

PAGE JUMP METHOD AND APPARATUS

[0001] The present application claims priority to Chinese Patent Application No. 201510894292.2, filed on November 27, 2015 and entitled "PAGE JUMP METHOD AND APPARATUS", which is incorporated here by reference in its entirety.

5

TECHNICAL FIELD

[0002] The present application relates to field of communications technologies, and in particular, to a page jump method and apparatus.

BACKGROUND

[0003] An application (APP) can provide a user with rapid and stable services
10 such as account recharging services, by using a native page. The native page is independently executable based on local code of the APP that does not rely on HyperText Markup Language (HTML5) code transmitted through a network.

[0004] However, when a new service is deployed, and is provided to the user by
using a native page, it is possible that the APP may need to be upgraded to a newer
15 version before the service can be provided, which can be inflexible.

SUMMARY

[0005] The present application provides a page jump method, where the method includes:

obtaining a page identifier of a page to be jumped from;
20 searching a page routing table for a corresponding jump route based on the obtained page identifier, where the jump route includes a mapping relationship between the page identifier and a jump page; and
jumping to the jump page based on the identified jump route.

[0006] Optionally, the method further includes:
25 periodically sending a first query request to a server, where the first query

request is used to trigger the server to deliver the latest jump route;

receiving the latest jump route delivered by the server; and

updating the page routing table based on the received latest jump route.

[0007] Optionally, the method further includes:

5 periodically sending a second query request to the server, where the second query request is used to trigger the server to deliver page data of the latest jump page offline;

receiving the page data of the latest jump page delivered by the server offline; and

10 updating a local page database based on the received page data of the latest jump page.

[0008] Optionally, the jumping to the jump page based on the identified jump route includes:

15 obtaining a jump page corresponding to the page identifier based on the identified jump route;

reading page data of the jump page; and

jumping to the jump page, and loading the read page data onto the jump page to complete a jump.

[0009] Optionally, the reading page data of the jump page includes:

20 determining whether a local page database stores the page data of the jump page offline;

reading the page data of the jump page from the page database when the page database stores the page data of the jump page offline; and

25 requesting the page data of the jump page from a server when the page database does not store the page data of the jump page.

[0010] The present application further provides a page jump apparatus, where the apparatus includes:

an acquisition module, configured to obtain a page identifier of a page to be jumped from;

30 a searching module, configured to search a page routing table for a corresponding jump route based on the obtained page identifier, where the jump route includes a mapping relationship between the page identifier and a jump page; and

a jump module, configured to jump to the jump page based on the identified jump route.

[0011] Optionally, the apparatus further includes:

a sending module, configured to periodically send a first query request to a server, where the first query request is used to trigger the server to deliver the latest jump route;

5 a receiving module, configured to receive the latest jump route delivered by the server; and

an updating module, configured to update the page routing table based on the received latest jump route.

[0012] Optionally, the sending module is further configured to:

10 periodically send a second query request to the server, where the second query request is used to trigger the server to deliver page data of the latest jump page offline;

the receiving module is further configured to:

receive the page data of the latest jump page delivered by the server offline;

15 and

the updating module is further configured to:

update a page database based on the received page data of the latest jump page.

[0013] Optionally, the jump module is configured to:

20 obtain a jump page corresponding to the page identifier based on the identified jump route;

read page data of the jump page; and

jump to the jump page, and load the read page data onto the jump page to complete a jump.

25 **[0014]** Optionally, the jump module is further configured to:

determine whether a local page database stores the page data of the jump page offline;

read the page data of the jump page from the page database when the page database stores the page data of the jump page offline; and

30 request the page data of the jump page from a server when the page database does not store the page data of the jump page.

[0015] In the present application, the page identifier of the page to be jumped from is obtained, the page routing table is searched for the corresponding jump route based on the obtained page identifier, and then the corresponding page is jumped to

based on the identified jump route, so as to dynamically and flexibly perform the page jump based on the jump route identified from the page routing table, so that a new service can be quickly and flexibly deployed without needing to upgrade a version of an APP.

