
JADE 6

 JADE Development Centre

HTML Thin Client - HTML Documents

In this paper . . .

HTML THIN CLIENT - HTML DOCUMENTS	1
INTRODUCTION	1
OVERVIEW	2
CREATING AN HTML THIN CLIENT APPLICATION	7
HTML THIN CLIENT FRAMEWORK	18
PERFORMANCE CONSIDERATIONS	20
USER EXITS	21
PROGRAMMATIC INTERFACES	22
FUTURES	25
APPENDIX A: APACHE SERVER CONNECTIONS	25
APPENDIX B: JADEHTTP INITIALIZATION FILE	26
APPENDIX C: CONFIGURING APACHE	29
APPENDIX D: INITIALIZATION FILE	40
APPENDIX E: DIRECTORY STRUCTURE	42
APPENDIX F: TYPICAL SETUP PROCESS	44
APPENDIX G: TROUBLESHOOTING TIPS	45
APPENDIX H: SAMPLE APPLICATION	47
APPENDIX I: COMPARISON	55



Introduction

The HTML Thin Client has been a feature of JADE since the release of JADE 4.0. The HTML Thin Client in JADE 6.0 introduces two programming models: one built around JADE controls - a model in which controls render their own user interfaces by generating HTML to return to Web browsers and firing events that are handled by JADE code, and the other built around external HTML documents - a model that provides for maximum flexibility and scalability. Since all of the action takes place at the JADE application end, virtually any browser can run an HTML Thin Client (in theory at least!).

This paper provides a detailed overview of the HTML Thin Client features using external HTML documents, its applicability, and how to optimize this interface to provide maximum performance.

Jade Software Corporation Ltd believes that the information furnished herein is accurate and reliable and has been prepared with care. However, no responsibility, financial or otherwise, can be accepted for any consequences arising out of the use of this material, including loss of profit, indirect, special or consequential losses or damages.

Overview

The HTML Thin Client technology is used to create programmable Web pages. They can present information in any browser, use code on the client to execute scripts, and in the JADE application to execute application logic.

With the HTML Thin Client the HTML documents are prepared externally and imported into JADE using the import wizard.

HTML Thin Clients:

- Allow the import of externally created HTML pages.
- Use the JADE environment for back-end processing
- Support a set of methods that allow developers to cleanly encapsulate Web logic.
- Allow for separation between code and content on a page, eliminating the "spaghetti-code" often found in client-side scripting or server-side scripting (for example, ASP).
- Provide session management features that preserve the state of a page between requests.

What The HTML Thin Client Helps You Accomplish

Web application programming presents challenges that do not typically arise when programming traditional client-based applications. Some of these challenges are:

- *Separation of client and server.* In a Web application, the client (browser) and server are different programs often running on different computers and maybe even on different operating systems. Consequently, the two halves of the application share very little information; they can communicate, but typically only exchange small chunks of simple information.
- *Stateless execution.* When a Web server receives a request for a page, it finds the page, processes it, sends it to the browser, and then, effectively, discards all page information. If the user requests the same page again, the server repeats the entire sequence, reprocessing the page from scratch. In other words, servers have no memory of pages that they have processed. If an application therefore needs to maintain information about a page, this becomes a problem that has to be solved in application code.
- *Unknown client capabilities.* In many cases, Web applications are accessible to many users using different browsers. Each of these browsers has different capabilities, making it difficult to create an application that will run equally well on all of them.

- *Data access.* Reading from and writing to a data source in traditional Web applications can be complicated and resource-intensive.

Meeting these challenges for Web applications can require substantial time and effort. The HTML Thin Client addresses these challenges in the following ways:

- *Browser-independent applications.* The HTML Thin Client provides a framework for creating all application logic in the JADE application, eliminating the need to explicitly code for differences in browsers. However, it still allows you to automatically take advantage of browser-specific features to provide improved performance and a richer client experience.
- *Abstract, intuitive, consistent object model.* The HTML Thin Client framework presents an object model that allows you to think of your HTML pages as a class, not as separate client and server pieces. In the model, you can program the page in a much more intuitive way than in traditional Web applications.
- *Session management.* The framework provides you with explicit ways to maintain the state of application-specific information. This is accomplished without heavy use of server resources and without sending cookies to the browser, two traditional means for storing state.
- *Scalable performance.* The framework allows you to scale your application from one computer with a single processor to a multi-processor multiple computer environment without any changes to the application's logic.

Processing Stages of an HTML Thin Client Document

The life cycle for an HTML Thin Client is, generally speaking, similar to that of any Web process that runs on the server. Certain characteristics of Web processing – information passed via HTTP protocol, the stateless nature of Web pages, and so on – apply to these applications just as they do to other Web applications.

However, the framework performs many Web application services for you. As one example, the framework captures the information posted with the page, extracts the relevant values, and makes it available to you in object properties. Even so, it is helpful to understand the sequence of events that occur when a page is processed. This knowledge will help you know how you can program your pages effectively.

Round Trips

One of the most important things to understand is the division of labor in an HTML Thin Client document. The browser presents the user with a page, and the user interacts with the page. However, all processing that interacts with JADE application components must occur in the application. This means that for each action that requires processing, the page must be sent back to the application, processed, and returned to the browser.

Note You can create client-script in these pages, which is useful for user input validation and for some types of UI programming. However, client script does not interact with the JADE application.

For example, if the user enters an order and you want to confirm sufficient inventory for the order, your application posts the page to the JADE application at an appropriate point in the user's order entry. The application logic examines the order, performs an inventory lookup, perhaps takes some action (such as modifying the page to indicate an error), and then returns the appropriate page to the browser for the user to continue.

In HTML Thin Client documents, most user actions such as clicking a button result in a round trip. The HTML document has full control over what events get returned to JADE for processing.

Updating the Page

In most Web processing scenarios, pages are created from scratch with every round trip. In effect, as soon as the server finishes processing a page and sends it to the browser, it discards the page information. The next time the page is posted, the server starts all over in creating and processing it. For this reason, Web pages are said to be *stateless*; that is, the values of a page's variables and controls are not preserved on the server.

In a traditional Web application, therefore, the only information that the server has about a page is the information that the user has filled into controls, because that information is sent to the server when the form is posted. Other information, such as variable values or property settings, has been discarded.

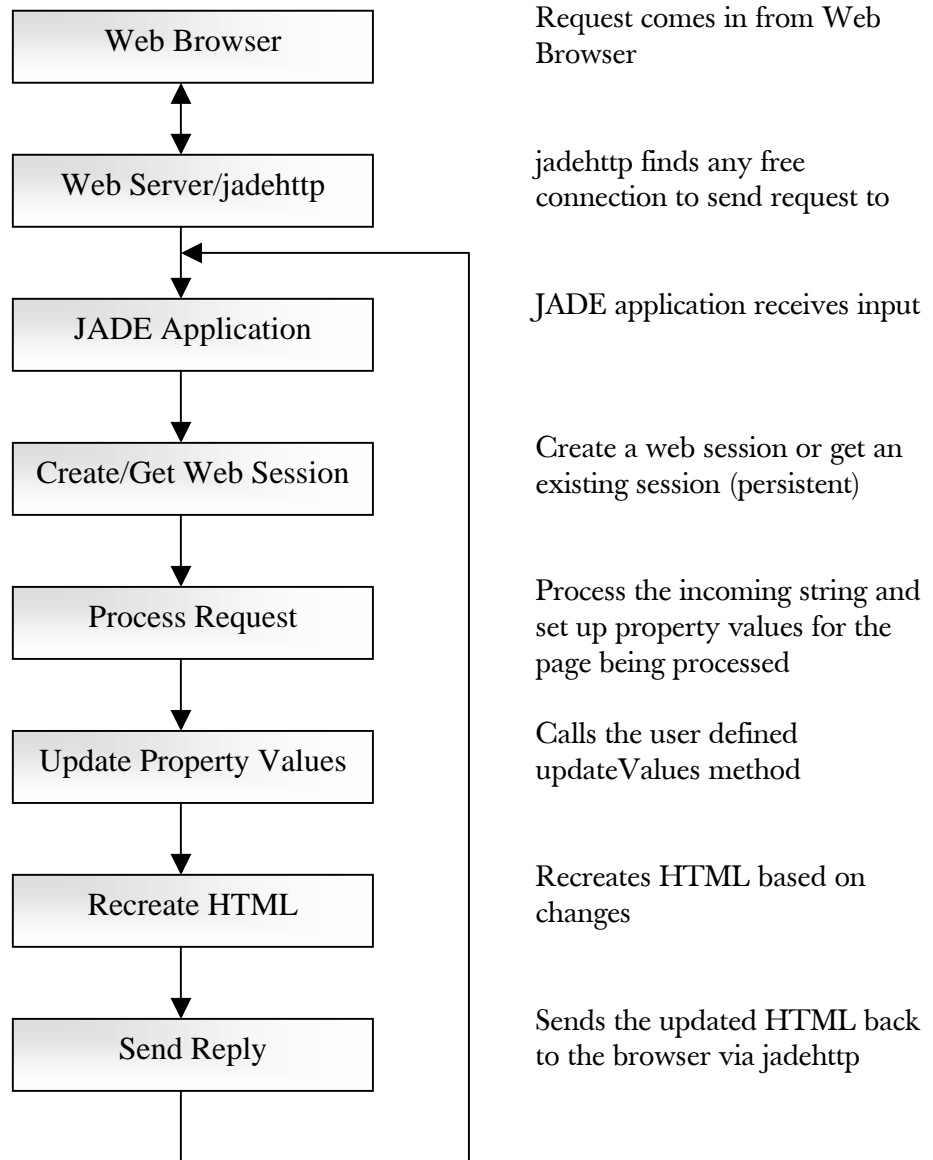
In the HTML Thin Client framework, session state may be saved in the session object. State information can also be saved on the page (by using hidden fields, for example). JADE saves state information on the page.

Stages in HTML Thin Client Processing

During forms processing, the form goes through a number of distinct stages. The following table lists the most commonly used stages of page processing. These stages are repeated at each round trip; that is, each time the form is posted.

Stage	Meaning
Process Request	Updates the form and control property values. Can be overridden by user logic.
Update Values	Calls the user defined method for setting up a page prior to being sent to the browser.
Reply	Sends a response back to the browser. Can be overridden by user logic.

The following diagram shows the message flow.



Architecture

The HTML Thin Client will work with either Microsoft's Web server, Internet Information Server (IIS), or the Apache server. For IIS, an ISAPI DLL (**jadehttp.dll**) is used for communicating between the server and the JADE application. For the Apache server, a similar module (**mod_jadehttp**) is used. The web application framework is shown in the following diagram.

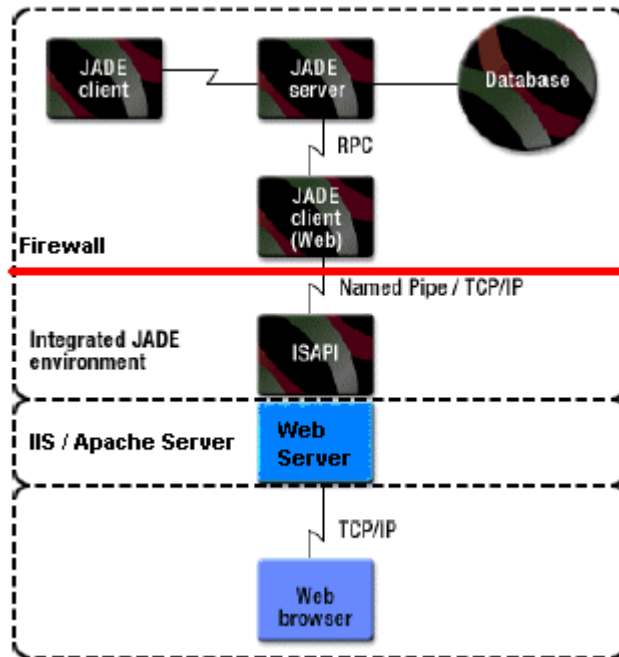


Figure 1

Named pipes are used by default for connection to JADE applications from the **jadehttp** module. Named pipes can only be used if the server is IIS. Alternatively, this module can handle multiple TCP/IP address connections to the same application, which allows the server to connect to multiple copies of the same JADE application on multiple hosts.

