METHODS AND APPARATUSES FOR CODE SNIPPET TRANSLATION, COMPUTER DEVICES AND STORAGE MEDIUMS

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority to Chinese Patent Application No. 2019102961275, filed with the Chinese Patent Office on April 12, 2019 and entitled "METHODS AND APPARATUSES FOR CODE SNIPPET TRANSLATION, COMPUTER DEVICES AND STORAGE MEDIUMS", which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present application relates to methods and apparatuses for code snippet translation, computer devices, and storage mediums.

BACKGROUND

[0003] With the development of internationalization, the developed software systems will face the needs of different regions and countries in the world, it is therefore necessary to translate the source code of the software system from the original language to a certain language. In the traditional way, it is usually the business analyst who collects translation requirements and writes the same into a document, and then delivers it to the programmer to modify the source code based on the document. However, the inventors realized that the conventional translation process of code internationalization requires the cooperation between the business analyst and the developer, the time cost and the manpower cost are high, and translation error is likely to happen when the developer implements the translation idea of the business analyst in practice, and it is difficult to ensure the unity of code translation by manual translation, the translation result is therefore unreliable.

SUMMARY

[0004] According to various embodiments disclosed in this application, there is

provided methods and apparatuses for code snippet translation, computer devices, and storage mediums.

[0005] A method for code snippet translation, applied to a code server, the method includes:

[0006] acquiring a to-be-translated analysis result set; the to-be-translated analysis result set includes a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

[0007] looking up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

[0008] identifying a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

[0009] calling a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code snippet of a corresponding target language type based on the translation policy.

[0010] An apparatus for code snippet translation includes:

[0011] an acquiring module, configured to acquire a to-be-translated analysis result set; the to-be-translated analysis result set includes a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

[0012] a looking up module, configured to look up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

[0013] an identification module, configured to identify a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

[0014] a translation module, configured to call a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code snippet of a corresponding target language type based

on the translation policy.

[0015] A computer device includes a memory and one or more processors, the memory store computer-readable instructions that, when executed by the one or more processors, cause the one or more processors to perform the steps of:

[0016] acquiring a to-be-translated analysis result set; the to-be-translated analysis result set includes a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

[0017] looking up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

[0018] identifying a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

[0019] calling a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code snippet of a corresponding target language type based on the translation policy.

[0020] One or more non-volatile computer-readable storage media stores computer-readable instructions that, when executed by one or more processors, cause the one or more processors to perform the steps of:

[0021] acquiring a to-be-translated analysis result set; the to-be-translated analysis result set includes a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

[0022] looking up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

[0023] identifying a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

[0024] calling a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code

snippet of a corresponding target language type based on the translation policy.

[0025] The details of at least an embodiment of the present disclosure will be presented with reference to the following drawings and description. Other characteristic and advantages of the present disclosure will be more apparent from the specification, drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0026] To illustrate the technical solutions according to the embodiments of the present disclosure more clearly, the accompanying drawings for describing the embodiments are introduced briefly in the following. Apparently, the accompanying drawings in the following description are only some embodiments of the present disclosure, and persons of ordinary skill in the art can derive other drawings from the accompanying drawings without creative efforts.

[0027] FIG. 1 is a schematic diagram of an application scenario of a method for code snippet translation according to one or more embodiments.

[0028] FIG. 2 is a schematic flowchart of a method for code snippet translation according to one or more embodiments.

[0029] FIG. 3 is a schematic flowchart of a method for code snippet translation according to another embodiment.

[0030] FIG. 4 is a block diagram of an apparatus for code snippet translation according to one or more embodiments.

[0031] FIG. 5 is a schematic diagram of a computer device according to one or more embodiments.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0032] The present disclosure will be described in details in combination with the accompanying drawings and embodiments such that the technical solution and advantages of the present disclosure will be more apparent. It should be understood that the particular embodiments are described for the purpose of illustrating as opposed to restricting the present disclosure.

[0033] The method for code snippet translation provided herein may be applied to the application environment shown in FIG. 1. The code server 102 communicates with the translation server 104 via the network. The code server 102 and translation server 104 can be implemented by a separate server or a cluster of servers composed of a plurality of servers. The code server 102 can look up a code processor matching the to-be-translated code snippet after acquiring the to-be-translated analysis result set containing the to-be-translated code snippet and the context code snippet corresponding to the to-be-translated code snippet. The code server 102 identifies a target language type and translation policy of the to-be-translated code snippet according to the context code snippet. The code server 102 can send the to-be-translated code snippet to a translation server 104 corresponding to a translator for translation by calling a translator interface through code processor to obtain a translated code snippet of the target language type corresponding to the to-be-translated code snippet and translated based on the translation policy.