5

BRIEF DESCRIPTION OF DRAWINGS

[0016] FIG. 1 is a flowchart illustrating a page jump method, according to an implementation of the present application;

[0017] FIG. 2 is a schematic diagram illustrating interactions between a client and a server, according to an implementation of the present application;

10 [0018] FIG. 3 is a logical block diagram illustrating a page jump apparatus, according to an implementation of the present application; and

[0019] FIG. 4 is a structural diagram of hardware illustrating a server that includes a page jump apparatus, according to an implementation of the present application.

DESCRIPTION OF IMPLEMENTATIONS

15 [0020] In related technologies, an APP can provide a user with fast and stable services such as account recharge services, by using native pages. However, when a new service is deployed on the APP, such as services operating in a special time period, if the services do not need to be run with high stability, and the service is provided for the user by using the native page, it is possible that update to the APP
20 may be needed to provide the services. In addition, some of the services are temporarily provided. After the operation service ends, the APP can require the user to restore to the original version. As such, the user can be frequently required to update the APP versions, and the user experience can be poor.

[0021] In related technologies, the previous problem can be resolved by
25 introducing a hybrid technology, that is, by mixing the native page and an HTML5 page in the APP. When a new service needs to be deployed on the APP, the APP can download an HTML5 page of the new service deployed on a server to the APP to dynamically replace an original HTML5 page, and then load the HTML5 page to run the new service. However, with this solution, because the native page in the APP
30 usually cannot be replaced, only the HTML5 page in the APP can be dynamically

replaced when the new service is deployed on the APP. However, the HTML5 page is less stable than the native page. As the number of HTML5 pages increase in the APP, operation stability of the APP can be affected to cause poor user experience.

[0022] Therefore, to improve user experience, the number of HTML5 pages in the APP needs to be reduced. When the new service is deployed on the APP, hard coded jump logic in the APP can be dynamically modified, so that the native page can be jumped to the HTML5 page of the new service, and a service on the native page can be replaced by the new service.

[0023] For example, assume that the new service is an operation service on the Taobao client APP in the Singles' Day. During operation of the service, a rich text page that includes a picture, a sound, and a text needs to be loaded onto a home page of the APP. However, this usually cannot be implemented by using a native page of the APP. Therefore, the APP can download, from the server, the rich text page that needs to be displayed. Then, an Lua script can be edited to patch the APP, and hard coded jump logic in the APP can be modified to jump to the rich text page. For example, assume that the original jump logic in the APP jumps from page A to page B, and the rich text page is page C, the Lua script can be edited to modify the hard coded jump logic from jumping from page A to page B to jumping from page A to page C.

[0024] However, to complete a page jump by dynamically modifying the hard coded jump logic in the APP, code of the APP needs to be dynamically modified. Consequently, a certain security risk can exist. For example, after the code of the APP is dynamically modified, the APP can be rejected due to security risks. In addition, to complete the page jump by dynamically modifying the hard coded jump logic in the APP, the APP needs to continuously load patches, which can be inflexible.

[0025] In view of this, the present application provides a page jump method, including: obtaining a page identifier of a page to be jumped from; searching a page routing table for a corresponding jump route based on the obtained page identifier, where the jump route includes a mapping relationship between the page identifier and a jump page; and jumping to a corresponding page based on the identified jump route. As such, a page jump can be dynamically and flexibly performed by using the jump route identified from the page routing table, so that a new service can be quickly and flexibly deployed without needing to update the corresponding APP. In addition, in the present application, when the page jump is performed, jump logic in the APP does not need to be modified. Therefore, the APP will not be rejected due to security risks.

[0026] The following describes the present application by using specific implementations and with reference to a specific application scenario.

[0027] Referring to FIG. 1, FIG. 1 is a page jump method according to an implementation of the present application. The method is applied to a client, and the method includes the following steps:

[0028] Step 101: Obtain a page identifier of a page to be jumped from.

[0029] Step 102: Search a page routing table for a corresponding jump route based on the obtained page identifier, where the jump route includes a mapping relationship between the page identifier and a jump page.

[0030] Step 103: Jump to a corresponding page based on the identified jump route.

