Data Backfilling Method and Means

Field of the Invention

The present application relates to a field of Internet technology. In particular, it relates to a data backfilling method and means.

Background of the Invention

When users at a web page execute operations such as account registration and filling in account information, they need to enter the appropriate data in input boxes on the web page. Examples include account name, account password, contact information and other such information. For information classification reasons, the various kinds of information that they enter are allocated to different web pages. For example, after filling in “Name” and “Age” in the first page, a user skips to the second page by clicking “Next” on the first page. At the same time, the content entered in the first page is submitted to a server. If, while performing an operation on the second page, the user wishes to modify the first page “Name,” the user may click “Previous” on the second page to return to the first page.

When the user returns to the first page, the browser needs to take “Name,” “Age” and other such content previously entered by the user and backfill them into the corresponding input boxes of the first page so that the user may conveniently view and modify them.

However, in actual applications, when Chinese characters occur in the content entered by the user, “garbled characters” will often appear when backfilling takes place. As a result, the user’s experience is adversely affected, for the user is unable to have a normal view of the content that was entered earlier.

Summary of the Invention

In view of the above, the present application provides a method and a means for data backfilling that can smoothly achieve backfilling operations for page data and which will not let garbled characters appear in the backfill content.

To achieve the objective stated above, the present application puts forward the following technical schemes:

in accordance with a first aspect of the present application, it puts forward a data backfilling method, which comprises:

associatively sending user input data in the first page of a terminal and the identifiers of the input boxes in said first page to a server for them to be stored by said server, the stored content mapping to the sessionID of said terminal and to the identifier of said first page;

when said first page is returned to from a second page, sending a data acquisition request to said server, said data acquisition request comprising the sessionID of said terminal and the identifier of said first page;

receiving said first page user input data and the associated identifiers of the input boxes in said first page sent back by said server according to said data acquisition request, and taking said first page user input data and backfilling it into the corresponding input boxes in said first page.

A data backfilling method put forward in accordance with a second aspect of the present application. It comprises:

associatively storing the first page user input data sent by the terminal and the identifiers of the input boxes in said first page, with the stored content mapping to the session ID of said terminal and the identifier of said first page;

receiving the data acquisition request sent by said terminal when said first page is returned to from the second page;

using the sessionID for said terminal and the identifier for said first page contained in said data acquisition request as a basis for sending back said first page user input data and the associated identifiers for the input boxes in said first page to said terminal so that said terminal can backfill said first page user input data into the corresponding input boxes in said first page.

A data backfilling means put forward in accordance with a third aspect of the present application. It comprises:

a sending unit, which associatively sends user input data in the first page of the terminal and the identifiers of the input boxes in said first page to a server for them to be stored by said server, with the stored content mapping to the sessionID of said terminal and to the identifier of said first page;

a requesting unit, which, when said first page is returned to from a second page, sends a data acquisition request to said server, said data acquisition request comprising the sessionID of said terminal and the identifier of said first page;

a backfilling unit, which receives said first page user input data and the associated identifiers of the input boxes in said first page sent back by said server according to said data acquisition request, and which takes said first page user input data and backfills it into the corresponding input boxes in said first page.

A data backfilling means put forward in accordance with a fourth aspect of the present application. It comprises:

a data storage unit, which associatively stores the first page user input data sent by the terminal and the identifiers of the input boxes in said first page, with the stored content mapping to the session ID of said terminal and the

identifier of said first page;

a request receiving unit, which receives the data acquisition request sent by said terminal when said first page is returned to from the second page;

a data returning unit, which uses the sessionID for said terminal and the identifier for said first page contained in said data acquisition request as a basis for sending back said first page user input data and the associated identifiers for the input boxes in said first page to said terminal so that said terminal can backfill said first page user input data into the corresponding input boxes in said first page.

As is clear from the technical schemes above, the present application uses associative storage of page data on a server, with the result that the server performs unified encoding and decoding operations on the page data and no garbled characters appear because of terminal-side browser type differences or because of incompatibilities between browser and server. It thus ensures smooth backfilling of page data.

Description of the Drawings

FIG. 1 presents a flowchart of a data backfilling method in the related art.

FIG. 2 presents a flowchart of a terminal side-based data backfilling method of an exemplary embodiment of the present application.

FIG. 3 presents a flowchart of a server side-based data backfilling method of an exemplary embodiment of the present application.