[0034] In an embodiment, as shown in FIG. 2, a method for code snippet translation is provided and is illustrated being applied to the code server 102 in FIG. 1, which includes the following steps:

[0035] At step 202, a to-be-translated analysis result set is acquired; the to-be-translated analysis result set includes a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet.

[0036] The to-be-translated analysis result set refers to the set of to-be-translated analysis results respectively corresponding to a plurality of to-be-translated code snippets. The analysis result includes the to-be-translated code snippets in the source code file and the related information. The to-be-translated analysis result set can contain one or more to-be-translated code snippet(s). The to-be-translated source code file refers to the to-be-translated source code files. Source code files are uncompiled text files written in accordance with certain programming language specifications. The to-be-translated code snippet refers to the code snippet in the to-be-translated source code file that is of the original language type and needs to be translated into the target language type. The context code snippet is the code

snippets of preset character length in the beginning and the ending of the to-be-translated code snippet in the to-be-translated source code file. For example, it is possible to acquire the code snippets of the first and last 50 character length of the to-be-translated code snippet as the context code snippet of the to-be-translated code snippet.

[0037] At step 204: a code processor matching the to-be-translated code snippet is looked up.

[0038] The code processor refers to the processor configured to perform grammatical recognition, risk recognition, translation and state tagging to the to-be-translated code snippets. Different code processors can be set up in advance for the to-be-translated code snippets of different programming languages, so that the code processor can perform more accurate and targeted operations on the translated code snippets to achieve more accurate code snippet translation.

[0039] In an embodiment, the looking up the code processor matching the to-be-translated code snippet includes: a programming language type of the to-be-translated code snippet is identified; a code processor set corresponding to the programming language type is looked up; the to-be-translated code snippet is matched with matching criteria corresponding to each code processor in the code processor set; and the code processor matching the matching criteria is regarded as the code processor matching the to-be-translated code snippet.

[0040] The programming language type includes but is not limited to JS (JavaScript), JSX (JavaScript XML, a XML-like grammar that organize internal build tag at React), and so on. It is possible to identify the programming language type of a to-be-translated code snippet based on a regular expression. A regular expression is a logical formula for a string operation, that is, to compose a "rule string" using a predefined specific characters and combination thereof, this "rule string" can be used to express a filter logic for a string. Each programming language type has one or more corresponding regular expressions.

[0041] For each kind of programming language type, the corresponding code processor set can be predefined. Each code processor has customized matching

criteria. The matching criteria include, but are not limited to, the string type involved in the to-be-translated code snippet, the business logic involved in the to-be-translated code snippet, and the configuration item attributes in the configuration file involved in the to-be-translated code snippet. Based on the matching criteria, the code processor matching the translation of the code snippet can be looked up from the code processor collection.

[0042] At step 206, a target language type and a translation policy of the to-be-translated code snippet are identified according to the context code snippet and by the code processor.

[0043] The code processor can combine context code snippets to analyze the syntax of the to-be-translated code snippets, thereby identifying translation policies corresponding to the to-be-translated code snippets. Different translation policies can be preset for different to-be-translated code snippets. Translation policies include whether to perform a replacement, and whether to perform different translations corresponding to to-be-translated code snippets of different syntaxes. Through the resulting context code snippets, a syntax analysis can be performed on the to-be-translated code snippets more accurately.

[0044] In an embodiment, the translation policy includes whether to perform a replacement operation. For example, when a to-be-translated code snippet is a code related to business logic, such as a to-be-translated code snippet involving business logic such as if/switch and split, no replacement operation is performed. When the to-be-translated code snippet is a related code meaningless to the user, such as code annotation and to-be-translated code snippet related to console output, no replacement operation is performed. Translation policies may also include performing different translation on to-be-translated code snippets corresponding to different syntax, such as string types in JS, included in double quotation marks, and string templates and JSX TEXT types in JS, not included in double quotation marks. The translation policies can also be based on the configuration item attributes in the involved profiles. For example, when a directReplace configuration item attribute set in the configuration file is true, the to-be-translated source code file is modified directly.

When the configuration item property is false, the description uses a translation dictionary instead of a direct replacement, using a pre-customized translation policy, such as storing translated code snippets in a resource configuration file. The translation policy can also be determined based on a third-party plug-in, such as for react (JSX) to-be-translated code snippets, the translation policy provided by the react-intl- universal plug-in is adopted, so that the replaced translated code snippet satisfies the grammatical rule corresponding to the third-party plug-in through the pre-customized replacement rules.