[0031] In this implementation, the client can be client software that provides a service for a user. For example, the client can be the Taobao client APP. The page identifier of the page to be jumped from can be included in jump logic predetermined in the client, and the jump logic can be hard coded in advance in a program of the client that does not change.

[0032] For example, assume that hard coded jump logic in the program of the client is jumping from page A to page B, and a page identifier of page B is viewid1. As such, viewid1 is a page identifier of a page to be jumped from that is included in the jump logic.

[0033] The page identifier of the page to be jumped from in the jump logic can correspond to different jump pages in different service scenarios.

[0034] In this implementation, a page routing table can be locally precreated in the client, and the page routing table can be used to store a jump route delivered by a server. A jump route in the page routing table can include the mapping relationship between the page identifier and a jump page.

[0035] For different services, the server can deliver different jump routes to the client, so that one page identifier can be associated to different jump pages.

[0036] For example, the client is the Taobao client. When the Taobao client executes a routing service, a local page routing table can store a jump route in the following that corresponds to the routine service:

Page identifier of a page to be jumped from	Jump page
viewid1	page A

[0037] The jump route indicates that when executing the routine service, the client jumps to page A when executing jump logic corresponding to viewid1.

[0038] When an operation service in the Singles' Day is executed in the Taobao client, a rich text page B that includes a picture, a sound, and a text needs to be loaded onto a home page of the client. As such, the server can deliver, to the client, a jump route in the following that corresponds to the operation service:

Page identifier of a page to be jumped from	Jump page
viewid1	page B

[0039] The jump route indicates that when executing the operation service, the client jumps to the rich text page B when executing the jump logic corresponding to viewid1.

[0040] In this implementation, a page database can further be locally precreated in the client, and the page database can be used to store page data of the latest jump page delivered by the server offline.

[0041] The client can interact with the server to update the local page routing table and the page database.

[0042] Referring to FIG. 2, FIG. 2 is a schematic diagram illustrating interactions between a client and a server, according to an implementation.

[0043] In this implementation, the server can include a routing table server and a service server.

[0044] The routing table server is responsible for interacting with the client to deliver the latest jump route to the client. The service server is responsible for interacting with the client to deliver page data of the latest jump page to the client. Certainly, during implementation, the server can alternatively not be divided into the routing table server and the service server. That is, a function of the routing table server and a function of the service server can be integrated into one physical server.

[0045] As shown in FIG. 2, the client can periodically send a first query message to the routing table server, and a time interval is not limited in this implementation. For example, the first query message can be periodically sent to the routing table server once every 15 seconds. The first query message can be used to query whether the routing table server has the latest jump route, and trigger the routing table server

to deliver the latest jump route to the client.

[0046] A new service can be predeployed on the service server. After deployment of the new service is completed, the service server can actively synchronize a jump route related to the new service to the routing table server. After receiving the jump route related to the new service and synchronized by the service server, the routing table server can locally store the received jump route. Certainly, during implementation, the jump route related to the new service can alternatively be directly deployed on the routing table server.

[0047] After receiving the first query message sent by the client, the routing table server can determine whether the jump route related to the new service is locally stored. If the jump route related to the new service is locally stored, the routing table server can immediately deliver the latest jump route corresponding to the new service to the client.

[0048] Certainly, in the previous description, the client sends the first query message to trigger the routing table server to deliver the latest jump route to the client. In addition, during implementation, the routing table server can actively deliver, to the client, the jump route related to the new service and synchronized by the service server to the routing table server.

[0049] For example, after receiving the jump route related to the new service and synchronized by the service server, the routing table server can be triggered to actively deliver the jump route related to the new service to the client. That is, each time after a new service is deployed on the service server, the routing table server is triggered to actively deliver a jump route related to the new service to the client.

[0050] In addition, to improve security, when the routing table server delivers the latest jump route to the client, the routing table server can further transmit the latest jump route in an encrypted way, or encrypt in advance the latest jump route related to the new service and stored in the background. After receiving the latest jump route delivered by the routing table server, the client can decrypt the received jump route, and then store the jump route in a page routing table to update the page routing table.

