(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 September 2001 (07.09.2001)

PCT

(10) International Publication Number
WO 01/65388 A1

(51) International Patent Classification[7]: G06F 15/00

(21) International Application Number: PCT/US01/06708

(22) International Filing Date: 28 February 2001 (28.02.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/514,483 28 February 2000 (28.02.2000) US

(71) Applicant: **CLICKSERVICES.COM** [US/US]; 39240 Liberty Street, Suite 250, Fremont, CA 94560 (US).

(72) Inventors: **NATARAJAN, Sundaram**; 38671 Chrisholm Place, Fremont, CA 94536 (US). **HA, Jack, T.**; 1560 Mission Springs Circle, San Jose, CA 95131 (US). **LEONG, Van, D.**; 586 Clyde Court, Milpitas, CA 95035 (US). **VERBIST, Luc, August, Jozef, Maria**; 2150 Park Boulevard, Palo Alto, CA 94306 (US).

(74) Agents: **MALLIE, Michael, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

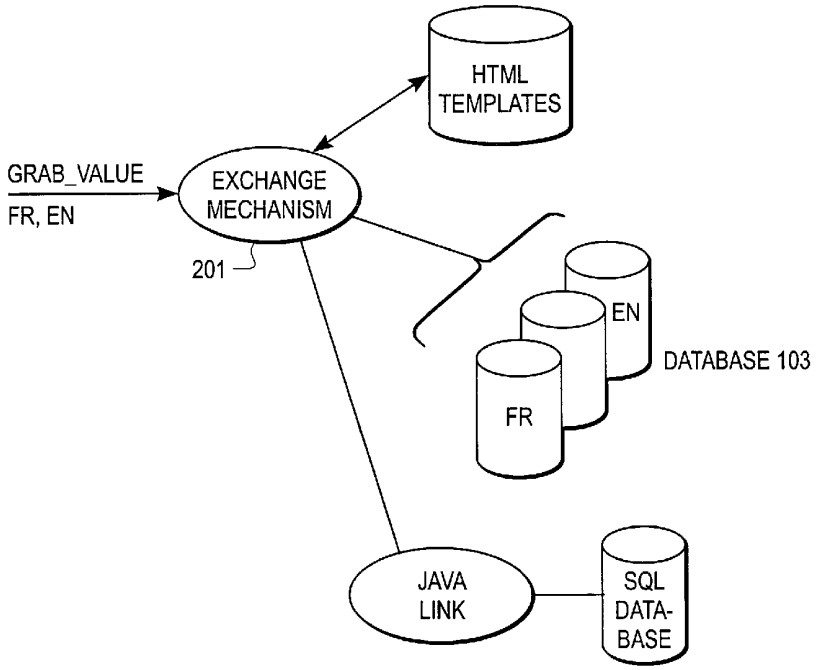
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: MULTI-LANGUAGE-MULTI TEMPLATE ARRANGEMENT



(57) Abstract: A multi-language multi-template architecture is described. In one embodiment, the architecture comprises a first memory storing a plurality of templates containing markup language for text (item HTML TEMPLATE), a second memory having stored therein the text in a plurality of languages (item DATABASE 103), and an exchange mechanism (item EXCHANGE MECHANISM 201) coupled to the first and second memories to generate a page by merging text in a particular language with one of the plurality of templates in response to an indication of the particular language.

WO 01/65388 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

MULTI-LANGUAGE - MULTI TEMPLATE ARRANGEMENT

FIELD OF THE INVENTION

The present invention relates to the field of data exchange; more particularly, the present invention relates to exchanging data between two programming environments, such as, for example, HTML and Java.

BACKGROUND

Exchange systems provide the mechanism by which information may be passed between two distinct programming environments. In other words, such exchange systems allows information that is in a form for use with one type of program to be accessible through another type of program.

In the prior art, the exchange systems are based on hardcoding of one type of programming in another. For example, in order to exchange information between the two programming environments, one prior art exchange system hardcoded HTML text in a Java program and output it to the user's browser. In such a system, the output of HTML may be hardcoded in a Java servlet. In another prior art exchange system, such as JSP, hardcoded Java programming inside an HTML page to enable the exchange of information.

In either of the two examples above, the hardcoding essentially ties the two different types of coding together, thereby causing problems. For example, with respect to JSP, if a change is desired in the Java code, the HTML environment must be used. If an individual does not want to use the HTML environment, they cannot duplicate the code and make the desired changes.

Problems also exist with hardcoding of HTML in a Java servlet. In addition to require that changes be made in the Java environment, these problems include a difficulty in maintenance and modification to the coding. With respect to maintenance, any correction that is necessary to the code requires a number of steps and eventually all of the code needs to be recompiled. The number of steps also increases the chance that an error will be introduced into the code. Other modifications to the code are also problematic. For

instance, if one desires to change the look and feel of their web site, the only way to make the change is to copy the existing code and redo the entire layout in cases where the new layout is going to ask for a different type of parameter for which no previous coding exists.

