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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 September 2007 (20.09.2007)

PCT

(10) International Publication Number
WO 2007/106273 A1

(51) International Patent Classification:
G06F 15/163 (2006.01)

(21) International Application Number:
PCT/US2007/003722

(22) International Filing Date:
13 February 2007 (13.02.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/276,536 3 March 2006 (03.03.2006) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **VON KOCH, Walter, V.**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **LYNDERSAY, Sean, O.**; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

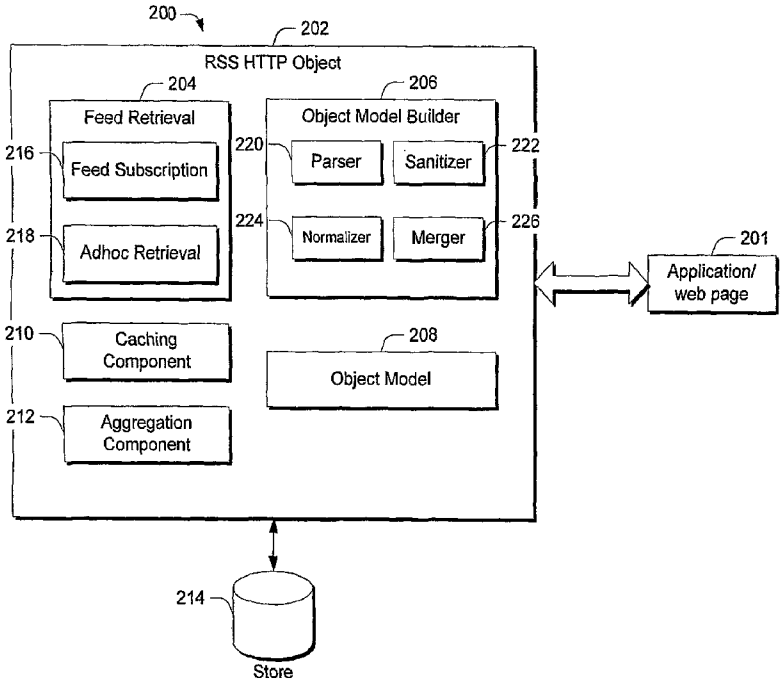
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:
— *with international search report*

[Continued on next page]

(54) Title: RSS DATA-PROCESSING OBJECT



(57) Abstract: Various embodiments utilize a special object referred to as an rsshttp object to acquire an RSS feed, process the feed and expose an object model to a web page or application. In at least some embodiments, the rsshttp object can parse through the feed's associated RSS data, normalize the feed data to a standard format, sanitize the feed data if necessary, and then present a standardized object model for interaction with web pages and applications.

WO 2007/106273 A1

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

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.

RSS DATA-PROCESSING OBJECT

BACKGROUND

RSS, which stands for ***R***eally ***S***imple ***S***yndication, is one type of web content syndication format. RSS web feeds have become more and more popular on the web and numerous software applications with RSS support are being developed. Typically, when web pages and other applications want to consume RSS data, the web pages or applications use some type of script, such as Javascript, to parse through the RSS feed and provide the data. What makes this scenario particularly challenging is that RSS comes in a variety of versions and different formats, e.g. RSS 0.91, 0.92, 1.0, 2.0 and Atom. This makes developing script or code for RSS parsing non-trivial and error prone.

SUMMARY

Various embodiments utilize a special object referred to as an rsshttp object to acquire an RSS feed, process the feed and expose an object model to a web page or application. The rsshttp object can parse through the feed's associated RSS data, normalize the feed data to a standard format, e.g. RSS 2.0, sanitize the feed data if necessary, and then present a standardized object model for interaction with web pages and applications. In at least some embodiments, the rsshttp object can be configured to work on an ad hoc basis, as by fetching and processing feeds when requested by the user, or on a scheduled basis in which feeds are fetched and processed on a scheduled basis. By using the object model, web pages and applications can access and meaningfully use associated feed data without having to understand the intricacies of the different feed formats.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates an exemplary system in which the inventive techniques can be employed in one embodiment.

Fig. 2 illustrates a system in accordance with one embodiment.

Fig. 3 illustrates individual objects of an object model in accordance with one embodiment.

Fig. 4 illustrates individual objects of an object model in accordance with one embodiment.

Figs. 5-7 are flow diagrams that describe steps in methods in accordance with one embodiment.

Fig. 8 illustrates one exemplary specific implementation or system in accordance with one embodiment.

DETAILED DESCRIPTION

Overview

Various embodiments utilize a special object to process RSS data. In this document, this special object is referred to as an rsshttp object and is configured to acquire an RSS feed, process the feed and expose an object model to a web page or application. It is to be appreciated and understood that while this object is referred to as an "rsshttp" object, such should not be construed to limit the object to applications only in connection with http. In one embodiment, the object can be implemented as a COM object.

The rsshttp object can parse through the feed's associated RSS data, normalize the feed data to a standard format, e.g. RSS 2.0, sanitize the feed data if necessary, and then present a standardized object model for interaction with web pages and applications. In at least some embodiments, the rsshttp object can be

configured to work on an ad hoc basis, as by fetching and processing feeds when requested by the user, or on a scheduled basis in which feeds are fetched and processed on a scheduled basis. By using the object model, web pages and applications can access and meaningfully use associated feed data without having to understand the intricacies of the different feed formats, such as the XML that describes the feed data. Of course, web pages and applications that wish to interact with the RSS data outside of the object model can do so if they so choose.

Exemplary System Overview

Fig. 1 illustrates an exemplary system, generally at 100, in which the inventive techniques can be employed in one embodiment. Here, system 100 includes one or more computing devices in the form of a client computing device 102 and one or more servers 104 that communicate with one another via a network such as the Internet. In this example, client computing device comprises one or more processors 106 and one or more computer-readable media 108 on which executable, computer-readable instructions reside. In this example, computer-readable media 108 includes code that implements a web browser 110.

In this example, the web browser 110 is configured to display one or more web pages 112 individual ones of which can support or contain Javascript, DHTML and the like. In addition, in at least some embodiments, browser 110 can comprise or otherwise make use of one or more rsshttp objects 114 and, optionally, one or more xmlhttp objects 116.

In this example, the rsshttp object is utilized to acquire RSS feeds, such as feeds that are required on an adhoc basis or feeds that are subscribed to by a user. In at least some embodiments, the rsshttp object can utilize xmlhttp object 116 as a means to acquire the feeds using known xml/http techniques.

When the rsshttp object acquires an RSS feed from, for example, servers 104, it can operate on the feed to perform a number of functions that permit applications and script to interact with the RSS data, without having to know anything about the underlying RSS format or the XML that describes the RSS data. More specifically, and as will be described in greater detail below, the rsshttp object processes the RSS feed to build an object model that is then exposed to applications and web pages.

