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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 October 2001 (11.10.2001)

PCT

(10) International Publication Number
WO 01/75597 A2

(51) International Patent Classification[7]: G06F 9/44

(21) International Application Number: PCT/US01/10626

(22) International Filing Date: 2 April 2001 (02.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/540,189 31 March 2000 (31.03.2000) US

(71) Applicant: SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, Palo Alto, CA 94303 (US).

(72) Inventor: CRAWFORD, Dan; 2-93 Murray Street, Ottawa, Ontario K1N 5M5 (CA).

(74) Agent: KOWERT, Robert, C.; Conley, Rose & Tayon, P.C., P.O. Box 398, Austin, TX 78767-0398 (US).

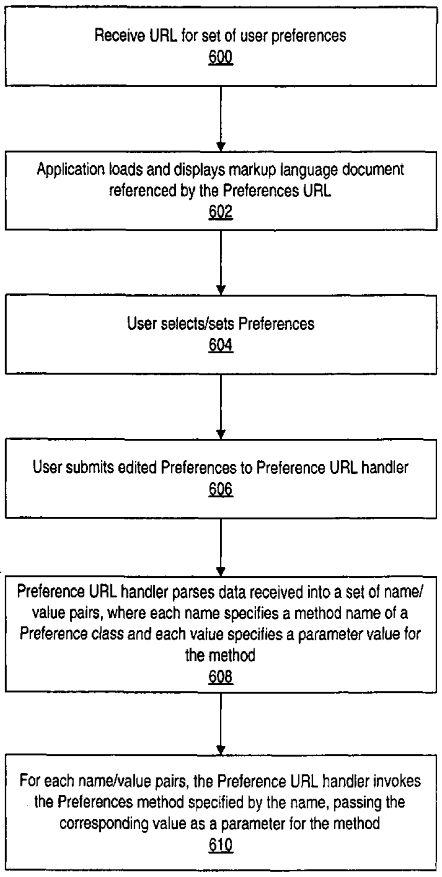
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, 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:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: EFFICIENT USER INTERFACE FOR SETTING APPLICATION PROGRAM USER PREFERENCES


Receive URL for set of user preferences
600
Application loads and displays markup language document referenced by the Preferences URL
602
User selects/sets Preferences
604
User submits edited Preferences to Preference URL handler
606
Preference URL handler parses data received into a set of name/value pairs, where each name specifies a method name of a Preference class and each value specifies a parameter value for the method
608
For each name/value pairs, the Preference URL handler invokes the Preferences method specified by the name, passing the corresponding value as a parameter for the method
610

(57) Abstract: An efficient graphical user interface to manage a set of user preferences for applications, such as browsers, running on a small footprint device. A user or an application may provide a URL to an application, such as a browser, to launch a graphical user interface to display a set of user preferences for the application. The graphical user interface may be programmed in a markup language to display the set of user preferences for the application such as the browser. The graphical user interface to display the set of user preferences may be programmed in a markup language such as HTML. The set of user preferences may comprise one or more user configurable parameters. The user may use the graphical user interface to edit the initial or existing values associated with the user configurable parameters included with the set of user preferences for the application. On completion of the editing process by the user, the application may save the edited parameter values for the user set of preferences in the memory of the small footprint device.

WO 01/75597 A2

TITLE: EFFECIENT USER INTERFACE FOR SETTING APPLICATION PROGRAM USER PREFERENCES

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to computer software programs and small footprint devices. More particularly, the present invention relates to a system and method for using an efficient graphical user interface to manage a set of user preferences for applications, such as browsers, running on a small footprint device.

2. Description of the Related Art

The field of "smart" small footprint devices is growing and changing rapidly. Small footprint devices include handheld computers, personal data assistants (PDAs), cellular phones, global positioning system (GPS) receivers, game consoles, set top boxes, and many more such devices. Small footprint devices are becoming increasingly powerful and may be capable of running software applications or services historically associated with general computing devices, such as desktop computers. For example, many small footprint devices are now capable of running browser applications. However, since memory, processing power, and other resources are typically very limited in small footprint devices, specialized, optimized versions of the services and/or application software are often necessary to execute in a small footprint device. A further goal of the optimization process may be to maintain a high degree of compatibility and 'look and feel' consistency between applications running on computer systems in general and on small footprint devices.

As small footprint devices have become more powerful and the software applications running on small footprint devices have become more complex, it has become increasingly feasible and desirable to enable users to easily interact with the applications through graphical user interfaces (GUIs). However, experience has shown that many techniques commonly used when implementing a GUI intended to run in a resource-rich environment such as a desktop computer have various drawbacks if applied to a GUI intended to run in a small footprint device. Small footprint devices may have various computing resource constraints, such as the limited availability of memory, processing power, display properties and other resources. For example, the resolution of a small footprint device display may be limited to a 400x400 pixel screen in comparison to a 1600x1280 pixel screen which is typical for desktop computers. This may limit the clarity and the amount of information that can be displayed.

For commonly used desktop computer platforms, users may use commercially available browsers. It is common practice for the user to customize the browsers by setting values to one or more user configurable parameters included in a set of user preferences. As an example, the user configurable parameters may include selecting the level of security desired, determining if cookies are to be accepted, preferring if graphics are to be loaded at all times, etc. Examples of values that may be set for the user configurable parameters may include, binary values such as Yes/No, 1/0, On/Off, etc. and other alphanumeric values such as 30, www.sun.com, etc. As an example, to set up the user configurable parameters for one browser, the user may select View → Internet Options from the browser menu or, in case of another browser, the user may select Edit → Preferences. On making these selections, a pop-up window, as shown in Figures 1and 2, may appear on the desktop computer screen allowing the user to select or edit the preferences for the particular browser application. Once the user preferences are selected and saved they would be used by the browser application program to modify its run time characteristics when executed. Unfortunately, many such application programs such as the browser, may require relatively large memory capacity and/or large displays and thus may have very limited use in small footprint devices.

Many application programs may use computer platform dependent programming commands and/or instructions to set the user configurable parameters included in the set of user preferences. The computer programming language which is used in generating and processing the display for user preferences may not have been specifically designed and/or optimized for running on a small footprint device. The source code for displaying and processing the set of user preferences may be typically a part of the application program code base and/or the operating system. For example, the source code, which display's the pop-up window and processes user inputs for setting the user preferences, for commercially available browsers, may be a separate part of the application program source code. The source code may also leverage certain operating system GUI features. The operating system software for many small footprint devices may not support programming instructions to display pop-up windows. The factors discussed above may result in the application program requiring additional computer system memory and processor. In addition, the application program source code may become optimized for a particular computing environment, thereby limiting its use.

