Providing Metadata Services on the World Wide Web

Yann Neveu, Youenn Guervilly,
Uffe K. Wiil, David L. Hicks

Technical Report CSE-01-01

Dept. of Computer Science and Engineering
Aalborg University Esbjerg
Niels Bohrs Vej 8
6700 Esbjerg
Denmark

Address

Dept. of Computer Science and Engineering
Aalborg University Esbjerg
Niels Bohrs Vej 8
6700 Esbjerg
Denmark

Email:	dept@cs.aue.auc.dk
Web:	www.cs.aue.auc.dk
Phone:	+45 7912 7666
Fax:	+45 7545 3643

Table of Contents

1. Introduction

Yann Neveu and Youenn Guervilly, two Socrates exchange students from the University of Bretagne in Brest, France, carried out the work described in this report during a three-month practicum at the Department of Computer Science and Engineering, Aalborg University Esbjerg, Denmark. Uffe K. Wiil and David L. Hicks supervised the project, which was carried out in the context of the Construct research project (www.cs.aue.auc.dk/construct).

The purpose of the project was to provide metadata services on the World Wide Web (WWW). We have extended Netscape 6 with additional functionality and made it an application of the Construct metadata structural service component.

The sidebar interface on Netscape has been designed with the XUL language, specially created in order to create the Netscape 6 user interface. The metadata services interface is located in 'My Sidebar' which is a frame located on the left side of the browser. It can be consulted at any time within all kinds of web pages. This will allow ordinary WWW users to add metadata (data about data) to arbitrary documents on the WWW. The JavaScript language combined with the LiveConnect technology permit communication between the XUL interface and the Java applet.

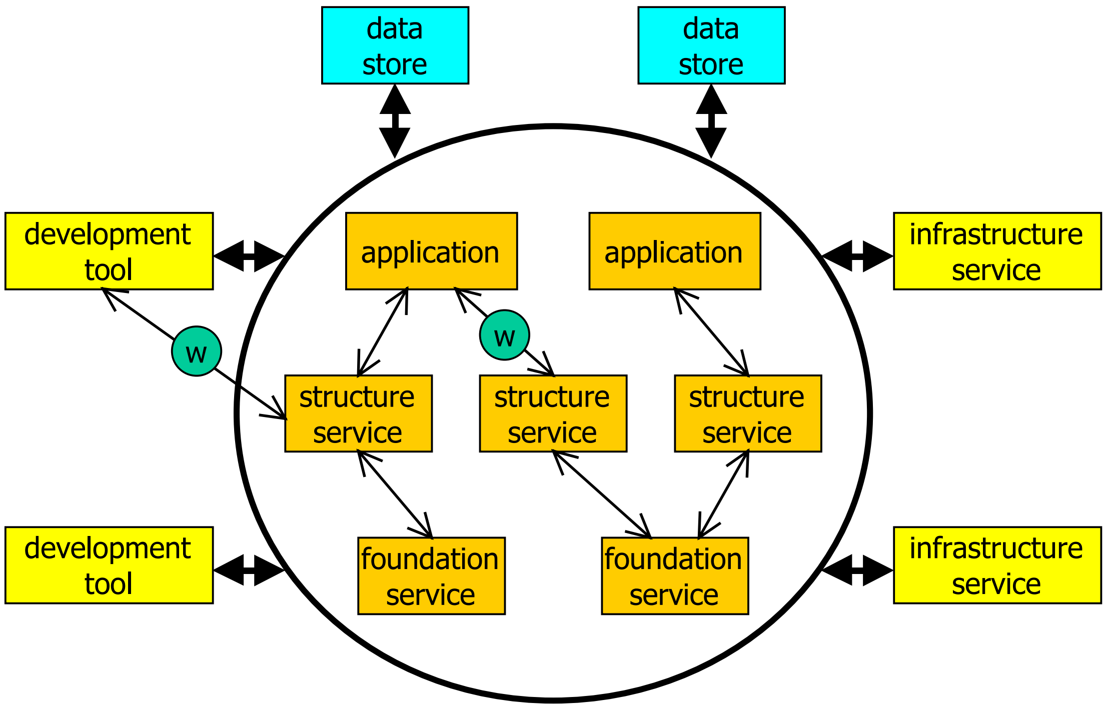
Users will also be able to access the metadata using a standalone Java application. This program uses the same code as the Java applet, but it has its own interface. In a future version, this application could be used with the XPCOM facilities to be linked to Netscape.

Both these solutions use the same wrapper as the Emacs integration with Construct. Thus, it will be easier to integrate all the applications in the Construct environment avoiding the multiplicity of wrappers.

The report continues with a short introduction to Construct that explains how this metadata project fits within the overall research project (Section 2). Sections 3 and 4 provide details about the analysis and design of the Netscape 6 integration, while Section 5 provides an overview of how to use the Construct metadata service in Netscape 6 and in the standalone metadata application. Section 6 concludes the report.

2. Construct

The Construct research project (www.cs.aue.auc.dk/construct) is concerned with providing different services to help users manage and structure their knowledge. The Construct environment consists of different categories of software components.



- *Applications*. This category includes desktop applications (e.g., Netscape, MS Word, and Emacs) that have been integrated (modified, extended, or wrapped) to be able to make use of the Construct structure services.

- *Wrapper services.* A wrapper is a service that allows a legacy application to be integrated with the Construct environment. Natively compliant applications do not need wrappers – all other applications do.

- *Structure services*. Integrated applications can make use of different types of structure services to organize data located in data stores. Each type of structure service provides a different set of structural abstractions (e.g., navigational, spatial, taxonomic, argumentation, workflow, and collaboration) that supports different application domains.

- *Foundation services.* Foundation services provide the very basic services in the environment that most other services depend on. Example foundation services are:

    - *Structure stores*. A structure store is a software component that manages storage and retrieval of structure. Structure stores can handle different types of structural abstractions.

    - *Multiuser and collaboration services.* This category includes services such as concurrency control, notification control, and access control.

    - *Versioning services.* Services that allow structural abstractions to be versioned.

- *Data stores*. A data store could be a file system, a database, a CD-ROM, etc.

- *Infrastructure services*. This category includes general services that enable the individual components of the environment to co-exist. Examples of these services are service discovery, naming, and location.

- *Development tools*. The development tools assist in the development of new services in the environment. Currently, three development tools exist:

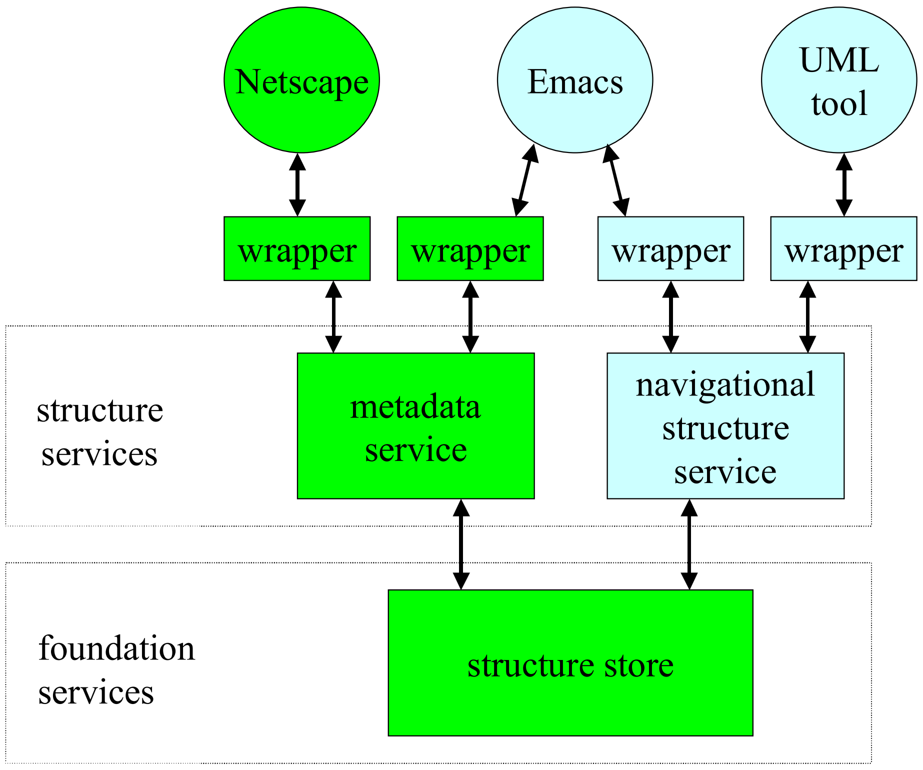
    - *UML Tool*. This tool allows the developer to specify new services by drawing UML diagrams in a graphical user interface. UML Tool translates the UML diagrams into IDL specifications.