As but one specific example of an rsshttp object, consider the section just below.

Exemplary RSSHTTP Object

Fig. 2 illustrates a system, generally at 200, in accordance with one embodiment. Here, system 200 includes one or more applications/web pages 201, one or more rsshttp objects 202 and a store 214.

In one embodiment, rsshttp object 202 includes functionality or components that include a feed retrieval component 204, an object model builder 206, an object model 208, a caching component 210 and an aggregation component 212.

In one embodiment, feed retrieval component 204 includes a feed subscription component 216 and an adhoc retrieval component 218. The feed retrieval component 204 is configured to enable feeds to be acquired and processed by the rsshttp object.

In this particular example, feed subscription component 216 enables a user to subscribe to a particular RSS feed. This can typically be done by a user, web site or application specifying an URL associated with the feed. Once the URL is specified, the subscription component 216 can subscribe to the feed and ensure that the most up-to-date information associated with that feed is available for

consumption by the user. It can do this by regularly checking the RSS feed for new information on a scheduled basis. The schedule can be one that the rsshttp object sets, or one that is negotiated with an entity, such as a server, that provides the RSS feed.

In addition to feed subscription component 216, feed retrieval component 204 also includes, in this example, an adhoc retrieval component 218. In this embodiment, adhoc retrieval component 218 is operable to acquire RSS feeds on an adhoc basis. For example, if a user sees a feed that is of particular interest, they may click on an associated link at which time adhoc retrieval component 218 can take steps to acquire the feed specified by the user.

In one embodiment, object model builder 206 includes a parser component 220, a sanitizer component 222, a normalizer component 224 and a merger component 226.

In this particular example, parser component 220 is configured to parse the XML associated with RSS feeds that are acquired. Any suitable parsing component can be utilized, as will be appreciated by the skilled artisan. When the parser component operates on a feed, it parses through the XML elements identifying the particular elements that comprise the feed. Recall that RSS feeds can have many different formats. Accordingly, the parser is able to identify all of the different elements that comprise the feed. Note also that some of these elements may be elements that have been used to extend a feed's basic schema.

In this particular example, sanitizer component 222 is configured to sanitize the feed of any undesirable features or characteristics that the feed may have. For example, a feed may contain certain active or executable content that is undesirable to have. In this case, sanitizer component 222 sanitizes or removes the active or executable content.

Normalizer component 224 operates on the RSS data to normalize it to a standard or common format. In this example, the common or standard format is RSS 2.0. Accordingly, those other formats that vary from the RSS 2.0 format are operated upon to remove or map elements to the RSS 2.0 elements. As such, having the RSS data in a standardized format leads to predictability in handling and processing.

In this example, merger component 226 processes the RSS data and does such things as register and store new content in data store 214, along with other relevant state data. This can enable the object model to provide or fire events to entities that register for them. For example, an application may register for a notification when new content is received from a particular feed. In this instance, the merger component can look for any such new content so that object 202 can fire an event to the application.

The output of the object model builder 206, in at least some embodiments, is a normalized, sanitized object model 208 that can be exposed to application/web page 201. These entities can interact with the object model in lieu of the XML that defines the RSS feed thus alleviating the entities from having to understand the intricacies of the different RSS feeds. Of course, for those entities that wish to interact directly with the XML, they are free to do so. An exemplary object model is described below under the heading "Object Model".

In one embodiment, caching component 210 is configured to perform caching duties that are designed to reduce the load that is experienced by servers that provide the RSS feeds. More specifically, the caching component can be configured to utilize conditional GET requests so that a request is not made unless it is necessary. For example, when requesting the feed data, the rsshttp object can send a timestamp of the last time it received data for the feed to the server. The

server can then respond with new data or quick respond with "no new data". In addition, the caching component can be configured to ensure that requests are small so that the cache can be quickly updated.

In one embodiment, aggregation component 212 is configured to aggregate content. That is, typically RSS feeds provide only the most recent items. Yet, there are instances when having a complete set of items is desirable. Aggregation component 212 is configured to acquire these different and sometimes dated items and aggregate the items in data store 214 so that an application or web page can access all of the items.

Object Model

Fig. 3 illustrates individual objects of an object model 300 in accordance with one embodiment. The object model about to be described constitutes but one example of an object model that can be utilized and is not intended to limit application of the claimed subject matter to only the object model that is described below. In at least some embodiments, the object model is exposed by an API that is callable by an application or web page.

In this particular object model, a top level object *feeds* 302 is of the type feed. Underneath the feeds object 302 is an item object 304 of the type item, and underneath the item object 304 is an enclosure object 306 of the type object.

The individual objects of the object model have properties, methods and, in some instances, events that can be utilized to manage received web content. The above-described object model permits a hierarchical structure to be utilized to manage and interact with feeds without necessarily having to be knowledgeable of the underlying XML that describes the RSS feed.

Considering the object model further, consider item and enclosure objects 304, 306 respectively. Here, these objects very much reflect how RSS is structured itself. That is, each RSS feed has individual items inside of which can optionally appear an enclosure. Thus, the structure of the object model is configured to reflect the structure of the syndication format.

From an object model perspective, there are basically two different types of methods and properties on an item. A first type of method/property pertains to data which is read only, and a second type of method/property pertains to data which can be both read and written.

As an example of the first type of method property, consider the following. Each feed can have data associated with it that is represented in an XML structure. This data includes such things as the title, author, language and the like. Data such as this is treated by the object model as read only. This prevents applications from manipulating this data.

On the other hand, there is data that is treated as read/write data, such as the name of a particular feed. That is, the user may wish to personalize a particular feed for their particular user interface. In this case, the object model has properties that are read/write. For example, a user may wish to change the name of a feed from "New York Times" to "NYT". In this situation, the name property may be readable and writable. The object model can also be extensible with "expando" properties which allow an application to add data/state to the feed dynamically. One example of this is storing foreign keys along side the rss items for easy matching of items with data in another database/store.

Fig. 4 illustrates a top level object or interface IFeed, along with objects or interfaces IItem and IEnclosure objects, along with their properties and methods in accordance with one embodiment. Other objects, interfaces, methods and

properties can be utilized without departing from the spirit and scope of the claimed subject matter.

Starting first with the IFeed object, consider the following. Many of the properties associated with this object come from the RSS feed itself, e.g., Title, Url, Webmaster, SkipHours, SkipDays, ManagingEditor, Homepage, ImageURL and the like, as will be appreciated by the skilled artisan. In addition, there is another set of properties of interest, i.e. the *Items property* which is a collection that has all of the items that are part of a feed and the *LocalEnclosurePath* property which provides the actual directory to which all of the enclosures are written. Thus, for an application, the latter property makes it very easy to access the enclosures.

In addition, this object supports a small set of methods such as Download() which are used to manage particular feeds. Further, this object supports a method XML(), which returns a feed's XML in a standard format. The XML data can be used for such things as creating a newspaper view of a feed.