It would be desirable to provide a method and a system for using an efficient graphical user interface to manage a set of user preferences for applications, such as browsers, running on a small footprint device. It would be desirable for an application, such as the browser, to utilize less computing resources such as memory, processor, etc. It would be also desirable for the graphical user interface to manage a set of user preferences, to be independent of the characteristics of the computing device such as processor, operating system, etc.

SUMMARY OF THE INVENTION

The problems outlined above may in large part be solved by a method and system for using an efficient graphical user interface to manage a set of user preferences for applications, such as browsers, running on a small footprint device or other device.

In one embodiment, a user may provide a location, e.g. a URL, to an application, such as a browser, to launch a graphical user interface to display a set of user preferences for the application. In another embodiment, a first application, such as a calendar manager, may provide a location, e.g. a URL, to a second application, such as a browser, to launch a graphical user interface to display a set of user preferences for the first application. Providing a location, e.g. a URL, to a browser may be accomplished in a variety of ways. In one embodiment, pressing a specific key coupled to the input mechanism for the small footprint device may result in the application receiving the location. In another embodiment, a user could access the application menu to launch the graphical user interface. Alternatively, if the application already has a markup language display capability, like a browser, it may receive the URL itself or be pre-set for a specific preferences GUI URL. If the application is not a browser itself, selecting a preference menu or a specific key for the application may launch the browser using a specified location, e.g. a URL.

The document indicated by the URL, which may represent one embodiment of a graphical user interface, may be authored (or programmed) in a markup language. Execution of the document may result in a browser type graphical user interface, which may display the user configurable parameters included in the set of user preferences for the application, such as the browser. One embodiment of a markup language may be HTML.

The user may use the graphical user interface to edit the initial or existing values for the user configurable parameters included with the set of user preferences for the application. On completion of the editing process by the user, the application may validate the data submitted by the user. If submitted data is found to be error free, then the application may save the edited parameter values, for the user set of preferences, in the memory of the small footprint device. The application for which the preferences were being set may be the application that displayed the markup language GUI document or it may be another application.

Thus, the markup language page display capabilities already included as a part of a browser running on a small footprint device, may be used for configuring preferences for the browser or for other applications loaded on the small footprint device. Furthermore, this may be accomplished without the need for small footprint device to support pop-up windows, etc. as used on systems that are more powerful. Of course, the method can be advantageously used in conjunction with applications on devices other than small footprint devices such as general purpose computers, workstations, personal computers and other devices as well.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects and advantages of the invention will become apparent upon reading the following detailed description and upon reference to the accompanying drawings in which:

Figures 1and 2 show a graphical user interface, using a pop-up window, for setting user preferences for two known browser application programs;

Figure 3 illustrates a block diagram of an exemplary system;

Figure 4 illustrates one embodiment of a graphical user interface for setting user preferences in a small footprint device or other device;

Figure 5 illustrates one embodiment of a markup language source code to generate the graphical user interface to set user preferences for an application in a small footprint device or other device; and

Figure 6 is a flow diagram illustrating one embodiment of a process for managing user preferences for an application running in a small footprint device or other device.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

Figure 3

In various embodiments, the system and method for using an efficient graphical user interface to manage a set of user preferences for applications, such as browsers, in accordance with the present invention, may be utilized within the environment of a small footprint device. Figure 3 is a block diagram illustrating aspects of one embodiment of a system 720 embodied within a small footprint device.

The system 720 may include a processor 710. Processor 710 may be any of various types, including an x86 processor, e.g., a Pentium class, a PowerPC processor, as well as other less powerful processors or processors developed specifically for small footprint devices, such as DSP's (digital signal processors) and/or embedded processors. The processor 710 may have various clock speeds, including clock speeds similar to those found in desktop computer-class processors, as well as lower speeds such as 16 MHz. It is noted that the small footprint device may have various different architectures, as desired.

The system 720 also includes a memory 712 coupled to the processor 710. The memory 712 may include any of various types of memory, including DRAM, SRAM, EDO RAM, etc., or a non-volatile memory such as a magnetic media, e.g., a hard drive, or optical storage. The memory 712 may include other types of memory as well, or combinations thereof.

As shown in Figure 3, the memory 712 may store one or more application programs 718, which includes code for implementing a method to use an efficient graphical user interface to manage a set of user preferences for applications, such as browsers, as described below. Included with the application program 718 may be, e.g. class definitions or modules, for implementing various objects and methods to manage a set of user preferences for applications, such as described in conjunction with Figures 3-6.

As shown in Figure 3, the system 720 also includes a display 716. The display 716 may be any of various types, such as an LCD (liquid crystal display), a CRT (cathode ray tube) display, etc. The processor 710 is configured to execute code and data from memory 712 to visually depict 722 and manage a set of user preferences for applications as described below.

The input mechanism 714 may be any of various types, as appropriate to a particular device. For example, the input mechanism may be a keypad, mouse, trackball, touch pen, microphone, modem, infrared receiver, etc.

As used herein, a small footprint device is a hardware device comprising computing resources such as a processor and a system memory, but having significantly greater constraints on one or more of these resources than a typical desktop computer has. For example, a typical small footprint device may have two megabytes of memory or less, whereas a typical desktop system may have 64 megabytes or more. Also a typical small footprint device may have significantly less processing power than a typical desktop computing system, either in terms of processor type, or processor speed, or both. For example, a personal data assistant device may have a 16 MHz processor, whereas a typical desktop system may have a processor speed of 100 MHz or higher. Also, a typical small footprint device may have a display size significantly smaller than the display screen of a desktop computing system. For example, the display screen of a handheld computer is typically small compared to the display screen of a desktop monitor. It is noted that the specific numbers given are exemplary only and are used for comparison purposes.

Small footprint devices may also have constraints on other resource types compared to typical desktop computing systems, besides the memory, processor, and display size resources described above. For example, a typical small footprint device may not have a hard disk, may not have a network connection, or may have an intermittent network connection, or may have a wireless network connection, etc.