[0051] Still referring to FIG. 2, if no new service is currently deployed on the service server, and the routing table server does not locally store a jump route related to the new service, the client can obtain no latest jump route by querying. The routing table server can return a null value to the client, indicating that no new service is currently deployed. As such, the client can load, onto the page routing table, a default

jump route or a previously received jump route delivered by the routing table server.

[0052] As such, it can be ensured that all jump routes stored in the page routing table are the latest jump routes. After a new service is deployed, the client can store, the latest jump route that corresponds to the new service in the page routing table
5 delivered by the routing table server, so as to update the page routing table.

[0053] Still referring to FIG. 2, the client can further periodically send a second query message to the service server, and the second query message can be used to query whether the service server has the page data of the latest jump page, and trigger the service server to deliver the page data of the latest jump page to the client.

10 **[0054]** Page data of a jump page related to the new service can be predeployed on the service server. After receiving the second query message, the service server can determine whether a new service is currently deployed. If a new service is deployed, the service server can immediately deliver page data of the latest jump page corresponding to the new service to the client offline. For example, the service server
15 can deliver the page data of the latest jump page to the client by using an offline data packet.

[0055] In the previous description, the client sends the second query message to trigger the service server to send the page data of the latest jump page to the client. Certainly, in addition, during implementation, the service server can actively deliver
20 the page data of the latest jump page related to the new service to the client.

[0056] For example, after the new service is deployed on the service server, the service server can be triggered to actively deliver the page data of the latest jump page related to the new service to the client. That is, each time after a new service is deployed on the service server, the service server is triggered to actively deliver page
25 data of the latest jump page related to the new service to the client.

[0057] In addition, to improve security, when delivering the page data of the latest jump page to the client offline, the service server can further transmit the page data of the latest jump page in an encrypted way, or encrypt in advance the page data of the latest jump page related to the new service and stored in the background. After
30 receiving the page data of the latest jump page delivered by the service server, the client can decrypt the received page data of the latest jump page, and then store the page data of the latest jump page in a page database to update the page database.

[0058] Still referring to FIG. 2, if no new service is deployed on the service server, the client can obtain no page data of the latest jump page by querying, and the service

server can return a null value to the client, indicating that no new service is currently deployed. As such, the client can load, onto the page database, page data of a default jump page or the latest page data of a jump page delivered by the service server.

[0059] As such, it can be ensured that all page data stored in the page database are page data of the latest jump page. After jump data of a jump page related to a new service is deployed on the service server, the client can store page data of the latest jump page corresponding to the new service delivered by the service server to the page database, so as to update the page database.

[0060] With reference to FIG. 2, the previous describes a process in which the client interacts with the server to update a locally created page routing table and a locally created page database.

[0061] The following describes a process in which the client completes a page jump of a new service based on the local page routing table and the local page database.

[0062] In this implementation, when performing the page jump, the client can still complete the page jump by executing specified jump logic.

[0063] The client can be triggered by monitoring a foreground operation of a user, to execute the specified jump logic.

[0064] In a shown application scenario, a corresponding jump control can be provided on an operation interface of the client in advance. For example, the jump control can include a jump button, a jump link, etc. When the user performs a corresponding service operation on the operation interface of the client, the client can monitor a trigger event of the user on the jump control on the operation interface of the client in real time. For example, the client is loaded onto a touchscreen terminal, and the trigger event can be a click event on the jump control. When the trigger event on the jump control on the operation interface of the client is monitored, the client can be triggered to invoke and execute the specified jump logic, so as to complete the corresponding page jump.

[0065] In addition, it is worthwhile to note that the specified jump logic described in this implementation is different from jump logic in existing implementation as to certain details.

[0066] In the existing implementation, jump logic specified in a client usually includes only a predetermined page to be jumped from. After executing the jump logic, the client can directly jump to a corresponding jump page based on the page to be

jumped from specified in the jump logic, so as to complete the jump.

[0067] However, in this implementation, in addition to a page identifier of a predetermined page to be jumped from, the jump logic specified in the client can include searching logic for searching the page routing table for a corresponding jump
5 route based on the page identifier of the page to be jumped from.