FIG. 4 presents a flowchart of a data backfilling method of an exemplary embodiment of the present application.

FIG. 5 presents a structural diagram of a terminal of an exemplary embodiment of the present application.

FIG. 6 presents a block diagram of a terminal side-based data backfilling means of an exemplary embodiment of the present application.

FIG. 7 presents a structural diagram of a server of an exemplary embodiment of the present application.

FIG. 8 presents a block diagram of a server side-based data backfilling means of an exemplary embodiment of the present application.

Specific Embodiments

"Backfilling" in the technical schemes of the present application should be understood as follows: when a terminal returns to a page in accordance with a user operation, it takes the data that the user previously input into this page and re-enters it in a one-to-one correspondence into the input boxes of the page so that the user may conveniently view it and perform editing operations. Because the data is not entered for the first time, but was rather re-entered back into this page from another page, this operation is called "backfilling."

Please refer to FIG. 1. FIG. 1 presents a flowchart of a data backfilling method in the related art. It comprises the steps below:

Step 102: Acquire the data input by the user in page 1 of the terminal browser. Assume that page 1 is the page where the user registers an account. For example, the user may fill in their own name "Zhang San" and their age "10" in the "Name" and "Age" input boxes on page 1.

Step 104: The user initiates a jump to page 2. For example, by clicking the "Next" operation button on page 1, the user triggers this go-to operation.

Step 106: Through a post request, the terminal browser submits the data input by the user in page 1 to the server.

Step 108: The server receives the data input by the user in page 1 and records it in the server.

Step 110: The server may process (e.g., perform a data check on) the recorded user input data.

Step 112: The server transmits the user input data in page 1 back to the terminal browser, and the terminal records it.

Step 114: If, while performing operations in page 2, the user executes the operation to go to page 1 by clicking, for example, the "Previous" button on page 2, the data backfilling operation consisting of steps 116 through 122 will be triggered for page 1.

Step 116: The terminal browser issues a get request to the server. This get request includes user input data recorded by the terminal in page 1 in step 112.

The get request is structured to include the URL (Uniform Resource Locator) of the target page and the data that needs to be transmitted. Moreover, this data directly follows the URL. For example:

“http://www.123456789.com?name=张三&age=10,” wherein “http://www.123456789.com” is the URL information, and “name=张三&age=10” is the data input by the user in page 1. The user input data that specifically corresponds to the input box “Name” is “张三” (“Zhang San”). The user input data that specifically corresponds to the input box “Age” is “10.”

In fact, the user may directly view the get request content (e.g., the above “http://www.123456789.com?name=张三&age=10”) in the terminal browser’s address bar. However, the user cannot directly receive the data “name=张三” and “age=10.” Because “张三” are Chinese characters, the terminal browser needs to perform an encoding operation using its own encoding method. For example, when the terminal browser encodes with UTF-8 (Unicode), it compiles “张三” as “&#x5F20&#x4E09.” Thus, the get request that the server receives may be “http://www.123456789.com?name=&#x5F20&#x4E09&age=10.”

Step 118: After receiving the get request from the terminal browser, the server performs decoding and encoding operations on the data “name=&#x5F20&#x4E09&age=10” according to its own encoding method.

Of course, in addition to the decoding and encoding operations on the data, the server will also subject the received data to an integrity check, a user identity check and other processing on the business layer.

Step 120: The server sends the decoded and encoded data back to the terminal browser through the get request.

Step 122: The terminal browser backfills the input boxes in page 1 with the corresponding received data.

However, after the terminal browser in step 112 receives and records the data transmitted back by the server, it still needs to take the recorded data and, by performing steps 116 through 120, “round-trip” transmit it once on the server, with the result that in this transmission process the data needs to undergo separate decoding and encoding operations on the terminal browser and on the server. Therefore, if either the terminal browser or the server is unable to recognize the other’s encoding method, the result will be that the data encoded by it will be garbled characters (a character string unrelated to the original meaning). Moreover, this will ultimately appear as backfill data in page 1, which will have an extremely negative impact on the user’s experience by making the user unable to easily view the data.

Clearly, the related art requires repeated transmission of the backfill data on the terminal browser and the server, meaning that the data which is ultimately for backfilling needs to undergo separate decoding and encoding operations by the terminal browser and the server.