Many small footprint devices are portable and/or are small compared to desktop computers, but are not necessarily so. Also, many small footprint devices are primarily or exclusively battery-operated. Also, small footprint devices may typically have a more limited or narrow range of usage possibilities than a typical desktop computing system. Thus, in various embodiments, the small footprint device illustrated in Figure 3 is illustrative of, but is not limited to, any one of the following: handheld computers, wearable devices (e.g., wristwatch computers), personal data assistants (PDAs), "smart" cellular telephones, set-top boxes, game consoles, global positioning system (GPS) units, electronic textbook devices, etc. Since new classes of consumer devices are rapidly emerging, it is not possible to provide an exhaustive list of small footprint devices. However, the term "small footprint device" is intended to include such devices as may reasonably be included within the spirit and scope of the term as described above.

Figure 3 is merely illustrative of some of the main functional components of the system 720 and is not meant to be exhaustive or to require any specific architecture. Note also that while some embodiments may advantageously be employed for a small footprint device, other embodiments may be implemented for other devices such as general purpose computers, workstations, personal computers and other devices.

Figure 4

Figure 4 illustrates one embodiment of a graphical user interface for setting the set of user preferences for an application in a small footprint device or other devices. As those skilled in the art may be familiar with acronyms like browser, URL, HTML, cookies, etc., it may be beneficial to describe them briefly.

A browser is any application that can display a markup language document. On example of a browser is a web browser which is a graphical user interface to the World Wide Web. It interprets hypertext links and lets a user view sites and navigate from one Internet node to another. Several companies develop and produce web browsers.

A URL (Uniform Resource Locator) is the distinct address or location of a markup language document or other resources to be accessed by a browser. A URL may specify a local file location on the same device on which the browser is running, or a World Wide Web address, for example. For web addresses, the first part of the address indicates what protocol to use (e.g. http:/), and the second part specifies the domain name where the resource is located (e.g. /www.sun.com). As noted above, a URL may also specify a file location on a device running an application like the browser.

On example of a markup language is the Hypertext Markup Language (HTML) which is the authoring language (or programming language) often used to create documents on the World Wide Web. HTML defines the page structure, fonts, graphic elements and hypertext links to other documents on the Web. When a user point the Browser to a URL, the browser interprets the HTML commands embedded in the page and uses them to format the page's text and graphic elements. Extensible Markup Language (XML) and Standard Generalized Markup Language (SGML) are other examples of markup languages.

Cookies are identifiers placed on a user's computer system by a web site that the users may have visited. Web sites may use cookies to identify and track a users movement through a web site. Cookies are an example of a user configurable parameter for the browser. A user may set preferences in a browser to disable cookies or warn before accepting a cookie.

One embodiment of the present invention may utilize browser technology to display the screen to be used in setting the user preferences for the browser or other applications in a small footprint device. Since the preferences GUI uses the already existing browser resources, this results in the application program requiring less code and hence less memory, an important consideration for small footprint device. The browser technology may include the use of a markup language, one embodiment of which may be the HTML mark up language. The graphical user interface may visually depict user preferences selection information. The information that may represent the set of user preferences may include one or more user configurable parameters. The user may assign one or more values to the one or more user configurable parameters. The specific number of user configurable parameters included in the set of user preferences may be dependent of the application. The process of changing initial, existing or default values for the user configurable parameters of the set of user preferences for an application may be referred to as configuring or setting or editing or specifying user preferences.

In one embodiment, the user may set preferences for the browser or other application in a small footprint device by clicking in a checkbox, using an input mechanism 714 in Figure 3, to select a corresponding user configurable parameter. Examples of user configurable parameters for a browser application may include, but not be limited to, defining a home page URL, acceptance of cookies, loading of all images, enabling Java VM, etc. On completion of the process to set user preferences, the user may select the OK or the Cancel button, using an input mechanism 714 in Figure 3, to initiate further processing.

Figure 5

Figure 5 illustrates one embodiment of a markup language source code to generate the graphical user interface to set user preferences for an application in a small footprint device or other device. One embodiment utilizes browser technology to display the screen used in setting or editing the user preferences for the browser in a small footprint device. The browser technology includes the use of a markup language, one embodiment of which may be the HTML mark up language. The markup language document HTML code, as shown in Figure 5, when executed generates a graphical user interface for visually depicting user preferences selection information shown in Figure 4. In one embodiment, the markup language document HTML code, as shown in Figure 5, is an integral part of the application program 718 in Figure 3.

In other embodiments, the browser may use other programming languages such as XML, Java Virtual Machine (JVM™), JavaScript™, etc. to generate a similar web display used in setting the user preferences.

Figure 6

Figure 6 illustrates one embodiment of a process for managing the set of user preferences for an application, running in a small footprint device or other device. In step 600, the user or the application provides a location, e.g. a URL, to a markup language display capable program to launch the initial display for specifying the set of user preferences for the application program. In another embodiment, a first application program, such as a calendar manager, provides a location, e.g. a URL, to a second application program to launch the initial display for specifying the set of user preferences for the first application program. In one embodiment, the display application program is a browser. The first application and the display application may be part of the same application or separate applications. As those skilled in the art may appreciate, the URL may be provided to the browser in one or more ways.

In one embodiment, a dedicated button or a key from the input mechanism 714 in Figure 1 may be defined to provide the URL to the web application for loading the user preference settings display. In another embodiment, a user may select an application menu to launch the display for the user preferences settings. In a third embodiment, the user may be provided with a URL link in an application help display to launch the display for specifying the set of user preferences. The URL may indicate a document location internal to the small footprint device or it may point to an external location e.g. on a server on the Internet.

In step 602, the browser loads and displays the markup language document in response to the user or application provided URL. In one embodiment, the markup language document may be authored in HTML language, as shown in Figure 5. In step 604, the user may modify default or pre-existing settings for the application preferences by changing the values of one or more user configurable parameters for the application. Examples of user configurable parameters for a browser application may include, but not be limited to, defining a home page

URL, acceptance of cookies, loading of all images, enabling Java VM, etc. Additional examples of information representing user configurable parameters included in a set of user preferences for a browser may be found in Figures 1 and 2. Examples of user configurable parameters for other applications such a calendar manager may include date format (DD/MM/YY, MM/DD/YY, etc.), display format (weekly, monthly, quarterly, etc.), daily schedule format (start time, end time, etc.) and similar other.