Note The main advantage of a TCP/IP connection over a named pipe connection is that the machine hosting the Web server can be different from the machine that is running the JADE application, to provide greater security via firewalls. A TCP/IP connection is also slightly faster than a named pipe connection.

Deployment

HTML Thin Client deployment is most useful in situations where a JADE application needs to be deployed on the Web; for example, in situations where the JADE Smart Thin Client is not a viable alternative. This feature is least useful in situations where you are designing a Web site, particularly where the Web site contains mostly static pages. In this case, the traditional approach of using a Web design tool to develop the pages would be the way to go. For the minimal set of pages where database storage and retrieval is required, the appropriate hooks can be put into the static pages to invoke an HTML Thin Client application and make use of user exits to customize the page, as required.

Creating an HTML Thin Client Application

The first step in creating an HTML Thin Client application is to import the HTML documents that are going to be the basis of your application. This can be done in a “batch mode” programmatically or by using the import wizard.

The HTML Document Browser enables you to add and maintain HTML documents in the current schema. Each schema in the JADE database can have a collection of HTML documents. The HTML Document browser has the following operations:

Command	Action	Result
Add	Add an HTML Document	Displays the HTML Wizard
Change	Changing an HTML Document	Displays the HTML Wizard
Edit HTML	Editing an HTML Document	Displays the HTML editor window
Extract	Extracting a Specific HTML Document	Displays the Extract dialog
Save HTML	Saving an HTML Document as a File	Displays the common Save As dialog
Reload	Reloading an HTML Document	Displays the Reload Document dialog
Remove	Removing an HTML Document	Deletes the selected document

Using the HTML Wizard to Add an HTML Document

Access the HTML Browser from the Browse menu and select the Add command from the HTML Browser menu.

The first page of the HTML Wizard, shown in the following diagram, is then displayed.

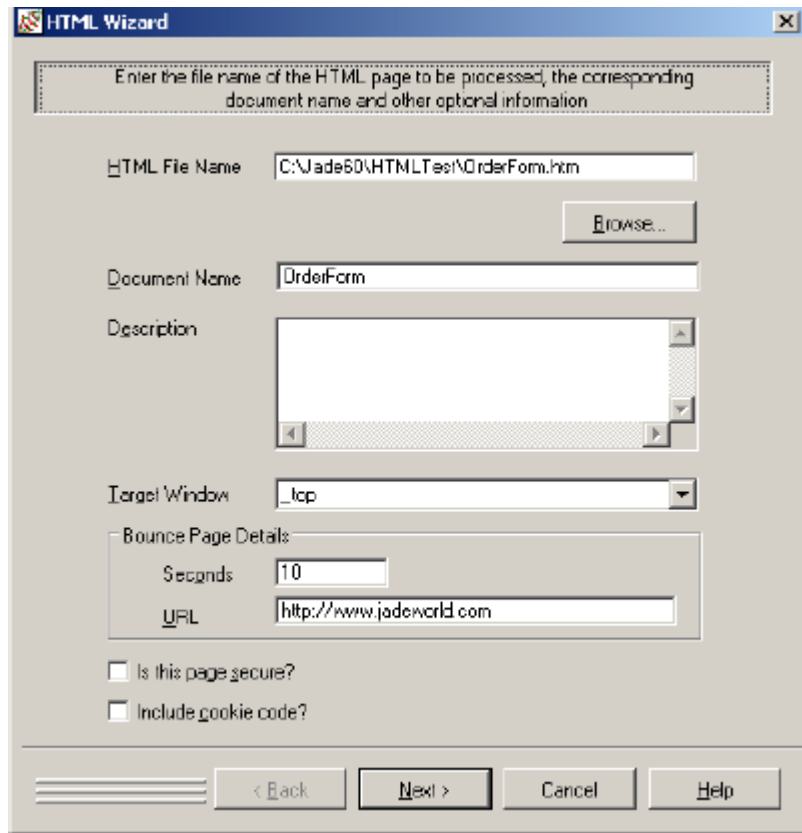


Figure 2

The HTML Wizard:

- Accepts an HTML file as input and processes the file to create a set of persistent objects that represent the file contents.
- Creates a class and properties corresponding to the **Input**, **Select**, and **JADE_TAG** tags that are present in the HTML.
- Enables you to change the default class and property names and to maintain a mapping of the HTML properties with the JADE properties.

The first page of the HTML Wizard enables you to define the name of your HTML document and its source.

In the **HTML File Name** text box, specify the name and location of the file from which to create the HTML document. Although this can be any valid file type, it is most often an HTML file (that is, **.html** or **.htm**) or click the **Browse** button to select the source file from the appropriate location.

In the **Document Name** text box, specify the name that you require for your HTML document. You must enter a document name, which can contain underscore characters but it cannot contain spaces. The document name length is restricted to 30 characters.

In the **Description** text box, enter the free-format text that describes your HTML document, if required.

In the **Target Window** drop-down list box, select the target attribute of the HTML document, if required. You can select one of the HTML target attributes listed in the following table.

Target	Loads the linked document into ...
_blank	A new blank window, which is not named
_media	The Media Bar (available in Internet Explorer 6 or later)
_parent	The immediate parent of the document containing the link
_search	The browser's search pane (available in Internet Explorer 5 or later)
_self	The window in which the link was clicked (the active window), which is the default target attribute
_top	The topmost window

In the **Seconds** text box, enter the number of seconds after which the Web page specified in the **URL** text box is displayed if there has been no user action.

In the **URL** text box, specify the URL of the Web page that is displayed when there has been no user action for the number of seconds specified in the **Seconds** text box. If seconds is specified but URL is not, then the request is sent back to the JADE application.

If the HTML page is secure, check the **Is this page secure?** check box. The form action and all hyperlinks are then prefixed with **https**. By default, the HTML page is not secure.

Cookies can be used to store information on the client machine. If you want to include cookie code in the HTML document, check the **Include cookie code?** check box. By default, cookie code is not included. Checking this will include the necessary script to add/change/remove cookies.

The second page of the wizard enables you to specify the class name, its superclass, and to change any JADE property names. An example of the second page of the HTML Wizard is shown in Figure 3.

In the **Class Name** text box, specify the JADE class name that you require for the HTML document. The name must be unique within the schema to which it is being added.

In the **Superclass** list box, select the **JadeHTMLClass** or one of its subclasses that you require for the superclass of your HTML document class. You can add your HTML document class only as a subclass of class of **JadeHTMLClass** or one of its subclasses.

In the **Jade Name** column, change the name of each of the JADE properties that you require, by clicking in the appropriate cell and specifying the valid JADE property name that meets your requirements. You can change values only in the **Jade Name** column.

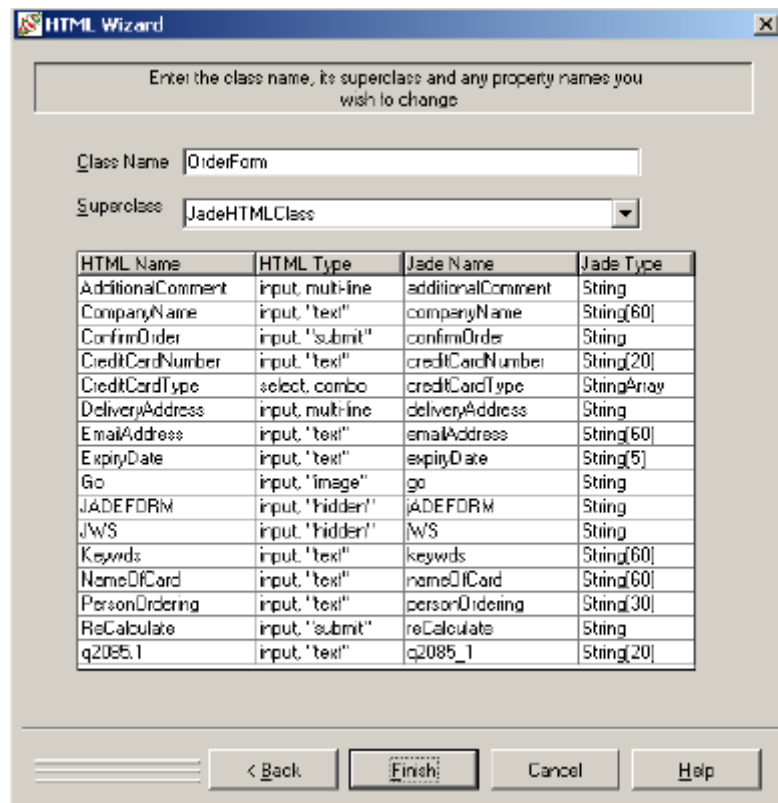


Figure 3

The following table lists the HTML tags that cause a JADE property to be created and the JADE types to which they are converted.

HTML Type	JADE Type
Input, radio (true, if set)	Boolean
Input, checkbox (true, if set)	Boolean
Input, submit	String
Input, image	String
Input, reset	String
Input, button	String
Input, text	String
Input, password	String
Input, hidden	String
Input, file	String
Select	String

HTML Type	JADE Type
Select, multiple	StringArray
TextArea	String
JADE_TAG	String / JadeHTMLClass

Each HTML document can include any number of **JADE_TAG** tags but each tag must have a **name** attribute. If the name is not unique, the same value is set for the tags. **JADE_TAG** tags can have one of the following formats.

```
<JADE_TAG name=headerTag>
```

This form of the **JADE_TAG** tag creates a property of type **String**. At generation time, this tag is replaced by the value of the property. Note that if this value is an HTML string that contains **Input** type tags, there is no equivalent property for the class and it is your responsibility to process any information that is returned for these tags.

```
<JADE_TAG name=includeHeader value=Header>
```

This form of the **JADE_TAG** tag creates a property of type **JadeHTMLClass**. The **value** attribute should resolve to a subclass of the **JadeHTMLClass** class. Use this mechanism to insert one HTML page into another. For example, if your system has a common header, create an HTML file containing only this header information and import this file into JADE. You can then insert this document into another imported document by using the **< JADE_TAG name=includeHeader value=Header>** syntax. At run time, the HTML will be inserted into the main page.

```
<JADE_TAG name=sayHelloDynamic _executeCode=JADE code>
```

This form of the **JADE_TAG** tag creates a property of type **String**. At generation time the JADE code in **JADE code** is compiled and executed at run time. This code must always have a return value. The return value is converted to a string and added to the output string that is to be returned to the browser. If the code does not compile, then nothing is output.

```
<JADE_TAG name=sayHelloStatic
_executeMethod=className::methodName>
```

This form of the **JADE_TAG** tag creates a property of type **String**. At generation time the method specified in **className::methodName** is executed. This code must always have a return value. The return value is converted to a string and added to the output string that is to be returned to the browser. If the method does not exist, is uncompiled or in error, then nothing is output. The receiver of this method will always be a transient instance of the class determined by the class name. This instance is created and deleted once the method execution is complete. If a transient instance of the class cannot be created, then nothing is output.