[0068] When executing the specified jump logic, the client can first read the page identifier of the page to be jumped from from the jump logic, and then search the page routing table for the corresponding jump route based on the page identifier of the page to be jumped from, and then complete the page jump based on the identified jump
10 route after identifying the corresponding jump route.

[0069] In this implementation, after the client identifies the corresponding jump route by searching the page routing table based on the page identifier of the page to be jumped from, if a new service is currently deployed on the service server, each jump route stored in the page routing table is the latest jump route. As such, the jump route
15 that corresponds to the page identifier and that is identified by the client is the latest jump route corresponding to the new service.

[0070] Certainly, if no new service is deployed on the server as such, a jump route stored in the page routing table is still a prestored default jump route, or the latest jump route delivered by the routing table server.

[0071] As such, after the client identifies the latest jump route from the page routing table, the client can read the latest jump route, and obtain a jump page corresponding to the page identifier of the page to be jumped from based on a mapping relationship between the page identifier of the page to be jumped from and a jump page and stored in the latest jump route. As such, the obtained jump page is a
20 jump page related to a new service.

[0072] After obtaining the jump page corresponding to the page identifier of the page to be jumped from, the client can jump to the jump page, and load, onto the jump page, page data that corresponds to the jump page and that is read from the local page database, so as to complete the page jump.

[0073] The jump page corresponding to the page identifier of the page to be jumped from can be a native page of an APP or can be an HTML5 page. This is not limited in this implementation. That is, in this implementation, a jump route delivered by the server can implement a page jump between a native page and an HTML5 page without any limitation.
30

[0074] In this implementation, when reading the page data corresponding to the jump page from the local page database, the client can search the page database for the corresponding page data by using a page identifier of the latest jump page as an index, so as to determine whether the local page database stores the page data
5 corresponding to the latest jump page.

[0075] If the corresponding page data is identified, the client can determine that the local page database stores the page data corresponding to the latest jump page offline. As such, the client can directly read the page data from the local page database.

10 [0076] Certainly, if the client determines that the page database does not store the page data corresponding to the latest jump page offline, the client can request the page data of the latest jump page from the server. For example, the client can send, to the server, a page data acquisition request that includes a URL address or another identifier of the jump page, to request the page data of the jump page from the server.

15 [0077] When the client directly reads the page data corresponding to the latest jump page from the local page database, the page data is page data of a new service deployed on the server, and the client can load the read page data of the new service onto the latest jump page, so as to complete the page jump. After the client completes the page jump based on the read page data of the new service, the client starts to
20 execute the new service deployed on the server.

[0078] It is worthwhile to note that when loading the page data of the new service onto the latest jump page, the client can transmit the page data by using a protocol, for example, by using an agreed protocol interface.

[0079] In this implementation, if another new service further needs to be deployed
25 on the server, a new jump route can be sent to the client to trigger the client to execute a specified jump logic to jump to a jump page of the new service, so as to execute the new service.

[0080] Certainly, if execution of the new service deployed on the server is completed, and no other new service is deployed on the server, the server can further
30 send, to the client, a trigger message that triggers the client to restore a routine service. After receiving the trigger message, the client can separately restore the local page routing table and the local page database by using the default jump route and the page data of the default jump page. When restoring succeeds, the client executes the same jump logic again, and then jumps to the default page to execute the routine service.

[0081] As such, when a new service is deployed on the server, by interacting with the server, the client can store a jump route of the new service in the local page routing table immediately, and store page data of a jump page of the new service in the local page database.

5 [0082] Therefore, after executing corresponding jump logic, the client can read a page identifier of a page to be jumped from the jump logic, and search the local page routing table for the latest jump route corresponding to the jump logic of the page identifier. After identifying the corresponding latest jump route, the client can read a corresponding jump page from the latest jump route, read page data of the jump
10 page from the local page database, and jump to the corresponding page based on the read page data.