[0045] At step 208, the translator interface is called by the code processor to obtain the translated code snippet of the target language type corresponding to the target language snippet.

[0046] The translator interface is an interface provided by a translator system and can be used to implement translation functions by calling. The translator system can be deployed on a translation server. The translator system includes but is not limited to one or more of Baidu or Google translators. The translator system can translate the to-be-translated code snippet into the translated code snippet of the corresponding target language type based on the translation policy. The target language types include but are not limited to Chinese, Khmer, Thai, Burmese and Laos.

[0047] In the described method for code snippet translation, after acquiring the result set of the translation analysis, the code server can look up the code processor matching the to-be-translated code snippet contained in the to-be-translated analysis result set, and identify the target language type and translation policy of the to-be-translated code snippet through the context code snippet corresponding to the to-be-translated code snippet. The code server can call the translator interface through the code processor to obtain the translated code snippet of the target language type corresponding to the to-be-translated code snippet based on the translation policy. The targeted translation policy is obtained by combining the syntax of the to-be-translated code snippet itself and processing the to-be-translated code snippet through the preset targeted code processor, thereby realizing an automatic and uniform translation of the to-be-translated code snippet by calling the translator

interface, and ensuring the reliability of the translation result.

[0048] In an embodiment, the to-be-translated analysis result set is acquired, which includes acquiring a to-be-translated source code file; reading a source code directory in the to-be-translated source code file; the directory of the to-be-translated source code includes a plurality of initial file names; traversing an initial code sub-file corresponding to each of the initial file names, and filtering out a to-be-translated code sub-file; the to-be-translated code sub-file matching a preset language type; looking up a parser matching the preset language type; obtaining a to-be-translated analysis result set by parsing the to-be-translated code sub-file by the parser; and the to-be-translated analysis result set contains the to-be-translated code snippets in the to-be-translated code sub-file and the context code snippets corresponding to the to-be-translated code snippets.

[0049] The to-be-translated source code can be pre-stored in the database of the code server, the code server can directly acquire the to-be-translated source code file from the local database. The code server can also receive to-be-translated code files uploaded by terminals or to-be-translated source code files sent by other servers. The initial code sub-file refers to the source code file in the to-be-translated source code file that is of the smallest granularity and cannot be further expanded. The to-be-translated code sub-file refers to the initial code sub-file matching the preset language type and needs to be translated into the target language type. The to-be-translated code snippet in the to-be-translated code can be obtained by parsing the to-be-translated code sub-file with the parser, and performing syntax analysis on the to-be-translated code in the to-be-translated code sub-file to obtain to-be-translated code snippet in the to-be-processed code.

[0050] In an embodiment, the code server may detect the to-be-translated code snippet in the to-be-translated code sub-file; segment snippet locating information of the to-be-translated code snippet in the to-be-translated code sub-file; the snippet locating information includes the start location information and end location information; extract the to-be-translated code snippet based on the start location information and the end location information; generate the to-be-translated analysis

result set based on the to-be-translated code snippet, the start location information, and the end location information. The to-be-translated analysis result set can contain a plurality of to-be-translated code snippets and snippet locating information corresponding to each to-be-translated code snippet. Each to-be-translated code snippet and the snippet locating information corresponding to the respective to-be-translated code snippet can be regarded as a translation analysis result.

[0051] In an embodiment, the to-be-translated analysis result set further includes snippet locating information; after identifying the target language type and the translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor, further includes: determining the translation operation according to the translation policy; correspondingly replacing the to-be-translated code snippet in the to-be-translated source code file with the translated code snippet when the translation operation is a replacement operation; outputting a translated source code file of the target language type.

[0052] Translation operation refers to the operation of processing the to-be-translated code snippets in the to-be-translated source code file. The translation operations can be one or more of a replacement operation, an annotation operation, etc. The code server can replace the to-be-translated code snippet in the translation source code file with the translated code snippet information based on the path of the to-be-translated source code file and the snippet locating information such as the location information of the to-be-translated code snippet in the to-be-translated source code file; and output the translated source code file corresponding to the target language type.

[0053] In an embodiment, a translator interface is called by the code processor to obtain a translated code snippet of the target language type corresponding to the to-be-translated code snippet based on the translation policy, including acquiring a user identifier; generating a translation request according to the user identifier, the to-be-translated code snippet, the target language type, and the translation policy; sending the translation request to the translation server through a translator interface; the translation request is configured to instruct the translation server to translate the

to-be-translated code snippet into the target language type using the translation policy after the user identifier is authenticated; and receiving a translated code snippet of the target language type and returned by the translation server.