The graphical user interface visually depicts user configurable parameters included in the set of user preferences, e.g. as shown in Figure 4. The user may assign one or more values to the one or user configurable parameters. The specific number of user configurable parameters included in the set of user preferences may be dependent on the application. The process of changing initial, existing or default values for the user configurable parameters of the set of user preferences for an application, may be referred to as configuration or setting or editing or specifying user preferences. In one embodiment, the user may set preferences for an application in a small footprint device or other device by clicking in a checkbox to select a corresponding user configurable parameter. In another embodiment, the user may enter a home URL for a browser using the input mechanism 714 in Figure 1. On completion of the process to set user preferences, the user may select the OK or the Cancel button to initiate further processing.

In step 606, in one embodiment, with the user action of selecting the OK button, the HTML language document, shown in Figure 5, submits the edited set of user preferences to a Preferences URL handler. The browser application sends the user preference information to a preferences URL handler associated with the application. In one embodiment, the Preferences URL handler, which utilizes object oriented technology including an object manager, performs processing of the data for the set of user preferences. In another embodiment, the Preferences URL handler may utilize traditional programming languages and databases to process the data for the set of user preferences. In one embodiment, submitting the user preference information includes submitting a list of name/value pairs representing the user preference information to the Preferences URL handler. In one embodiment, the list of name/value pairs includes information specified in the markup language document, e.g. HTML. The Preferences URL handler, e.g. located at prefs://usersettings, is accessed to process the edited data for the set of user preferences submitted by the user.

In one embodiment, the initiation of the Preferences URL handler software is performed by the <FORM METHOD=GET ACTION="prefs://usersettings">
HTML command shown in Figure 5. Sending the user preference information to the preferences URL handler may include performing an HTTP GET operation. In one embodiment, the Preferences URL handler uses 'prefs:/' protocol to process Preference class objects, using an object manager, and uses '/usersettings' to get the name of the class object to be loaded from the URL, i.e. usersettings in Figure 5. If the user selects the Cancel button then the browser returns to the previous web display prior to user providing user preference URL to browser.

In step 608, the processing of data for the set of user preferences is initiated by the Preferences URL handler. In one embodiment, Preferences URL handler parses data received from the markup language document authored in HTML language, shown in Figure 5. The parsed data may be stored in memory in a set of name/value pairs. In one embodiment, each name/value pair in the list of name/value pairs includes a method name and an associated parameter value i.e. each name specifies a method of a Preference object class and each value specifies a parameter value for the method. In one embodiment, shown in Figure 5, the name of each name/value pair includes information specified by a "NAME" attribute for an HTML form element included in the HTML document and the value of each name/value pair comprises information specified by a corresponding "VALUE" attribute for the

HTML form element. One embodiment of a name/value pair is <name1, value1>, <name2, value 2>, etc. For example, referring to the HTML program in Figure 5, in one embodiment, one name/value pair for the parsed data is <imageLoading, off>. The specified method is 'imageLoading' operating on the 'usersettings' object class. Another example of a name/value pair is <useCookies, off>. The specified method is 'useCookies' operating on the 'usersettings' object class.

In step 610, the Preferences URL handler software uses the parsed data to dynamically load and execute code at run time, using the reflection API. The reflection API represents, or reflects, the classes, interfaces and objects in the Java Virtual Machine (JVM™). In one embodiment, processing the list of name/value pairs includes the application invoking each method specified in the name/value pair list i.e. a specified method and passing the corresponding parameter value to each method. Thus for each name/value pair, the Preferences URL handler software invokes the Preferences method specified by the name i.e. the specified method, operating on a Preferences object, i.e. specified object, passing the corresponding value as a parameter. For example, referring to the HTML program in Figure 5, in one embodiment, the Preferences URL handler invokes the 'imageLoading' method i.e. the specified method, which operates on the 'usersettings' object class i.e. the specified object. The argument sent to the method is 'off'. As another example, the Preferences URL handler, using an object manager, invokes the 'useCookies' method that operates on the 'usersettings' object class. The argument sent to the method is 'off'.

By invoking the Preferences method, an object manager may check for validity of the specified method and may check for the validity of the data provided. The object manager may store new values if the operation was successful, e.g. no errors were found. For example, invoking the 'imageLoading' method operating on a 'usersettings' object class, may result in storing the edited values for user preference settings. If errors were encountered then the invocation process may display one or more error messages. Once the user preferences have been saved they would be used by the application program to modify its run time characteristics when executed.

In accordance with the foregoing description, an efficient graphical user interface may be advantageously utilized to manage a set of user preferences for applications, such as browsers, running on a small footprint device or other device. Furthermore, since additional program code to display and manage a set of user preferences does not need to be a part of the application program, memory requirements for the application program may also be reduced.

It is noted that, various embodiments further include receiving or storing instructions and/or data implemented in accordance with the foregoing description upon a carrier medium. Suitable carrier media include memory media or storage media such as magnetic or optical media, e.g., disk or CD-ROM, as well as signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as networks and/or a wireless link.

Although the system and method of the present invention have been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the spirit and scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method for managing a set of user preferences for a computer program application, the method comprising:

a browser receiving a location corresponding to a markup language document, wherein the markup language document specifies a graphical user interface which visually depicts user preferences selection information;

the browser displaying the user preferences selection information in response to said browser receiving said location;

receiving data for said set of user preferences for the computer program application;

processing said data for the set of user preferences, wherein completion of the processing of said data for the set of user preferences specifies a processed set of user preferences; and

the computer program application using said processed set of user preferences to modify its operations.

2. The method of claim 1, wherein said computer program application is a web browser and includes said browser.

3. The method of claim 1, wherein said computer program application and said browser are executed on a handheld computer, a wristwatch computer, a personal data assistant (PDA), a cellular telephone, a set-top box, a video game console, a global positioning system (GPS) unit or an electronic textbook device.

4. The method of claim 1, wherein said user preferences selection information comprises one or more user preference parameters included in the set of user preferences, wherein one or more user preference parameters are configurable by a user by assigning one or more values to the corresponding user preference parameters.

5. The method of claim 1, wherein said markup language is HTML.

6. The method of claim 1,wherein said receiving comprises submitting the data for said set of user preferences to a preferences handler associated with the application and the preferences handler receiving said data.

7. The method of claim 1, wherein said processing comprises:

parsing said data;

invoking a specified method to operate on a specified object oriented programming object using said data; and

storing the set of user preferences for the computer program application in response to invoking said specified method.