Notes:

If a **JADE_TAG** tag is to be used as an attribute of another tag, the **JADE_TAG** tag must be enclosed in single quote characters (' ') or in double quote characters (" "). For example, when using this tag with the <A> tag, you could specify the following.

```
<A href = "<JADE_TAG name=updateAccount>"> Edit </A>
```

At run time, the value of the **updateAccount** property is used to generate the HTML code; for example, the following is generated if **updateAccount := http://www.jadeworld.com**; in your JADE code.

```
<A href = "http://www.jadeworld.com"> Edit </A>
```

Changing an HTML Document

You can modify the name and characteristics of an existing HTML document in the current schema. When you modify an HTML document, you can change the document name but you cannot change the class name. The HTML Wizard is used to make the changes.

Editing the HTML

You can modify the HTML for an existing document. When you edit the HTML, you are editing the HTML in the JADE database, not the original file from which it was loaded.

The **Edit HTML** command displays the HTML editor window, shown in the following diagram:

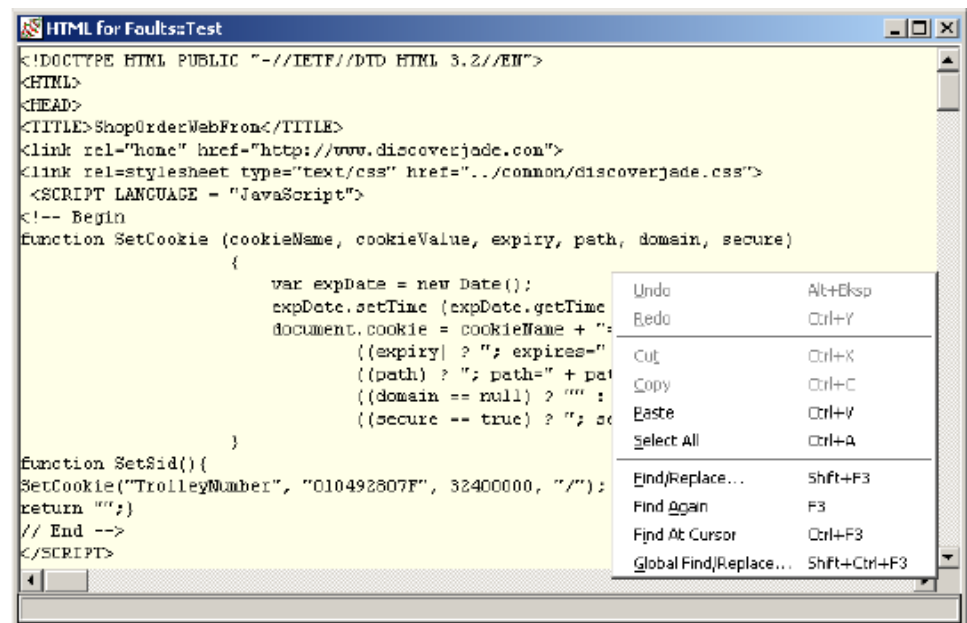


Figure 4

Edit the HTML to suit your requirements and save it. Saving the HTML will re-parse the HTML and add, change or remove property definitions, as required.

When a property is changed or removed, methods that reference these properties will be recompiled.

Extracting and Loading an HTML Document

You can extract HTML documents individually (that is, a partial schema definition) or as part of a complete schema extract into schema (.**scm**) and forms definition (.**ddb**) files.

When loading a file that contains HTML documents, the internal structures are rebuilt by using the extracted information that you load. Existing HTML documents are replaced by incoming documents in the load for documents that have the same name.

Extracting a Specific HTML Document

The **Extract** command from the HTML Document menu enables you to extract an individual HTML document.

Saving an HTML Document as a File

Although the HTML document that you add, change, or edit is saved in the JADE database itself, you can save the HTML document outside the JADE database. To save the HTML document externally, select the **Save HTML** command from the HTML menu. The file name defaults to the name of the current HTML document, with an **.htm** suffix. The location defaults to the directory that contains the file from which the HTML document was created. The file that is created will contain the HTML.

Reloading an HTML Document

You can update an existing HTML document by reloading it from an external HTML file. This replaces the document, retaining common information, and removing information that does not exist in the incoming file. Use the **Reload** menu option to reload the document. This reload can cause property types to change and properties to be deleted. Methods that reference these properties will be recompiled.

Removing an HTML Document

The **Remove** command from the menu enables you to remove (delete) the HTML document that is currently selected. You cannot delete the created class or any of its properties. An HTML document class is deleted only when you delete its corresponding HTML document from the HTML Document Browser. Methods that reference this class or any of its properties will be marked as being in error.

Batch Load of HTML Documents

The `loadHTMLDocuments` method of the `Schema` class provides a mechanism for bulk loading of HTML documents. Its signature is as follows.

```
loadHTMLDocuments(pathName: String;  
                  reload: Boolean): String;
```

The method processes all files in the folder specified in the `pathName` parameter, to enable you to load or reload multiple HTML documents. The names of the HTML documents are obtained from the file names, excluding the extension and the path. For example, if the file name is `c:\documents\header.htm`, the document name is **Header** (with an uppercase first character). If this results in the name being longer than 30 characters, it is truncated to 30 characters. If a document already exists in the schema and the `reload` parameter is set to `true`, the document is updated. If the document exists and the `reload` parameter is `false`, then the document is not updated. If a document does not exist in the schema, the document is created and updated.

An exception is raised if the `pathName` parameter contains a null value or the specified directory cannot be located. As no validation is done to determine if the file is a valid HTML file, it is your responsibility to ensure that the files in the specified folder are valid HTML documents.

Defining The Application

Once all the documents have been loaded, the application needs to be set up. Figure 5 shows the Application Options dialog.

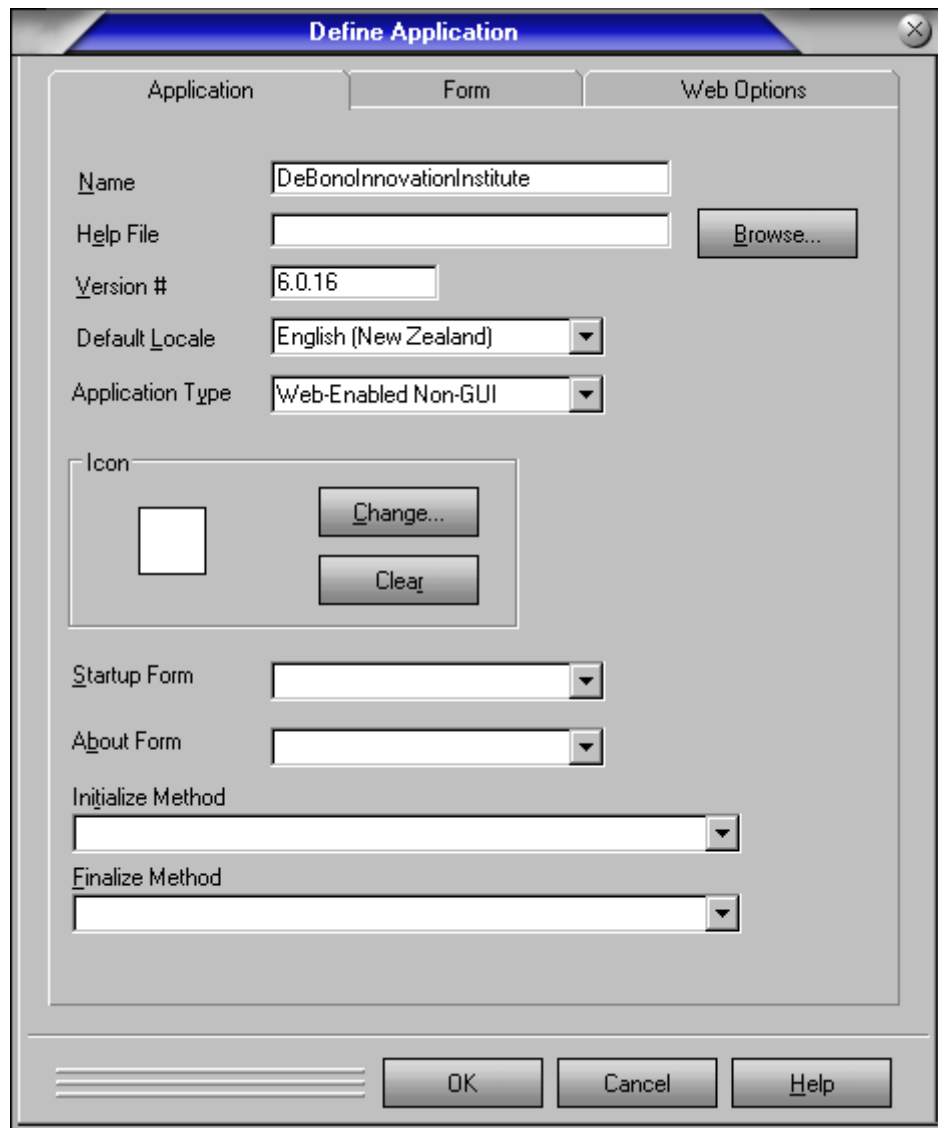


Figure 5

The application type can be **Web-Enabled** or **Web-Enabled Non-GUI**. If **Web-Enabled** is selected, the web application monitor window is displayed when the application is launched. If **Web-Enabled Non-GUI** is selected, the monitor window is not shown. If the application type is Web-Enabled Non-GUI, then the application can be run on any of the JADE supported platforms.

The startup form must be left blank. If a startup form is specified, JADE assumes that the application uses JADE forms and will use dynamic generation to display the Web pages.

All other options are as per a standard JADE application. In addition to this change, there is also a **Web Options** sheet where HTML Thin Client-specific options can be set, as shown in Figure 6. The **JADE Forms** sheet is disabled (when the startup form is null).

Figure 6.

By default communication between the JADE application and the Web server (via jadehttp) is by using a named pipe. Note that named pipes only work with IIS. The pipe name defaults to the application name. If you wish to use TCP/IP, in the **Connection Name** text box you enter the host name or host IP address followed by a : followed by the starting port number. For example, if the machine name was **cnwchcs36** and the starting port number was **50000**, then the following would be entered:

```
cnwchcs36:50000
```

Application copies determines the number of copies of the application that will be started up when the application is run. When using TCP/IP, the first copy will open a port with the specified port number and each subsequent copy will have the number incremented by 1. If 5 copies were therefore started up, then the port numbers for the above example will be 50000, 50001, 50002, 50003, and 50004.

As the **jadehttp** module needs this information as well in order to communicate with the JADE application, and the module can be outside a firewall, the information must also be present in the **jadehttp.ini** file. See the section “Jadehttp Ini File Settings”, later in this document, for further information.

The **session timeout** parameter, if specified, will start a timer for the session when a response is sent back to the browser and if no request is received within the given time period, the session is removed from the list of active sessions. If the user tries to reconnect after the timeout period, a default message is sent back to the browser informing the user that the session had timed out. You can override the default message. See “User Exits”, later in this document, for more. The default setting of zero (0) implies that there is no timeout, that is, the session will stay connected until the application is terminated.

Note that when a session is removed, either programmatically by calling the **WebSession::removeSession** method or automatically by using the session timeout parameter, an HTML Thin Client license is freed. For security and resource usage reasons, it is recommended that a reasonable timeout parameter be set.

The **Minimum Response Time** value, if specified, will start a timer for the session when a request is received from the browser. If the HTML generation does not complete within this time, a default message is sent back to the browser informing the user that the response time has been exceeded. You can override the default message. See the section “User Exits”, later in this document, for more. The default setting of zero (0) implies that there is no minimum response time; that is the processing will continue until it completes. Note that if this processing takes more than 5 minutes, the **jadehttp** module will time out and the connection will be lost. In unusual circumstances where the processing is going to take more than 5 minutes, a background application that does all this processing should be started and a response should be sent back to the user immediately. The user can then choose to do some other operation (or wait) and when the background application has completed, the user should be informed of this or the results of the processing sent back to the browser on the next response cycle. Alternatively, an auto refresh page can be set up to inform the user of the progress of the operation. It is generally not a good idea to hold up the main application thread for any length of time, as there could be other requests queued.