    - *Emacs.* This well-known editing tool is used to create IDL specifications and Java code as well as documentation of various kinds. Emacs is considered a development tool of the Construct environment due to its integration with the navigational structure service.

    - *Construct Service Compiler (CSC)*. The CSC takes IDL specifications as input and generates service skeletons in Java. Service skeletons are wrapped in components that can automatically run as part of the Construct environment. The developer must add some code to the skeletons to define the semantics of the generated operations.

    Development tools can also be clients of structure services. For example, UML Tool and Emacs are clients of the navigational structure service. Thus, it is possible to link from classes or fields in the UML diagrams to, for example, design rational located in a text file under the control of Emacs.

The specific instance of the Construct environment in which the present WWW metadata project has been taking place is depicted below. The darker shaded boxes (green boxes when printed in color) indicate the components directly involved in the project.



When the present project began, the Construct metadata service was already in place. It was being used by the Emacs editor, through a wrapper service, to provide metadata support for Emacs files. Briefly stated, the goal of this project was to integrate Netscape 6 into the Construct environment using a similar wrapper and to then use the metadata service to provide metadata support for web pages.

3. Analysis

3.1 Needs

One part of the construct project is to provide metadata on different documents to users, which consult them using different software. One type of document is the web page, which is consulted via a browser.

The functionality, which has to be added to the browser, is an extension, which must permit the consultation of the Construct metadata server.

The Netscape 6 browser has been chosen because of its availability on most platforms. Another advantage of this browser is the possibility to get the source code.

The Java language permits to make an extension, which can run on most of platforms. Moreover it keeps a certain integrity with the other parts of the Construct project, which are Java components.

3.2 Means

3.2.1 How to extend Netscape 6 with Java

The Mozilla Project. Since 1998, the main part of the Netscape development has been done in the Mozilla project. The Mozilla project is based on the open source code of the former Netscape 5 project. At the end of 2000, the first version of the Communicator suite was issued from the Mozilla code (i.e., browser + mail client) and has been distributed; it is called Netscape 6.

Netscape 6 status. Netscape 6 is based on the status of Mozilla at the end of 2000. Today, in March of 2001, Mozilla's version is 0.8. According to Netscape, the Mozilla code has been optimized and then stabilized before being called and distributed as the Netscape 6 product. However, when we tested some development possibilities as they are described on the Netscape developer web site, we realized that some are not really possible with the current version.

3.2.1.1 Blackwood API

The Blackwood Project is a part of the Mozilla Project. It deals with use of the Java language with Mozilla and Netscape. The Java facilities of Netscape are based on the Open JVM integration facility, which allows different Java engines to be plugged in easily. The advantages of this facility alone are obvious, allowing users the freedom to choose whatever Java engine they want.

Pluglets or Java plugins. Another possibility given by the Blackwood API is pluglets. These are plugins and components that are implemented in Java, giving the advantages of Microsoft's ActiveX controls while remaining cross-platform. A plugin is a code module that behaves as though it is part of the Netscape browser. We can use the plugin API to create plugins that extend Navigator with interactive and multimedia capabilities that handle one or more data MIME types. Generally, plugins are developed in the C++ language and for the MS Windows platforms. Plugins can use the Netscape Java Runtime Interface (JRI) to access Java. Communication with Java, and through Java, with JavaScript, is possible with a LiveConnect connection.

The Java plugin is useful when you have to use a singular data type. It could be integrated to the web page.

XPCOM. The Blackwood API would permit, when it will be finished to connect standalone Java applications and Netscape, using a COM like interface. Netscape 6 should bring a COM-compatible cross-platform component object model called XPCOM. It follows that Netscape components would be scriptable. Component methods may be invoked from any language for which there exists a binding to the corresponding XPCOM interface. In Netscape, the primary language supported is JavaScript, and the binding to XPCOM components is provided by a library called XPConnect. The Java Virtual Machine communicates with Mozilla/Netscape through the Object Java Interface.

Applet. An applet would allow the functionality to be easily integrated to a web page situated in the Netscape 'My Sidebar' for example. This solution requires a Swing or AWT Java interface. It will be easier if necessary, to port this extension to Internet Explorer but it seems to be a less integrated solution.

For JavaScript to be able to control a Java applet, the Java applet must provide public methods. The methods of an applet are the functions, which are associated with its class. Public methods are functions, which are externally visible and can be called by other Java classes and by JavaScript. Well-written Java applets will provide public methods to make basic properties readable and writeable by other code. This makes it possible to customize and reuse existing Java applets without modifying the applet's source code.

3.2.1.2 The better choice at the moment: The applet

The three solutions found have advantages and drawbacks. Integrating a Java plugin is more useful when you have to use a specific data format (especially for multimedia and interactive). Applet extension is more portable if you need in the future to develop the Construct services for another browser. The Java application seems to permit a good integration to the Netscape 6 navigator. Furthermore, it allows the application to be used without any browser if you just have to consult documents' metadata.

The standalone application, which would communicate, with the Netscape 6 application would probably be the best solution. The fact that it would be a real application would let the user use it without the need to have to launch Netscape 6 to consult metadata. He could consult them just by typing a URL in the application. Use of XPCOM would enable future integration with the Microsoft Internet Explorer browser, which uses the COM system to communicate with other applications.

Netscape 6 limitations and the Mozilla project. There are many subprojects in Mozilla, which should improve the Netscape Communicator browser. The Blackwood Project is one of those subprojects of the Mozilla Project, which is the open source development part of Netscape. Some parts of the API are here but can be used only with very simple examples like some tests that are bundled with Mozilla source code.

Unfortunately, the Netscape 6 browser cannot use the XPCOM capability, which should be integrated in the next release of the product.

The problem is the same with the pluglet solution, which can run, but the API is not fully implemented and there are few possibilities today.

So for the project, the last solution has been chosen, the Java applet solution, which is the only one, which is not affected by Netscape 6 limitations.

3.2.2 My Sidebar

Description. My Sidebar is a mini window within Netscape 6 that can contain a Java applet or an HTML page. It allows you to get the content of the web sites you choose. It can also offer easy access to data from the web sites you want.

My Sidebar tabs can be constantly updated to keep you connected to the information you need. My Sidebar tab is a web page formatted to fit in the My Sidebar window. It supports languages like HTML 4, JavaScript, XML, CSS, and DOM. It allows tabs to be dynamic, customizable, and interactive. Anything that can be done in a standard web page can be done in a Sidebar tab. Therefore, it is easy to integrate an interactive Java applet, which could display metadata of the Netscape main frame.

Sidebar's advantages. For developers, it is easy to build, update, and manage the sidebar. You can create the Sidebar tab like a web page and integrate a Java applet. You can easily analyze the content of the web page in the main frame. There is no need to modify the Netscape sources.

For web users, it is convenient, customizable, and dynamic. With My Sidebar tabs, the web user can easily customize the metadata services applet. It is very easy to install and use. You just have to click on a link that runs a JavaScript function, which installs the package needed on your computer. Metadata are easily accessible at any time. My Sidebar tabs can be updated automatically. Tab content itself can be personalized to suit an individual's needs. A user can interact with the tabs without losing track of what he is browsing in the main window. Multi-task is permitted. Because it uses a Java applet, users can separate the program from the sidebar and drop it where he prefers on his desktop.