Moving to the Item object, this object has a set of properties that represent regular RSS elements, e.g. Description, Url, Title, Author and the like. In addition, there is a Parent property that points back to the associated actual feed, and an Id property so that an application can identify items versus having to iterate over all items. In addition, there is an Enclosures property which is the collection of the item's enclosures of the type IEnclosure. Further, an IsRead property enables an application to indicate whether a particular item has been read.

Moving to the Enclosure object, consider the following. This object has properties that include a Type property (e.g. mp3) and Length property that describes the length of a particular enclosure for example in bytes. There is also the LocalAbsolutePath to a particular enclosure. The Download() method allows individual enclosures to be downloaded and used by applications.

By exposing the object model described above to applications and web pages, the applications and web pages can interact with the objects and hence data of the feed, without having to be knowledgeable of or deal with any of the underlying XML that describes the feed. Of course, applications and web pages that wish to deal with the underlying XML are still free to do so.

Exemplary Methods

Figs. 5-7 are flow diagrams that illustrate steps in a method in accordance with one embodiment. The method can be implemented in connection with any suitable hardware, software, firmware or combination thereof. In at least some embodiments, the methods can be implemented in connection with systems such as those shown and described above and below. It is to be appreciated and understood that systems different from the ones described in this document can be utilized to implement the described methods without departing from the spirit and scope of the claimed subject matter.

In the illustrated methods, the flow diagrams are organized to illustrate which entities can perform the various acts. Accordingly, those acts that can be performed by an application or web page are designated as such. Similarly, those acts that can be performed by an rsshttp object or RSS source (server) are designated as such.

Fig. 5 illustrates an exemplary method for building an object model in accordance with one embodiment.

At step 500, an application or web page makes a request on the rsshttp object for an RSS feed. This request can be made via a call to a suitably exposed application program interface and can constitute any suitable type of request, such as an adhoc request, subscription request and the like.

The rsshttp object receives the request at 502 and makes a corresponding request, at 504, on an associated RSS source such as a server. The server receives the request at 506 and provides or sends RSS feed data to the rsshttp object at 508.

The rsshttp object then normalizes, sanitizes and merges the data into the store if it's not an ad-hoc feed.

The rsshttp object receives the RSS feed data at 510 and builds an object model at 512. Examples of how an object model as well as a specific instance of an object model are provided above. Once the object model is built, the rsshttp object exposes the object model to the application/web page at 514.

Once the object model is exposed, an application/web page can interact with and consume the RSS data. This can be done by making calls to various methods exposed by the object model. In this manner, an application or web page does not need to be concerned with the underlying XML that describes the feed. This is because the XML was abstracted away when the object model was built. In addition to the object model, in at least some embodiments, various error messages can be generated for the application or web page in the event of an error. Errors can include, for example, failed download, failed to normalize, failed to sanitize, invalid feed format and the like. Further, various statuses can be updated or saved in a suitable data store. For example, statuses associated with items such as "not updated", "new", "updated" and "removed" can be recorded.

Fig. 6 illustrates an exemplary method for making scheduling requests in accordance with one embodiment.

At step 600, an application or web page makes a scheduling request on the rsshttp object. This request can be made via a call to a suitably exposed application program interface. In this example, an application or web page may wish to periodically receive feed updates. Thus, according to a schedule provided by the

application or web page, or negotiated with the server, regular checks can be made. In at least some embodiments, the application or web page does not make the request every time. It simply sets up the request/schedule initially, then the rsshttp object will automatically, in the background, make the requests.

The rsshttp object thus receives the request at 602 and makes a corresponding request, at 604, on an associated RSS source such as a server, according to the schedule. The server receives the request at 606 and provides or sends RSS feed data to the rsshttp object at 608 if there is any data that meets the request.

The rsshttp object receives the RSS feed data at 610 and makes the RSS feed data available at 612. This step can be performed in a number of ways. For example, when new feed data is received responsive to the request, an event can be fired and the user can be notified. Notification can take place in any suitable way. For example, a user interface element in the user's browser may be activated to indicate that a new item has been received. Alternately or additionally, a notification can be sent to the user, such as an email or instant message, popup window or application/web page UI can update itself with the new item.

Fig. 7 illustrates an exemplary method for registering for events in accordance with one embodiment.

At step 700, an application or web page registers for an event with the rsshttp object. Any suitable event or type of event can be the subject of registration. For example, an application or web page may be interested in receiving notifications when new feed items are added or feed items are changed, deleted or read.

The rsshttp object receives the registration request at 702 and listens for the particular event at 704. This step can be implemented in any suitable way. For

example, the rsshttp object may poll the RSS source or server at regular intervals to ascertain whether a particular event has occurred. Alternately or additionally, an RSS source or server may notify subscribers when certain events occur.

If an event occurs at step 706, the rsshttp object notifies the application or web page at step 708. Any suitable notification can be provided examples of which are described above. If an event does not occur, then step 706 returns to step 704 and listens for the event of interest.

Exemplary Implementation

Fig. 8 illustrates one exemplary specific implementation or system in accordance with one embodiment generally at 800. It is to be appreciated and understood that the example about to be described constitutes but one example of how one can implement the above-described functionality. As such, other different implementations can be utilized without departing from the spirit and scope of the claimed subject matter.

In this example, a web page/user interface 802 allows a user to interact with the system. This interface can be provided by any suitable application. For example, in some embodiments, a browser interface can be used. System 800 also includes an rsshttp object 804, an optional xmlhttp object 806, an optional RSS platform component 808, a WinINET component 810 and a store database 812.

Here, rsshttp object 804 can use xmlhttp object 806 to access XML data in a manner which will be understood by the skilled artisan. Additionally, in this particular implementation, these objects can leverage an RSS platform 808 to acquire RSS feed data. An exemplary platform is described in U.S. Patent Publication No. US-2007-0011665-A1, published on January 11, 2007.

WinINET component 810 is utilized to make the network requests, as will be appreciated by the skilled artisan, and store database 812 is used to store all of the individual items and state information.

A typical operation utilizing this implementation example will occur as follows. In some instances, a piece of jscript code executing as part of an application will instantiate rsshttp object 804. The jscript code can then make a request on the rsshttp object using an URL associated with a particular RSS feed. The rsshttp object 804 can then make a request on the server. If xmlhttp object 806 and/or RSS platform are present, the rsshttp object can leverage these components to acquire the RSS feed data.

The server then gives back the RSS feed data to the rsshttp object 804. Now, the rsshttp object can do things such as sanitize the data, normalize the data, and merge the data in store database 812. This can include, by way of example and not limitation, updating state information associated with the RSS data. Once this is done, the rsshttp object can build an object model and expose the object model to the application or web page. In this example, store database 812 stores not only the state of individual feed items, but various subscription lists as well. This allows the rsshttp object to keep feed items fresh even when an application or web page 802 is not loaded.