Certain operations (for example, when shutting down the HTML Thin Client application) can cause a message box to be displayed if there are active sessions. The **Disable Messages** value, if checked, will cause these message boxes not to be displayed. In the case of a shutdown, for example, the application will go away silently.

The home page of the Web application must be specified by selecting an entry from the **Home Page** combo box.

The **machine name** and **virtual directory**, if specified, are used for dynamic generation of hyperlinks as well as for the **POST** method. If these are not specified, the machine name and virtual directory are obtained from the Web Server or initialization file settings. These attributes can also be set programmatically. See the sections “Jadehttp Ini File Settings” and “Programmatic Interfaces”, later in this document, for further information.

HTML Thin Client Framework

The framework consists of the following.

- Jadehttp
- Connection Classes
- Web Session Manager
- Web Session
- Web Application Monitor

Jadehttp

The Web server invokes **jadehttp** when a Web browser user connects to an HTML Thin Client application. Its principal functions are as follows.

- Handling requests from a browser. As no transient state is held between requests, the module finds any available connection and sends the information to the JADE application. If there are no free connections, the request is queued until one is available.
- Handling file uploads using the MIME-compliant content type multipart/form-data. The module receives the file contents, and depending on the firewall setting, it writes directly to the transfer directory (**firewall=false**) or sends it through to the JADE application where the framework assembles this file and writes it to the appropriate directory. Note that users who do not use the standard framework may need to handle this in their code.

- The module also provides a way of initiating extra logging for debugging purposes and statistics by way of the **Status_Request** message on a browser request.

The module will be referred to as **jadehttp** throughout this document, although its name can be changed. One reason for changing its name would be, in the case of IIS, to allow multiple copies of the module to be attached to the same Web server. This is useful, for example, where the one server is servicing both a development and a UAT environment.

Connection Classes

The **Connection** classes hierarchy in the **RootSchema** is defined as follows:

```
Connection
  INFOConnection
  NamedPipe
    InternetPipe
  TCPIPConnection
    JadeInternetTCPIPConnection
```

The classes of interest are **InternetPipe** and **JadeInternetTCPIPConnection**. Depending on the application setting, transient instances of **InternetPipe** (when named pipes are to be used) or transient instances of **JadeInternetTCPIPConnection** (if TCP/IP is to be used) are created, setting up as the callback function the **openPipeCallback** method. When **jadehttp** opens the pipe, this method is called, which then sets up the **readPipeCallBack** method as the callback function for reading input. When the input arrives, it is read and processed by this method.

The **sendReply** method is used for sending responses back to the browser.

Web Session Manager

The Web Session Manager is an internal object that is created when the first HTML Thin Client application is started. This object keeps track of web sessions and ensures that the correct Web session object is retrieved for a given request.

Web Session

A **WebSession** instance is created for every new Web session. When this instance is created, a unique identifier is created for it and this identifier is transferred with the Web page as a means of retrieving the **WebSession** instance when the page is returned from the browser. The **WebSession** instance is an instance of the **WebSession** subclass for that schema. You can add properties to save state information for a Web session and methods to do specialised processing, if required. A special system variable, **currentSession**, will always refer to the currently active Web session. For non-HTML Thin Client applications, this system variable will be set to **null**. This variable can therefore be used to test for the presence of HTML Thin Client in code.

As the **WebSession** instance is a persistent instance, all property updates must be done in transaction state. This differs from the HTML Thin Client JADE Forms implementation.

Note No transient state is maintained between requests. Each request may be serviced by any of the processes that are running. If you therefore create transient instances, they will not be available between requests (unless you only have one process running).

Web Application Monitor

Figure 7 shows details about the browser and the contents of the request string that are returned from the browser.

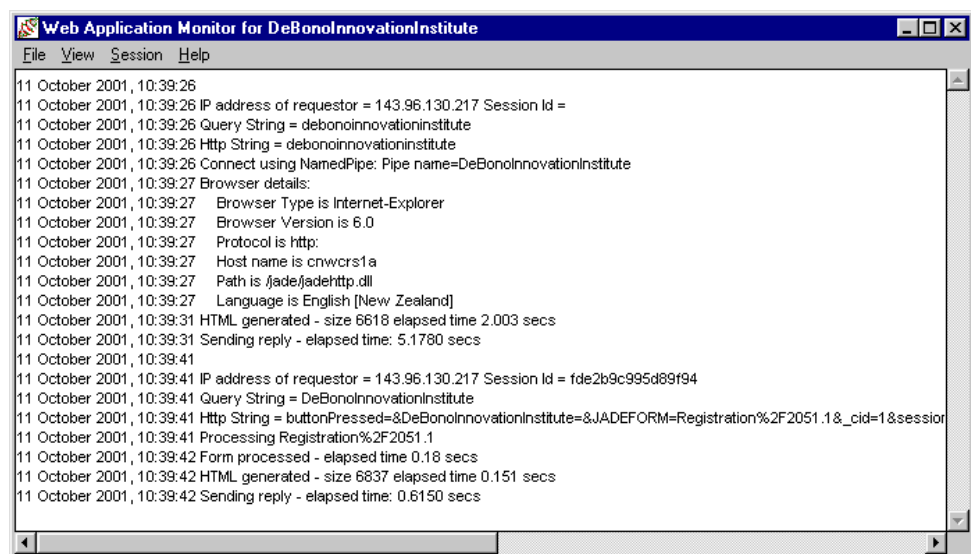


Figure 7

The Web Application Monitor window is the window that is shown when an HTML Thin Client application is run (when the application type is set to Web-Enabled). The window shows details about Web sessions in progress. Currently active sessions can be viewed or removed from this window.

Performance Considerations

Web Application Monitor

The Web Application monitor window records all request and response details. Because it is appending text to the window, it is a relatively slow operation. The display should be turned off for production systems and used only for debugging purposes. This can be turned off using a menu item on this window or using an initialization file setting (see “HTML Thin Client Ini File Settings” later in this document). Running the application as a **Web-Enabled Non-GUI** application is another option.

Large Output

When executing a user query, for example, it can produce a large result set. Rather than displaying all of the results in one page, which can incur significant generation time, only display a subset of this result, providing **Next** and **Back** buttons or alternatively use the Page Bounce feature to update the page with the progress of the request.

Application Copies

For better throughput, run multiple copies of the application. This will obviously only be of real benefit when running in a multi-processor machine.

User Exits

The HTML Thin Client framework allows you the freedom to use all, some or a minimal subset of its features to implement a Web-enabled JADE application. For example, you can subclass the **Connection** subclasses, reimplement some or all of the methods and have full control over the communication with **jadehttp** and not use any of the other features at all.

In this section, the discussion centres around you being able to provide user hooks as part of the standard framework.

JadeHTML Class Methods

If you want full control of the HTML pages in your applications, you can then reimplement the appropriate **JadeHTMLClass** methods to suit your requirements by reimplementing the **generateHTMLString**, **processRequest**, **reply**, and **updateValues** methods for every **JadeHTMLClass** subclass. These methods are described in the “JadeHTMLClass Definition” section.

Connection Methods

Similarly at a higher level, the **InternetPipe** or the **JadeInternetTCPIPConnection** classes or both, can be subclassed and the **readPipeCallback** and the **sendReply** methods can be reimplemented (not recommended if you are using the rest of the standard framework, unless **inheritMethod** is called).

_executeMethod

Another user exit is provided so that if one of the parameters returned from the browser has a name of **_executeMethod**, the framework assumes that a user method is to be executed. The value of this parameter must have the format **<class name>::<method name>**, where method name has to be prefixed with **web_** (to prevent the execution of unauthorized methods). This user exit can be used, for example, to generate XML data to return to a client browser that made this request using scripting. Executing code this way bypasses the framework’s session handling so the **currentSession** system variable will be null.

If this method does not exist, a default error page is returned. This error page can be customized by reimplementing the **Application::executeMethodNotFoundMessage** method in the current schema's application subclass.

Programmatic Interfaces

JadeHTMLClass Definition

The **JadeHTMLClass** class implements the behavior required to support HTML pages in your JADE applications. A subclass of the **JadeHTMLClass** class is automatically created in your schema when you import an HTML document into JADE using the HTML Wizard.

JadeHTMLClass Properties

Name	Type	Description
goToPage	HTMLClass	This property contains the next page to send to the Web browser. Set this property to a JadeHTMLClass subclass.
httpString	String	This property contains the request string from the Web browser.

JadeHTMLClass Methods

Method	Description
addCookie	Adds cookie information on the client node for the HTML page when the HTML string is generated
addHiddenField	Dynamically adds hidden input tags to the HTML page
addOptionTag	Returns the HTML representing an option tag for a <select> tag
addPageBounce	Returns a string containing page bounce code
buildFormAction	Returns the action attribute for a form tag
buildLink	Generates a hyperlink
clearCookies	Removes all cookies from the HTML page
generateHTMLString	Generates and returns the HTML string based on the information saved when the page was imported
getAttributes	Returns the attributes that have been set up for a specified property on the HTML page
getCookies	Returns all of the cookies set for the current session
getHttpValue	Processes the httpString property value and returns the value for the specified name
processRequest	Processes the input from the Web browser and sets the properties to the appropriate values
queryIncludePage	Determines if the included page is inserted into the HTML string
reply	Sends the response back to the Web browser

Method	Description
setAttributes	Allows additional attributes to be set on an input tag prior to the HTML generation
tableAddData	Returns a <td> tag with attributes and data set appropriately
tableAddInput	Returns a <td> tag with attributes, input type, and values set appropriately
tableAddItem	Sets up data for a table, with all data for a table row specified in the value string
tableBegin	Returns a string containing the opening tag for a table
tableBeginRow	Returns a string containing the HTML for the row tag
tableEnd	Returns a string containing the HTML for the end table tag
tableEndRow	Returns a string containing the tag for the end of a table row
updateValues	When reimplemented, sets up the required property values
wasPageBounced	Specifies whether the HTML page was bounced to another Web page

Application Methods

The following methods defined in the **Application** class are for detecting certain error conditions. All of these methods can be reimplemented.

Method	Purpose
executeMethodNotFoundMessage	This method is called when JADE receives an _executeMethod request and the method does not exist. The following default response is returned to the browser. <i>Method Not Found</i> <i>The method that you are trying to execute is not available or is invalid. Please try a different operation.</i>
invalidWebSessionMessage	This method is called when a request is made with a session id that is not valid. The following default response is returned to the browser. <i>Invalid Web Session</i> <i>The session id is no longer valid. Please refresh your browser.</i>
licencesExceededMessage	This method is called when there are no more licences to start up a new HTML Thin Client session. The following response is returned to the browser: <i>Number of Licenses Exceeded</i> <i>Please try again later.</i>

Method	Purpose
minimumResponseTimeExceededMsg	This method is called when the processing of the request exceeds the minimum response time specified in the application options. The following default response is returned to the browser. <i>Minimum Response Time Exceeded</i> <i>The allowed response time has been exceeded.</i> <i>Please retry your operation.</i>
htmlPageNotFoundMessage	This method is called when a requested document could not be found. The following default response is returned to the browser. <i>Page Not Found</i> <i>The page that you are trying to load is not available or is invalid.</i>
timedOutSessionMessage	This method is called when the time difference between the time of the last request and the time of the current request exceeds the timeout specified in the application options. The following default response is returned to the browser: <i>Session Timed Out</i> <i>Your session has timed out. Please sign on again.</i>
removeSessionMessage	This method is called when a session has been removed programmatically. When the next request for this session is made, the following default response is sent to the browser. <i>Session Ended</i> <i>Your session has now ended.</i>
The following methods cannot be reimplemented:	
getWebMachineName	Returns a string representing the machine name as set in the Application Options dialog, from the ini file or programmatically using setWebMachineName
setWebMachineName	Sets the machine name
getWebVirtualDirectory	Returns a string representing the virtual directory as set in the Application Options dialog, from the ini file or programmatically using setWebVirtualDirectory
setWebVirtualDirectory	Sets the virtual directory

Futures

Web Services

Support for Web Services will be incorporated in the next release of JADE.