Hardcoding in the prior art is also limiting when multiple natural speaking languages are to be supported for particular content over the Internet. In the prior art, to support multiple natural speaking languages, separate and distinct web pages had to be created and stored for each distinct language. If a change to the content is to be made, then all the web pages having that content for each separate language had to be corrected. Making changes to a large number of pages increased the chance that an error is made in one or more of the pages. What is needed is a way of supporting multiple natural speaking languages while reducing the chance or affect of such errors when changes are necessary.

SUMMARY OF THE INVENTION

A multi-language multi-template architecture is described. In one embodiment, the architecture comprises a first memory storing a plurality of templates containing markup language for text, a second memory having stored therein the text in a plurality of languages, and an exchange mechanism coupled to the first and second memories to generate a page by merging text in a particular language with one of the plurality of templates in response to an indication of the particular language.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

Figure 1 illustrates one embodiment of a networked environment.

Figure 2 illustrates another embodiment of a networked environment.

Figure 3 is a block diagram of one embodiment of a network environment.

Figure 4 is a block diagram of an exemplary computer system.

Detailed Description

An exchange system is described. Such an exchange system may be used to support multiple markup languages (e.g., HTML, WML, HDML, etc.), multiple natural languages (e.g., English, French, Chinese, etc.), multiple user interaction flows (such as, e.g., various clicks on a web site, various button presses on a WAP phone, etc.), all of which may be based on a common underlying server (e.g., a Java server).

In the following description, numerous details are set forth. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

Some portions of the detailed descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven

convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be

appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

Overview of An Exchange Format

An exchange mechanism is provided to exchange information between a markup language execution environment and a Java programming execution environment, where "environment" refers to the programming language utilized therein. In one embodiment, the markup language may be an HTML, WML or another markup language. In such a case, the exchange mechanism translates variables that are in an HTML, WML, or other markup language into information that may be processed using Java and vice versa. In one embodiment, the exchange mechanism is implemented in XML.

A benefit of such an approach is that coding done in the HTML environment may operate independent of the programming in the Java environment (as well as the exchange environment) apart from an interface thereto.

In one embodiment, the HTML environment includes multiple HTML templates. Each HTML template contains text or other data that may be substituted from the Java environment through the exchange mechanism. A complete HTML web page may, in turn, contain multiple templates that are substituted into the web page through the exchange mechanism. In this manner, a web page may be constructed in a modular fashion, thereby simplifying the creation and modification of a web site.

With the HTML template, the look and feel of the display is present without the dynamic data (e.g., individual's name, phone number, etc.). The

dynamic data is obtained from a database via the exchange mechanism. The exchange mechanism translates the database information into a table. The table maintains a reference to allow access to the Java content that was requested by the HTML environment by associating a variable in the HTML environment with the actual data in the Java environment. In one embodiment, the table provides a reference between the information in Java objects and a specific substantiation of an HTML template. In this manner, using variables in the HTML environment may be substituted for real values from the Java environment and a relationship is maintained between the substituted text in the HTML document and its representation in the Java environment.

The exchange mechanism operates using software that provides a set of functions to access Java content and allow building up the table. The functions define the method by which the data from the database is accessed and stored in the table.

In one embodiment, the table is a key value table that may be based on property values that are well known in the art for use in Java programming. The key value table associates a key word with its specific data (e.g., the key being "phone number" and the value being the actual number itself).

Once the data has been entered into the table, other functions of the exchange mechanism translate the table and merge it with the HTML template to build an HTML page. In one embodiment, the exchange mechanism uses a display function to cause the merge operation to occur. Thus, use of the exchange mechanism provides increase flexibility in creating a web page because the HTML template can refer to some or all the data in the table. In one embodiment, the same display can occur different depending on what data is found in the table. In such a case, a function executed by the exchange mechanism is a comparison operation that compares data and selects certain data for entry in the table based on the outcome of the comparison.

Thus, an HTML page may be displayed as a response of the exchange mechanism.

By entering information on the web page associated with one of the web page templates, the individual causes a request to be made to the exchange mechanism, specifying variables that are inputs to the request. The exchange mechanism transfers those variables to the Java environment according to the exchange definition being used. The exchange mechanism also executes the underlying Java functionality specified in the exchange definition. The result of the execution is transferred from the Java environment and used by the exchange mechanism to construct a new web page as a response.

More specifically, the exchange mechanism handles a request based on the exchange definition that may take the input information from the HTML page and store it in the table. Then the exchange mechanism calls functions to assign the information correctly into the database. The HTML page need only indicate what functions to call to obtain a particular piece of data and assign a variable to it. For example, if the HTML page provides a paging number, then the HTML page need only add an assignment statement assigning the paging number to a variable recognized and used by an exchange definition in the exchange mechanism. If a voice number is to be added, the HTML need only add another assign statement to cause another variable to be assigned.

One benefit of this approach is the HTML programmer is allowed to change the flow of what data appears on web pages. Specifically, because an individual HTML template may be set up with a variety of references and each of the references are linked to executable Java routines, the selection of which information to put in the template essentially controls the flow of information to a user in the HTML environment.