[0054] A user identifier is an identifier used to identify the code server identity, the user identifier includes but is not limited to a user account registered by the code server party in a translator system. After generating the translation request, the code server calls the translator interface so that the translation server can process the translation code snippet based on the translation request. The translation server can return the translated code snippet of the target language type obtained according to the translation policy to the code server.

[0055] In an embodiment, a translation request is generated according to the user identifier, the to-be-translated code snippet, the target language type, and the translation policy. The user identifier, to-be-translated code snippet, target language type, and translation policy are concatenated to generate the initial string; the initial string is encrypted to obtain a request string; and a translation request is generated according to the request string.

[0056] For example, the initial string can be encrypted according to the MD5 (Message-Digest Algorithm) encryption algorithm, thereby preventing the risk of leakage of the initial string and ensuring the security of the initial string. When the analysis result set contains a plurality of to-be-translated code snippets, a translation request can be generated according to each of the to-be-translated code snippets and sent to the translation server one by one for translation. Translation requests can also be generated from the plurality of to-be-translated code snippets to reduce the total number of callings to translation requests and improve the execution stability of the plurality of to-be-translated code snippets.

[0057] In an embodiment, the code processor generates a unique processing step identifier after processing each piece of the matched to-be-translated code snippets, and may also label the processed to-be-translated code snippet in the translation analysis result set by the processing step identifier to facilitate subsequent processing step queries. For example, when a search keyword is acquired, it is possible to look

up a unique search result corresponding to the search keyword. The processing step identifier may be a string consisting of one or more of a letter, a number, a punctuation, or the like. For example, the processing step identifier can start with P, with an incremented integers in the middle, and end up with E. Ending with E can improve the search efficiency. For example, when the search keyword is P10, the search result can be P10, P101, P1000, etc., which reduces the search efficiency. When the search keyword is P10E, the search result is a single P10E.

[0058] In an embodiment, after performing the processing by the code processor on the to-be-translated code snippets in the translation analysis result set, a risk level assessment may be performed on the translation results of each of the to-be-translated code snippet, and a processing state may be recorded. Based on custom risk rules corresponding to the selected code processor, a risk level assessment is performed on this code translation result.

[0059] The risk levels include, but are not limited to, low risk level, potential risk level, and high risk level. When the context code snippet is checked, it is found that the to-be-translated code snippet is part of a string concatenation, which indicates that the automatic translation in this case is often incorrect and that the to-be-translated code snippet can be marked as of a potential risk level. Or when the translation results are checked, it is found that the translated code snippet still contains a string of original language type, which indicates that the translator system failed to translate accurately and that the to-be-translated code snippet can be marked as of a potential risk level. When a translation failure due to a failure of the translator interface calling is detected, the to-be-translated code snippet can be marked as of a high risk level. Or when a code processor matching the to-be-translated code snippet is not found, mark the to-be-translated code snippet as of a high risk level.

[0060] The processing states include but are not limited to a unprocessed state, a unreplaced state, an alert state, and a processed state. The unprocessed state includes that the code processor matching the to-be-translated code snippet is not found. The unreplaced state includes that the to-be-translated code snippet is code annotation and console output and no replacement operation is required. The alert

state includes that the to-be-translated code snippet involves business logic and cannot be easily replaced. The processed state includes that the to-be-translated code snippet has already been translated and replaced.

[0061] In an embodiment, the translated code snippets may be labelled. Each translated code snippet processed by the processor can be labelled, the labelling can be code annotation for the purposes of prompting and fast locating. The labelled information includes but is not limited to a processing step identifier, a code processor identifier, a processing state, and a risk level etc. With this annotation, developers can quickly understand the processing of translated code snippets.

[0062] In an embodiment, a customized end-processing may also be performed after translation. Different end-processings can be preset for the to-be-translated source code files of different programming language types. For example, for the react (JSX) language, the following end-processing can be performed: determining whether to use a resource file-based customized scheme according to the parameter input value; if not, no processing; if yes, write the preset resource file according to the preset path, each resource file corresponds to a language type, and the resource file stores all the translated code snippets. There can be two resource files, one for the original language type and the other for the original language type.

[0063] In an embodiment, after translating the plurality of to-be-translated code snippets in the to-be-translated analysis result set into the translated code snippets of the target language type, the translated source code file of the target language type and the translation detail report may be outputted according to the translation results. The translation detail report can contain the context code snippets, the translated code snippet, the risk level, the processing state, and so on corresponding to each to-be-translated code snippet.