Sidebar's drawbacks. For developers, there are some restrictive rules to respect. Although any file format that the Navigator 6 browser can display can appear in My Sidebar, these files will have to display well in a window that is smaller than 170 pixels wide by 150 pixels tall. When designing the My Sidebar tab, we must be sure that information is presentable in a small space and responds well to being resized by the user. The default width of My Sidebar is 170 pixels wide. The content area is 162 pixels wide (144 pixels wide when a scroll bar is included). You can effectively utilize the space within the My Sidebar area by decreasing font sizes. The tab title width is approximately 20 variable width western characters or 10 double-byte characters.

A good integration to Netscape 6. The interface between users and the Construct metadata structural service could be implemented in many ways. We have decided to design two different interfaces for many reasons.

The first one is made in XUL. This language is used in Netscape 6 to display the browser interface. The sidebar gets the same look as the browser, so it is the default interface.

In this case, we have to use a hidden applet located in the sidebar. The URL of the web page and metadata will be transmitted to this applet thanks to the LiveConnect protocol.

The second solution is the Swing interface of the Java applet. This interface uses the most important part of the sidebar to display metadata. We have discovered two problems with this solution. It is not possible to use a Java applet in the sidebar, so we have decided to create a frame in the sidebar in order to resolve this bug. Unfortunately, we cannot use a local applet for some security reasons that are why the applet is located on a distant web server.

The only data transmitted from the JavaScript to the applet is the URL of the current web page because metadata are directly fill and display via the Swing interface.

This kind of interface has been created for two reasons: The applet could be run as a standalone application without Netscape 6, so a Swing interface is necessary. It will be easier to integrate the applet to Internet Explorer if we want to make it an application of the Construct metadata structural service.

3.2.3 Security issues

Java applet security issues. The original security policy of Java 1.0 uses the Sandbox model. This model is very simple and restrictive. The "Sandbox" represents a small part of the system resources the Java Virtual Machine (JVM) has access to.

Under this model, applets were always assumed to be hostile so they were restricted to use "Sandbox" resources and nothing else. Any attempt to leave the "Sandbox" would cause the applet to throw a security exception and exit. An applet cannot load libraries or define native methods.

An applet cannot ordinarily read or write files on the host that is executing it. An applet cannot make network connections except to the host that it came from. An applet cannot start any program on the host that is executing it. An applet cannot read certain system properties.

Java 1.1 introduced the concept of signed applets. Signed applets allowed a user to verify the origin of the applet and its integrity.

This was done by digitally signing JAR files. This model allowed the user to trust certain applets depending on whether they originated from a known trusted source. Certain restrictions could be removed with this signing process. However, Java 1.1 provides only two levels of security, safe or not safe.

Java 2 security policy. The Java 2 security policy keeps the notion of trusted and un-trusted code to a minimum. Instead, code runs at different permission levels.

This system improves the signed applet capabilities of Java 1.1, but this time a specific security level must also be supplied for a specific signed applet. We can define, for a specific applet, files that it can read or write for example.

Construct Java applet and security. In the Construct extension, the applet needs to establish a connection with the metadata structure server. This server could not have a web server to supply the Construct applet, but the applet can only establish connection with a host, which belong to the same domain that it is from.

The way to allow the applet to communicate with another host is to trust it:

- This can be by signing the applet, so the operation can be transparent for the user who would like to use it.
- Another way is to install the applet on the user's local file system. The applet must be placed in a *CLASSPATH* directory.
- The Java 2 security policy tool allows us to choose to trust applets from specific domains.

4. Design

4.1 Description

4.1.1 Two applications in one

The system can be seen as two different parts, even though there is only one source code. In fact, the final code can be launched as an applet, in an HTML or XUL page or as a standalone Java application.

4.1.1.1 Applet

The applet is launched as a normal applet, it is called by a XUL or an HTML page between *<applet></applet>* tags. Therefore, the applet has, like others applets, an *init()* method, which calls the *jbInit()* method.

4.1.1.2 Standalone application

Because we have to load and run Netscape 6 each time we want to test the applet, we have decided to make, in parallel with the applet code, a standalone application, which permits us to test the main functionality. This standalone application has its own swing Java interface.

Advantages of such a solution. Having a standalone application in addition to the Java applet offers other advantages in addition to those for the development phase.

For the moment, until the next release of the browser, the XPCOM capabilities are not available for a Netscape external Java application. However, when it is available, this standalone program will be linked with Netscape without having to develop another user interface.

The user may choose to consult metadata about a specific URL without having to launch the Netscape 6 browser, which is a very heavy application, in terms of memory and processor use.

Swing or AWT interface. We had to choose between two types of Java GUI API. In fact, there is the old one, AWT, and the layer above it, more recent, the Swing API. The Swing API is available only in the more recent releases of the Java runtime environment.

The main goal of the project is to make an extension for the Netscape 6 browser. With this browser is bundled the Java 2 runtime environment.

Because this version of Java was available with Version 6 of the browser, we have decided to use the Swing API.

4.1.1.3 The same code for the applet and for the application

Using the Inprise Jbuilder integrated development environment, there is an option, during the project creation process that provides the opportunity to add a Boolean parameter to the skeleton code of the Core class, which is called *isStandalone*. This attribute will be initialized at the Core instance creation.

If the code is launched as an applet, the *init()* method will be called at the beginning. So in the *init()* method, the *isStandalone* parameter will be set to *false*.

However, if it is as a standalone application, the parameter will be set to *true*, through the *main()* method.

Each method, *init()* or *main()* will later call the *jbinit()* method, which contains the other instructions needed by the application.

4.1.2 A wrapper

Emacs wrapper. A metadata extension has already been done for the Emacs software. Due to the software implementation language, Emacs' metadata extension has been written in the Lisp language. This extension provides the ability to consult metadata for documents, which are opened in the text editor. One Emacs buffer is used to edit the text, and, below it, there is another buffer to display the metadata. An entry in the main menu has been added to manipulate those data.

For communication between Emacs and the data server, TCP/IP has been chosen. A wrapper, in Java, links the Emacs extension and the metadata server. The Emacs extension sends, via TCP/IP, queries to the wrapper, which use Construct services to manipulate the data in the structure storage component.

Reuse of the wrapper. We decided to keep the same wrapper to extend Netscape 6. In fact, we have begun to test the first Netscape extension functionality with this wrapper. Later we realized that this wrapper would be a good one for the Netscape extension project too. It can manage the metadata with simple commands, which are easy to understand and may be extended in the future.