An Exemplary Data Exchange Environment

Figure 1 illustrates one embodiment of a networked environment showing an HTML template 101, an exchange mechanism 102, and a Java link 103. The programming of HTML template 101 includes statements that call for substitution of Java content. For example, HTML template 101 may include the following statement:

<subst data="fname"/>

where data that is being entered by the user is associated with the variable "fname."

Exchange mechanism 102 includes an exchange definition (e.g., code) to handle the substitution requests from individual users. In one embodiment, exchange mechanism 102 includes a submit function that is responsive to submission of one instantiation of the HTML template. For example, the exchange mechanism may include the following:

```
<submit
  <response ...
      <assign frame="firstn" name="fname" type="string"/>
...
  <response>
```

When the user in the HTML environment specifies content, these response operations used by exchange mechanism 102 translate the request into a request for a Java variable to assign to a specific instance of HTML content. The submit function performed by the exchange mechanism 102 invokes the translation between the HTML environment and the Java environment. In response to a substitute request by the HTML environment, exchange mechanism 102 causes the assignment of a specific Java content obtained from a database, such as database 104, via Java link 103, and the Java content is made part of a Java object with a substitute data reference. Thus, using the assign statement, the Java content is accessed from database 104 and a Java object is created for the submit function. For example, the assign statement above specifies that the Java variable "firstn" is to be assigned to the HTML variable

"fname". The assign statement also specifies that the Java content used is a string.

In the example of above, the assign function is executed to obtain the first name. The execution of the function obtains the "Joe" out of the database. (Note that the user in the HTML environment specified one or more parameters that were submitted to exchange mechanism 102 to help identify the specific Java content "Joe" to be accessed.) Once "Joe" is accessed from database 104, a Java object is created and is treated as a string.

Thus, when the information is to be potentially input into the web page, an association is made between the substitute functions in the HTML template of the information input by the user. The exchange information receives programming statements that substitute data for a certain user specified information.

In the Java environment, the result of execution of the functions is to generate Java objects when accessing information in the database. Thus, when information from the HTML environment is received by the exchange mechanism, the exchange mechanism determines which Java object is being called and which Java object is to be associated with the specific template reference tag in the HTML template.

Besides assigning, comparison statements are also available as statements that detect whether a variable in Java is equal to a particular input. An example of one such comparison statement is given below in the exchange examples.

Exchange mechanism 102 creates a table to indicate assignments between the created Java object and its HTML reference. In this manner, a reference in the HTML environment to the same variable will be automatically associated with the correct Java object without having to access database 104 again. The table is maintained by exchange mechanism 102 to allow the Java content for references in the HTML document to be accessed quicker and repeatedly. Thus, the exchange mechanism 102 provides an interface that allows the building up of a table and how to retrieve information from that table.

In essence, exchange mechanism 102 includes a first API to allow Java objects to be created when requested and a second API that allows execution of Java code in order to generate web pages in an HTML environment.

After the table has been created, the exchange mechanism can, as a response, invoke a merge operation to create a web page by merging the table into the HTML template or, as a request, can cause data in the table to be received into the database 104.

An Exemplary Portion of an Exchange Definition

The following are examples of the exchange files that are used.

1. The user browses to a page that should show a previously scheduled appointment. This fragment of an exchange file causes the appointment information to be retrieved from the database, and displayed as an HTML page in the browser:

```
<exch template="evedit">

 <!-- First, retrieve the appointment from the database -->
 <submit class="$evMethod" method="get"
      name="displayEvEdit"
      next="displayEv">
  <request class="$ev" method="setEv">
   <assign from="id" name="id" encrypt="decode"/>
  </request>
 </submit>

 <!-- Then, display a page with that appointment -->
 <submit class="$evMethod" method="display"
      name="displayEv">
  <response class="$ev" method="getEv">
   <assign name="location" from="location"/>
```

```
    <assign name="description" from="description"/>
    <!-- Check an appropriate radio button -->
    <compare from="isrecurring" with="Y"
         name="rrule_yes" value="checked"/>
    <compare from="isrecurring" with="N"
         name="rrule_no" value="checked"/>
  </response>
 </submit>

</exch>
```

2. Below is a fragment of an HTML template which is used in conjunction with the exchange file to display the appointment.

```
<html>
 <head>
  <title><subst lang="i_evedit_title"/></title>
 </head>
 <body>
  <form name="form1" method=post
     action="<subst data="servlet"/>">

   <input type=hidden name=id
      value="<subst data="id"/>">

   <subst lang="i_location"/> :
   <input type="text" name=location
      value="<subst data="location"/>">

   <subst lang="i_description"/> :
```

```
<input type="text" name=description
    value="<subst data="description"/>">

<input type=radio name=rrule value=Y
    <subst data="rrule_yes"/>>
<subst lang="i_recurring"/>

<input type=radio name=rrule value=N
    <subst data="rrule_no"/>>
<subst lang="i_not_recurring"/>

<input type=submit name=save
    value="<subst lang="i_save"/>">
</form>
</body>
</html>
```

In various places, <subst lang...> is used to display fixed text in the user's natural language.