For example, a web application can, per domain (url domain), subscribe to N number of feeds. This allows a web application to always have up-to-date data when it is launched.

Security

In at least some embodiments, the rsshttp object enforces a per domain RSS feed security model which means that a web page from a specific domain can only

access a subset of the user's feed subscription for which it has received permission from the user. This makes it possible, for example, for a user to allow access to his family picture feed subscription to a new slideshow web page that displays images in a new and engaging way, while at the same time limit the access of this page to a feed of the user's recent credit card transactions.

Conclusion

Various embodiments utilize a special object referred to as an rsshttp object to acquire an RSS feed, process the feed and expose an object model to a web page or application. The rsshttp object can parse through the feed's associated RSS data, normalize the feed data to a standard format, e.g. RSS 2.0, sanitize the feed data if necessary, and then present a standardized object model for interaction with web pages and applications. In at least some embodiments, the rsshttp object can be configured to work on an ad hoc basis, as by fetching and processing feeds when requested by the user, or on a scheduled basis in which feeds are fetched and processed on a scheduled basis. By using the object model, web pages and applications can access and meaningfully use associated feed data without having to understand the intricacies of the different feed formats.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

CLAIMS

1. A system comprising:

one or more computer-readable media;

computer-readable instructions on the one or more computer-readable media which, when executed, implement an object configured to:

acquire (510) an RSS feed;

process the RSS feed to provide an object model (512) associated with the feed;

expose (514) the object model to entities so that such entities can interact with associated feed data without having to understand XML that describes the feed data.

2. The system of claim 1, wherein the object is configured to fetch feeds on an ad hoc basis.

3. The system of claim 1, wherein the object is configured to fetch feeds on a scheduled basis.

4. The system of claim 1, wherein the object is configured to enable a user to subscribe to an RSS feed.

5. The system of claim 1, wherein the object is configured to parse XML associated with RSS feeds that are acquired by the object and normalize the feed's format to a standard format, wherein the object model represents the standard format.

6. The system of claim 1, wherein the object is configured to cache RSS feed data.

7. The system of claim 1, wherein the object is configured to aggregate RSS feed data.

8. The system of claim 1, wherein the object model comprises:

a feeds object associated with a particular RSS feed;

an items object associated with particular items of a feed; and

an enclosure object associated with particular enclosures of a feed, wherein individual object of the object model have associated methods and properties.

9. The system of claim 1, wherein said object comprises part of a browser.

10. A system comprising:

one or more computer-readable media;

computer-readable instructions on the one or more computer-readable media which, when executed, implement a web browser configured to:

receive (502) a request for an RSS feed;

make (504) a corresponding request on an associated RSS source;

receive (510) RSS feed data associated with the request;

parse XML associated with the RSS feed data and build (512) a normalized object model, wherein the normalized object model comprises individual objects at least some of which have callable methods and properties associated with the RSS feed data; and

expose (514) the object model to applications or web pages.

11. The system of claim 10, wherein the object model comprises:

a feeds object associated with a particular RSS feed;

an items object associated with particular items of a feed; and

an enclosure object associated with particular enclosures of a feed.

12. The system of claim 10, wherein the web browser is configured to receive a scheduling request for an RSS feed or item and make requests on an RSS source according to an associated schedule.

13. The system of claim 10, wherein the web browser is configured to receive registration requests and listen for associated events.

14. The system of claim 10, wherein the web browser is configured to receive registration requests and listen for associated events, and wherein the web browser is configured to generate notifications associated with events that occur.

15. A computer-implemented method comprising:

receiving (502) a request for an RSS feed;

making (504) a corresponding request on an associated RSS source;

receiving (510) RSS feed data associated with the request;

parsing XML associated with the RSS feed data;

building (512) a normalized object model, wherein the normalized object model comprises individual objects at least some of which have callable methods and properties associated with the RSS feed data; and

exposing (514) the object model to applications or web pages.

16. The method of claim 15, wherein the object model comprises:

a feeds object associated with a particular RSS feed;

an items object associated with particular items of a feed; and

an enclosure object associated with particular enclosures of a feed.

17. The method of claim 15 further comprising:

receiving a scheduling request for an RSS feed or item; and

making requests on an RSS source according to an associated schedule.

18. The method of claim 15 further comprising:

receiving registration requests; and

listening for associated events.

19. The method of claim 15 further comprising:

receiving registration requests;

listening for associated events; and

generating notifications associated with events that occur.