Thus, because of mismatched encoding/decoding methods between the terminal and the server, it is extremely easy for backfill data containing Chinese to involve the appearance of garbled characters when backfilled into a page.

What needs to be explained:

In the related art, let us take the example of step 108. Although the server allocated the appropriate cache space for the terminal browser (the corresponding sessionID), the life cycle for stored data in the cache space is determined according to "requests." That is, each time the server receives a request, the data contained in the request is recorded in the cache space, and the previously stored data is deleted. Therefore, the server may be regarded as merely "recording" received data and not storing it in the true sense, for the terminal browser cannot perform search or acquisition operations as it might intend. This also makes it necessary for the terminal browser and the server to conduct between themselves the repeated data transmissions shown in FIG. 1.

However, the related art data backfilling process only involves the transmission of data between the terminal browser and the server, and the server in the process needs only "record" the data. As a result, although the entire scheme appears to have a complex process flow, it is in fact a "light-volume" processing process and is thus broadly applied to data backfill processing for terminal browsers.

To avoid the garbled character problem in the related art, the present application makes improvements to the technical scheme described above. By avoiding repeated transmission of data, it can smoothly achieve backfill operations for page data. The following embodiments are provided in order to further explain the present application.

1. Terminal side

Please refer to FIG. 2. FIG. 2 presents a flowchart of a terminal side-based data backfilling method of an exemplary embodiment of the present application. This method applies to the terminal and may comprise the steps described below:

Step 202: Associatively send user input data in the first page of the terminal and the identifiers of the input boxes in said first page to a server for them to be stored by said server, with the stored content mapping to the sessionID of said terminal and to the identifier of said first page.

In the present embodiment, it is possible to package the first page user input data and the associated input box identifiers in a preset manner and to send the packaged data to the server. The packaged data may make use of JSON (JavaScript Object Notation) format so as to minimize the volume of transmitted data.

In the present embodiment, when the aforesaid packaging operation is executed, it is possible to package the input box identifier and the entered data whenever a user has completed an input operation for an input box and thus to establish an association between the input box identifier and the entered data. Or it is possible to associatively execute a packaging operation on all user input data in the first page with the input box identifiers.

In the present embodiment, sessionID is the unique identifier allocated by the server for the terminal. Specifically, it is allocated by the server when the terminal first accesses it, and it is stored by the terminal in a local cookie. Thus, whenever the terminal accesses the server, it declares its own identity through the sessionID stored in the carried cookie. The identifier for the first page may be the first page's URL or URI (Uniform Resource Identifier) information. Therefore, the server may use a combination of the sessionID and URL/URI information to accurately distinguish and identify the stored content corresponding to each page.

Step 204: When said first page is returned to from a second page, send a data acquisition request to said server. Said data acquisition request includes the sessionID of said terminal and the identifier of said first page.

In the present embodiment, the second page and the first page may have an "adjacent" relationship. That is, when the user clicks the "Next" button on the first page, it goes directly to the second page, and when the user clicks the "Previous" button on the second page, it goes directly to the first page. Or the second page may not be adjacent to the first page. For example, the first page is "Page 1," but the second page is "Page 3." Thus, when the destination directly selected from the second page is "Page 1," it can directly go to the first page.

Step 206: Receive said first page user input data and the associated identifiers of the input boxes in said first page sent back by said server according to said data acquisition request, and backfill said first page user input data into the corresponding input boxes in said first page.

In the present embodiment, the user input data that the terminal subjects to backfill operations comes directly from the server. There is no need for the server to transmit the user input data back to the terminal browser to be recorded there (corresponding to step 112 shown in FIG. 1) when the terminal browser goes to "Next" from the first page, nor is it necessary to repeatedly transmit the user input data back and forth between the terminal and the server (corresponding to steps 116 to 120 shown in FIG. 1).

In fact, after the server in step 202 stores the present application user input data, it is transmitted directly from the server to the terminal in step 206. And it is only when the terminal needs to execute a backfilling operation that it issues a data acquisition request (i.e., step 204) to the server, with the result that the user input data need only undergo a unified encoding and decoding operation on the server, without having to consider compatibility issues between the server and the terminal browser. Therefore, it is not possible that backfilled data would appear with garbled characters because of incompatibility between the server and the terminal browser.