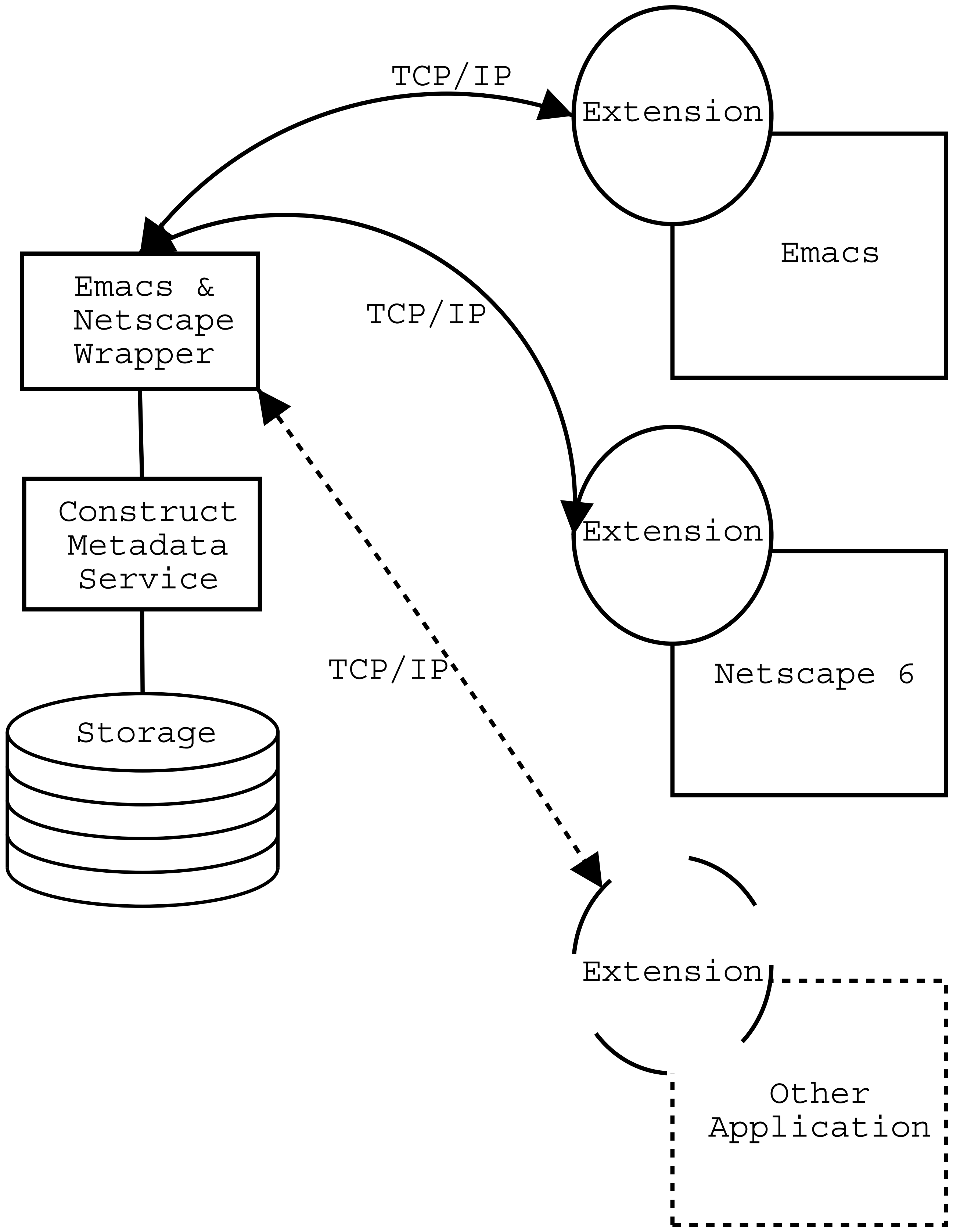
For example, when the user wants to get a metadata record for a specific URL, an order like this one is transmitted to the wrapper:

```
Message from Netscape: Get metadata record http://www.google.com
```

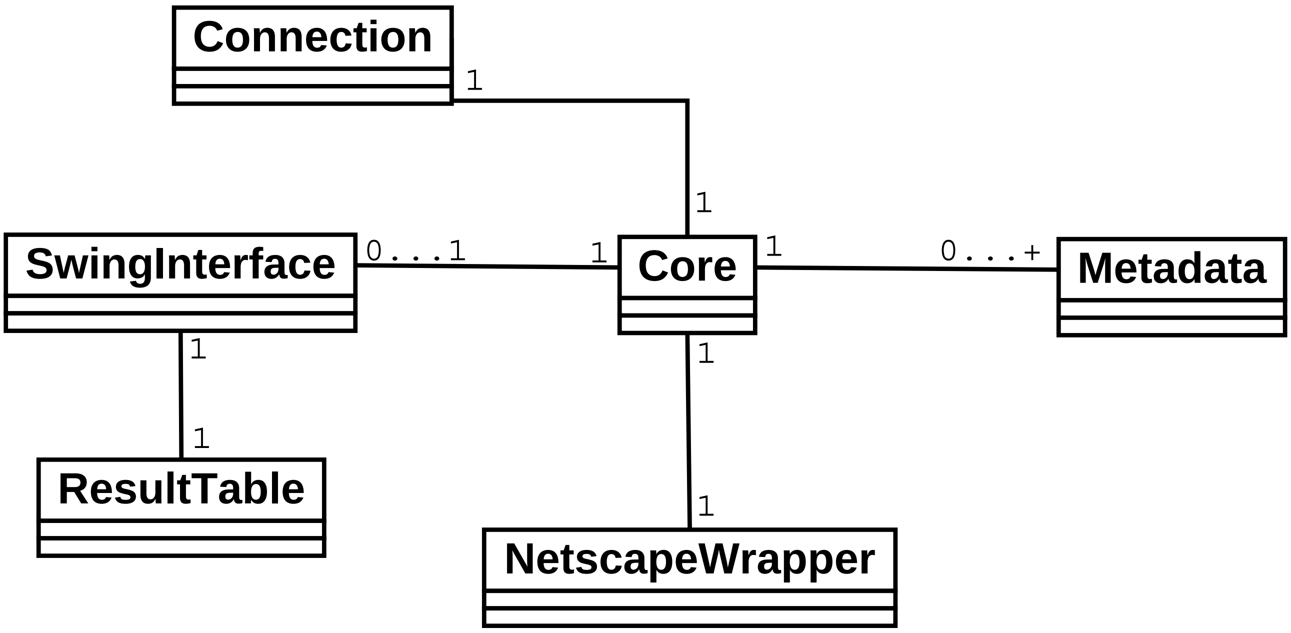
And the wrapper will answer to Netscape, giving it another simple string where tokens can be easily found by special character sequences, like #@key: and #@value:

```
Message to Netscape: Metadata record http://www.google.com #@key: test1
#@value: test1 #@key: machine #@value: super #@key: truc #@value: muchejuh
#@key: rtrytrytr #@value: ngjv
```

A great advantage of the fact that we have chosen the same wrapper is that it will be easier to integrate Netscape into the Construct environment. Indeed, there is only one wrapper for two applications, and we can suppose that other extensions will be able to use it also.


TCP/IP
Extension
Emacs
Emacs & Netscape Wrapper
TCP/IP
Construct Metadata Service
Extension
Netscape 6
TCP/IP
Storage
Extension
Other Application

4.2 Class diagram



4.2.1 NetscapeWrapper Class

The NetscapeWrapper class is a simple class with the commands, which can be sent to the server as final static parameters.

Here are the final static attributes, which can be used when calling the NetscapeWrapper class instance:

GET_RECORD "Get metadata record"

CREATE_RECORD "Create metadata record"

UPDATE_RECORD "Update metadata record"

DELETE_RECORD "Delete metadata record"

GET_KEY "Get metadata key"

ADD_KEY "Add metadata key"

UPDATE_KEY "Update metadata key"

DELETE_KEY "Update metadata key"

4.2.2 Connection class

The Connection class is the class, which knows parameters about the metadata server. The two main attributes are the hostname and the port number. This is the class that will make the connection with the server and which will try to reconnect if there is a problem. Connections will be done using TCP/IP; two attributes of the class are socket descriptors. By default, connections will be established with the "localhost" address and on the port 16000, but there are different constructors, which allow to choose the ports at the instantiation instant.

4.2.3 Metadata class

The Metadata class represents metadata handled by the application. Normally it is a character string being used as key, associated with another one corresponding to the value of the metadata.

We can imagine that in the future other data, like pictures or video sequences, could be used instead of characters. Normally, in this case, the metadata class would be the only class to change, the other classes deal with the data via the getMetadataValue() method, which will return an Object class derived object.

4.2.4 SwingInterface class

The SwingInterface class provides the methods used by the Swing interface – that is, by the standalone application. The java code should be usable with two types of interface, a XUL one and a Java Swing one. If the user wants to use the standalone application, the Swing interface will be used. In applet use situations, the XUL interface in the sidebar will be shown to the user.

The SwingInterface class contains all the attributes relating to the design of the interface. It implements also the methods necessary to handle the actions of the user on an element of the interface, when he presses a button for example.