20. The method of claim 15, wherein said acts are performed by a web browser.

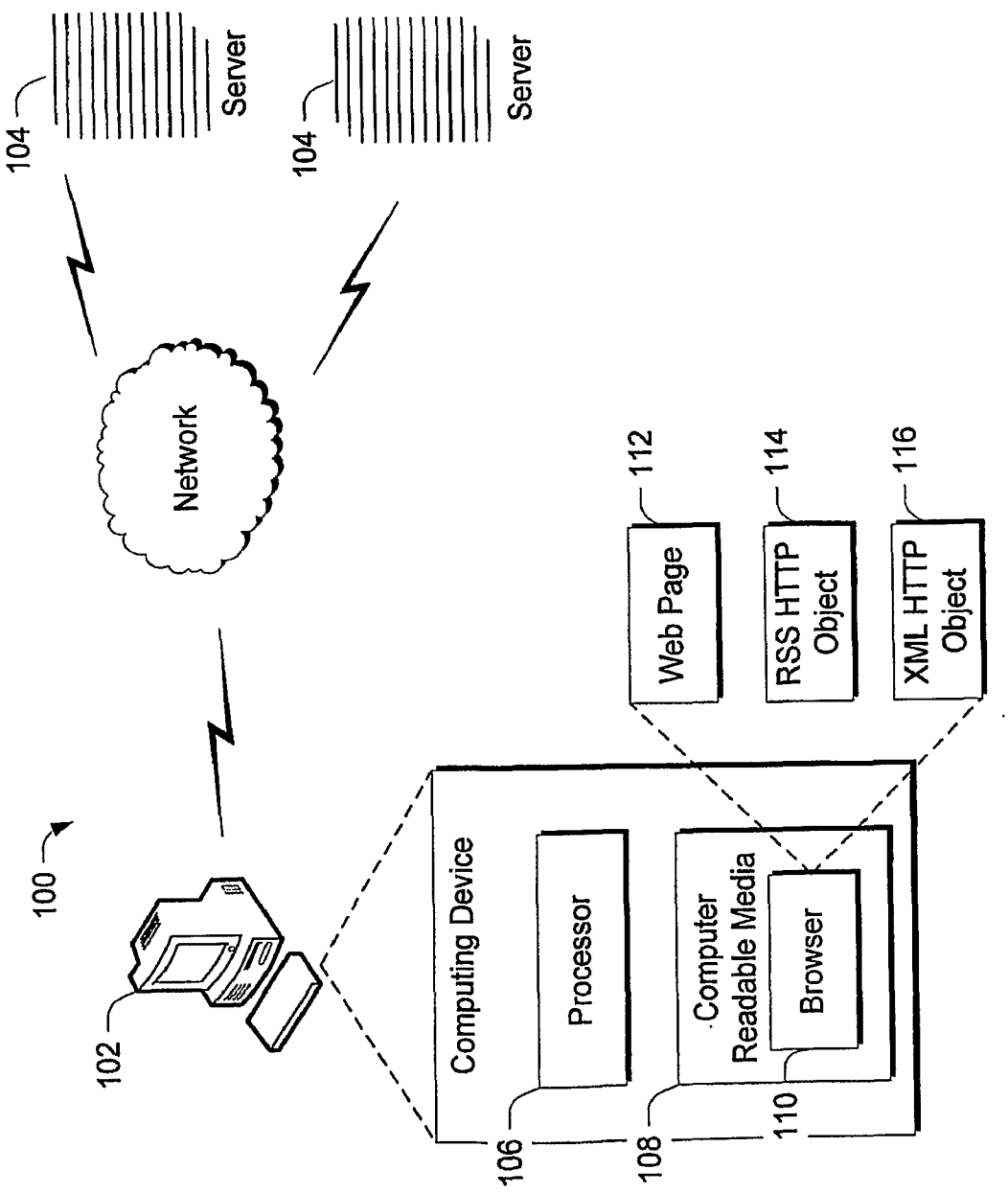

Server
Server
104
104
Network
100
102
Computing Device
Processor
Computer Readable Media
Browser
106
108
110
Web Page
RSS HTTP Object
XML HTTP Object
112
114
116

Fig. 1

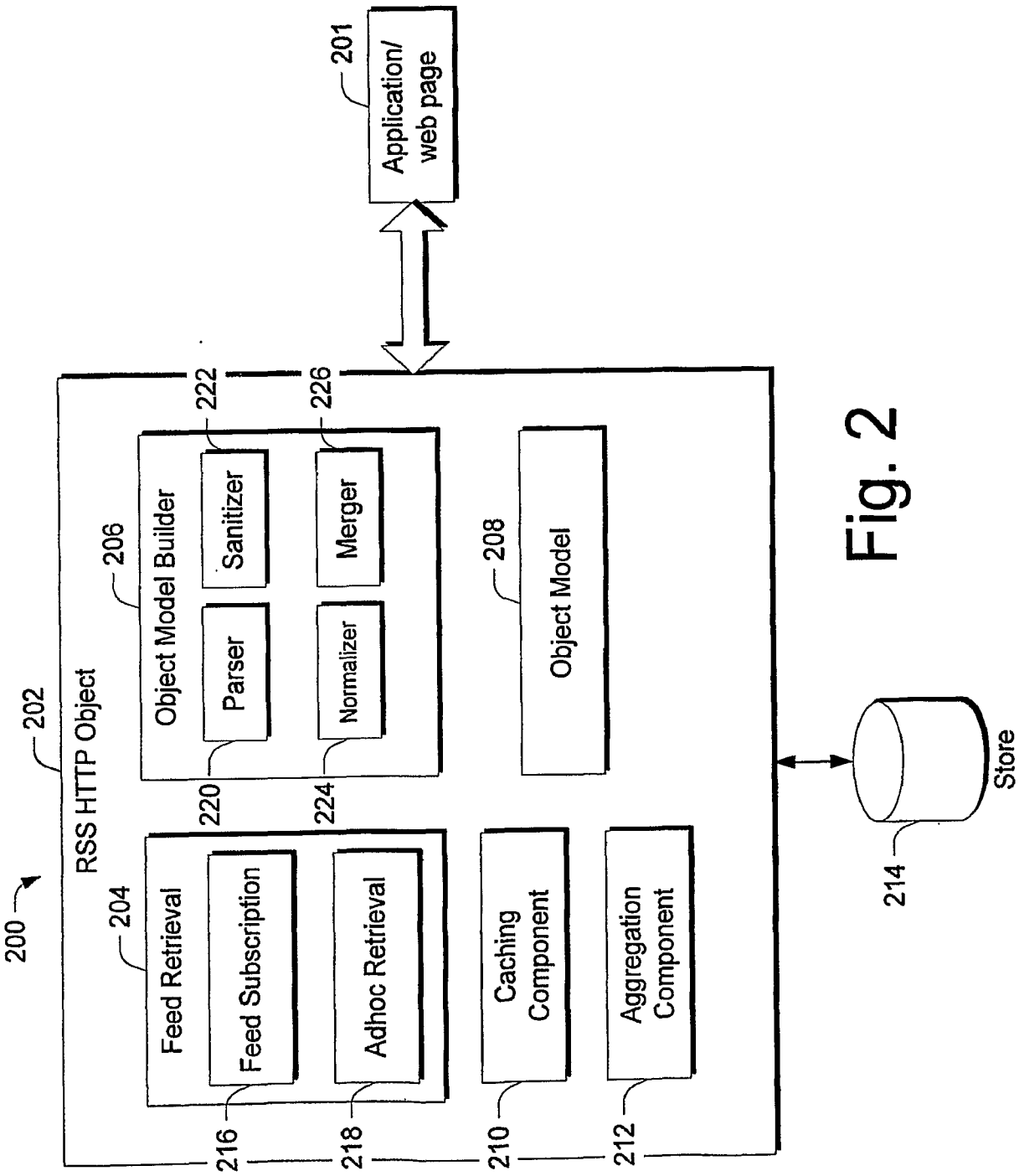

200
202
RSS HTTP Object
204
Feed Retrieval
Feed Subscription
Adhoc Retrieval
206
Object Model Builder
Parser
Normalizer
Sanitizer
Merger
208
Object Model
210
Caching Component
212
Aggregation Component
214
Store
216
218
220
222
224
226
201
Application/ web page