The appointment id stored in a hidden field in the page, so it can be read back when the user hits the save button. The <subst data...> is replaced with the corresponding data from the table prepared by the exchange mechanism.

The location and description fields initially display data from the database, but they can be changed by the user.

There is a radio button with two possible states, either "recurring" or "not recurring". One of these states will be selected initially, because the table produced by exchange will contain either "rrule_yes = checked", or "rrule_no = checked".

At the bottom of the page is a "save" button, which will invoke the fragment of exchange given below, to save the changed fields in the database.

3. The user changes some HTML fields on the appointment page and hits a "save" button. This fragment of an exchange file causes the new information to be saved in the database.

```
<exch template="evedit">

  <submit class="$evMethod" method="edit"
      name="save"
      next="displayEv">
    <request class="$ev" method="setEv">
      <constant name="type" value="EVENT"/>
      <assign from="id" name="id" encrypt="decode"/>
      <assign from="location" name="location"/>
      <assign from="description" name="description"/>
      <assign from="rrule" name="isrecurring"/>
    </request>
  </submit>

</exch>
```

Multi-language and Multi-template Arrangement

In one embodiment, the exchange mechanism may be used to facilitate multi-language and multi-template operation. Figure 2 illustrates another embodiment of a networked environment. Referring to Figure 2, the environment comprises exchange mechanism 201 that obtains a language indication, referred to herein as the grab_language value 202, from the user. Exchange mechanism 201 obtains the language value automatically. In one embodiment, the language value is set by the user as part of their browser preferences. In an alternative embodiment, the language indication is set dynamically (e.g., based on user's location, location of service provider, etc.).

In response to the language indication, exchange mechanism 201 selects one of the databases 203. Each of databases 203 is dedicated to a particular language and stores phrases that have been translated into that particular language. In one embodiment, the databases 203 are property files.

The environment also includes HTML template storage 204 and Java link 205. Java link 205 provides access to Java content database 206, which stores information. Each template has substitute language to receipt of information from database 206.

When a page is requested, exchange mechanism 201 causes text in the language indicated by the grab-language value 202 to be substituted into the page. That is, templates (from the HTML template storage) include information that causes the exchange mechanism to substitute text into an HTML document. In one embodiment, the substitution is performed by merging the template and text in the same manner as merging is described above. In one embodiment, exchange mechanism 201 performs the substitution in response to the following command:

<subst lang = ?phrase_keyword? type ?kind?

where the term "phrase_keyword" identifies a particular phrase that has been translated into multiple languages. The term "kind" may be static, dynamic, or final. If the kind is static, then the field does not change once the information has been set. If the kind is dynamic, the fields may be regularly updated and the statement indicates to the exchange mechanism that updated information is to be used and/or retrieved (which the exchange mechanism does). If the kind is final, the field may be substituted with information that was subject to dynamic updating but is now is to be made static. For example, in the case of a table that is being substituted, the data in the table may be updated every time the substitute command is executed. However, once the field has been made final, substitution no longer occurs. Note that the information may be received from external sources, in a manner described in U.S. Patent Application Serial No. _____, entitled "Information Services," filed concurrently herewith, assigned to the corporate assignee, and incorporated herein by reference.

In one embodiment, the substitute command may be an "if" statement which provides alternative actions to be performed dependent on whether a condition precedent has been met.

Being able to use multiple languages allows for communication to occur with individuals who communicate using different languages.

An Exemplary Network

Figure 3 is a block diagram of one embodiment of a network environment 301 that may be used in the translation technique describe above. In one embodiment, a server computer system 300 is coupled to a wide-area network 310. Wide-area network 310 may include the Internet or other proprietary networks including, but not limited to, America On-Line™, CompuServe™, Microsoft Network™, and Prodigy™. Wide-area network 310 may include conventional network backbones, long-haul telephone lines, Internet and/or Intranet service providers, various levels of network routers, and other conventional mechanisms for routing data between computers. Using network protocols, server 300 may communicate through wide-area network 310 to client computer systems 320, 330, 340, which are possibly connected through wide-area network 310 in various ways or directly connected to server 300. For example, client 340 is connected directly to wide-area network 310 through direct or dial-up telephone or other network transmission line.

Alternatively, clients 330 may be connected through wide-area network 310 using a modem pool 314. Modem pool 314 allows multiple client systems to connect with a smaller set of modems in modem pool 314 for connection through wide-area network 310. Clients 331 may also be connected directly to server 300 or be coupled to server through modem 315. In another alternative network typology, wide-area network 310 is connected to a gateway computer 312. Gateway computer 312 is used to route data to clients 320 through a local area network 316. In this manner, clients 320 can communicate with each other

through local area network (LAN) 316 or with server 300 through gateway 312 and wide-area network 310. Alternatively, LAN 317 may be directly connected to server 300 and clients 321 may be connected through LAN 317.