4.2.5 ResultTable class

The ResultTable class is an AbstractModelTable derived class, which is used by the Interface class to display data in a scrollable table. It has some methods, which are called when core hash table has been altered.

4.2.6 Core class

The Core class is the central class, the one which will join the interface part of the applet and the metadata server. It will control the Connection class, asking it to establish a connection at the beginning, giving it a new server address or port. The Core class will also instantiate the Connection and the Interface class.

Attributes. After the first query to the server for a URL’s metadata, a hash table will be filled. This hash table enhances performance because the client has the results in memory. All the methods to consult the data will manipulate firstly the hash table. After that, if modifications have to be written on the server, a query will be sent to it. Therefore, in a simple consultation mode (read only), only one query will be done.

Methods. Core class' methods can be divided into two parts

- Internal methods: these are private methods, which are used to establish the connection and deal with the server.

- Interface methods: these methods, which are public, are presented to the Interface instance and to the LiveConnect protocol.

Thanks to these methods, LiveConnect is able to make the applet connected with the browser, providing it queries data or document location for example.

4.2.7 Communication between XUL and Java

With Netscape, it is possible to communicate between Java and JavaScript by using *LiveConnect*. It is enabled by default in the browser. For *LiveConnect* to work, both Java and JavaScript must be enabled. This protocol lets us perform the following tasks:

- Use JavaScript to access Java variables, methods, classes, and packages directly.

- In JavaScript, a wrapper is an object of the target language data type that encloses an object of the source language. On the JavaScript side, a wrapper object is used to access methods and fields of the Java object; calling a method or accessing a property on the wrapper results in a call on the Java object. When a JavaScript object is sent to Java, the runtime engine creates a Java wrapper of type *JSObject*. There are only two restrictions if you want to use a Java method, the applet has to be loaded at the time of calling the Java method and this method must be public. We use the NAME attribute of the <applet> tag, for example, the format for code is:

```
document.applets("theAppletName").theMethod();
```

This technique is use to transmit the URL of the web page to the applet and the metadata when the user has chosen an XUL interface.

Use Java code to access JavaScript methods and properties. On the Java side, JavaScript objects are wrapped in an instance of the class *Netscape.Javascript.JSObject* and passed to Java. If we want to include JavaScript objects in Java, we must import this *netscape.javascript* package into our Java file. It contains the additional object model *JSObject* that can be used from within a Java applet to access JavaScript objects in a document. When a *JSObject* is sent from Java to JavaScript, the runtime engine unwraps it to its original JavaScript object type. The general format for code is:

```
JSObject.objectname.method();
```

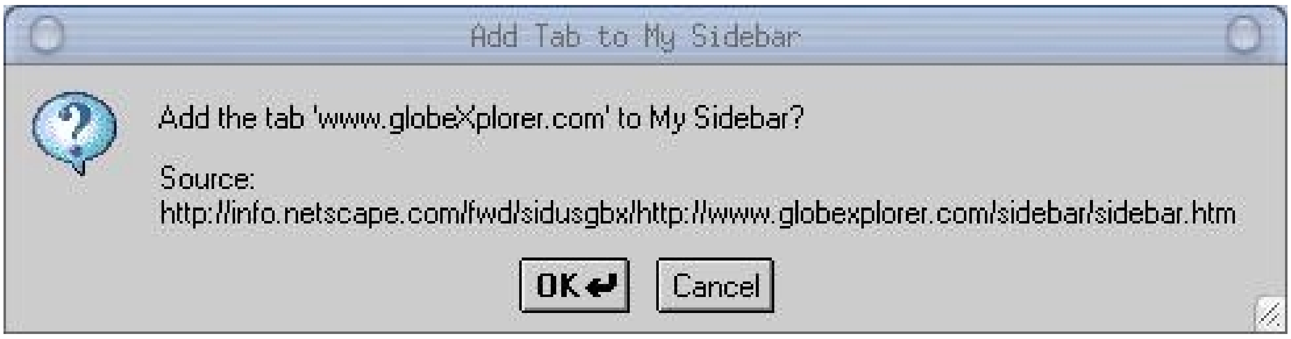
or

```
JSObject.objectname.property();
```

It provides the ability to transmit metadata from the server to the JavaScript document by using the Java applet. These metadata could be easily displayed thanks to the XUL interface in the sidebar.

4.2.8 Installation procedure, XPI package

A set of UI objects (including the description of its structure, appearance, behavior, and localizable strings) is called a package. Sample packages in the Communicator product include Navigator, Messenger, Preferences, Bookmarks and Composer. Packages can communicate with one another and may share functionality (which can be abstracted into a common package). Generally, a package is a Jar archive.


Add Tab to My Sidebar

Add the tab 'www.globeXplorer.com' to My Sidebar?

Source:
http://info.netscape.com/fwd/sidusgbx/http://www.globexplorer.com/sidebar/sidebar.htm

A package can be added to an existing application, without modifying it in any way. It is also possible to use an installation procedure. This kind of package is called an XP Installer package (XPI). A JavaScript function provides the ability to trigger it from a web page.

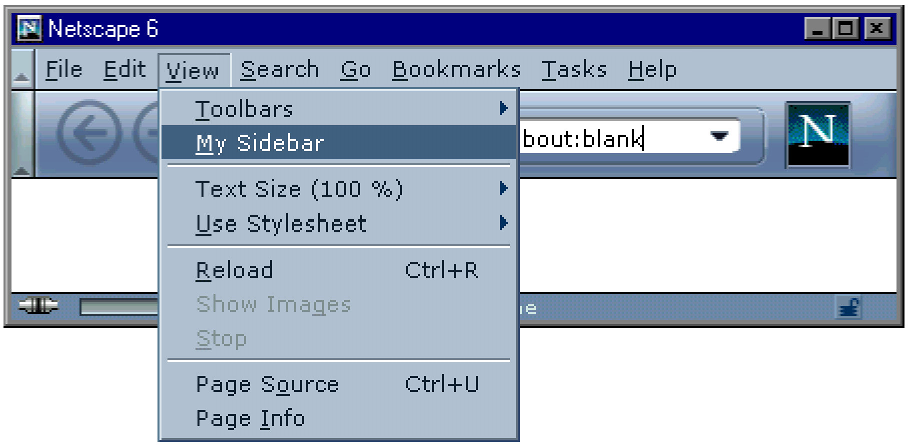
5. User Manual

The Construct metadata service functionality can be used in two different ways. It can be launched in the Netscape 6 sidebar or run as a standalone application. We shall see how to use these two interfaces, which are quite similar.

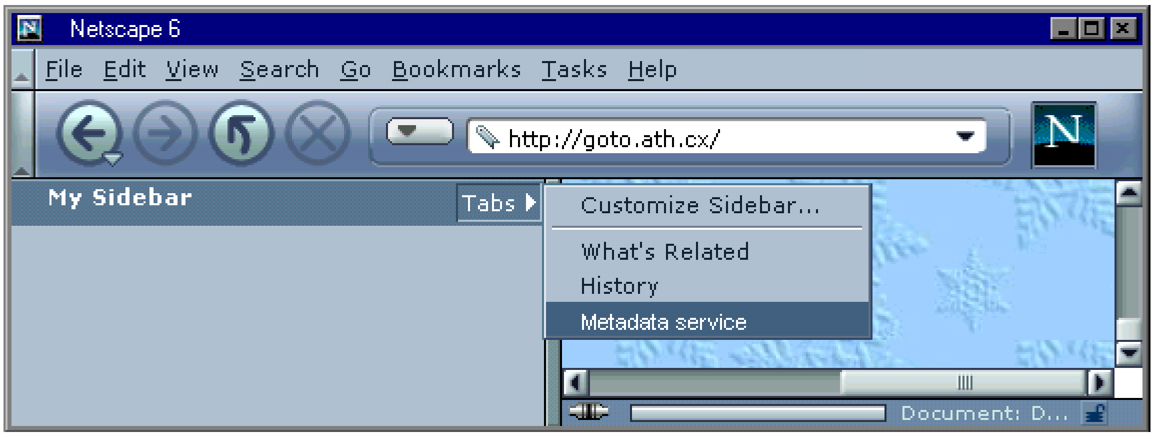
5.1 The Netscape 6 interface

5.1.1 Launch the metadata service

The metadata services on the WWW have been included as a panel in the Netscape 6 'My Sidebar'. To view the sidebar, you have to select the 'My Sidebar' item in the 'view' menu. Then, it will appear on the left side of your browser.