In the present embodiment, when the server sends the first page user input data and the associated input box identifiers back to the terminal, it can employ a streaming form and thus avoid having an excessively long get request URL truncated when there is too much data. Parameter errors in such cases make it impossible to backfill data correctly.

2. Server side

Please refer to FIG. 3. FIG. 3 presents a flowchart of a server side-based data backfilling method of an exemplary embodiment of the present application. This method is applied to servers, and it may comprise the steps below:

Step 302: Associatively store the first page user input data sent by the terminal and the identifiers of the input boxes in said first page, with the stored content mapping to the session ID of said terminal and the identifier of said first page.

In the present embodiment, the server may employ the memcache caching framework to associatively store user input data with input box identifiers so as to achieve high-speed access of data.

Step 304: Receive the data acquisition request sent by said terminal when said first page is returned to from the second page.

In the present embodiment, after going from one page to another on the terminal browser, the server does not have to transmit user input data and other such data to the terminal (corresponding to step 112 shown in FIG. 1), but only needs to respond to the data acquisition request sent by the terminal.

In the present embodiment, the data acquisition request may take the form of a get request. However, since the server does not send information such as user input data and input box identifiers back to the terminal in advance (corresponding to step 112 shown in FIG. 1), the get request need not contain information such as user input data and input box identifiers, but only contains the first page URL and carries the sessionID used by the server to perform terminal identification.

Step 306: Use the sessionID for said terminal and the identifier for said first page contained in said data acquisition request as a basis for

sending back said first page user input data and the associated identifiers for the input boxes in said first page to said terminal so that said terminal can backfill said first page user input data into the corresponding input boxes in said first page.

In the present embodiment, the user input data sent by the server to the terminal is the stored content in step 302 and not the get request from the terminal. Thus, the data needs to undergo a character encoding or decoding operation only when saved or extracted by the server and does not need to be encoded or decoded by the terminal browser. Thus, no garbled characters will appear as a result of incompatibility between the terminal browser and the server.

As is clear from the embodiment described above, the present application associatively stores user input data and input box identifiers on a server. The server "stores" the received data in the true sense and does not "record" it as in the related art. That is, the life cycle of the data stored on the server is based on "sessions" (equivalent to the effective term of the sessionID currently allocated to the terminal browser). As a result, when the terminal performs a page data backfill operation, there is no need to transmit the user input data and input box identifiers repeatedly back and forth between the terminal and the server. Rather, the server simply directly sends its stored content to the terminal. That is, the server performs unified encoding and decoding operations on stored content such as user input data, without need for terminal browser encoding and decoding operations. Moreover, no garbled characters will appear as a result of incompatibility between the server and the terminal browser.

The following is a detailed explanation of the exchange of information between a terminal and a server in light of FIG. 4. FIG. 4 presents a flowchart of a data backfilling method of an exemplary embodiment of the present application. As shown in FIG. 4, the method may comprise the steps below:

Step 402: Acquire the user input data in page 1 of the terminal browser. Assume that page 1 is the web page where the user executes account registration. For example, the user could fill in their own name "Zhang San" and age "10" in the "Name" and "Age" input boxes in page 1.

Step 404: The user initiates a jump to page 2. For example, by clicking the "Next" operation button on page 1, the user triggers this go to operation.

Step 406: Through a post request, the terminal browser submits the data input by the user in page 1 to the server.

Step 408: The server receives the page 1 user input data and

associatively stores the user input data and the input box identifiers.

In the present embodiment, the server can regard the terminal sessionID and the page 1 URL or URI as key words and store the user input data and associated input box identifiers in a cache, such as a memcache cache. Or the server may store the received data into another form of storage space, such as a database. The present application imposes no restrictions in this regard.

Step 410: The server may process (e.g., perform a data check on) the recorded user input data.

Step 412: If, while performing an operation in page 2, the user executes a jump to page 1 (if, for example, the user clicks the "Previous" button on page 2), the page 1 data backfilling operation consisting of steps 414 through 420 will be triggered.

Step 414: The terminal browser issues a get request to the server.

In the present embodiment, because the server did not send the page 1 user input data back to the terminal earlier, and because the present application does not require repeated transmission of the user input data back and forth between the terminal browser and the server, the get request need only contain the page 1 URL and does not have to add the user input data, the input box identifiers and so on after the URL and thereupon send it to the server, unlike in step 116 shown in FIG. 1. Therefore, the information that the user sees in the address bar in the terminal browser is different from "http://www.123456789.com?name=张三&age=10" shown above; it is simply "http://www.123456789.com."