[0083] During the whole process, the page identifier of the page to be jumped from in the jump logic does not change, the server can dynamically deliver a new jump route, so that the page identifier of the page to be jumped from dynamically
15 corresponds to different jump pages. Therefore, a service can be more flexibly deployed on the server without needing to update the APP to a different version.

[0084] In addition, in the present application, when the client performs the page jump, the page jump is implemented by dynamically delivering a jump route by the server, and jump logic in the APP does not need to be modified. Therefore, the APP
20 does not need to continuously load patches, which makes it more flexible. In addition, because the jump logic in the APP is not modified any more, the APP will not face a risk of being rejected due to security risks.

[0085] The following describes the technical solutions in the previous implementation by using a specific application instance.

25 [0086] Assume that the client is the Taobao client. When the Taobao client executes a routine service, a local page routing table can store a jump route in Table 1 that corresponds to the routine service:

Table 1

Page identifier of a page to be jumped from	Jump page
viewid1	page A

30 [0087] The jump route indicates that when executing the routine service, the client

jumps to page A when executing jump logic corresponding to viewid1.

[0088] In addition, a local page database can also correspondingly store page data of page A.

[0089] Assume that a new service is deployed on the server, and the new service
5 is an operation service in the Singles' Day, the client can periodically send the first query message and the second query message to the server, so as to trigger the server to deliver a jump route of the new service and page data of a jump page corresponding to the jump route to the client.

[0090] When the client executes the new service, a rich text page B that includes a
10 picture, a sound, and a text needs to be loaded onto a home page of the client. As such, the server can deliver a jump route in Table 2 that corresponds to the operation service to the client:

Table 2

Page identifier of a page to be jumped from	Jump page
viewid1	page B

15 [0091] The jump route indicates that when executing the operation service, the client jumps to the rich text page B when executing the jump logic corresponding to viewid1. As such, jump logic in the client does not change, and a jump page corresponding to a page identifier of a page to be jumped from specified in the jump logic changes.

20 [0092] After receiving the jump route of the new service delivered by the server as shown in Table 2, the client can update, by using the jump route of the new service, the jump route that corresponds to the routine route and that is stored in the local page routing table as shown in Table 1.

[0093] In addition, the server can further send page data of page B to the client.
25 After receiving the page data of page B, the client can update the page data of page A stored in the local page database.

[0094] When both the local page routing table and the local page database are updated, the client executes the same jump logic again. After reading viewid1 from the jump logic, the client can identify the jump page B corresponding to viewid1 from
30 the local page routing table. After identifying page B, the client can jump to page B,

read the page data of page B from the page database, load the read page data of page B onto page B, and then jump to page B. It can be seen that during the whole process, a process of delivering a jump route of a new service is completely unperceived by the user. This helps to improve user experience.

- 5 **[0095]** When execution of the new service is completed, if another new service is redeployed on the server, the server can deliver a jump route of the another new service to the client again in the same way. Certainly, if execution of the new service is completed, and no other new service is deployed on the server, the server can also send, to the client, a trigger message that triggers the client to restore a routine service.
- 10 After receiving the trigger message, the client can restore the jump route shown in Table 1 from the jump route stored in the local page routing table and shown in Table 2. In addition, after receiving the trigger message, the client can further restore the page data of page A from the page data of page B stored in the local page database. When restoring is completed, the client executes the same jump logic again, and then
- 15 jumps to page A to execute the routine service.

- [0096]** In the previous implementation, the page identifier of the page to be jumped from is obtained, the page routing table is searched for the corresponding jump route based on the obtained page identifier, and then the corresponding page is jumped to based on the identified jump route, so as to dynamically and flexibly
- 20 perform the page jump based on the jump route identified from the page routing table. As such, the new service can be quickly and flexibly deployed without needing to upgrade the version of the APP. In addition, in the present application, when the page jump is performed, the jump logic in the APP does not need to be modified. Therefore, the APP will not face a risk of being rejected due to the security risk.

- 25 **[0097]** Corresponding to the previous method implementation, the present application further provides an apparatus implementation.