8. The method of claim 7, wherein said parsing comprises:

reading said data for the set of user preferences for the computer program application in response to receiving said data, wherein said data includes one or more name/value pairs representing said set of user preferences for the application; and selecting a name from each said name/value pair to specify the specified method to operate on the specified object oriented programming object.

9. The method of claim 8, wherein the list of name/value pairs comprises information specified in accordance to the markup language document.

10. A small footprint device comprising:

a processor;

a memory operatively coupled to said processor;

a computer program application stored in said memory, and wherein said computer program application is executable to:

receive a location corresponding to a markup language document, wherein the markup language document specifies a graphical user interface which visually depicts user preferences selection information;

display the user preferences selection information in response to receiving said location;

receive data for said set of user preferences an application; and

process said data for the set of user preferences, wherein the processing of said data for the set of user preferences specifies a processed set of user preferences to be used by the application to modify its operations.

11. A carrier medium for computer program instructions, wherein the program instructions are executable by a device to implement a method of:

a browser receiving a location corresponding to a markup language document, wherein the markup language document specifies a graphical user interface which visually depicts user preferences selection information;

the browser displaying the user preferences selection information in response to said browser receiving said location;

receiving data for said set of user preferences for an application;

processing said data for the set of user preferences, wherein the processing of said data for the set of user preferences specifies a processed set of user preferences; and

the application using said processed set of user preferences to modify its operations.

12. The Carrier Medium of claim 11, wherein the application is the browser.

13. The Carrier Medium of claim 11, wherein the application executes on a handheld computer, a wristwatch computer, a personal data assistant (PDA), a cellular telephone, a set-top box, a video game console, a global positioning system (GPS) unit or an electronic textbook device.

14. The Carrier Medium of claim 11, wherein said user preferences selection information comprises one or more user preference parameters included in the set of user preferences, wherein one or more user preference

parameters are configurable by a user by assigning one or more values to the corresponding user preference parameters.

15. The Carrier Medium of claim 11, wherein said markup language is HTML.

16. The Carrier Medium of claim 11, wherein said processing comprises:
parsing said data;
invoking a specified method to operate on a specified object oriented programming object using said data; and
storing the set of user preferences for the application in response to invoking said specified method.

17. A method for managing user preference settings, the method comprising:
an application receiving a preferences uniform resource locator (URL), wherein the preferences URL specifies a markup language document;
in response to said application receiving the preferences URL, the application displaying the markup language document specified by the preferences URL, wherein said displaying the markup language document comprises the application displaying a graphical user interface described in the markup language document;
a user utilizing the graphical user interface to specify user preference information;
the user submitting the user preference information to the application, wherein said submitting the user preference information comprises submitting a list of name/value pairs representing the user preference information to the application;
the application processing the list of name/value pairs representing the user preference information in order to store the user preference settings.

18. The method of claim 17, wherein each name/value pair in the list of name/value pairs comprises a method name and an associated parameter value, and wherein said processing the list of name/value pairs comprises the application invoking each method specified in the name/value pair list and passing the corresponding parameter value to each method.

19. The method of claim 18, wherein said method names comprise names of methods of a preferences class, wherein said application invoking each method specified in the name/value pair list comprises the application invoking methods of the preferences class.

20. The method of claim 17, wherein the list of name/value pairs comprises information specified in the markup language document.

21. The method of claim 20, wherein the markup language document is a hypertext markup language (HTML) document, and wherein the name of each name/value pair comprises information specified by a "NAME" attribute for an HTML form element included in the HTML document and the value of each name/value pair comprises information specified by a corresponding "VALUE" attribute for the HTML form element.

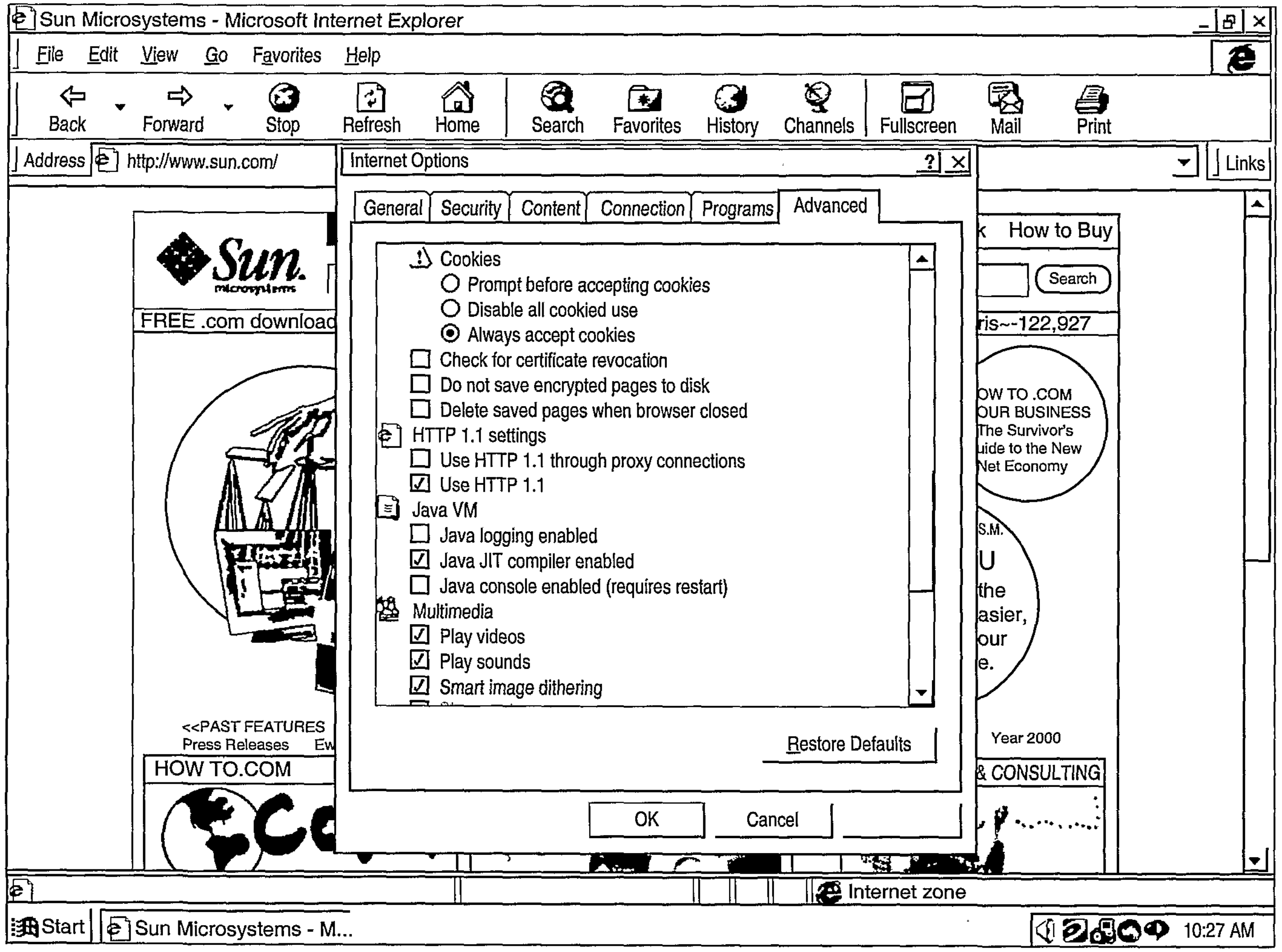
22. The method of claim 17, wherein said user submitting the user preference information to the application comprises the application sending the user preference information to a preferences URL handler associated with the application.