Step 416: Use the page 1 URL or URI contained in the get request and the session ID transmitted together with it when the terminal sent the get request to search the cache for the stored content corresponding to the "sessionID + URL" or "sessionID + URI" combination, i.e., corresponding to the page 1 user input data and the corresponding input box identifiers.

Step 418: Transmit the found page 1 user input data and other found stored content back to the terminal.

Step 420: The terminal browser backfills the user input data in the data transmitted back by the server into the corresponding input boxes in page 1.

In the embodiment described above, the server implements associative storage of data submitted by the terminal, and the server transmits data according to the terminal-issued get request back to the terminal in order to achieve data backfill,

with the result that the user input data received by the terminal is subjected by the server alone to unified character encoding and decoding operations in step 406 (server encodes received data for storage in cache) and step 418 (after decoding located data, server sends it back to terminal) and does not require encoding and decoding operations by the terminal browser. Thus, there is no need for terminal browser and server encoding operations to be compatible, and no garbled characters will result in the backfill data.

Step 422: After completing a page 1 viewing or editing operation, the user triggers an operation to go to another page, e.g., to page 3.

Step 424: Similar to step 406, through a post request, the terminal browser associatively sends the latest user input data and input box identifier to the server.

Step 426: The server uses the terminal's sessionID and page 1 identifier to search for stored user input data and associated input box identifiers and matches it with currently received data. It thus determines whether the stored user input data corresponding to the same input box identifier differs from the most recent user input data.

Step 428: In the case of a difference in data, the server's stored content will be updated.

In the present embodiment described above, the server subjects page 1 user input data to difference assessments and data updating. In addition to the server implementing data difference assessments, the terminal may also test the update status of the user input data. For example, it compares the latest user input data with the user input data received in step 418. If there is a data difference, it associatively sends the appropriate latest user input data and the input box identifiers to the server so that the server can perform data updating.

As is clear from the embodiment described above, the present application implements "storage" in the true sense on the server. That is, the server associatively stores all the data it has received with keys. As a result, the terminal browser may at any time use a key to obtain the corresponding stored data. Thus, the number of data transmissions between the terminal browser and the server is reduced so as to solve the garbled character problem arising from the data transmission process.

FIG. 5 presents a structural diagram of a terminal of an exemplary embodiment of the present application. Please refer to FIG. 5. In the hardware layer, the terminal includes a processor, an internal bus, a network interface, memory and non-volatile memory. Of course, it may also include other hardware needed for services. The processor fetches the corresponding computer program from the non-volatile memory into the memory and then runs it. A data backfilling means is formed in the logical layer.

Of course, in addition to realizing the means in the form of software, the present application may also realize it in other forms, e.g., as a logic device or as a combination of hardware and software. In other words, the executing entities of the processing flow below are not limited to the various logic units, but may also be hardware or logic devices.

Please refer to FIG. 6. In a software implementation, the data backfilling means may comprise a sending unit, a requesting unit and a backfilling unit, wherein:

a sending unit, which associatively sends user input data in the first page of the terminal and the identifiers of the input boxes in said first page to a server for them to be stored by said server, with the stored content mapping to the sessionID of said terminal and to the identifier of said first page;

a requesting unit, which sends a data acquisition request to said server when said first page is returned to from a second page, said data acquisition request comprising the sessionID of said terminal and the identifier of said first page;

a backfilling unit, which receives said first page user input data and the associated identifiers of the input boxes in said first page sent back by said server according to said data acquisition request and takes said first page user input data and backfills it into the corresponding input boxes in said first page.

Optionally, said sending unit specifically is for:

taking said first page user input data and said first page input box identifiers and associatively generating them as packaged data according to a preset format;

sending said packaged data to said server.

Optionally, said preset format is JSON format.

Optionally, it further comprises:

a first updating unit, which, when said first page is switched to another page, associatively sends the latest user input data in said first page and the identifiers for the input boxes in said first page to said server so that said server can update storage content mapped to said terminal’s sessionID and said first page identifier.

Optionally, it further comprises:

a comparing unit, which, when said first page is switched to another page, compares the latest user input data in said first page to the user input data sent back by said server;

a second updating unit, which, if the comparison result is not identical, associatively sends the differing user input data and corresponding input box identifiers to said server so that said server will update said storage content.