Appendix A: Apache Server Connections

Your connections from the Apache HTTP Server to JADE applications can only be via a TCP/IP connection. The design and configuration of the **mod_jadehttp** module fully conforms to Apache HTTP Server 2.0 module usage and configuration practices.

JADE supports the Apache HTTP Server by supplying **mod_jadehttp** for Linux, AIX, and Windows operating systems. Only TCP/IP connections are supported for all versions of **mod_jadehttp**, regardless of the operating system. The **mod_jadehttp** library module supplied by JADE for the Apache HTTP Server implements support for the following Multi-Processing Modules (MPMs). The selection of the default MPM is based on the operating system on which the Apache server is running, as follows.

- **mpm_winnt**, which is the default for Microsoft-based operating systems.
- **prefork**, which is the default for UNIX-based operating systems (that is, Linux and AIX).

If you build the Apache server from source code, you can build and select whichever MPM you want to use, on the operating system of your choice. Some relevant MPMs with respect to JADE and the **mod_jadehttp** library are listed in the following table.

Multi-Processing Module (MPM)	Implements a ...
mpm_winnt	Multiple-threaded single process Web server
prefork	Non-threaded, pre-forking Web server
worker	Hybrid multiple-threaded multiple-process Web server

When using the **mpm_winnt** MPM or IIS and the **jadehttp** library to access JADE applications over the World Wide Web, only one connection from the Web server to each JADE application is opened. When a new user accesses the Web site, a connection to an available JADE application is allocated and all HyperText Markup Language (HTML) traffic between the user and the JADE application uses this specific connection (that is, it is a multiple-threaded single process). This allows the JADE application logic to open and manage a single network connection for each running application instance.

The Apache **prefork** and **worker** MPMs are implemented using multiple operating system processes. Because there are different processes, they cannot share the same connection to JADE. In addition, each time a user causes data to be sent from the MPMs Web browser to the Web server, it may be handled by a different MPM process.

The **mod_jadehttp** module ensures that the same destination IP address and port number combination are used for a specific logical connection, by using the hidden fields that both the **jadehttp** and the **mod_jadehttp** libraries insert into the HTML data. JADE applications therefore support multiple connections.

Appendix B: Jadehttp Initialization File

[*application-name*] Section

To ensure that an application specified in a Web browser cannot cause an attachment to a non-JADE environment within Windows NT or Windows 2000, you must specify the name of the Web application to which users can connect, by defining an [*application-name*] section for each Web-enabled JADE application. **jadehttp** attempts to attach only to the applications whose names are specified as section names in the **jadehttp.ini** file.

The [*application-name*] section of the **jadehttp.ini** file contains parameters that define TCP/IP connections that are used instead of the default named pipe connections. To do this, the [*application-name*] section can contain the following parameters.

FirstTcpPort[*n*]

This parameter is an integer that specifies the first valid TCP/IP port number for connection to the JADE application.

GroupSharesConnections

This parameter is a Boolean that specifies whether all connections in a group are considered equal so that Web browser replies are directed to any connection within a connection group. The **jadehttp** module inserts hidden fields into the reply but any connection in the identified group can be used. This setting cannot and must not be used for a standard JADE Web-generated application.

LastTcpPort[*n*]

This integer parameter specifies the last valid TCP/IP port number for connection to the JADE application.

MaximumPipes

This integer parameter specifies the maximum number of named pipe connections to the JADE application from **jadehttp**.

MaxQueueDepth

This integer parameter specifies the maximum number of entries that can be queued at any time. The default value of zero (0) indicates that there is no queue limit.

When you specify a maximum number of queued entries, any requests that would also be queued when the maximum queued entries is reached will be rejected. The user will be sent the contents of a file named **busy.htm** from the same directory as the **jadehttp.ini** file.

If the **busy.htm** file is not available, the following text response will be sent.

```
The application is too busy
This request cannot be processed at this time due to
heavy usage - please try again shortly.
```

TcpConnection

This parameter specifies a valid TCP/IP address or machine name of the port to connect to on the server (remote) node, to enable Web client nodes to connect to the specified server node across the network through a TCP/IP connection.

VirtualDirectory

This parameter enables you to specify a string containing the physical name of the virtual directory for the HTML-enabled application. The value specified for this parameter is used if the **fileName** parameter value of the **WebSession::createVirtualDirectoryFile** method does not include a directory. If neither this parameter nor the file name in the **createVirtualDirectoryFile** method contains a directory, the files are written into the same directory as the **jadehttp** module. This parameter is only used when the **firewall** parameter is set to **true**.

[Jadehttp Files] Section

This section of the initialization file contains parameters that control firewall security and the location of files uploaded from a Web-enabled application.

Caution To prevent malicious use of files uploaded to Web-enabled applications, the files are removed as soon as the event that resulted in their upload has completed. You should therefore process the file immediately or move it into a directory that is not available from the Web, if you require that file for future processing.

FileTransferDirectory

This parameter specifies the directory to which files are uploaded during Web sessions from JADE applications that accept file input in text boxes on Web pages.

Note This parameter applies only when the **Firewall** parameter is set to **false**.

Firewall

This parameter controls whether firewall separation is used when transferring files to JADE from Web browsers or the reverse. When this parameter is set to the default value of **false**, files are written to the transfer directory. When this is to **true**, then the file contents are passed in chunks back to the Jade application and the file is recreated on the client.

Sample jadehttp.ini File

```
[Jadehttp Files]
FileTransferDirectory = c:\jade52\bin_jadehttp\transfer
Firewall              = true

[DeBonoInnovationInstitute] ← Application name
GroupSharesConnections = true
VirtualDirectory = c:\web\debono\files
MaxQueueDepth = 0
MaximumPipes = 10 // ignored, as TCP/IP connections are
                  // defined
TcpConnection      = cnwchcs63
FirstTcpPort       = 6001
LastTcpPort        = 6010
TcpConnection2     = cnwchcs64
FirstTcpPort2      = 6001
LastTcpPort2       = 6010
TcpConnection3     = cnwchcs65
FirstTcpPort3      = 6010
LastTcpPort3       = 6019
```

This example provides connections to three hosts for the **CustomerWeb** application, as follows.

- cnwchcs63 ports 6001 through 6010
- cnwchcs64 ports 6001 through 6010
- cnwchcs65 ports 6010 through 6019

[Jadehttp Logging] Section

The [Jadehttp Logging] section of the initialization file controls logging by **jadehttp** and can contain the following parameters.

Trace

This Boolean parameter controls logging by **jadehttp**.

You can use this parameter to debug data passing through the **jadehttp**.

TraceFile

This parameter specifies an optional path and a file name that overrides the default **jadehttp.log** file name so that log files can be placed in other directories.

Sample [Jadehttp Logging] Section

```
[Jadehttp Logging]
Trace      = true
TraceFile = c:\jade52\bin_jadehttp\logs\jadehttp.log
```

Appendix C: Configuring Apache

Only TCP/IP connections can be used for connections to JADE applications running under a Windows, Linux, or AIX operating system from the Apache HTTP Server.

Apache Configuration Directives

The configuration directives in this section are available to configure the JADE **mod_jadehttp** module. It uses the standard Apache configuration syntax, rather than implementing configuration via an initialization file.

Most of the directives specific to **mod_jadehttp** have the same meaning as their IIS counterparts as specified in the **jadehttp.ini** file.

These configuration directives are as follows.

The following **mod_jadehttp** directives apply between the core Apache **<Location>** and **</Location>** directives. Apache processes the **<Location>** directives in the order in which they are found in the configuration file, following any **<Files>** and **<Directory>** sections and after **.htaccess** files have been read and processed.

```
<Location /jade>
SetHandler jadehttp-handler
Application WebApplication
</Location>
```

This allows **mod_jadehttp** directives to be placed in higher locations and their values to become the defaults for lower locations unless you explicitly override them. In the following example, the handler and **FaultDocument** specified in **<Location /jade>** also apply to **<Location /jade/subloc>**.

```
<Location /jade>
SetHandler jadehttp-handler
FaultDocument PostTooLarge "http://server/ohmama.html"
</Location>
<Location /jade/subloc>
Application WebApplication
</Location>
```

SetHandler

The standard **SetHandler** Apache directive, whose characteristics are listed in the following table, selects the **mod_jadehttp** module to handle this location.

Characteristic	Value
Action	Forces all matching files to be processed by a handler
Syntax	SetHandler <i>handler-name</i>
Context	server config, virtual host, directory, .htaccess
Module	Core
Examples	SetHandler jadehttp-info and SetHandler jadehttp-handler

The **jadehttp-handler**, which is the primary handler inside the **mod_jadehttp** module, formats and passes data to JADE and returns the results to the Web browser. The **jadehttp-info** handler reports internal information about the configuration and current status of the **mod_jadehttp** module back to the Web browser. This handler is similar to the standard Apache **mod_info** module that provides a comprehensive overview of the server configuration. Access to locations that use this handler should be restricted to trusted sites.

JadeHttp_Trace

The **JadeHttp_Trace** directive, whose characteristics are listed in the following table, controls where any **mod_jadehttp** tracing is written.

Characteristic	Value
Action	Specifies the log file to which JADE trace messages are sent
Syntax	JadeHttp_Trace <i>bit-mask</i> [<i>file-name</i> [<i>size</i> [<i>K</i> <i>M</i>]]]
Context	Server config, virtual host, Location
Module	mod_jadehttp
Example	JadeHttp_Trace 0x000000ff logs/jade.log 10M

The **JadeHttp_Trace** directive is similar to the **Trace** and **TraceFile** parameters in the [Jadehttp Logging] section of the **jadehttp.ini** file, discussed earlier in this document.

If the *file-name* value is relative, it is created relative to the **ServerRoot** location. If you do not specify the size, it defaults to 5M bytes. The *size* value can have a multiplier appended to it. The **K** multiplier multiplies by 1,024, the **M** multiplier multiplies by 1,048,576 bytes, and all other values are treated as bytes. The bit set values for the *bit-mask* value are listed in the following table.

Bit Set	Description
0x00000001	When a redirect occurs
0x00000002	When a special command is issued

Application

The **Application** directive, whose characteristics are listed in the following table, specifies the name of the Web-enabled JADE application.

Characteristic	Value
Action	Specifies the JADE application to use
Syntax	Application <i>application-name</i>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	Application ErewhonShop

This directive is similar to the functionality of the [*application-name*] section of the **jadehttp** library module for IIS, described earlier in this document.

LocalInterface

The **LocalInterface** directive, whose characteristics are listed in the following table, specifies the local network interface and optionally the port number to use when connecting from the **mod_jadehttp** module to the JADE Web-enabled application.

Characteristic	Value
Action	Specifies the TCP/IP connection to JADE
Syntax	LocalInterface <i>handler-name</i>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	LocalInterface 0.0.0.0
Example	LocalInterface 10.1.1.1

If your Web server host has multiple network interface cards, you can specify which one **mod_jadehttp** uses when initiating connections to the Web-enabled JADE application. The default value of **0.0.0.0** allows the operating system to choose the network interface card. You would normally require this directive only when there are specific network security or routing issues to be addressed.