Using one of a variety of network connection mechanisms, server computer 300 can communicate with client computers 350. In one embodiment, a server computer 300 may operate as a web server if the World-Wide Web ("WWW") portion of the Internet is used for wide area network 310. Using the HTTP protocol and the HTML coding language, such a web server may communicate across the World-Wide Web with clients 350. In this configuration, clients 350 use a client application program known as a web browser such as the Netscape™ Navigator™, the Internet Explorer™, the user interface of America On-Line™, or the web browser or HTML translator of any other conventional supplier. Using such browsers and the World Wide Web, clients 350 may access graphical and textual data or video, audio, or tactile data provided by the web server 300.

In one embodiment, server 300 contains the exchange mechanism and the database storing Java content.

An Exemplary Computer System

Figure 4 is a block diagram of an exemplary computer system. Referring to Figure 4, computer system 400 may comprise an exemplary client 150 or server 100 computer system. Computer system 400 comprises a communication mechanism or bus 411 for communicating information, and a processor 412 coupled with bus 411 for processing information. Processor 412 includes a microprocessor, but is not limited to a microprocessor, such as, for example, Pentium™, PowerPC™, Alpha™, etc.

System 400 further comprises a random access memory (RAM), or other dynamic storage device 404 (referred to as main memory) coupled to bus 411 for storing information and instructions to be executed by processor 412. Main

memory 404 also may be used for storing temporary variables or other intermediate information during execution of instructions by processor 412. In one embodiment, main memory 404 has a portion of its memory allocated to a ad database for storing Java content.

Computer system 400 also comprises a read only memory (ROM) and/or other static storage device 406 coupled to bus 411 for storing static information and instructions for processor 412, and a data storage device 407, such as a magnetic disk or optical disk and its corresponding disk drive. Data storage device 407 is coupled to bus 411 for storing information and instructions.

Computer system 400 may further be coupled to a display device 421, such as a cathode ray tube (CRT) or liquid crystal display (LCD), coupled to bus 411 for displaying information to a computer user. An alphanumeric input device 422, including alphanumeric and other keys, may also be coupled to bus 411 for communicating information and command selections to processor 412. An additional user input device is cursor control 423, such as a mouse, trackball, trackpad, stylus, or cursor direction keys, coupled to bus 411 for communicating direction information and command selections to processor 412, and for controlling cursor movement on display 421.

Another device which may be coupled to bus 411 is hard copy device 424, which may be used for printing instructions, data, or other information on a medium such as paper, film, or similar types of media. Furthermore, a sound recording and playback device, such as a speaker and/or microphone may optionally be coupled to bus 411 for audio interfacing with computer system 400. Note that any or all of the components of system 400 and associated hardware may be used in the present invention. However, it can be appreciated that other configurations of the computer system may include some or all of the devices.

Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular

embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as essential to the invention.

CLAIMS

We claim:

1. An architecture comprising:

a first memory storing a plurality of templates containing markup language for text;

a second memory having stored therein the text in a plurality of languages;

an exchange mechanism coupled to the first and second memories to generate a page by merging text in a particular language with one of the plurality of templates in response to an indication of the particular language.

2. The architecture defined in Claim 1 wherein the exchange mechanism obtains the indication automatically.

3. The architecture defined in Claim 2 wherein the exchange mechanism obtains the indication via receipt of a browser preference.

4. The architecture defined in Claim 2 wherein the exchange mechanism obtains the indication via a user entry.

5. The architecture defined in Claim 1 wherein the second memory comprises a plurality of partitions of storage, each of the plurality of partitions being for a different language, wherein the exchange mechanism selects one of the plurality of partitions based on the indication.

6. The architecture defined in Claim 5 wherein each of the plurality of partitions comprises a database.

7. The architecture defined in Claim 1 wherein the exchange mechanism substitutes text in the particular language into the template.

8. The architecture defined in Claim 7 wherein the exchange mechanism executes a command to substitute the text using static information.

9. The architecture defined in Claim 7 wherein the exchange mechanism executes a command to substitute the text using dynamic information that is determined at substitution time.

10. A method comprising:

storing a plurality of templates in a first memory, the templates containing markup language for text;

storing text in a plurality of languages in a second memory;

generating a page by merging text in a particular language with one of the plurality of templates in response to an indication of the particular language.

11. The method defined in Claim 10 obtaining the indication automatically.

12. The method defined in Claim 11 wherein obtaining the indication comprises receiving the indication from a browser preference.

13. The method defined in Claim 11 wherein obtaining the indication comprises receiving the indication via a user entry.

14. The method defined in Claim 10 further comprising selecting one of a plurality of partitions of the second memory based on the indication, each of the plurality of partitions being for a different language.

15. The method defined in Claim 14 wherein each of the plurality of partitions comprises a database.

16. The method defined in Claim 10 further comprising substituting text in the particular language into the template.

17. The method defined in Claim 16 further comprising executing a command to substitute the text using static information.

18. The method defined in Claim 16 further comprising executing a command to substitute the text using dynamic information that is determined at substitution time.

19. A computer software product having one or more recordable media with executable instructions stored thereon which, when executed by a processing device, cause the processing device to:

store a plurality of templates in a first memory, the templates containing markup language for text;

store text in a plurality of languages in a second memory; and

generate a page by merging text in a particular language with one of the plurality of templates in response to an indication of the particular language.