23. The method of claim 22, wherein the preferences URL handler is operable to receive data from a hypertext transfer protocol (HTTP) GET operation, and wherein said sending the user preference information to the preferences URL handler comprises performing an HTTP GET operation.

24. The method of claim 23,wherein the markup language document is a hypertext markup language (HTML) document, and wherein the name of each name/value pair comprises information specified by a "NAME" attribute for an HTML form element included in the HTML document and the value of each name/value pair comprises *information specified by a corresponding "VALUE" attribute for the HTML form element.*

25. The method of claim 22, wherein the markup language document is a hypertext markup language (HTML) document, and wherein the HTML document comprises an HTML form element specifying the preferences URL handler.

26. The method of claim 25, wherein the preferences URL handler is operable to receive data from a hypertext transfer protocol (HTTP) GET operation; and wherein said sending the user preference information to the preferences URL handler comprises performing an HTTP GET operation



Internet Options

General | Security | Content | Connection | Programs | Advanced

Cookies
- ○ Prompt before accepting cookies
- ○ Disable all cookied use
- ● Always accept cookies
- ☐ Check for certificate revocation
- ☐ Do not save encrypted pages to disk
- ☐ Delete saved pages when browser closed

HTTP 1.1 settings
- ☐ Use HTTP 1.1 through proxy connections
- ☑ Use HTTP 1.1

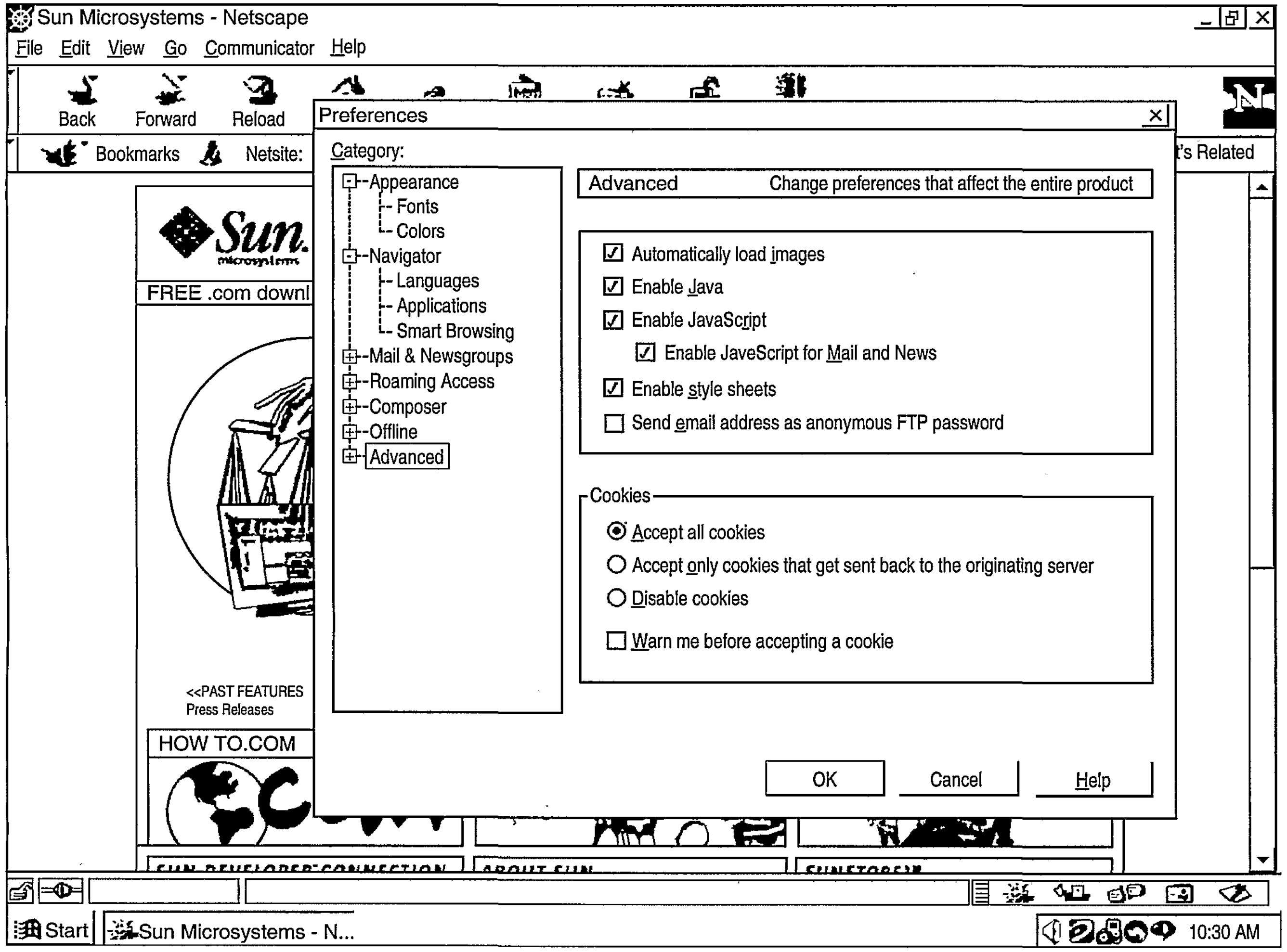
Java VM
- ☐ Java logging enabled
- ☑ Java JIT compiler enabled
- ☐ Java console enabled (requires restart)