Caution: Although you can also specify the local outgoing port number, you should take great care when doing so and only when you have a specific requirement. Specifying a **LocalInterface** port number limits you to a maximum of one TCP/IP connection to JADE for each Apache **<Location>** directive. In addition, if the Web server terminates abnormally and the port is not properly closed, it may take several minutes before this port becomes available again for use.

TcpConnection

The **TcpConnection** directive, whose characteristics are listed in the following table, specifies the TCP/IP connection details between the **mod_jadehttp** module and the Web-enabled JADE application.

Characteristic	Value
Action	Specifies the TCP/IP connection to the JADE application
Syntax	TcpConnection[<i>n</i>] <i>host-name</i> <i>first-port</i> [<i>last-port</i>]
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Examples	TcpConnection server1 6008 and TcpConnection1 10.1.1.99 51030 51039

The **TcpConnection** directive is similar to the combined values of the **TcpConnection[*n*]**, **FirstTcpPort[*n*]**, and **LastTcpPort[*n*]** parameters in the [*application-name*] section of the **jadehttp.ini** file for IIS servers, described earlier in this document.

You can specify the *host-name* value as a host name or as an IP address. If you specify a host name, it must resolve to a single IP address. If you append the optional *n* value to the **TcpConnection** directive, it indicates the group to which the specified host name port number range is assigned. If you do not specify the optional *n* group suffix, JADE treats the value as **1**. The maximum value is **9**. When you set the **GroupSharesConnection** directive to **true**, all connections within a group are treated equally.

GroupSharesConnection

The **GroupSharesConnection** directive, whose characteristics are listed in the following table, specifies that HTML input from the Web browser can be sent to any TCP/IP connection within the same group.

Characteristic	Value
Action	Specifies the JADE application to use
Syntax	GroupSharesConnection true false
Default	GroupSharesConnection false
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	GroupSharesConnection true

The **GroupSharesConnection** directive is similar to the **GroupSharesConnections** parameter in the [*application-name*] section of the **jadehttp.ini** file for IIS servers, described earlier in this document. A group is defined by specifying all **TcpConnection** directives with the same numeric suffix.

PhysicalDirectory

The **PhysicalDirectory** directive, whose characteristics are listed in the following table, specifies the name of a local directory to use when files are transferred from JADE to the Web server by using the **WebSession** class **createVirtualDirectoryFile** method without specifying a directory in the **fileName** parameter.

Characteristic	Value
Action	Specifies the directory to use as a physical directory
Syntax	PhysicalDirectory <i>local-directory</i>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	PhysicalDirectory /var/tmp/appname

The **PhysicalDirectory** directive is similar to the **VirtualDirectory** parameter in the [*application-name*] section of the **jadehttp.ini** file for IIS servers, described earlier in this document.

PurgeDirectoryFrequency

The **PurgeDirectoryFrequency** directive, whose characteristics are listed in the following table, specifies how often the physical directory is checked to see if files require purging.

Characteristic	Value
Action	Specifies the frequency at which the physical directory is purged
Syntax	PurgeDirectoryFrequency <i>time-unit</i> [<i>H</i>] <i>[M]</i>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	PurgeDirectoryFrequency 1H
Example	PurgeDirectoryFrequency 5H

You can append a multiplier to the *time-unit* value. The **H** multiplies by 3,600, the **M** multiplier multiplies by 60, and all other *time-unit* values are treated as seconds. Specify zero (**0**) for the *time-unit* value to turn off the purging of the physical directory. The minimum that you can specify for the *time-unit* value is **5** minutes and the maximum is **24H** (hours). Values outside this range are forced to their respective limits.

PurgeFileAge

The **PurgeFileAge** directive, whose characteristics are listed in the following table, specifies the length of time since a file was last modified before it is purged from the physical directory.

Characteristic	Value
Action	Specifies the minimum modified age before the files are purged
Syntax	<code>PurgeFileAge <i>time-unit</i>[D H M]</code>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	PurgeFileAge 12H
Example	PurgeFileAge 1D

You can append a multiplier to the *time-unit* value. The **D** multiplier multiplies by 86,400, the **H** multiplier by 3,600, the **M** multiplier multiplies by 60, and all other *time-unit* values are treated as seconds. Specify zero (**0**) for the *time-unit* value to turn off the purging of the physical directory. The minimum that you can specify for the *time-unit* value is **30** seconds and the maximum is **35D** (days). Values outside this range are forced to their respective limits.

MaxMessageSize

The **MaxMessageSize** directive, whose characteristics are listed in the following table, specifies the largest size of a **POST** URL before an error is generated.

Characteristic	Value
Action	Specifies the maximum size of a POST string
Syntax	<code>MaxMessageSize <i>size</i>[M K]</code>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	MaxMessageSize 1M
Example	MaxMessageSize 50K

You can append a multiplier to the *size* value. The **M** multiplier multiplies by 1,048,756, the **K** multiplier multiplies by 1,024, and all other values are treated as bytes.

Firewall

The **Firewall** directive, whose characteristics are listed in the following table, specifies whether there is a firewall between the **mod_jadehttp** module and the JADE application.

Characteristic	Value
Action	Indicates if there is a firewall between mod_jadehttp and JADE
Syntax	Firewall True False
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	Firewall True

The **Firewall** directive is similar to the **Firewall** parameter in the [[Jadehttp Files](#)] section of the **jadehttp.ini** file for IIS servers, described earlier in this document. If a firewall is present, file transfers between JADE and the Web server must be done by using the **mod_jadehttp** module.

FileTransferDirectory

The **FileTransferDirectory** directive, whose characteristics are listed in the following table, specifies the name of a local directory to use when files are uploaded from the Web browser to the JADE application.

Characteristic	Value
Action	Specifies the directory to use as a virtual directory
Syntax	FileTransferDirectory <i>local-directory</i>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Example	FileTransferDirectory /var/tmp/appname

The **FileTransferDirectory** directive is similar to the **FileTransferDirectory** parameter in the [[Jadehttp Files](#)] section of the **jadehttp.ini** file for IIS servers, described earlier in this document. As the **mod_jadehttp** module must put the transfer files somewhere before transferring them to JADE, this directive specifies the directory to which the file or files are written.

JadeServer

The **JadeServer** directive, whose characteristics are listed in the following table, is reserved for future implementation.

Characteristic	Value
Action	Specifies the JADE server configuration
Syntax	JadeServer <i>jade-version byte-order character-size</i>
Context	Location
Handler	jadehttp-handler
Module	mod_jadehttp
Default	JadeServer 6.0 <i>localEndian oneByte</i>
Example	JadeServer 6 Intel utf-16

When implemented, this directive allows a single **mod_jadehttp** module to connect to JADE, regardless of the combination of Web server architecture and JADE server architecture, allowing **mod_jadehttp** to adapt to the installed release of the JADE server. The *jade-version* value allows for the protocol to between JADE and the **mod_jadehttp** module to be changed. The version number can be in the range 1 through 4, with each number separated with numbers inclusive separated with the dot operator (.) notation; for example, **6.0.10.7**. The *byte-order* value allows for binary numbers and wide characters to be converted to the native byte ordering of the JADE server, which may be different from the byte ordering of the Web server. In the default example in the previous table, the *localEndian* value defaults to **little-Endian** when the **mod_jadehttp** module is running on Intel-based hosts and **bigEndian** when on PowerPC-based hosts. The entered value is not case-sensitive. The primary names and synonyms for the *byte-order* value are listed in the following table.

Primary Name	Synonym
Little-Endian	Intel
Big-Endian	PowerPC

The *character-size* value indicates the size of a JADE character, which is ANSI or Unicode, depending on your JADE installation. If the character size is Unicode where the JADE server is running, the physical size of a JADE character can be one of 1, 2, or 4 bytes in size. This allows **mod_jadehttp** to send JADE characters of the correct size and type to the JADE server.

The primary names and synonyms for the entered value, which is not case-sensitive, are listed in the following table.

Primary Name	Synonym
OneByte	ANSI
TwoByte	Unicode, Utf16, or Utf-16
FourByte	Utf32 or Utf-32

FaultDocument

The **FaultDocument** directive specifies what the **mod_jadehttp** module does when it encounters an error.

Characteristic	Value
Action	Specifies the URL or path name for redirection when an error occurs
Syntax	<code>FaultDocument <i>cause redirect-url</i></code>
Context	Location
Handler	<code>jadehttp-handler, jadehttp-info</code>
Module	<code>mod_jadehttp</code>
Example 1	<code>FaultDocument TcpConnectFailed http://secondary.example.com:80/page.html</code>
Example 2	<code>FaultDocument PostTooLarge bigpost.html</code>

You can specify the *cause* value in numeric or text form. Valid values for the *cause* are listed in the following table.

Numeric Name	Text Name
600	TcpConnectFailed
601	PostTooLarge
602	ServiceFailed
603	DataReadFailed

You must specify a valid URL or path name for the *redirect-url* value, which can be on the local Web server or on a remote Web server. If you do not specify a *redirect-url* value and an error occurs, an error page is generated internally in the **mod_jadehttp** module.

Apache Configuration Examples

In the configuration examples in this section, the following is assumed to have been added to the **httpd.conf** file, which is the default main configuration file for Apache.

```
LoadModule jadehttp_module modules/mod_jadehttp.so
Include conf/jade.conf
```

The **jade.conf** file in this example of the **httpd.conf** file is called by the **httpd.conf** file. (Alternatively, you could integrate all information from the **jade.conf** examples in the following subsections into the **httpd.conf** file instead of branching to one or more separate **jade.conf** files.)

Each of the examples in the following subsections has a different **jade.conf** file.

Minimal Configuration Example

The following is a minimal example of the **jade.conf** file that enables you to connect to the JADE example **Erewhon** system and access the Web shop.

```
<IfModule mod_jadehttp.c>
<Location /jade-info>
Set-Handler jadehttp-info
</Location>
<Location /Erewhon>
Set-Handler jadehttp-handler
Application WebShop
TcpConnection jadeserver 6007
PhysicalDirectory "/pictures"
</Location>
<IfModule>
<Directory "/pictures">
Order deny,allow
Allow from All
</Directory>
Alias /jadeImages "/pictures"
```

From your Web browser, use the **http://webserver/Erewhon?WebShop** URL to connect to the example **Erewhon** system. To find out status information about what is happening inside the **mod_jadehttp** module, use the **http://webserver/jade-info** URL.

Extended Configuration Example

Add the following to the **jade.conf** file, referred to in the **httpd.conf** file under “[Apache Configuration Examples](#)”, earlier in this document (or you can include it in the **httpd.conf** file itself).

```
<IfModule mod_jadehttp.c>
<Location /jade>
Set-Handler jadehttp-handler
FaultDocument TcpConnectFailed "/messages/CanNotConnect.html"
</Location>
<Location /jade/info>
Set-Handler jadehttp-info
</Location>
<Location /jade/BankOfJade>
Application BankApp
TcpConnection cnwjdcla 6007
</Location>
<Location /jade/Erewhon>
Application Erewhon
TcpConnection cnwjdcla 6007
</Location>
<Location /jade/Erewhon/user>
Application UserApp
TcpConnection cnwjdcla 6007 6017
</Location>
<Location /jade/Erewhon/admin>
Application AdminApp
TcpConnection cnwjdcla 6018
</Location>
</IfModule>
```

Extended Configuration Example with Additional Apache Directives

Add the following to the **jade.conf** file, referred to in the **httpd.conf** file under “[Apache Configuration Examples](#)”, earlier in this document (or you can include it in the **httpd.conf** file itself).

```
s
<IfModule mod_jadehttp.c>
<Location /jade-info>
JadeHttp-Trace 0xffff Logs/jadehttp.log 5M
Set-Handler jadehttp-handler
FaultDocument TcpConnectFailed "/messages/CanNotConnect.html"
ErrorDocument 503
"http://localhost:8080/messages/Custom503.html"
</Location>
<Location /jade/info>
Set-Handler jadehttp-info
Order deny,allow
Deny from all
Allow from 127.0.0.0/8 143.96.129.225
</Location>
<Location /jade/BankOfJade>
JadeHttp-Trace 0xff
```

```
Application BankApp
TcpConnection cnwjdc1a 6007
Order deny,allow
Deny from all
Allow from 143.96.0.0/16
</Location>
<Location /jade/Erewhon>
Application Erewhon
TcpConnection cnwjdc1a 6007
Order deny,allow
Deny from all
Allow from 143.96.0.0/16
</Location>
<Location /jade/Erewhon/user>
Application UserApp
TcpConnection cnwjdc1a 6007 6017
Order allow,deny
Allow from all
</Location>
<Location /jade/Erewhon/admin>
Application AdminApp
TcpConnection cnwjdc1a 6018
Order deny,allow
Deny from all
Allow from 127.0.0.0/8
</Location>
</IfModule>
```

Apache Considerations

When using the Apache HTTP Server to connect to your JADE applications from the World Wide Web, consider the following.