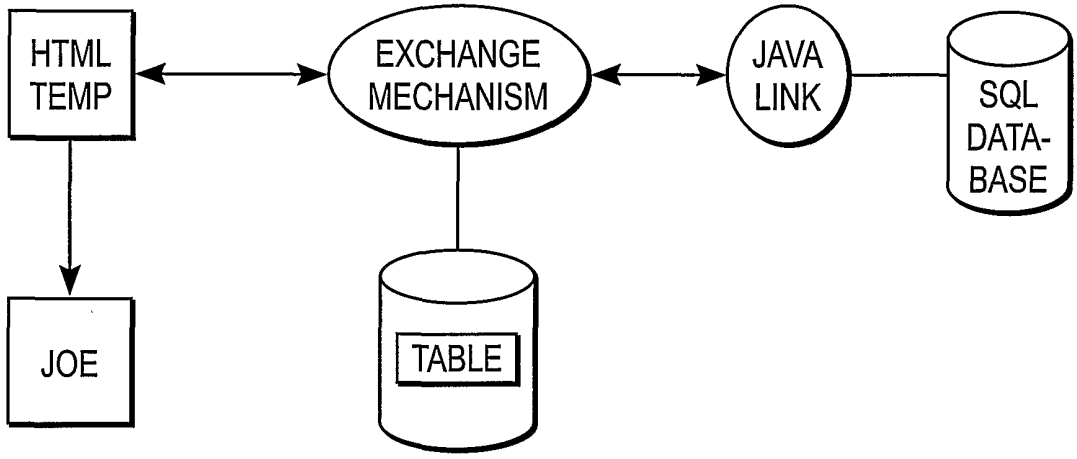

HTML
TEMP
EXCHANGE
MECHANISM
JAVA
LINK
SQL
DATA-
BASE
JOE
TABLE

FIG. 1

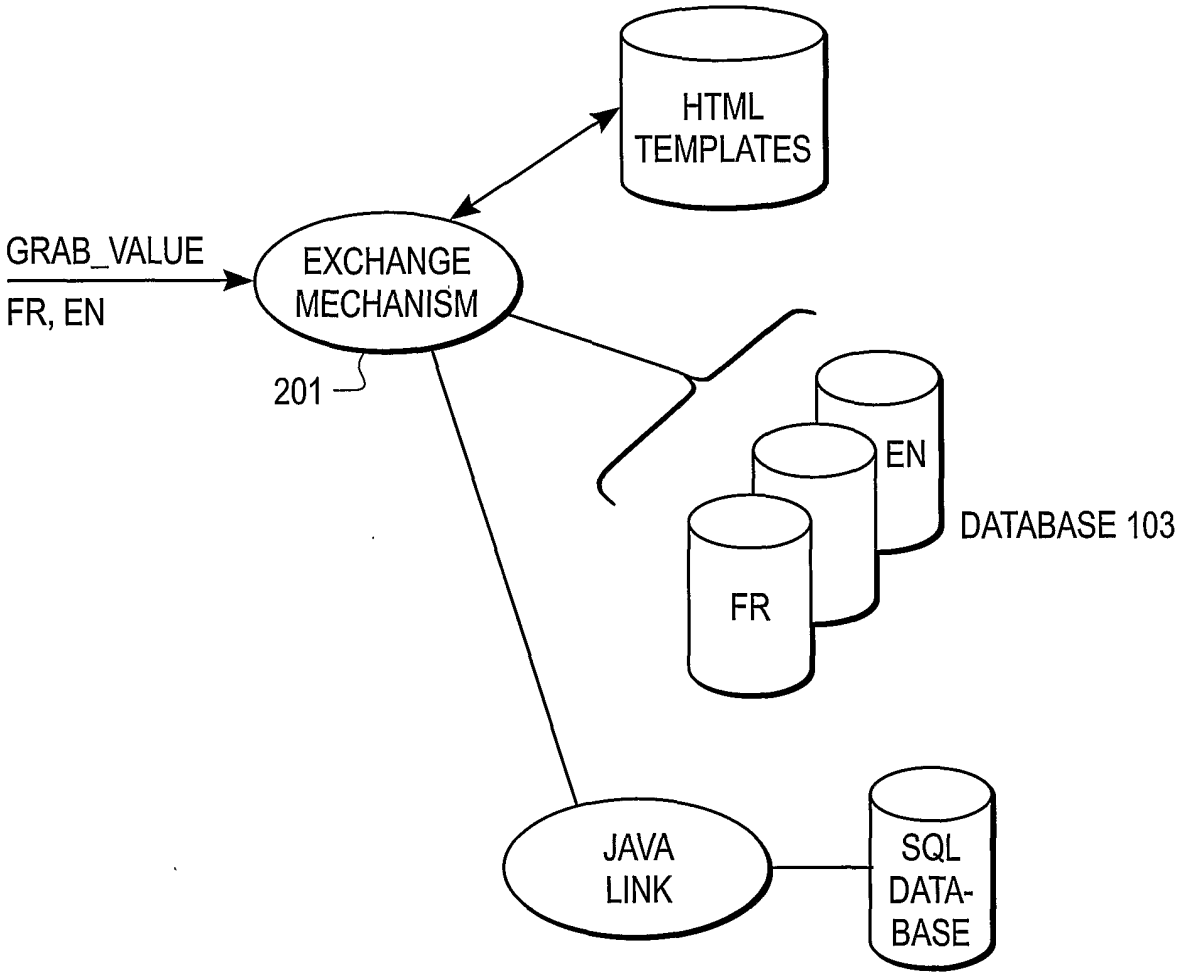

HTML
TEMPLATES
GRAB_VALUE
FR, EN
EXCHANGE
MECHANISM
201
EN
FR
DATABASE 103
JAVA
LINK
SQL
DATA-
BASE