If you want to run the Metadata service, you just have to select the 'Metadata service' item in the 'Tabs' menu. Then it will automatically display the application interface in the 'My Sidebar'.

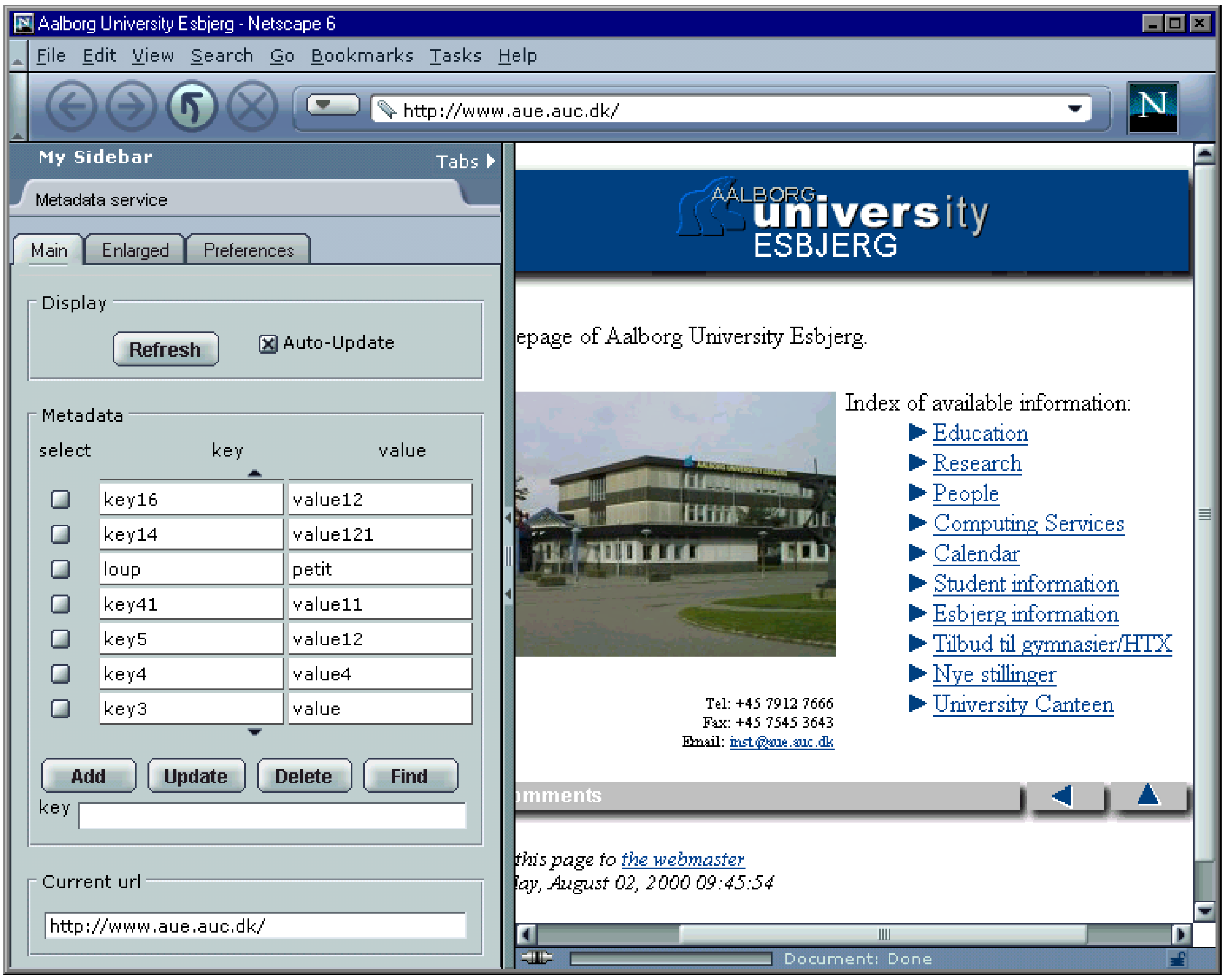


The application will be include in a panel like all the other 'My Sidebar' functionality.

You can see three different tabs (main, enlarged, preferences) on the top of this panel. They allow you to choose what kind of operation you want to execute.

5.1.2 The main panel

The main panel is the default panel. It allows you to do all the needed operations on the metadata. It is separated into three different parts.



key	value
key16	value12
key14	value121
loup	petit
key41	value11
key5	value12
key4	value4
key3	value

http://www.aue.auc.dk/

epage of Aalborg University Esbjerg.

Index of available information:
- Education
- Research
- People
- Computing Services
- Calendar
- Student information
- Esbjerg information
- Tilbud til gymnasier/HTX
- Nye stillinger
- University Canteen

Tel: +45 7912 7666
Fax: +45 7545 3643
Email: inst@aue.auc.dk

mments

this page to the webmaster
lay, August 02, 2000 09:45:54

5.1.2.1 The display section

This section permits you to choose if you want to see new metadata each time you load a new web page. If you check the 'auto-update' button, it will be done automatically. Otherwise, you will have to use the refresh button to get metadata for the new web page.

5.1.2.2 The metadata section

This section shows you all keys and values of the web page. When there are too many keys, you can put your mouse over two scroll buttons at the top and bottom of the list and it will automatically show you the other data.

Add a key. If you want to add a key, you enter the name of the new key in the key field and press the add button. Then, it will switch to the enlarged panel where you will have to enter the value of your key and to confirm your operation.

Update a key. A key can be updated by selecting it or by entering its name in the key field. Then, when you press the update button, you will see the value of this key in the enlarged panel and it will be possible to modify it.

Delete one or more key(s). You can delete one or more keys by checking their select buttons and pressing the delete button. It is also possible to enter the name of the key you want to delete in the key field and then press delete.