- The dynamic updating facilities of IIS and **jadehttp.dll** that change port number ranges, and so on, are not implemented in the **mod_jadehttp** module because JADE uses Apache configuration files.
- Only **JadeInternetTCPIPConnection** class TCP/IP communications are supported. This applies to all versions of **mod_jadehttp**, regardless of the operating system.
- The security implications for **mod_jadehttp** are the same as that of the IIS **jadehttp.dll**. Access to JADE data from the Web server can be controlled by Apache security directives.
- Apache and the HTTP generally use UTF-8 as the encoding scheme for Unicode data on the Web. As **mod_jadehttp** does not allow for this and passes the data directly to JADE, only ANSI data can be read or written to the **JadeInternetTCPIPConnection** object.
- As the **JadeServer** configuration directive that indicates details about the JADE application to which **mod_jadehttp** is connected is not yet implemented, it is assumed that the value of this directive is **JadeServer 6.0 littleEndian Ansi**.

Appendix D: Initialization File

The [WebOptions] section of the JADE initialization file contains parameters that enable you to specify options for your Web pages and can contain the following parameters.

DisableLogging

This parameter specifies whether information is logged in the JADE Web Application Monitor window. By default, monitor information is logged.

Set this parameter to **true** if you want to stop the display of logging information in the window (for example, when you are running an application as a service and the Web Application Monitor window View menu item cannot be used).

This setting is available in JADE 6.0 for backwards compatibility only. In JADE 6.0, the application type should be set to **Web-Enabled Non GUI**, to achieve the same result.

ImageType

This parameter specifies the type of image that is displayed on your Web pages. The valid values are the default **jpg** (Joint Photographic Experts Group) file type and the **png** (Portable Network Graphics) file type. Set this parameter to **png** if you want to use images with lossless compression on your Web pages.

LogFileName

This parameter specifies the file name and the full path to which output displayed in the Web Application Monitor is directed to for later analysis.

If the specified file name cannot be written to or it is not valid, a file called **websession.log** is created in the physical directory specified in the Application Options or in the **PhysicalDirectory** parameter.

PhysicalDirectory

This parameter enables you to specify the physical directory for Web-enabled applications. Specify this parameter if you want to override the physical directory specified in the Application Options for the application.

To specify multiple physical directories in the same JADE initialization file, prefix the **PhysicalDirectory** parameter with the application name followed by an underscore (**_**) character. For example, if the application name is **CustomerInvoice**, you would specify the following parameter and value.

```
CustomerInvoice_PhysicalDirectory=c:\customer\system
```

VirtualDirectory

This parameter enables you to specify the virtual directory for Web-enabled applications.

Specify this parameter if you want to override the virtual directory specified in the Application Options for the application.

To specify multiple virtual directories in the same JADE initialization file, prefix the **VirtualDirectory** parameter with the application name followed by an underscore (`_`) character. For example, if the application name is **CustomerInvoice**, you would specify the following parameter and value.

```
CustomerInvoice_VirtualDirectory=/customer
```

URLSpecifications | *application-name*_URLSpecifications

This parameter enables you to specify Internet server virtual directories for HTML documents for all of your Web-enabled applications or for a specific Web-enabled application. This parameter can have one of two forms, as follows.

```
URLSpecifications  
application-name_URLSpecifications
```

Use the *application-name*_URLSpecifications form of this parameter if you want different applications to have different URL specifications. The values for the *application-name*_URLSpecifications form of this parameter are as follows.

```
protocol, machine-name, virtual-directory-name
```

In this format, the *protocol* value must be **http** or **https**, the *machine-name* value is the name of the target host for hyperlinks and POST actions, and the *virtual-directory* value is the name of the virtual directory for hyperlinks and POST actions. The parameter is ignored if the *protocol* value is not **http** or **https**. The following is an example of the *application-name*_URLSpecifications form of this parameter.

```
CustomerWeb_URLSpecifications = https,JadeAppsServer,  
/customer
```

If JADE locates both forms of this, the specific form (that is, *application-name*_URLSpecifications) takes precedence. If you do not specify the **URLSpecifications** or *application-name*_URLSpecifications parameter in the JADE initialization file, the Internet server directory information is obtained from the Web server.

UseHTML4ForNetscape

The **UseHTML4ForNetscape** parameter specifies whether HTML generation for browsers that identify themselves as Netscape is similar to that for Internet Explorer. For non-Internet Explorer browsers, the version number returned must be 5 or greater. For Internet Explorer browsers, there is no change to the current implementation.

Sample [WebOptions] Section

```
[WebOptions]
CustomerInvoice_PhysicalDirectory = c:\customer\system
CustomerInvoice_VirtualDirectory = /customer
DisableLogging = false
ImageType = png
LogFileName = s:\jade52\webtest\logs\webactivity.txt
CustWeb_URLSpecifications = https,AppsServer,/customer
UseHTML4ForNetscape=true
```

Appendix E: Directory Structure

One aspect of HTML Thin Client applications that may be confusing to users new to this feature is the directory set up and its relationship to the Web server. The following brief discussion attempts to explain this.

When first initiating an HTML Thin Client application from a browser, typically the format of a URL will be entered as follows.

<http://cnwchcs36/jade/jadehttp.dll?DeBonoInnovationInstitute>

In this example:

- **http** is the protocol to use
- **cnwchcs36** is the machine name of the Web server
- **jade** is the name of the virtual directory
- **jadehttp.dll** is the ISAPI dll being invoked
- **DeBonoInnovationInstitute** is the name of the HTML Thin Client Application

When the Web server receives this request, it needs to know the physical location of the **jadehttp** module. In order for the Web server to know this, you have to set up a virtual directory mapping. With IIS, you use the Internet Service Manager to do this. If the virtual directory **jade** maps to a physical location of **u:\jade52**, this mapping must be defined to IIS. At this time, the directory security and access permissions must also be set up. The access permissions must have at least read and execute permissions. Refer to the IIS documentation for details. In the case of the Apache server, these configurations are set up in the configuration file.

Virtual directory mappings must also be set up for the virtual directory and physical directory specified in the application options or in the initialization file. The set up for this is the same as above. If this is not specified, the application uses the working directory to create the generated image files. If there is no virtual directory mapping defined in the Web server for the working directory, the browser will not be able to find the images. It is therefore important to set up this mapping.

The **jadehttp** module creates a directory with three subdirectories (if they do not already exist). It creates the directory at the same level as the directory where the module resides. The name of the directory is the name of the directory where the module resides, suffixed with the name of the module. For example, if the name of the directory where the module is located is **u:\jade52**, a directory is created with the name **u:\jade52_jadehttp**. The three subdirectories that are created under this directory are **ini**, **logs**, and **transfer**.

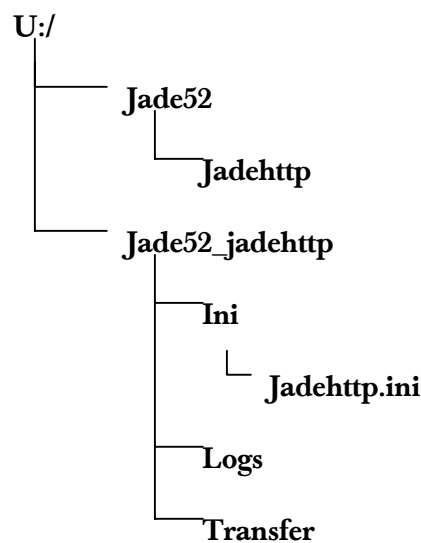
The **ini** directory contains the **jadehttp.ini** file. This is the only location where **jadehttp** module looks for this file. When an HTML Thin Client session is started from a browser, **jadehttp** checks for a section name that corresponds to the application name. If there is no such section, the connection is refused.

The **logs** directory is where the **jadehttp.log** is created. This directory will not be created or used if the [Jadehttp Logging] section contains a valid **TraceFile** name.

The **transfer** directory is where files uploaded from the web are created. This directory will not be created or used if the [Jadehttp Files] section contains a valid **FileTransferDirectory** name.

Example Directory Structure

Machine **cnwchcs36** has a drive called U. On drive U, the following is set up:



Appendix F: Typical Setup Process

Once the HTML Thin Client application has been written, the following steps (typically) will need to be taken to deploy the application onto the WWW.

1. Create a directory where the **jadehttp** module will be located and copy the module to this directory.
2. Create a directory where the image files will be located and copy any static images to this directory.
3. Start up Internet Service Manager and define virtual directory mappings for the two directories defined in the steps above, setting up directory security and access permissions as required.
4. Create a directory with the same name as the directory created in step 1, suffixed with **_jadehttp**.
5. Create a subdirectory of this directory and call this directory **ini**.
6. Bring up Notepad and create a file **called jadehttp.ini** (*not jadehttp.ini.txt*) and add a section with the name of your application. If you are using TCP/IP, you will need to specify **TcpConnection**, **FirstTcpPort**, and **LastTCPPort** parameters as well. Save this file.
7. If the Web Server has not been started, start it now.
8. Start up your HTML Thin Client application.
9. From a browser, start a session to the application by entering an URL using the format detailed in Appendix C.
10. You should now see browser details on the Application Monitor window and the start up form will be shown on the browser. You will find that the very first connection will typically take longer than subsequent connections because all the initialization happens on the first connection.

Appendix G: Troubleshooting Tips

Service Unavailable

If you see the following message on the browser:

```
Service Unavailable
This Service is currently unavailable at this time.
```

One of the following caused this.

- Your application is not running
- The TCP parameters specified in the ini file do not match those of the application

HTTP 400 – Bad Request or No Application Named

Depending on the browser, you may see one of the following messages.

```
The page cannot be found
(this is equivalent to HTTP 400 error)
```

```
No Application Named
The URL must include the Application name as parameter
```

One of the following caused this.

- You have not included the application name in the URL
- The application name does not have a section in the initialization file

Invalid Application Name

If you see the following message,

```
Invalid Application Name
The specified Application name is too long.
```

the name of the application is too long. In JADE, the application name cannot be longer than 30 characters.

HTTP Error 404

If you see the following message,

```
HTTP Error 404
404 Not Found
```

this is caused when the URL is invalid.

Images do not appear on the browser

If images do not appear on the browser, one of the following reasons is the cause.