FIG. 2

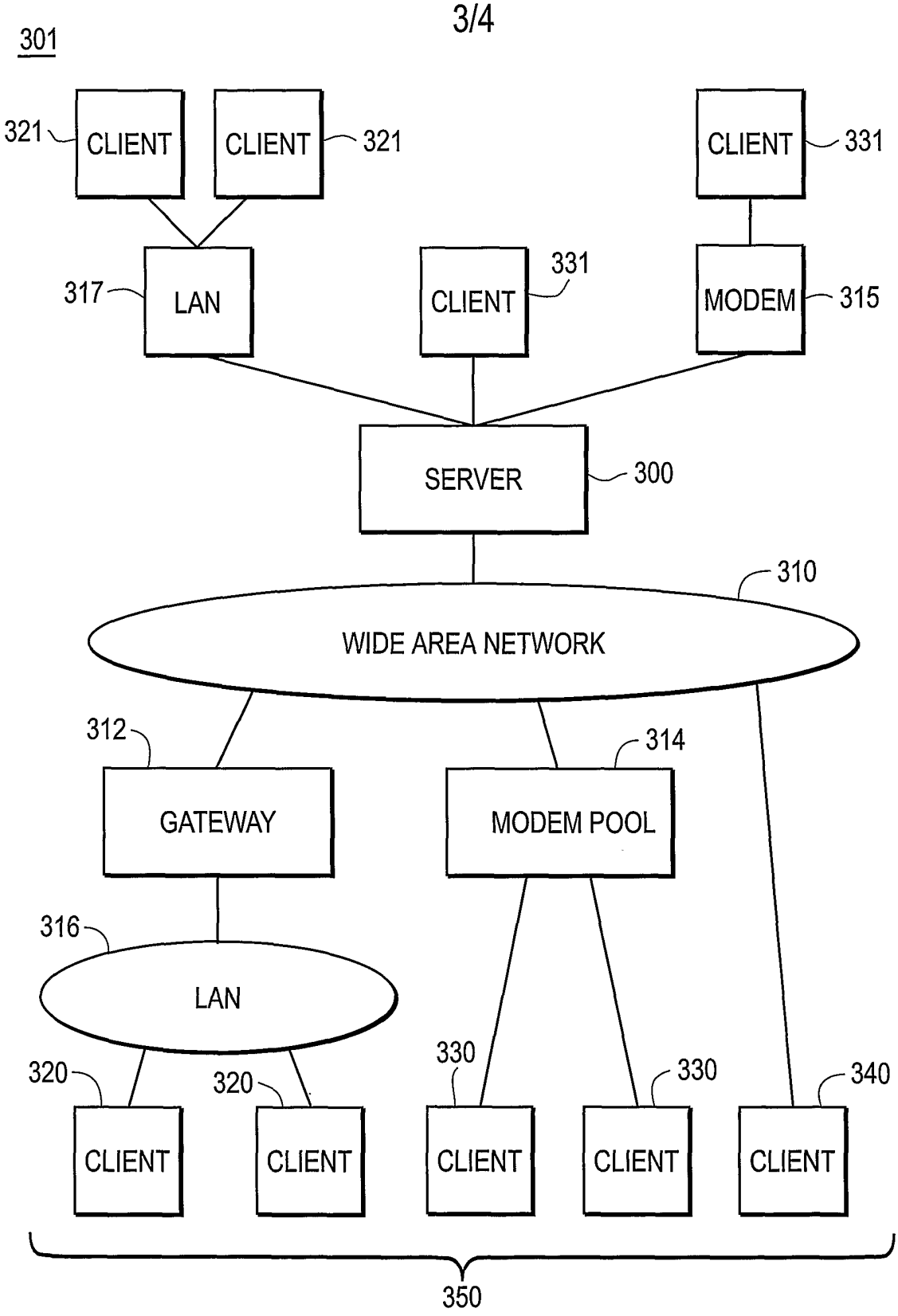

301
321
CLIENT
CLIENT
321
CLIENT
331
317
LAN
CLIENT
331
MODEM
315
SERVER
300
310
WIDE AREA NETWORK
312
GATEWAY
314
MODEM POOL
316
LAN
320
320
330
330
340
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
350