Fig. 2

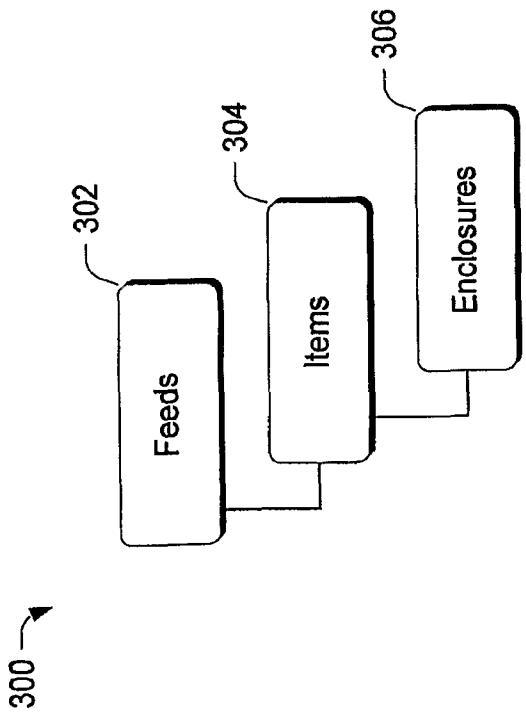


Fig. 3

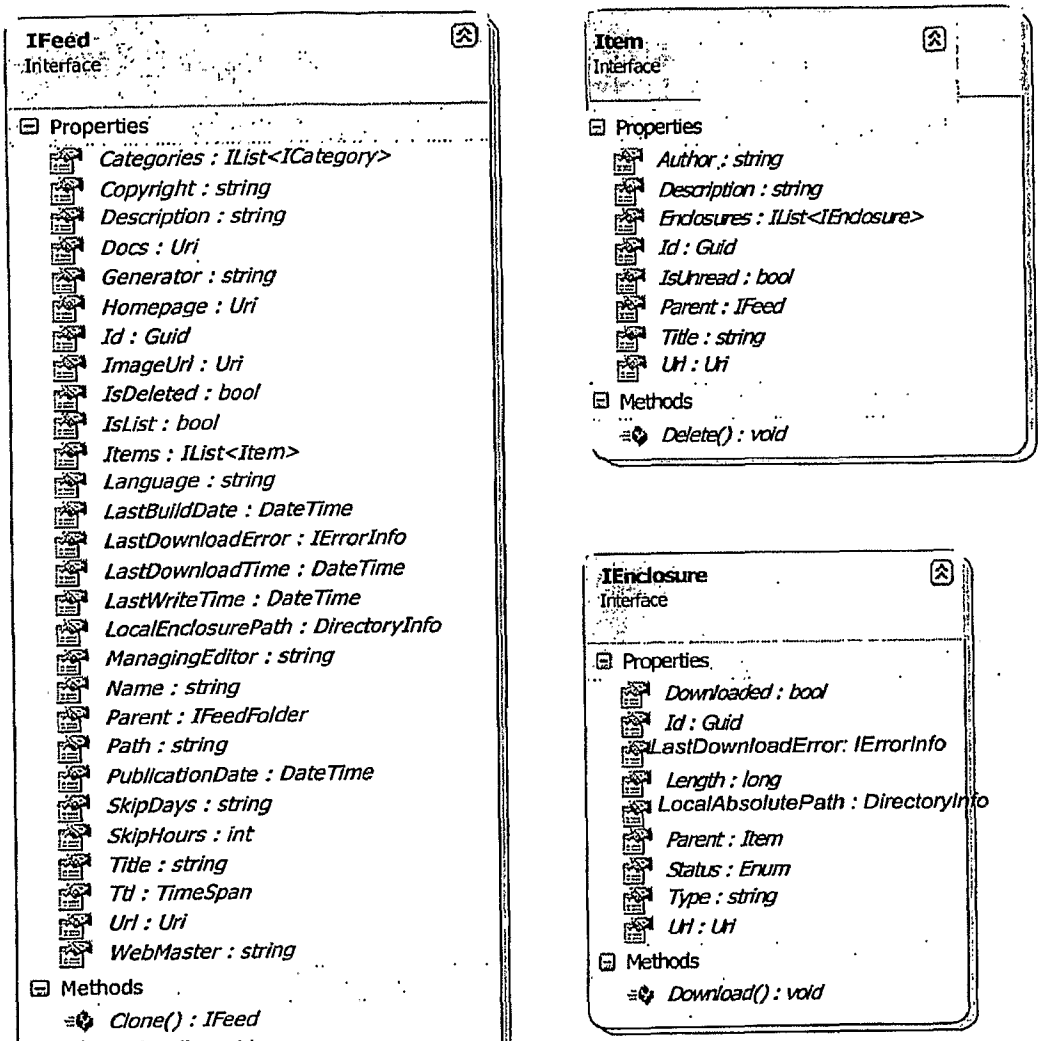

IFeed
Interface
Properties
Categories : IList<ICategory>
Copyright : string
Description : string
Docs : Uri
Generator : string
Homepage : Uri
Id : Guid
ImageUrl : Uri
IsDeleted : bool
IsList : bool
Items : IList<Item>
Language : string
LastBuildDate : DateTime
LastDownloadError : IErrorInfo
LastDownloadTime : DateTime
LastWriteTime : DateTime
LocalEnclosurePath : DirectoryInfo
ManagingEditor : string
Name : string
Parent : IFeedFolder
Path : string
PublicationDate : DateTime
SkipDays : string
SkipHours : int
Title : string
Ttl : TimeSpan
Url : Uri
WebMaster : string
Methods
Clone() : IFeed
Delete() : void
Download() : void
Rename() : void
Xml() : IStream
Events
ItemNotification : EventHandler
Item
Interface
Properties
Author : string
Description : string
Enclosures : IList<IEnclosure>
Id : Guid
IsUnread : bool
Parent : IFeed
Title : string
Url : Uri
Methods
Delete() : void
IEnclosure
Interface
Properties
Downloaded : bool
Id : Guid
LastDownloadError: IErrorInfo
Length : long
LocalAbsolutePath : DirectoryInfo
Parent : Item
Status : Enum
Type : string
Url : Uri
Methods
Download() : void