Multimedia
- ☑ Play videos
- ☑ Play sounds
- ☑ Smart image dithering

Restore Defaults

OK | Cancel

Figure 1
(PRIOR ART)



Preferences

Category:

- Appearance
  - Fonts
  - Colors
- Navigator
  - Languages
  - Applications
  - Smart Browsing
- Mail & Newsgroups
- Roaming Access
- Composer
- Offline
- Advanced

Advanced Change preferences that affect the entire product

☑ Automatically load images
☑ Enable Java
☑ Enable JavaScript
  ☑ Enable JavaScript for Mail and News
☑ Enable style sheets
☐ Send email address as anonymous FTP password

Cookies
[✓ Accept all cookies]
Accept only cookies that get sent back to the originating server
Disable cookies

☐ Warn me before accepting a cookie

OK Cancel Help

Figure 2
(PRIOR ART)

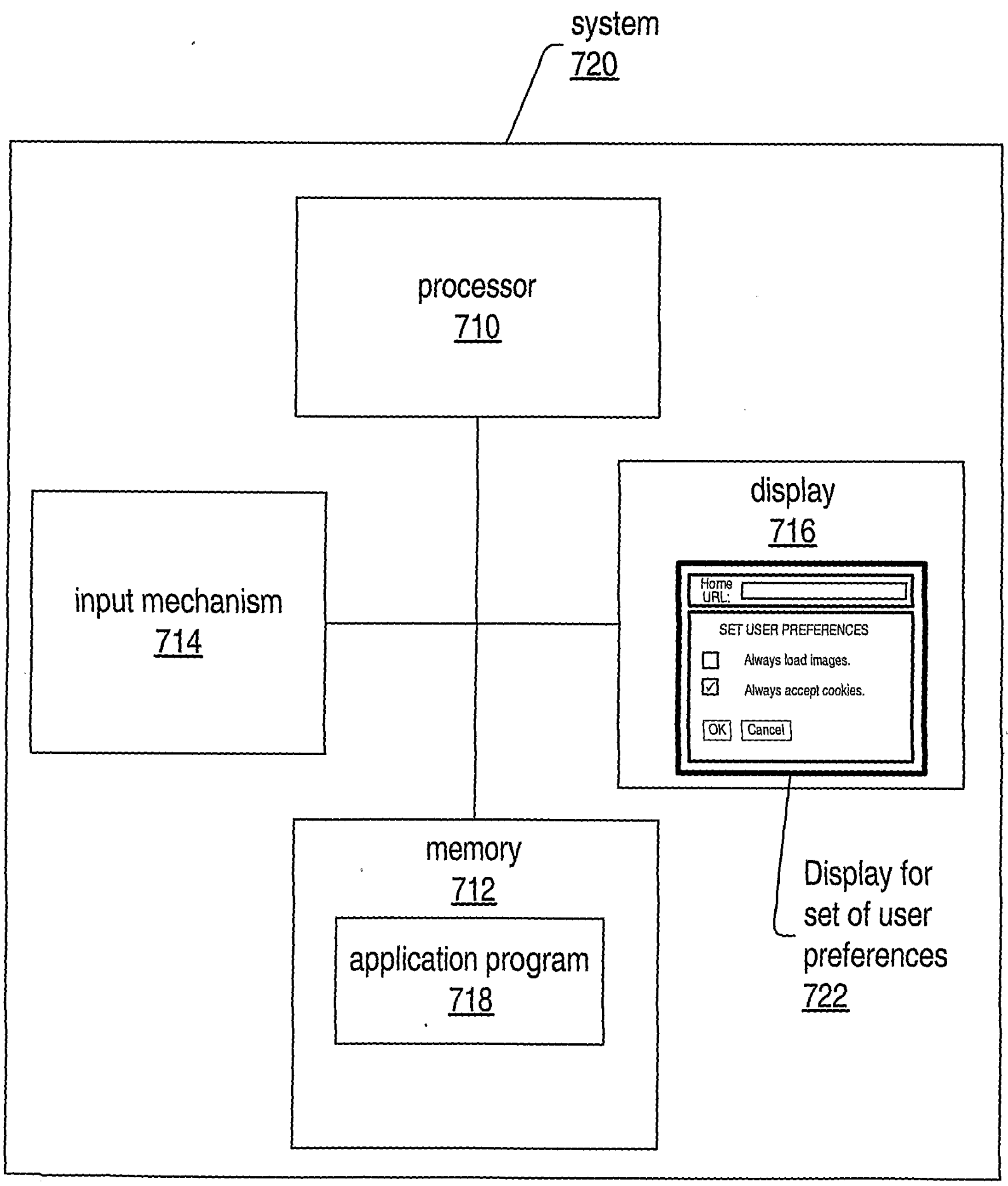

system
720
processor
710
input mechanism
714
display
716
Home URL:
SET USER PREFERENCES
☐ Always load images.
☑ Always accept cookies.
OK
Cancel
memory
712
application program
718
Display for set of user preferences
722