FIG. 3

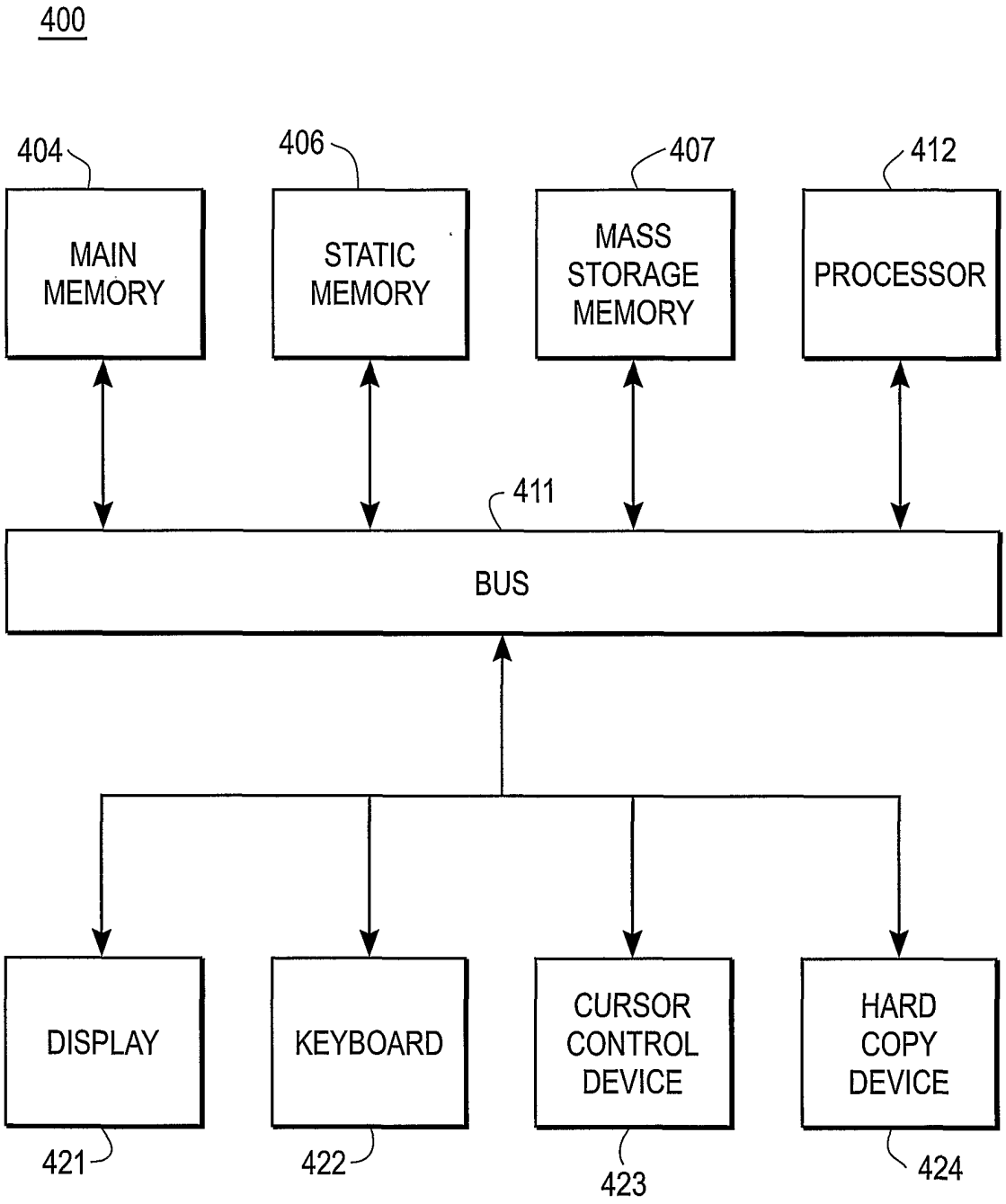

400
404
406
407
412
MAIN MEMORY
STATIC MEMORY
MASS STORAGE MEMORY
PROCESSOR
411
BUS
DISPLAY
KEYBOARD
CURSOR CONTROL DEVICE
HARD COPY DEVICE
421
422
423
424

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/06708

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 15/00
US CL : 707/513, 514, 536

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/513, 514, 536

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
East, IEEE

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X, P	US 6,161,139 A (WIN et al.) 12 December 2000 (12.12.2000), col 5, lines 55-62; col 11, lines 57-62; col 20, lines 30-38; col 21, lines 29-35; col 27, lines 47-51.	1-19
A	US 5,946,458 A (AUSTIN et al.) 31 August 1999 (31.08.1999), col 4, lines 49-67 to col 5, lines 1-2; col 9, lines 52-67.	7-9, 16-18
A	US 5,535,120 A (CHONG et al.) 09 July 1996 (09.07.1996), abstract, col 4, lines 40-60.	1-19

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
19 April 2001 (19.04.2001)

Date of mailing of the international search report
04 MAY 2001

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Heather Herndon James R. Matthews
Telephone No. 703-305-5186

Form PCT/ISA/210 (second sheet) (July 1998)