FIG. 7 presents a structural diagram of a server of an exemplary embodiment of the present application. Please refer to FIG. 7. In the hardware layer, the server includes a processor, an internal bus, a network interface, memory and non-volatile memory. Of course, it may also include other hardware needed for services. The processor fetches the corresponding computer program from the non-volatile memory into the memory and then runs it. A data backfilling means is formed in the logical layer.
Of course, in addition to realizing the means in the form of software, the present application may also realize it in other forms, e.g., as a logic device or as a combination of hardware and software. In other words, the executing entities of the processing flow below are not limited to the various logic units, but may also be hardware or logic devices.

Please refer to FIG. 8. In a software implementation, the data backfilling means may comprise a data storage unit, a requesting unit, and a data returning unit, wherein:

a data storage unit, which associatively stores the first page user input data sent by the terminal and the identifiers of the input boxes in said first page, with the stored content mapping to the session ID of said terminal and the identifier of said first page;

a request receiving unit, which receives the data acquisition request sent by said terminal when said first page is returned to from the second page;

a data returning unit, which uses the sessionID for said terminal and the identifier for said first page contained in said data acquisition request as a basis for sending back said first page user input data and the associated identifiers for the input boxes in said first page to said terminal so that said terminal can backfill said first page user input data into the corresponding input boxes in said first page.

Optionally, it further comprises:

a data receiving unit, which receives said first page user input data and the associated identifiers for the input boxes in said first page sent by said terminal when said first page is switched to another page;

a comparing unit, which compares the latest user input data in said first page to already stored said first page user input data;

a content updating unit, which, if the result is not identical, updates the storage content mapping to the sessionID of said terminal and the identifier of said first page.

Therefore, the present application uses associative storage of page data on a server, with the result that the server performs unified encoding and decoding operations on the page data and no garbled characters appear because of terminal-side browser type differences or because of incompatibilities between browser and server. It thus ensures smooth backfilling of page data.

In one typical configuration of the present application, the computing equipment comprises one or more processors (CPUs), input/output interfaces, network interfaces and memory.

Memory may include such forms as volatile memory in computer-readable media, random access memory (RAM) and/or non-volatile memory, such as read-only memory (ROM) or flash memory (flash RAM). Memory is an example of a computer-readable medium.

Computer-readable media, including permanent and non-permanent and removable and non-removable media, may achieve information storage by any method or technology. Information can be computer-readable commands, data structures, program modules, or other data. Examples of computer storage media include but are not limited to phase-change memory (PRAM), static random access memory (SRAM), dynamic random access memory (DRAM), other types of random access memory (RAM), read-only memory (ROM), electrically erasable programmable read-only memory (EEPROM), flash memory or other memory technology, compact disk read-only memory (CD-ROM), digit multifunction disk (DVD) or other optical storage, magnetic cassettes, magnetic tape or magnetic disc storage, or other magnetic storage equipment or any other non-transmission media that can be used to store information that is accessible to computers. As defined in this document, computer-readable media does not include temporary computer-readable media, (transitory media), such as modulated data signals and carrier waves.

Furthermore, the term “comprise” or “contain” or any of their variants are to be taken in their non-exclusive sense. Thus, processes, methods, products, or devices that comprise a series of elements not only comprise those elements, but also comprise other elements that have not been explicitly listed or elements that are intrinsic to such processes, methods, products or devices. In the absence of further limitations, elements that are limited by the phrase “comprises a(n)...” do not exclude the existence of additional identical elements in processes, methods, products or devices that comprise said elements.

The above-described are merely preferred embodiments of the present application and do not serve to limit the present application. Any modifications, equivalent substitutions, or improvements that are performed within the spirit and principles of the present application shall be contained within the protective scope of the present application.

CLAIMS

1. A method for backfilling data, characterized in that it comprises:

associatively sending user input data in the first page of the terminal and the identifiers of the input boxes in said first page to a server for them to be stored by said server, with the stored content mapping to the sessionID of said terminal and to the identifier of said first page;

when said first page is returned to from a second page, sending a data acquisition request to said server, said data acquisition request comprising the sessionID of said terminal and the identifier of said first page;

receiving said first page user input data and the associated identifiers of the input boxes in said first page sent back by said server according to said data acquisition request, and taking said first page user input data and backfilling it into the corresponding input boxes in said first page.