Figure 3

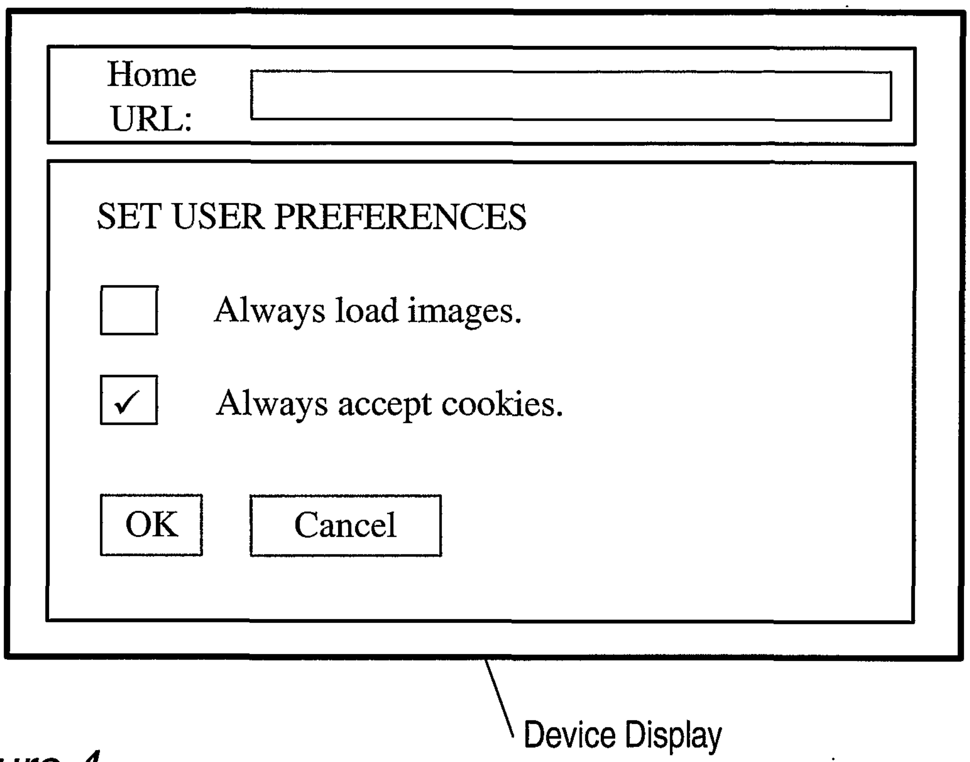


Figure 4

```
<HTML>
<BODY>
<H4 align=left>SET USER PREFERENCES</H4>
<BR>
<FORM METHOD=GET ACTION="prefs://usersettings">

<P><INPUT TYPE=checkbox NAME="imageLoading" VALUE="off">
Always load images</P>

<P><INPUT TYPE=checkbox NAME="useCookies" VALUE="off">
Always accept cookies</p>

<INPUT TYPE=submit VALUE=OK>
<INPUT TYPE=button VALUE=Cancel>

</FORM>
</BODY>
</HTML>
```

Figure 5

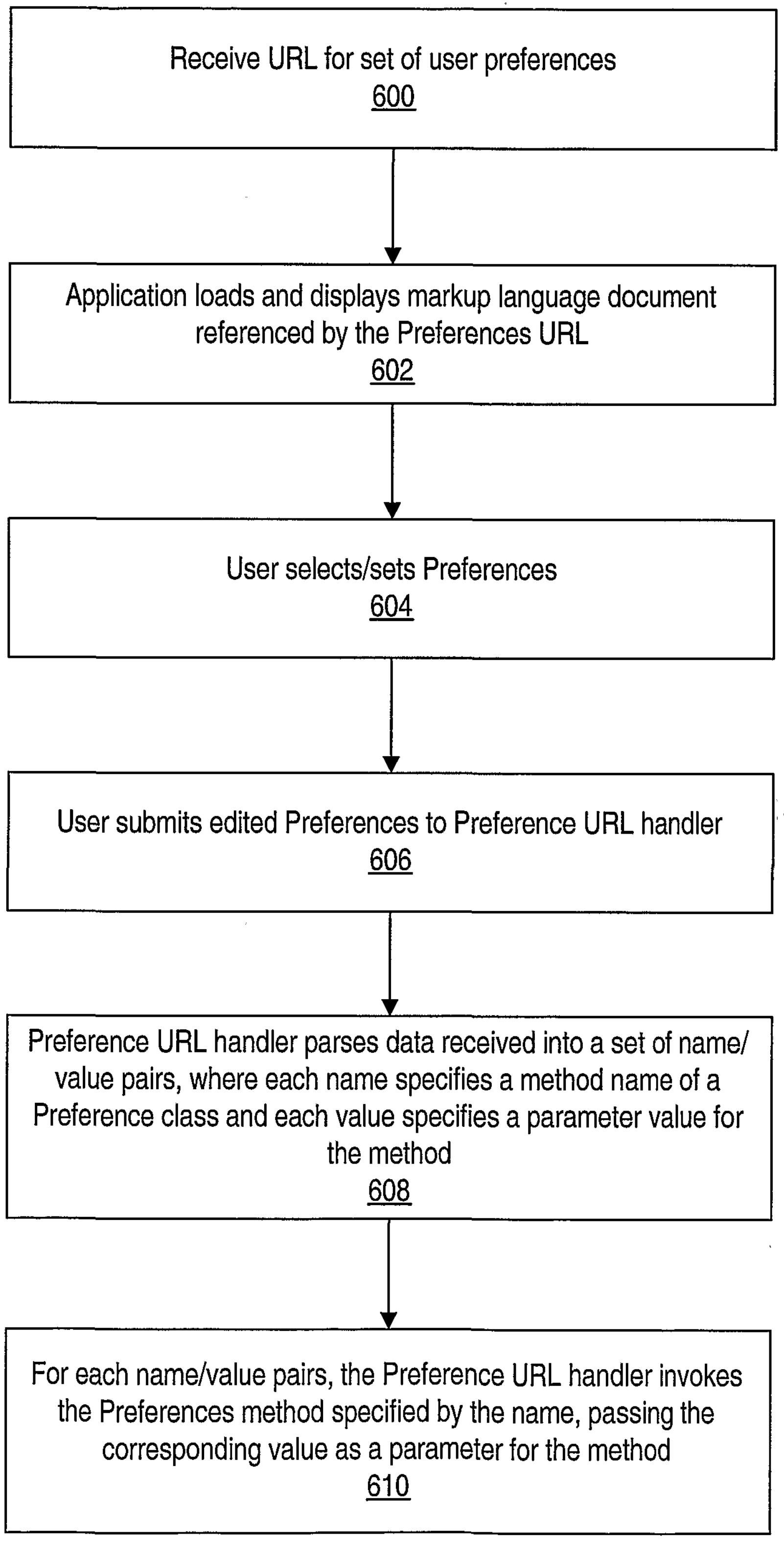

Receive URL for set of user preferences
600
Application loads and displays markup language document referenced by the Preferences URL
602
User selects/sets Preferences
604
User submits edited Preferences to Preference URL handler
606
Preference URL handler parses data received into a set of name/value pairs, where each name specifies a method name of a Preference class and each value specifies a parameter value for the method
608
For each name/value pairs, the Preference URL handler invokes the Preferences method specified by the name, passing the corresponding value as a parameter for the method
610

Figure 6