In addition, the process depicted in the accompanying drawings does not necessarily need a particular sequence or consecutive sequence to achieve the desired results. In some implementations, multitasking and parallel processing are possible or may be advantageous.

[0105] In conclusion, after reading the detailed disclosure, a person skilled in the art can understand that the detailed disclosure can be presented only in an example manner, and can

impose no limitation. Although it is not explicitly stated herein, a person skilled in the art can understand that the requirements of this specification include various proper changes, improvements, and modifications to the embodiments. These changes, improvements, and modifications are intended to be presented in this specification, and fall within the spirit and scope of the example embodiments of this specification.

[0106] In addition, some terms in this specification are used to describe the embodiments of this specification. For example, "one embodiment", "an embodiment", and/or "some embodiments" mean that particular features, structures, or characteristics described with reference to the embodiments can be included in at least one embodiment of this specification. Therefore, it can be emphasized and understood that two or more references to "embodiments", "one embodiment", or "alternative embodiments" in various parts of this specification do not necessarily refer to a same embodiment. In addition, specific features, structures, or characteristics can be properly combined in one or more embodiments of this specification.

[0107] It should be understood that in the above-mentioned descriptions of the embodiments of this specification, to help understand a feature, for the purpose of simplifying this specification, various features are combined in a single embodiment, accompanying drawing, or description thereof in this specification. However, this does not mean that the combination of these features is necessary. It is entirely possible for a person skilled in the art to mark some devices and understand the devices as separate embodiments when reading this specification. That is, the embodiments of this specification can also be understood as an integration of a plurality of secondary embodiments. Content of each secondary embodiment is also true when features are less than all features in one of the above-mentioned disclosed embodiments.

[0108] Each patent, patent application, publication of a patent application, and other materials, for example, articles, books, instructions, publications, documents, and products, that are cited in this specification and that are other than any document that is inconsistent or conflict with this document or that imposes limiting impact on the widest scope of the claims can be incorporated herein by reference and used for all purposes now or later associated with this document. In addition, the term in this document shall prevail when there is any inconsistency or conflict between the description, definition and/or use of a related term in any material and the description, definition and/or use of a related term in this document.

[0109] Finally, it should be understood that the principles of the implementation solutions of

this specification are described in the implementation solutions of the application disclosed in this specification. Other modified embodiments also fall within the scope of this specification. Therefore, the embodiments disclosed in this specification are merely examples and not limitations. A person skilled in the art can use an alternative configuration according to the embodiments of this specification to implement the application in this specification. Therefore, the embodiments of this specification are not limited to the embodiments accurately described in the application.

CLAIMS

What is claimed is:

1. A data processing method for a middle platform system, wherein the middle platform system is configured to provide a target service to a plurality of tenants, the target service comprises a general engine layer and a personalized functional layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer comprises a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the method comprises:

receiving a target request of a target tenant for the target service, wherein the plurality of tenants comprise the target tenant;

invoking, based on the target request, the general engine layer and a target computing module corresponding to the target tenant at the personalized functional layer to execute the target request, to obtain a target result, wherein the plurality of computing modules comprise the target computing module; and

outputting the target result.

2. The method according to claim 1, wherein the interface layer comprises an SPI extension interface.

3. The method according to claim 1, wherein the target request carries a target tenant identifier of the target tenant, and the invoking, based on the target request, the general engine layer and a target computing module corresponding to the target tenant at the personalized functional layer to execute the target request, to obtain a target result comprises:

identifying the target computing module based on the target tenant identifier;

invoking, based on the target request, the general engine layer to execute a general function of the target service;

invoking, by using the interface layer, the target computing module to execute a function corresponding to the target tenant in the target service; and

obtaining the target result.

4. The method according to claim 3, wherein the target computing module comprises

submodules in a plurality of scenarios, the target request further comprises a target scenario identifier corresponding to a target scenario, the plurality of scenarios comprise the target scenario, and the invoking, by using the interface layer, the target computing module to execute a function corresponding to the target tenant in the target service comprises:

invoking, by using the interface layer, a submodule corresponding to the target scenario identifier in the target computing module to execute a function that is of the target tenant and that corresponds to the target scenario in the target service.

5. The method according to claim 4, wherein the submodules in the plurality of scenarios are classified based on a classification rule preset by the target tenant.

6. The method according to claim 5, wherein code of the submodule is implemented by using script.

7. The method according to claim 1, wherein the computing module and a corresponding tenant identifier are stored in a key-value pair.