2. The method as described in claim 1, characterized in that associatively sending user input data in the first page of the terminal and the identifiers of the input boxes in said first page to a server comprises:

taking said first page user input data and said first page input box identifiers and associatively generating them as packaged data according to a preset format;

sending said packaged data to said server.

3. The method as described in claim 2, characterized in that said preset format is JSON format.

4. The method as described in claim 1, characterized in that it further comprises:

when said first page is switched to another page, associatively sending the latest user input data in said first page and the identifiers for the input boxes in said first page to said server so that said server can update storage content mapped to said terminal’s sessionID and said first page identifier.

5. The method as described in claim 1, characterized in that it further comprises:

when said first page is switched to another page, comparing the latest user input data in said first page to the user input data sent back by said server;

if the comparison result is not identical, associatively sending the differing user input data and corresponding input box identifiers to said server so that said server can update said storage content.

6. A method for backfilling data, characterized in that it comprises:

associatively storing the first page user input data sent by a terminal and the identifiers of the input boxes in said first page, with the stored content mapping to the session ID of said terminal and the identifier of said first page;

receiving a data acquisition request sent by said terminal when said first page is returned to from the second page;

using the sessionID for said terminal and the identifier for said first page contained in said data acquisition request as a basis for sending back said first page user input data and the associated identifiers for the input boxes in said first page to said terminal so that said terminal can backfill said first page user input data into the corresponding input boxes in said first page.

7. The method as described in claim 6, characterized in that it further comprises:

receiving said first page user input data and the associated identifiers for the input boxes in said first page sent by said terminal when said first page is switched to another page;

comparing the latest user input data in said first page to already stored said first page user input data;

if the result is not identical, updating the storage content mapping to the sessionID of said terminal and the identifier of said first page.

8. A means for backfilling data, characterized in that it comprises:

a sending unit, which associatively sends user input data in the first page of the terminal and the identifiers of the input boxes in said first page to a server for them to be stored by said server, with the stored content mapping to the sessionID of said terminal and to the identifier of said first page;

a requesting unit, which, when said first page is returned to from a second page, sends a data acquisition request to said server, said data acquisition request comprising the sessionID of said terminal and the identifier of said first page;

a backfilling unit, which receives said first page user input data and the associated identifiers of the input boxes in said first page sent back by said server according to said data acquisition request and takes said first page user input data and backfills it into the corresponding input boxes in said first page.

9. The means as described in claim 8, characterized in that said sending unit is specifically for:

taking said first page user input data and said first page input box identifiers and associatively generating them as packaged data according to a preset format;

sending said packaged data to said server.

10. The means as described in claim 9, characterized in that said preset format is JSON format.

11. The means as described in claim 8, characterized in that it further comprises:

a first updating unit, which, when said first page is switched to another page, associatively sends the latest user input data in said first page and the identifiers for the input boxes in said first page to said server so that said server can update storage content mapped to said terminal's sessionID and said first page identifier.

12. The means as described in claim 8, characterized in that it further comprises:

a comparing unit, which, when said first page is switched to another page, compares the latest user input data in said first page to the user input data sent back by said server;

a second updating unit, which, if the comparison result is not identical, associatively sends the differing user input data and corresponding input box identifiers to said server so that said server can update said storage content.

13. A means for backfilling data, characterized in that it comprises:

a data storage unit, which associatively stores the first page user input data sent by the terminal and the identifiers of the input boxes in said first page, with the stored content mapping to the session ID of said terminal and the identifier of said first page;

a request receiving unit, which receives the data acquisition request sent by said terminal when said first page is returned to from the second page;

a data returning unit, which uses the sessionID for said terminal and the identifier for said first page contained in said data acquisition request as a basis for sending back said first page user input data and the associated identifiers for the input boxes in said first page to said terminal so that said terminal can backfill said first page user input data into the corresponding input boxes in said first page.

14. The means as described in claim 13, characterized in that it further comprises:

a data receiving unit, which receives said first page user input data and the associated identifiers for the input boxes in said first page sent by said terminal when said first page is switched to another page;

a comparing unit, which compares the latest user input data in said first page to already stored said first page user input data;

a content updating unit, which, if the result is not identical, updates the storage content mapping to the sessionID of said terminal and the identifier of said first page.