[0064] In an embodiment, as shown in FIG. 3, another method for code snippet translation is provided being applied to the code server 102 in FIG. 1, which includes the following steps:

[0065] At step 302, a to-be-translated source code file is acquired.

[0066] At step 304, a source code directory in the to-be-translated source code file

is read; the directory of the to-be-translated source code contains a plurality of initial file names;

[0067] At step 306, an initial code sub-file corresponding to each of the initial file names is traversed, and a to-be-translated code sub-file is filtered out; the to-be-translated code sub-file matches a preset language type;

[0068] At step 308, a parser matching the preset language type is looked up.

[0069] At step 310, the to-be-translated code sub-file is parsed by the parser and a to-be-translated analysis result set is obtained; the to-be-translated analysis result set contains the to-be-translated code snippet in the to-be-translated code sub-file and the context code snippet corresponding to the to-be-translated code snippet.

[0070] At step 312, a programming language type of the to-be-translated code snippet is identified.

[0071] At step 314, a code processor set corresponding to the programming language type is looked up.

[0072] At step 316: the to-be-translated code snippet is matched with matching criteria corresponding to each code processor in the code processor set.

[0073] At step 318, the code processor matching the matching criteria is regarded as the code processor matching the to-be-translated code snippet.

[0074] At step 320, a target language type and a translation policy of the to-be-translated code snippet are identified according to the context code snippet and by the code processor.

[0075] At step 322: a translator interface is called by the code processor to obtain the translated code snippet of the target language type corresponding to the to-be-translated code snippet based on the translation policy.

[0076] In the described method for code snippet translation, after acquiring the to-be-translated source code file, the to-be-translated analysis result set containing the to-be-translated code snippets is obtained by parsing the to-be-translated code sub-file by the parser. According to the programming language type of the to-be-translated code snippet, the code processor matching the to-be-translated code snippet contained in the to-be-translated analysis result set, is looked up in the code processor set, and

the target language type and translation policy of the to-be-translated code snippet are identified through the context code snippet corresponding to the to-be-translated code snippet. The code server can call the translator interface through the code processor to obtain the translated code snippet of the target language type corresponding to the to-be-translated code snippet based on the translation policy. The targeted translation policy is obtained by combining the syntax of the to-be-translated code snippet itself and processing the to-be-translated code snippet through the preset targeted code processor, thereby realizing an automatic and uniform translation of the to-be-translated code snippet by calling the translator interface, and ensuring the reliability of the translation result.

[0077] It should be appreciated that although the various steps in the flowcharts of FIGS. 2 to 3 are shown sequentially as indicated by the arrows, these steps are not necessarily performed sequentially in the order indicated by the arrows. The performance of these steps is not critical to the order in which they are performed unless explicitly stated herein, and the steps may be performed in other orders. Moreover, at least a part of the steps in FIGS. 2 to 3 may include a plurality of sub-steps or stages, which are not necessarily performed at the same time, but may be performed at different times, and the order in which the sub-steps or stages are performed is not necessarily performed sequentially, but may be performed in turns or alternatively with other steps or at least a part of sub-steps or stages of other steps.

[0078] In an embodiment, as shown in FIG. 4, an apparatus for code snippet translation 400 is provided, which includes an acquiring module 402, configured to acquire a to-be-translated analysis result set; the to-be-translated analysis result set includes a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet; a looking up module 404, configured to look up a code processor matching the to-be-translated code snippet; an identification module 406, configured to identify a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and a translation module 408, configured to call a translator interface provided by a translator system by the code processor to obtain a translated code

snippet of a target language type corresponding to the to-be-translated code snippet based on the translation policy.

[0079] In an embodiment, the acquiring module 402 is further configured to acquire a to-be-translated source code file; reading a source code directory in the to-be-translated source code file; the directory of the to-be-translated source code includes a plurality of initial file names; traverse an initial code sub-file corresponding to each of the initial file names, and filter out a to-be-translated code sub-file; the to-be-translated code sub-file matches a preset language type; look up a parser matching the preset language type; obtain a to-be-translated analysis result set by parsing the to-be-translated code sub-file by the parser; and the to-be-translated analysis result set contains the to-be-translated code snippets in the to-be-translated code sub-file and the context code snippets corresponding to the to-be-translated code snippets.

[0080] In an embodiment, the to-be-translated analysis result set further includes snippet locating information; the apparatus further includes an output module, configured to determine a translation operation according to the translation policy; the to-be-translated code snippet in the to-be-translated source code file is replaced with the translated code snippet based on the snippet locating information when the translation operation is a replacing operation; and a translated source code file of the target language type is outputted.