- [0098]** Referring to FIG. 3, the present application provides a page jump apparatus 30, applied to a client. Referring to FIG. 4, a hardware architecture related to the client that includes the page jump apparatus 30 usually includes a CPU, a memory, a nonvolatile memory, a network interface, a local bus, etc. For example, during
- 30 software implementation, the page jump apparatus 30 can usually be understood as a logical apparatus that combines software and hardware and that is formed after a computer program loaded into memory runs in the CPU. The apparatus 30 includes:

an acquisition module 301, configured to obtain a page identifier of a page

to be jumped from; a searching module 302, configured to search a page routing table for a corresponding jump route based on the obtained page identifier, where the jump route includes a mapping relationship between the page identifier and a jump page; and a jump module 303, configured to jump to the jump page based on the identified
5 jump route. In this implementation, the page identifier of the page to be jumped from can be included in jump logic predetermined in the client, and the jump logic can be hard coded in advance in a program of the client that does not change.

[0099] The page identifier of the page to be jumped from in the jump logic can separately correspond to different jump pages in different service scenarios.

10 **[0100]** In this implementation, the apparatus 30 further includes:

a sending module 304, configured to periodically send a first query request to a server, where the first query request is used to trigger the server to deliver the latest jump route;

a receiving module 305, configured to receive the latest jump route
15 delivered by the server; and

an updating module 306, configured to update the page routing table based on the received latest jump route.

[0101] In this implementation, a page routing table can be locally precreated in the client, and the page routing table can be used to store the jump route delivered by
20 the server. A jump route in the page routing table can include the mapping relationship between the page identifier and a jump page. For different services, the server can deliver different jump routes to the client, so that one page identifier separately corresponds to different jump pages.

[0102] The client can interact with the server by sending the first query request to
25 the server, so as to update the local page routing table.

[0103] Optionally, the sending module 304 is further configured to:

periodically send a second query request to the server, where the second query request is used to trigger the server to deliver page data of the latest jump page offline;

30 the receiving module 305 is further configured to:

receive the page data of the latest jump page delivered by the server offline;

and

the updating module 306 is further configured to:

update a page database based on the received page data of the latest jump

page.

[0104] In this implementation, a page database can further be locally precreated in the client, and the page database can be used to store page data of the latest jump page delivered by the server offline.

5 **[0105]** The client can interact with the server by sending the second query request to the server, so as to update the local page routing table.

[0106] In this implementation, the jump module 303 is configured to:

obtain a jump page corresponding to the page identifier based on the identified jump route;

10 read page data of the jump page; and

jump to the jump page, and load the read page data onto the jump page to complete a jump.

[0107] In this implementation, when performing a page jump, the client can still complete the page jump by executing specified jump logic. In a shown application
15 scenario, a corresponding jump control can be provided on an operation interface of the client in advance. For example, the jump control can include a jump button, a jump link, etc. When the user performs a corresponding service operation on the operation interface of the client, the client can monitor a trigger event of the user on the jump control on the operation interface of the client in real time. For example, the
20 client is loaded onto a touchscreen terminal, and the trigger event can be a click event on the jump control. When the trigger event on the jump control on the operation interface of the client is monitored, the client can immediately be triggered to invoke and execute the specified jump logic, so as to complete the corresponding page jump.

[0108] After the client identifies the corresponding jump route by searching the
25 page routing table based on the page identifier of the page to be jumped from, if a new service is currently deployed on the service server, each jump route stored in the page routing table is the latest jump route. As such, the jump route that corresponds to the page identifier and that is identified by the client is the latest jump route corresponding to the new service.

30 **[0109]** As such, after the client identifies the latest jump route from the page routing table, the client can read the latest jump route, obtain a jump page corresponding to the page identifier of the page to be jumped from based on a mapping relationship between the page identifier of the page to be jumped from and a jump page and stored in the latest jump route, jump to the jump page, and load, onto

the jump page, page data that corresponds to the jump page and that is read from the local page database, so as to complete the page jump.

[0110] In this implementation, the jump module 303 is further configured to:

5 determine whether a local page database stores the page data of the jump page offline;

read the page data of the jump page from the page database when the page database stores the page data of the jump page offline; and

request the page data of the jump page from a server when the page database does not store the page data of the jump page.