Fig. 4

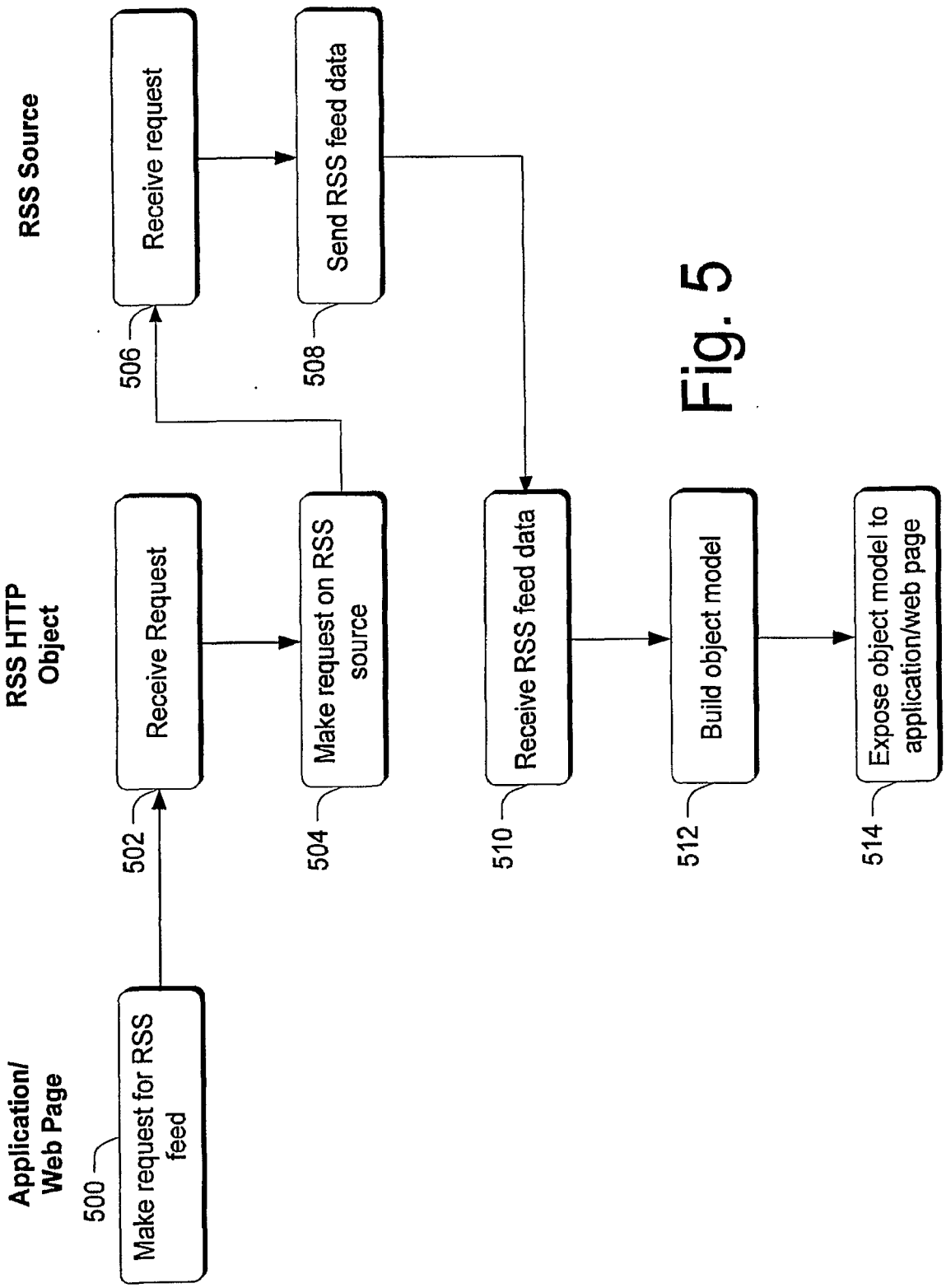

RSS Source
RSS HTTP Object
Application/ Web Page
Receive request
Send RSS feed data
Receive Request
Make request on RSS source
Receive RSS feed data
Build object model
Expose object model to application/web page
Make request for RSS feed
506
508
502
504
510
512
514
500

Fig. 5

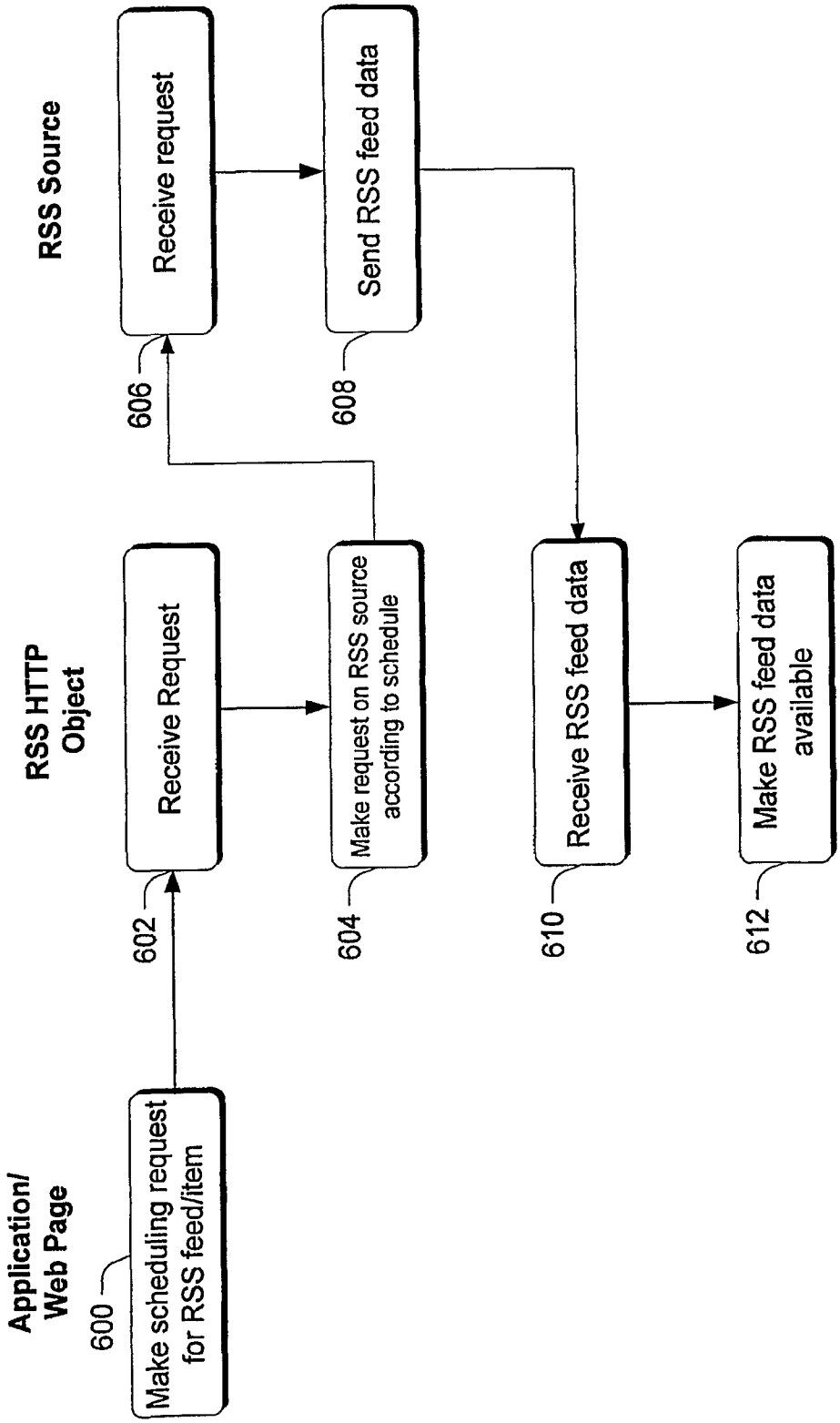

Application/
Web Page
RSS HTTP
Object
RSS Source
Make scheduling request for RSS feed/item
600
Receive Request
602
Make request on RSS source according to schedule
604
Receive request
606
Send RSS feed data
608
Receive RSS feed data
610
Make RSS feed data available
612