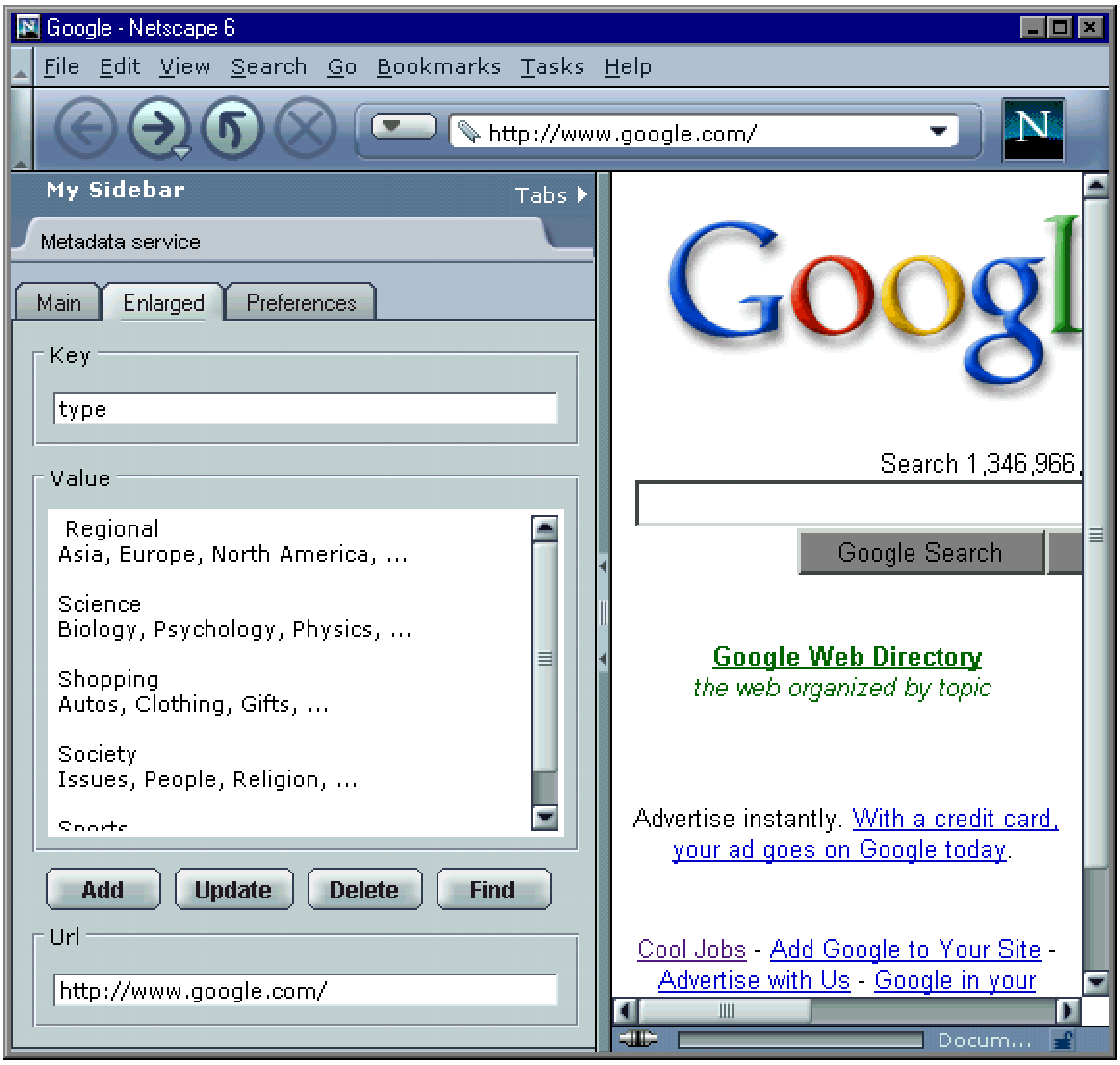
Find or enlarge the display of a key value. The main panel displays all the keys and values of a web page, so it could be difficult to find a key in the list or to read a value because of the size of the cells. If you select a key or enter its name and press the find button, then the key and its value will be displayed in the enlarged panel and it will be easy to read the value of this key.

5.1.2.3 The current URL section

The field of this section just reminds the user of the URL of the current metadata. If the 'auto-update' button is checked, this URL will change when a new web page is loaded.

5.1.3 The enlarged panel

The second panel has three parts. It allows you to do the same operations as the main panel. You can directly add or update a key if you enter its name and its value in the two fields key and value. You are also able to find the value of a key or delete it by giving its name. The URL of the key is displayed at the bottom of the interface like in the main panel. However, this URL will always be the same because it depends on the key and not on the current URL.


Google - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
http://www.google.com/
My Sidebar
Tabs
Metadata service
Main
Enlarged
Preferences
Add
Update
Delete
Find
Search 1,346,966,
Google Search
Google Web Directory
the web organized by topic
Advertise instantly. With a credit card, your ad goes on Google today.
Cool Jobs - Add Google to Your Site - Advertise with Us - Google in your
Docum...

Key

type

Value

Regional
Asia, Europe, North America, ...

Science
Biology, Psychology, Physics, ...

Shopping
Autos, Clothing, Gifts, ...

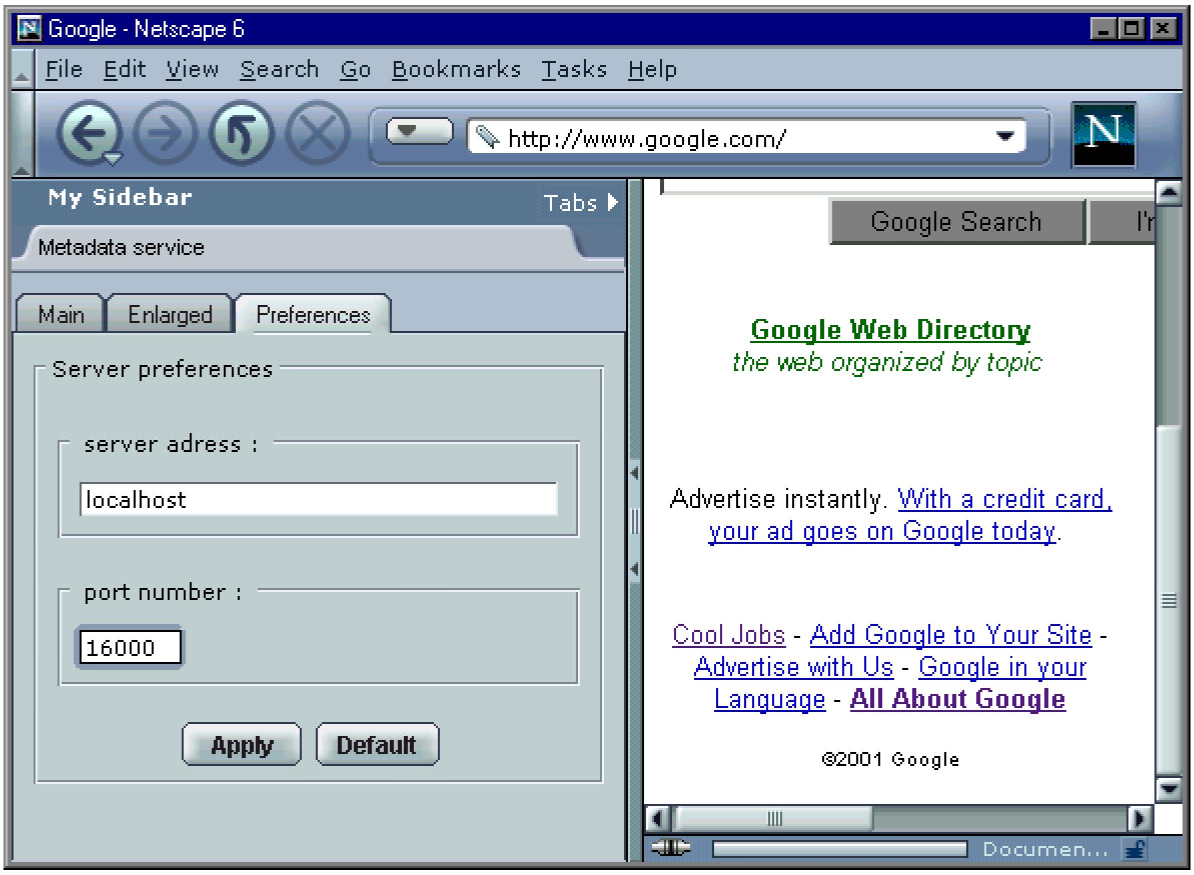
Society
Issues, People, Religion, ...

Url

http://www.google.com/

5.1.4 The preferences panel

The third panel allows you to choose which metadata server you want to use. You just need to enter the address and the port number of this server into the two corresponding fields. It is also possible to restore the default parameters if you prefer to be connected to the default server.



server adress : localhost

port number : 16000

Advertise instantly. With a credit card, your ad goes on Google today.

5.2 The standalone application

If you do not need to consult any web pages, you can run the Java applet as a standalone application. The interface is quite similar as the Netscape 6 metadata service sidebar. It still has three panels with the same functionality, but there are also some differences.