10 **[0111]** In this implementation, when reading the page data corresponding to the jump page from the local page database, the client can search the page database for the corresponding page data by using a page identifier of the latest jump page as an index, so as to determine whether the local page database stores the page data corresponding to the latest jump page.

15 **[0112]** If the corresponding page data is identified, the client can determine that the local page database stores the page data corresponding to the latest jump page offline. As such, the client can directly read the page data from the local page database.

[0113] Certainly, if the client determines that the page database does not store the page data corresponding to the latest jump page offline, the client can request the page data of the latest jump page from the server. For example, the client can send, to the server, a page data acquisition request that includes a URL address or another identifier of the jump page, to request the page data of the jump page from the server.

20 **[0114]** When the client directly reads the page data corresponding to the latest jump page from the local page database, the page data is page data of a new service deployed on the server, and the client can load the read page data of the new service onto the latest jump page, so as to complete the page jump.

25 **[0115]** A person skilled in the art can easily figure out another implementation solution of the present application after considering the specification and practicing the disclosure disclosed here. The present application is intended to cover any variations, functions, or adaptive changes of the present application. These variations, functions, or adaptive changes comply with general principles of the present application, and include common knowledge or a commonly used technical means in the technical field not disclosed in the present application. The specification and the

implementations are merely considered as examples, and the actual scope and the spirit of the present application are pointed out by the following claims.

[0116] It should be understood that the present application is not limited to the accurate structures described above and shown in the accompanying drawings, and
5 modifications and changes can be made without departing from the scope of the present application. The scope of the present application is limited only by the appended claims.

[0117] The previous descriptions are merely example implementations of the present application, but are not intended to limit the present application. Any
10 modification, equivalent replacement, or improvement made without departing from the spirit and principle of the present application should fall within the protection scope of the present application.

CLAIMS

1. A computer-implemented method executed by one or more processors, the method comprising:
 - receiving, by the one or more processors, a request for a first service;
 - determining, by the one or more processors, a first jump page associated with the first service, wherein the first jump page is associated with a page identifier;
 - retrieving, based on the page identifier and by the one or more processors, a first jump route from a page routing table, wherein the first jump route is associated with the first service, and wherein the first jump route includes the page identifier and represents a route for retrieving the first jump page;
 - in response to retrieving the first jump route, loading, by the one or more processors, the first jump page based on the first jump route;
 - receiving, by the one or more processors, a second jump route from a jump route server, wherein the second jump route includes the page identifier and represents a route for retrieving a second jump page, the second jump route being associated with a deployed second service that is different from the first service; and
 - updating, by the one or more processors, the page routing table based on the second jump route.
2. The method of claim 1, further comprising
 - loading, by the one or more processors, the second jump page based on the second jump route.
3. The method of claim 1, wherein the second jump route is received in response to sending a first query to the jump route server, wherein the jump route server is configured to transmit the second jump route based on the deployed second service and in response to the first query.
4. The method of claim 1, wherein the one or more processors are associated with a client computing system.

5. The method of claim 1, wherein loading the first jump page comprises:
determining, by the one or more processors, that a local page database lacks page data of the first jump page, and in response, requesting the page data of the first jump page from a page data server, wherein the local page database is configured to store the page data of the first jump page.
6. The method of claim 5, further comprising:
receiving, by the one or more processors, the page data of the first jump page from the page data server; and
updating, by the one or more processors, the local page database to include the page data of the first jump page.
7. The method of claim 1, wherein loading the first jump page comprises:
determining, by the one or more processors, that page data of the first jump page is stored in a local page database, and in response, retrieving the page data of the first jump page from the local page database.
8. A non-transitory, computer-readable medium storing one or more instructions executable by one or more processors to perform the method of any one of claims 1 to 7.
9. A computer-implemented system, comprising:
one or more processors; and
one or more computer memory devices interoperably coupled with the one or more processors and having tangible, non-transitory, machine-readable media storing one or more instructions that, when executed by the one or more processors, perform the method of any one of claims 1 to 7.