8. A middle platform system, wherein the middle platform system is configured to provide a target service to a plurality of tenants, the target service comprises a general engine layer and a personalized functional layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer comprises a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the middle platform system comprises:

at least one storage medium, storing at least one instruction set, to provide the target service for the plurality of tenants; and

at least one processor, communicatively connected to the at least one storage medium, wherein when the middle platform system runs, the at least one processor reads the at least one instruction set, and performs the following operations based on an indication of the at least one instruction set:

receiving a target request of a target tenant for the target service, wherein the plurality of tenants comprise the target tenant;

invoking, based on the target request, the general engine layer and a target computing module corresponding to the target tenant at the personalized functional layer to execute the target request,

to obtain a target result, wherein the plurality of computing modules comprise the target computing module; and

outputting the target result.

9. The system according to claim 8, wherein the interface layer comprises an SPI extension interface.

10. The system according to claim 8, wherein the target request carries a target tenant identifier of the target tenant, and to invoke, based on the target request, the general engine layer and the target computing module corresponding to the target tenant at the personalized functional layer to execute the target request, to obtain a target result, the at least one processor performs the following operations:

identifying the target computing module based on the target tenant identifier;

invoking, based on the target request, the general engine layer to execute a general function of the target service;

invoking, by using the interface layer, the target computing module to execute a function corresponding to the target tenant in the target service; and

obtaining the target result.

11. The system according to claim 10, wherein the target computing module comprises submodules in a plurality of scenarios, the target request further comprises a target scenario identifier corresponding to a target scenario, the plurality of scenarios comprise the target scenario, and to invoke, by using the interface layer, the target computing module on a storage path to execute the function corresponding to the target tenant in the target service, the at least one processor performs the following operation:

invoking, by using the interface layer, a submodule corresponding to the target scenario identifier in the target computing module on the storage path to execute a function that is of the target tenant and that corresponds to the target scenario in the target service.

12. The system according to claim 11, wherein the submodules in the plurality of scenarios are classified based on a classification rule preset by the target tenant.

13. The system according to claim 12, wherein code of the submodule is implemented by using script.

14. The system according to claim 8, wherein the computing module and a corresponding tenant identifier are stored in a key-value pair.

15. A program modification method for a middle platform system, wherein the middle platform system is configured to provide a target service to a plurality of tenants, the target service comprises a general engine layer and a personalized functional layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer comprises a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the method comprises:

receiving a modification request of a target tenant for a program of the target service, wherein the plurality of tenants comprise the target tenant; and

modifying code of a target computing module corresponding to the target tenant at the personalized functional layer based on the modification request.

16. The method according to claim 15, wherein the modification request carries a target tenant identifier of the target tenant, and the modifying code of a target computing module corresponding to the target tenant at the personalized functional layer based on the modification request comprises:

identifying the target computing module based on the target tenant identifier; and

modifying the code of the target computing module.

17. A middle platform system, wherein the middle platform system is configured to provide a target service to a plurality of tenants, the target service comprises a general engine layer and a personalized functional layer, the general engine layer is configured to provide the target service to the plurality of tenants, the personalized functional layer comprises a plurality of computing modules corresponding to the plurality of tenants, a computing module is configured to provide the target service to a corresponding tenant, the computing module is configured by the corresponding tenant, invoking between the general engine layer and the personalized functional layer is implemented by using an interface layer, and the middle platform system comprises:

at least one storage medium, storing at least one instruction set, to perform program modification; and

at least one processor, communicatively connected to the at least one storage medium, wherein when the middle platform system runs, the at least one processor reads the at least one instruction set, and performs the following operations based on an indication of the at least one instruction set:

receiving a modification request of a target tenant for a program of the target service, wherein the plurality of tenants comprise the target tenant; and

modifying code of a target computing module corresponding to the target tenant at the personalized functional layer based on the modification request.

18. The system according to claim 17, wherein the modification request carries a target tenant identifier of the target tenant, and to modify the code of the target computing module corresponding to the target tenant at the personalized functional layer based on the modification request, the at least one processor performs the following operations:

identifying the target computing module based on the target tenant identifier; and

modifying the code of the target computing module.

19. A computer-readable non-volatile storage medium, wherein the computer-readable non-volatile storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the data processing method according to any one of claims 1 to 8 is implemented.

20. A computer-readable non-volatile storage medium, wherein the computer-readable non-volatile storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the program modification method according to either claim 15 or claim 16 is implemented.