Fig. 6

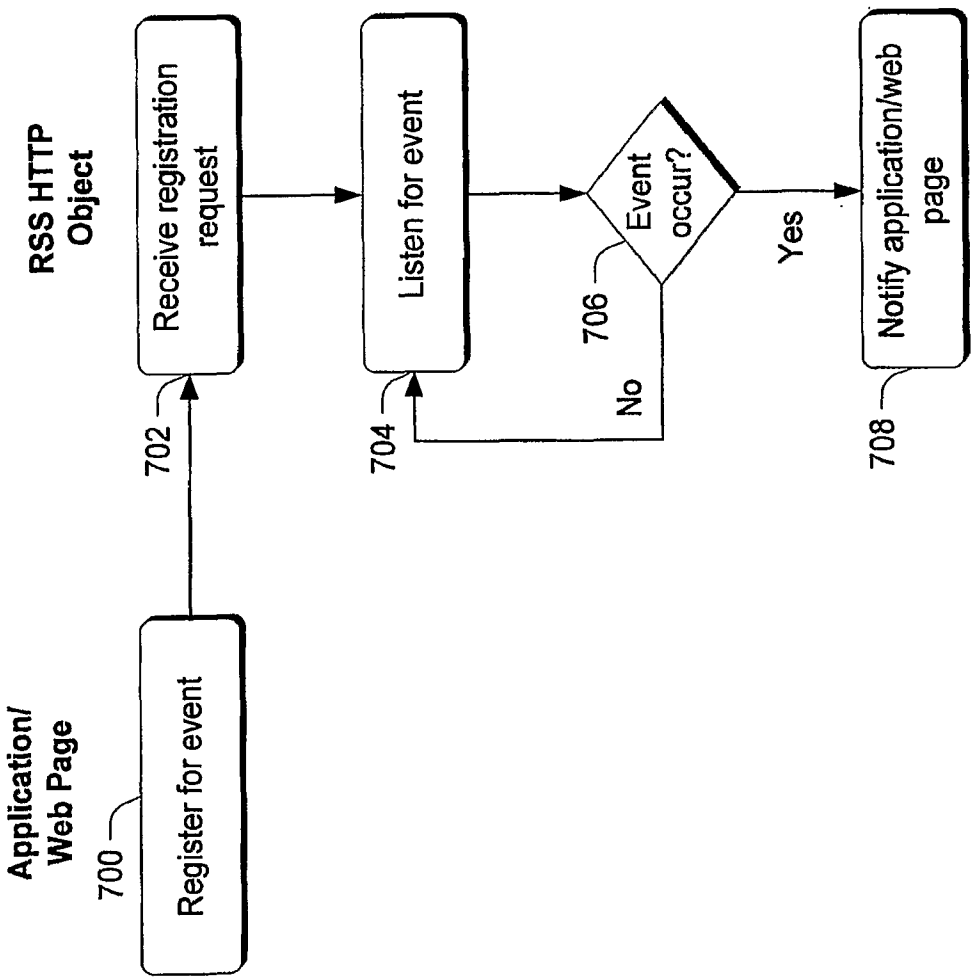

Application/
Web Page
RSS HTTP
Object
Register for event
Receive registration request
Listen for event
Event occur?
Yes
No
Notify application/web page
700
702
704
706
708

Fig. 7

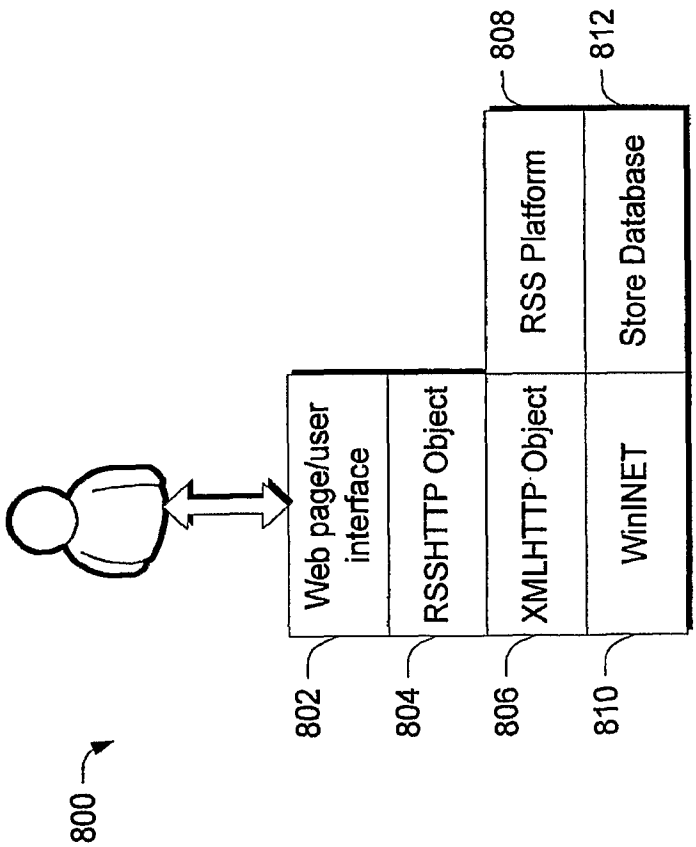

800
802
804
806
810
808
812
Web page/user interface
RSSHTTP Object
XMLHTTP Object
WinINET
RSS Platform
Store Database

Fig. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2007/003722

A. CLASSIFICATION OF SUBJECT MATTER

G06F 15/163(2006.01)i

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)

IPC8 G06F

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

Korean Utility models and applications for Utility models since 1975

Japanese Utility models and applications for Utility models since 1975

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

eKIPASS(KIPO internal) "RSS feed object"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2004/0225749 A1 (GREGORY PAVLIK et al.) 11 November 2004 See abstract & Fig. 2	1-20
A	US 2005/0165615 A1 (NELSON MINAR) 28 July 2005 See abstract	1-20
A	DE SUTTER R. et al. 'Enhancing RSS Feeds: Eliminating Overhead through Binary Encoding' In: Information Technology and Applications, 2005. ICITA 2005. Third International Conference on Volume 1, 4-7 July 2005 Page(s): 520-525 vol.1 See the whole document	1-20

☐ 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 but 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 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	Date of mailing of the international search report
24 JULY 2007 (24.07.2007)	**24 JULY 2007 (24.07.2007)**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office 920 Dunsan-dong, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	YEO, Won Hyeon Telephone No. 82-42-481-5696

Form PCT/ISA/210 (second sheet) (April 2007)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2007/003722

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US2004/0225749A1	11.11.2004	JP2004334866A2	25.11.2004
US2005/0165615A1	28.07.2005	AU2004311794AA	21.07.2005
		BR200418271A	02.05.2007
		CA2552183A1	21.07.2005
		EP01716512A2	02.11.2006
		KR2006130157A	18.12.2006
		WO2005065237A2	21.07.2005