[0081] In an embodiment, the looking up module 404 is further configured to identify the programming language type of the to-be-translated code snippet; look up the code processor set corresponding to the programming language type; match the to-be-translated code snippet with matching criteria corresponding to each code processor in the code processor set; and regard the code processor matching the matching criteria as the code processor matching the to-be-translated code snippet.

[0082] In an embodiment, the translation module 408 is further configured to acquire a user identifier; generate a translation request according to the user identifier, the to-be-translated code snippet, the target language type, and the translation policy; send the translation request to the translation server; the translation request is

configured to instruct the translation server to translate the to-be-translated code snippet into the target language type using the translation policy after the user identifier is authenticated; and receive a translated code snippet of the target language type and returned by the translation server.

[0083] In an embodiment, the translation module 408 is further configured to concatenate the user identifier, to-be-translated code snippet, target language type, and translation policy to generate an initial string; encrypt the initial string to obtain a request string; and generate a translation request according to the request string.

[0084] For specific definition of the apparatus for code snippet translation, reference may be made to the definition of the above method for code snippet translation, and details are omitted for brevity. Each of the modules in the apparatus for code snippet translation described may be entirely or partially implemented by software, hardware, or a combination thereof. Each of the modules may be embedded in or independent of the processor in the computer device in the form of hardware, or may be stored in the memory in the computer device in the form of software, so that the processor calls and executes the operations corresponding to each of the modules.

[0085] In an embodiment, a computer device is provided, which may be a server and an internal structure thereof may be as shown in FIG. 5. The computer device includes a processor, a memory, a network interface, and a database connected through a system bus. The processor of the computer device is configured to provide calculation and control capabilities. The memory of the computer device includes a non-transitory storage medium and a random access memory (RAM). The non-transitory storage medium stores an operating system, computer-readable instructions and the database. The RAM provides an environment for the operation of the operating system and the computer-readable instructions in the non-transitory storage medium. The database of the computer device is configured to store data such as the to-be-translated analysis result set. The network interface of the computer device is configured to communicate with an external terminal through a network connection. The computer-readable instructions are executed by the

processor to implement the method for code snippet translation.

[0086] Those skilled in the art will appreciate that the structure shown in FIG. 5 is merely a block diagram of a portion of the structure associated with the solution of the present disclosure, and does not constitute a limitation on the computer device to which the solution of the present disclosure is applied, a particular computer device may include more or less components, or combine certain components, or with a different arrangement of components.

[0087] In an embodiment, a computer device is provided, which includes a memory and one or more processors, the memory store computer-readable instructions that, when executed by the one or more processors, cause the one or more processors to perform the steps of the method for code snippet locating according to any of the foregoing embodiments.

[0088] In an embodiment, one or more non-volatile computer-readable storage media is provided, which stores computer-readable instructions that, when executed by one or more processors, causing the one or more processors to perform the steps of the method for code snippet locating according to any of the foregoing embodiments.

[0089] Persons of ordinary skill in the art understand that all or part of the processes in the methods of the foregoing embodiments may be implemented by a computer program instructing relevant hardware. The computer program may be stored in a non-transitory computer-readable storage medium. When the computer program is executed, flows of embodiments of the methods as described above may be included. Any references to memory, storage, databases, or other media used in the various embodiments provided herein may include non-transitory and/or transitory memory. The non-transitory memory may include a read only memory (ROM), a programmable ROM (PROM), an electrically programmable ROM (EPROM), an electrically erasable programmable ROM (EEPROM), or a flash memory. The transitory memory may include a random access memory (RAM) or an external cache memory. By way of illustration and not limitation, RAM is available in a variety of forms such as static RAM (SRAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), double data rate SDRAM (DDRSDRAM), enhanced SDRAM

(ESDRAM), Synchlink DRAM (SLDRAM), Memory Bus (Rambus) Direct RAM (RDRAM), Direct Memory Bus Dynamic RAM (DRDRAM), Memory Bus Dynamic RAM (RDRAM) etc.

[0090] The foregoing respective technical features involved in the respective embodiments can be combined arbitrarily, for brevity, not all possible combinations of the respective technical features in the foregoing embodiments are described, however, to the extent they have no collision with each other, the combination of the respective technical features shall be considered to be within the scope of the description.

[0091] The foregoing implementations are merely specific embodiments of the present disclosure, and are not intended to limit the protection scope of the present disclosure. It should be noted that any variation or replacement readily figured out by persons skilled in the art within the technical scope disclosed in the present disclosure shall all fall into the protection scope of the present disclosure. Therefore, the protection scope of the present disclosure shall be subject to the protection scope of the claims.