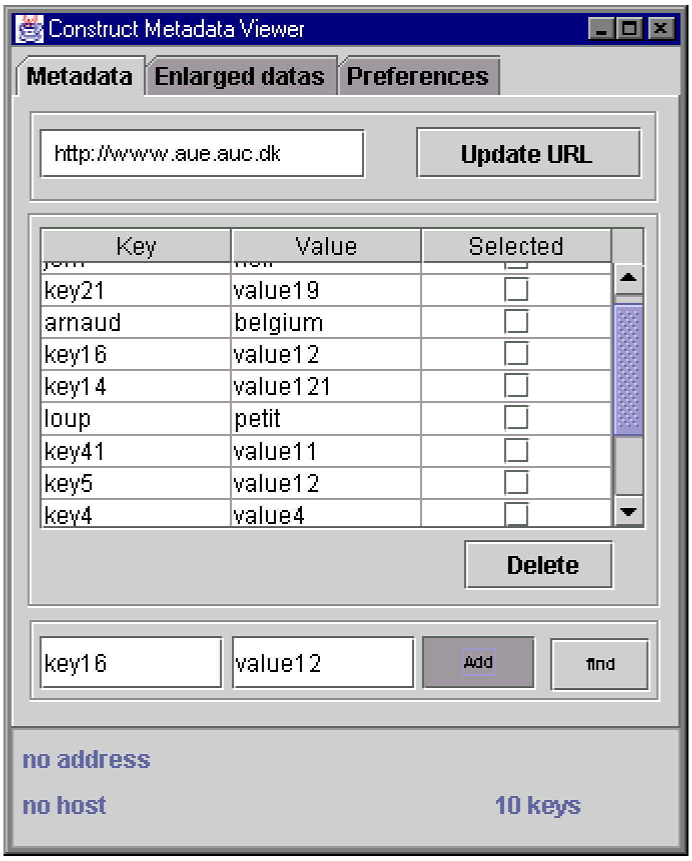
5.2.1 The Metadata panel

First of all, you have to enter the URL of the web page for which you want to consult the metadata. When you have filled in the field, press the 'update URL' button. Then the metadata and their values will appear as a list.

It is possible to select one or more keys by checking their 'Selected' button and to delete them by pressing the ‘delete’ button.

You can add a value by giving its name and value and pressing the 'add' button. If you want to update a key, it is the same method, a confirm box will just ask you if the update should be done.

The value of a key could be found by filling in the name field and pressing the 'find' button. If this key is recorded in the database, the value will be displayed in the value field.



http://www.aue.auc.dk

Key	Value	Selected
key21	value19	☐
arnaud	belgium	☐
key16	value12	☐
key14	value121	☐
loup	petit	☐
key41	value11	☐
key5	value12	☐
key4	value4	☐

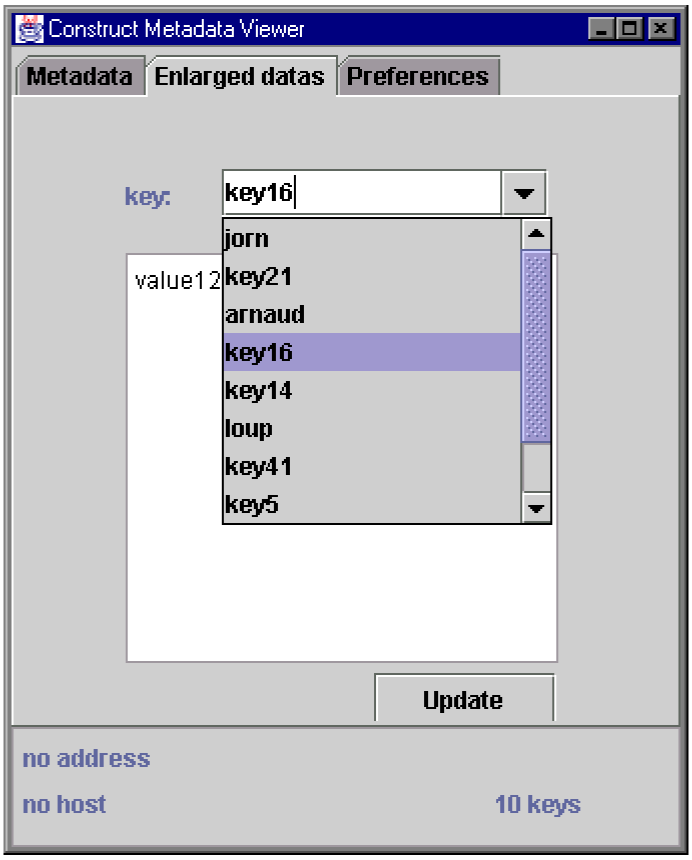
key16 value12

no address

no host 10 keys

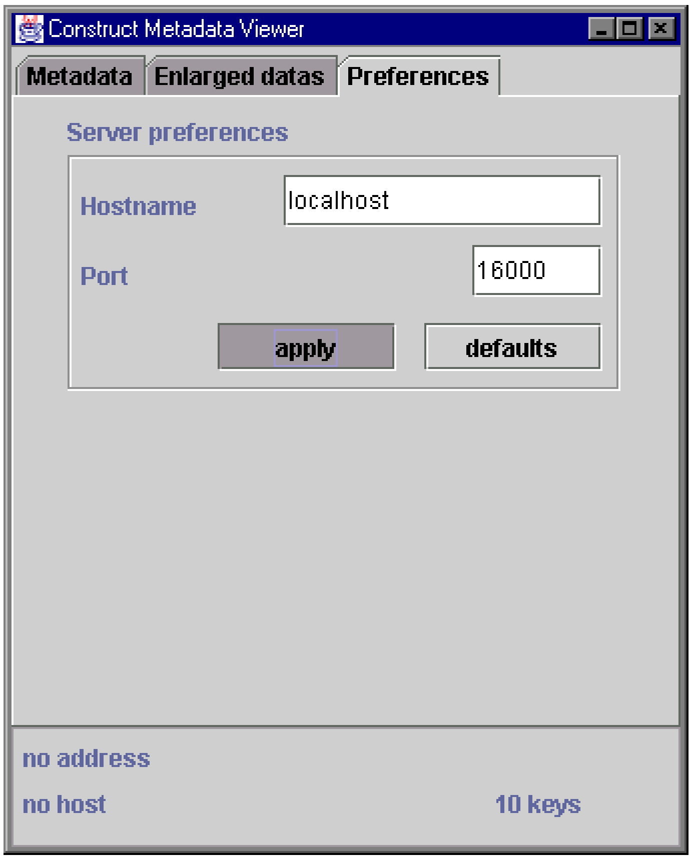
5.2.2 The enlarged data panel

If the value of a key is very large, it is easier to read it in the 'enlarged data' panel. You can select the name of the key of your choice in a scroll menu. Then, the value of the selected key will automatically be displayed in a large field. This value can be changed and updated by pressing the 'update' button.



5.2.3 The preferences panel

The third panel is exactly the same as the Netscape 6 one. You can choose which metadata server to use. You should fill in the address and the port number fields and press the 'apply' button. If you want to restore the default parameters in order to be connected to the default server, you just have to press the 'default' field.



Hostname localhost

Port 16000

no address

no host 10 keys

6. Conclusion

The Netscape 6 metadata project has been a success. The Construct metadata services are now available on the World Wide Web both as an integral part of Netscape 6 using an applet and as a standalone Java application.

The Netscape 6 integration software has been developed in a general way that will allow additional Construct structure services to be made available on the World Wide Web in the future using the general code base developed in this project.

References

1. Blackwood Project API. http://mozilla.org/projects/blackwood/java-util/api/index-all.html

2. JavaScript Developer Central. http://developer.netscape.com/tech/javascript/

3. MySidebar's Developer's Guide. http://developer.netscape.com/docs/manuals/browser/sidebar/index.html#creating

4. Netscape Plugins Developer's guide. http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm

5. XUL Reference. http://www.mozilla.org/xpfe/xulref/XUL_Reference.html

6. XUL tutorial. http://www.xultutorial.com

7. Sun Java Reference. http://java.sun.com

8. JavaScript FAQ. http://developer.irt.org

9. Java developer connection. http://developer.javasoft.com