What is claimed is:

1. A method for code snippet translation, applied to a code server, the method comprising:

acquiring a to-be-translated analysis result set; the to-be-translated analysis result set comprises a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

looking up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

identifying a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

calling a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code snippet of a corresponding target language type based on the translation policy.

2. The method according to claim 1, wherein the acquiring the to-be-translated analysis result set comprises:

acquiring a to-be-translated source code file;

reading a source code directory in the to-be-translated source code file; the directory of the to-be-translated source code comprises a plurality of initial file names;

traversing an initial code sub-file corresponding to each of the initial file names, and filtering out a to-be-translated code sub-file; the to-be-translated code sub-file matching a preset language type;

looking up a parser matching the preset language type; and

acquiring a snippet locating information of the to-be-translated code snippet in the to-be-translated code sub-file by the parser, extracting the to-be-translated code snippet based on the snippet locating information, and generating a to-be-translated analysis result set according to the to-be-translated code snippet and the snippet

locating information; the to-be-translated analysis result set comprises the to-be-translated code snippet in the to-be-translated code sub-file and the context code snippet corresponding to the to-be-translated code snippet.

3. The method according to claim 2, after the identifying the target language type and the translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor, the method further comprising:

determining a translation operation according to the translation policy by the code processor;

correspondingly replacing the to-be-translated code snippet in the to-be-translated source code file with the translated code snippet based on the snippet locating information when the translation operation is a replacing operation; and

outputting a translated source code file of the target language type.

4. The method according to claim 1, wherein the looking up the code processor matching the to-be-translated code snippet comprises:

identifying a programming language type of the to-be-translated code snippet based on a regular expression;

looking up a code processor set corresponding to the programming language type;

matching the to-be-translated code snippet with matching criteria corresponding to each code processor in the code processor set; and

regarding the code processor matching the matching criteria as the code processor matching the to-be-translated code snippet.

5. The method according to claim 1, wherein the calling the translator interface provided by the translator system by the code processor comprises:

acquiring a user identifier;

generating a translation request according to the user identifier, the to-be-translated code snippet, the target language type, and the translation policy;

sending the translation request to the translation server through a translator interface; the translation request is configured to instruct the translation server to translate the to-be-translated code snippet into the target language type using the translation policy after the user identifier is authenticated; and

receiving a translated code snippet of the target language type and returned by the translation server.

6. The method according to claim 5, wherein the generating the translation request according to the user identifier, the to-be-translated code snippet, the target language type, and the translation policy comprises:

concatenating the user identifier, to-be-translated code snippet, target language type, and translation policy to generate an initial string;

encrypting the initial string to obtain a request string; and

generating a translation request according to the request string corresponding to a plurality of to-be-translated code snippets when the to-be-translated analysis result set contains the plurality of to-be-translated code snippets.

7. The method according to any one of claims 1 to 6, the method further comprising:

performing a risk level assessment on a translation result of each to-be-translated code snippet by examining the context code snippet; and

recording a processing state of each to-be-translated code snippet.

8. An apparatus for code snippet translation, comprising:

an acquiring module, configured to acquire a to-be-translated analysis result set; the to-be-translated analysis result set comprises a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

a looking up module, configured to look up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

an identification module, configured to identify a target language type and a

translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

a translation module, configured to call a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code snippet of a corresponding target language type based on the translation policy.

9. A computer device comprising a memory and one or more processors, the memory storing computer-readable instructions that, when executed by the one or more processors, causing the one or more processors to perform the steps of:

acquiring a to-be-translated analysis result set; the to-be-translated analysis result set comprises a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

looking up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

identifying a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

calling a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code snippet of a corresponding target language type based on the translation policy.

10. The computer device according to claim 9, wherein the one or more processor further execute(s) the instructions to perform the steps of:

acquiring a source code directory in the to-be-translated source code file; the directory of the to-be-translated source code comprises a plurality of initial file names;

traversing an initial code sub-file corresponding to each of the initial file names,

and filtering out a to-be-translated code sub-file; the to-be-translated code sub-file matching a preset language type;

looking up a parser matching the preset language type; and

acquiring a snippet locating information of the to-be-translated code snippet in the to-be-translated code sub-file by the parser, extracting the to-be-translated code snippet based on the snippet locating information, and generating a to-be-translated analysis result set according to the to-be-translated code snippet and the snippet locating information; the to-be-translated analysis result set comprises the to-be-translated code snippet in the to-be-translated code sub-file and the context code snippet corresponding to the to-be-translated code snippet.

11. The computer device according to claim 10, wherein the one or more processor further execute(s) the instructions to perform the steps of:

determining a translation operation according to the translation policy by the code processor;

correspondingly replacing the to-be-translated code snippet in the to-be-translated source code file with the translated code snippet based on the snippet locating information when the translation operation is a replacing operation; and

outputting a translated source code file of the target language type.

12. The computer device according to claim 9, wherein the one or more processor further execute(s) the instructions to perform the steps of:

identifying a programming language type of the to-be-translated code snippet based on a regular expression;

looking up a code processor set corresponding to the programming language type;

matching the to-be-translated code snippet with matching criteria corresponding to each code processor in the code processor set; and

regarding the code processor matching the matching criteria as the code processor matching the to-be-translated code snippet.

13. The computer device according to claim 9, wherein the one or more processor further execute(s) the instructions to perform the steps of:

acquiring a user identifier;

generating a translation request according to the user identifier, the to-be-translated code snippet, the target language type, and the translation policy;

sending the translation request to the translation server through a translator interface; the translation request is configured to instruct the translation server to translate the to-be-translated code snippet into the target language type using the translation policy after the user identifier is authenticated; and

receiving a translated code snippet of the target language type and returned by the translation server.

14. The computer device according to claim 13, wherein the one or more processor further execute(s) the instructions to perform the steps of:

concatenating the user identifier, to-be-translated code snippet, target language type, and translation policy to generate an initial string;

encrypting the initial string to obtain a request string; and

generating a translation request according to the request string corresponding to a plurality of to-be-translated code snippets when the to-be-translated analysis result set contains the plurality of to-be-translated code snippets.

15. The computer device according to any one of claims 9 to 14, wherein the one or more processor further execute(s) the instructions to perform the steps of:

performing a risk level assessment on a translation result of each to-be-translated code snippet by examining the context code snippet; and

recording a processing state of each to-be-translated code snippet.

16. One or more non-volatile computer-readable storage media storing computer-readable instructions that, when executed by one or more processors,

causing the one or more processors to perform the steps of:

acquiring a to-be-translated analysis result set; the to-be-translated analysis result set comprises a to-be-translated code snippet and a context code snippet corresponding to the to-be-translated code snippet;

looking up a code processor matching the to-be-translated code snippet according to a programming language type of the to-be-translated code snippet;

identifying a target language type and a translation policy of the to-be-translated code snippet according to the context code snippet and by the code processor; and

calling a translator interface provided by a translator system by the code processor, the translator system being deployed on a translation server, and the translator system translating the to-be-translated code snippet into a translated code snippet of a corresponding target language type based on the translation policy.

17. The computer-readable storage medium of claim 16, wherein the one or more processors further execute(s) the instructions to perform the steps of:

acquiring a to-be-translated source code file;

reading a source code directory in the to-be-translated source code file; the directory of the to-be-translated source code comprises a plurality of initial file names;

traversing an initial code sub-file corresponding to each of the initial file names, and filtering out a to-be-translated code sub-file; the to-be-translated code sub-file matching a preset language type;

looking up a parser matching the preset language type; and

acquiring a snippet locating information of the to-be-translated code snippet in the to-be-translated code sub-file by the parser, extracting the to-be-translated code snippet based on the snippet locating information, and generating a to-be-translated analysis result set according to the to-be-translated code snippet and the snippet locating information; the to-be-translated analysis result set comprises the to-be-translated code snippet in the to-be-translated code sub-file and the context

code snippet corresponding to the to-be-translated code snippet.

18. The computer-readable storage medium of claim 16, wherein the one or more processors further execute(s) the instructions to perform the steps of:

identifying a programming language type of the to-be-translated code snippet based on a regular expression;

looking up a code processor set corresponding to the programming language type;

matching the to-be-translated code snippet with matching criteria corresponding to each code processor in the code processor set; and

regarding the code processor matching the matching criteria as the code processor matching the to-be-translated code snippet.

19. The computer-readable storage medium of claim 16, wherein the one or more processors further execute(s) the instructions to perform the steps of:

acquiring a user identifier;

generating a translation request according to the user identifier, the to-be-translated code snippet, the target language type, and the translation policy;

sending the translation request to the translation server through a translator interface; the translation request is configured to instruct the translation server to translate the to-be-translated code snippet into the target language type using the translation policy after the user identifier is authenticated; and

receiving a translated code snippet of the target language type and returned by the translation server.

20. The computer-readable storage medium according to any one of claims 16 to 19, wherein the one or more processors further execute(s) the instructions to perform the steps of:

performing a risk level assessment on a translation result of each to-be-translated code snippet by examining the context code snippet; and

recording a processing state of each to-be-translated code snippet.