- Physical directory/virtual directory mapping has been specified incorrectly in the application or to the web server
- The image type is not supported by the browser
- The **VirtualDirectory** parameter in the **jadehttp.ini** file does not match
- The **PhysicalDirectory** as specified in the Application options or the jade.ini file

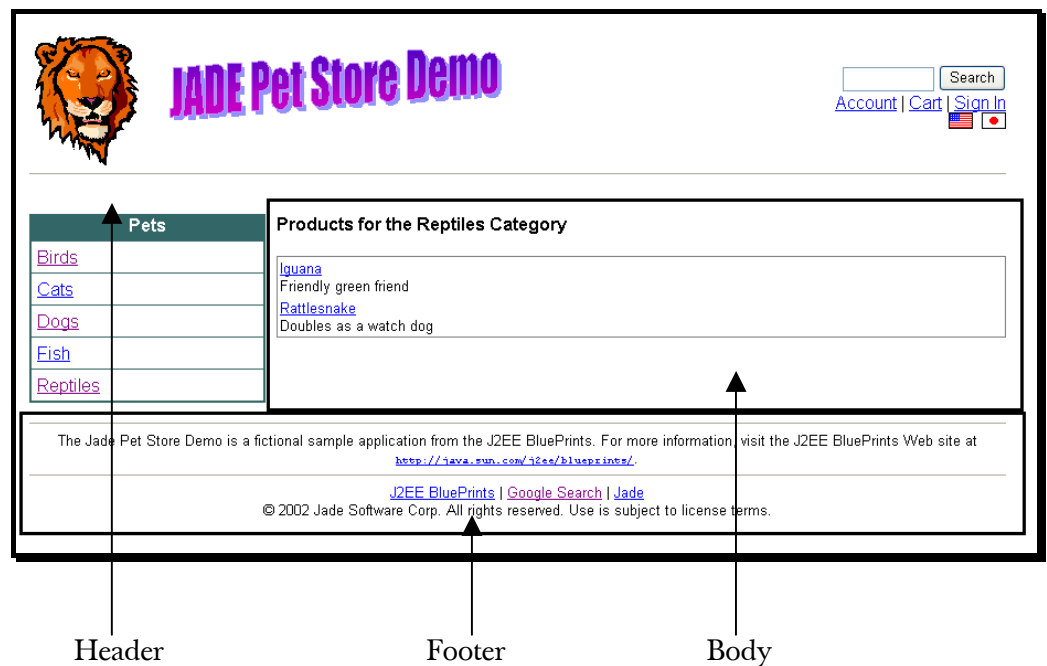
JADE Support

If you are unable to resolve the problem, contact JADE Support if your licence includes support.

Appendix H: Sample Application

The following example uses portions of the JAVA Pet Store Application example to illustrate the usage of this feature.

In this abbreviated example, we will import 3 HTML documents: a header, a footer, and a body. The following image shows the completed page as displayed in IE6.



The HTML for the header document is as follows:

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title><jade_tag name=pagetitle></title>
<meta http-equiv=content-type content="text/html; charset=utf-8">
<style type=text/css>
.petstore {
FONT-SIZE: small; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_title {
FONT-WEIGHT: bold; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_footer {
FONT-SIZE: x-small; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_listing {
FONT-SIZE: x-small; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_form {
FONT-SIZE: x-small; FONT-FAMILY: Helvetica, Arial, sans-serif}
</style>
</head>

<body bgcolor=#ffffff>
<table cellspacing=0 cellpadding=5 width="100%" border=0>
<tbody>
<tr>
<td colspan=2>
<form action="<JADE_TAG name=search>" method=post>
<table cellspacing=0 cellpadding=0 width="100%" bgcolor=#ffffff
border=0><tbody>
```

```

<tr>
  <td valign=center align=left><a
    href="<JADE_TAG name=mainScreen>"></a></td>
  <td class=petstore valign=center align=right>
<input class=petstore_listing size=8 name=keywords>
  <input class=petstore_listing type=submit value=search
    name=submitsearch>
  <br>
  <a href="<JADE_TAG name=customerScreen>">Account</a>
  |
  <a href="<JADE_TAG name=cartDo>">Cart</a>
  |
  <a href="<JADE_TAG name=signon_welcomeScreen>">
  <jade_tag name=signonmessage></a> <br>
  <a href="<JADE_TAG name=changeLocaleEN>">
  </a> &nbsp;
  <a href="<JADE_TAG name=changeLocaleJP>">
  </a>
  </td>
</tr>
<tr>
  <td colspan=2>
  <hr noshade size=1>
  </td>
</tr></tbody></table></form></td></tr>
<tr>
  <td valign=top width="20%">
  <table cellspacing=0 cellpadding=1 width="100%" border=0>
  <tbody>
  <tr>
  <td class=petstore_title align=middle bgcolor=#336666><font
    color=#ffffff>Pets</font> </td></tr>
  <tr>
  <td bgcolor=#336666>
  <table cellspacing=1 cellpadding=5 width="100%" border=0>
  <tbody>
  <tr>
  <jade_tag name=categories>
  </tr></tbody></table></td></tr></tbody></table></td>

```

JADE_TAGS are highlighted.

Note the following.

- JADE_TAGS that are used as attributes (e.g. the **A href=**) are always in double quotes.
- The HTML document is incomplete. This is because the body and footer documents are appended to it to form the complete document.

Loading this document into the wizard will create a class (**Header**, for example) with the following properties:

Name	Jade Type	HTML Type
cartDo	String	JADE_TAG
categories	String	JADE_TAG
changeLocaleEN	String	JADE_TAG
changeLocaleJP	String	JADE_TAG
customerScreen	String	JADE_TAG
keywords	String	JADE_TAG
mainScreen	String	JADE_TAG

Name	Jade Type	HTML Type
pageTitle	String	JADE_TAG
search	String	JADE_TAG
signonMessage	String	JADE_TAG
signon_welcomeScreen	String	JADE_TAG
submitSearch	String	SUBMIT

In the `updateValues` method of the `Header` class, add the following code.

```
updateValues(): Boolean updating;

vars
    attr: String;
    cat: Category;
begin
    if currentSession.user = null then
        signon_welcomeScreen := buildLink(SignOn);
        signonMessage := 'Sign In';
    else
        signon_welcomeScreen := buildLink(SignOff);
        signonMessage := 'Sign Out';
    endif;
    search := buildFormAction('');
    attr := 'class=petstore bgColor=#ffffff';
    categories := null;
    foreach cat in app.myRoot.allCategories do
        categories := categories & tableBeginRow('') &
            tableAddData(attr, '<a href="" & buildLink(Product) &
                '&categoryId=' & cat.catid & "">' & cat.catid & '</a>') &
            tableEndRow;
    endforeach;
    cartDo := buildLink(Cart);
    customerScreen := buildLink(Customer);
    return inheritMethod();
end;
```

The HTML fragment for the `Footer` class is as follows.

```
<table cellspacing="0" cellpadding="0" width="100%" bgcolor=
"#ffffff">
<tbody>
<tr>
<td>
<hr noshade size="1">
</td>
</tr>

<tr>
<td class="petstore_footer" align="middle">The Jade Pet Store Demo
is a fictional sample application from the J2EE BluePrints. For
more information, visit the J2EE BluePrints Web site at <a href=
"http://java.sun.com/j2ee/blueprints/">
<code>http://java.sun.com/j2ee/blueprints/</code></a>.<br>
</td>
</tr>

<tr>
<td>
<hr noshade size="1">
</td>
</tr>

<tr>
<td class="petstore_footer" align="middle"><a href=
"http://java.sun.com/j2ee/blueprints/">J2EE BluePrints</a> | <a
href="http://www.google.com/">Google Search</a> | <a href=
"http://www.discoverjade.com/">Jade</a><br>
&Aacute;&copy; 2002 Jade Software Corp. All rights reserved. Use is
subject to license terms.</td>
```

```

</tr>
</tbody>
</table>
</body>
</html>

```

As there are no input tags for this document, there are no properties. This is a static page and hence no code is required in **updateValues**.

The HTML code fragment for the **Body** class is as follows:

```

<jade_tag name=header value=header>
  <td valign=top width="60%">
    <p class=petstore_title>Products for the <jade_tag name=catname>
Category</p>
  <table cellspacing=0 cellpadding=1 width="100%" border=0>
    <tbody>
      <tr>
        <td bgcolor=#808080>
          <table cellspacing=0 cellpadding=2 width="100%" bgcolor=#ffffff
border=0>
            <tbody>
              <jade_tag name=productdetails>
              </tbody>
            </table>
          </td></tr></tbody></table>
        <div align=right>
          <p class=petstore_listing></p></div></td>
      <td valign=top></td></tr>
    <tr>
      <td colspan=2></td></tr>
    <tr>
      <td colspan=2><
<jade_tag name=footer value=footer>

```

The wizard will create the **Body** class with the following properties.

Name	Jade Type	HTML Type
header	JadeHTMLClass	JADE_TAG
catName	String	JADE_TAG
productDetails	String	JADE_TAG
footer	JadeHTMLClass	JADE_TAG

In the **updateValues** method of the **Body** class, add the following code:

```
updateValues(): Boolean updating;

vars
  attr,
  link:      String;
  cat:       Category;
  prod:      AnimalProduct;
begin
  link := buildLink(items);
  attr := 'class=petstore_listing';
  cat := app.myRoot.allCategories[currentSession.categoryId];
  if cat <> null then
    catName := cat.catid;
    foreach prod in cat.allProducts do
      productDetails := productDetails &
      tableBeginRow('') & tableAddData(attr, '<A
        href="" & link & '&productId=' & prod.productId & '>' &
        prod.name & '</A><BR>' & prod.descn) & tableEndRow;
    endforeach;
  endif;
  header.Header.pageTitle := 'Products';

  return inheritMethod();
end;
```

At run time, the following HTML will be created when the **Body** document is requested.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<meta name="generator" content="HTML Tidy, see www.w3.org">
<title>Products</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<style type="text/css">
.petstore {
  FONT-SIZE: small; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_title {
  FONT-WEIGHT: bold; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_footer {
  FONT-SIZE: x-small; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_listing {
  FONT-SIZE: x-small; FONT-FAMILY: Helvetica, Arial, sans-serif}
.petstore_form {
  FONT-SIZE: x-small; FONT-FAMILY: Helvetica, Arial, sans-serif}
</style>
</head>
<body bgcolor="#ffffff">
<table cellpadding="5" cellspacing="0" width="100%" border="0">
<tbody>
<tr>
<td colspan="2">
<form action="http://cnwcrsla/jade/jadehttp.dll?jadepetstoredemo"
method="post"><input type="hidden" name="_ts" value=
"15 April 2003, 16:29:58"> <input type="hidden" name=
"_jadeReferenceClass" value="Header"> <input type="hidden" name=
"_jadeReferenceDocument" value="Header"> <input type="hidden" name=
"_session" value="f14f8c337be99d41">

<table cellpadding="0" cellspacing="0" width="100%" bgcolor=
"#ffffff" border="0">
<tbody>
<tr>
<td valign="center" align="left"><a href=""><img alt=
"Jade Pet Store Demo logo" border="0" src=
"images/Jade_banner.png"></a></td>
<td class="petstore" valign="center" align="right"><input class=
"petstore_listing" name="keywords" size="8"> <input type="submit"
class="petstore_listing" name="submitSearch" value="Search"><br>
```

```

<a href=
"http://cnwcrsla/jade/jadehttp.dll?jadepetstoredemo&_jadereferenceclass=
header&_jadereferencedocument=header&_session=f14f8c337be99d41&_
gotopage=customer">
Account</a> | <a href=
"http://cnwcrsla/jade/jadehttp.dll?jadepetstoredemo&_jadereferenceclass=
header&_jadereferencedocument=header&_session=f14f8c337be99d41&_
gotopage=cart">
Cart</a> | <a href=
"http://cnwcrsla/jade/jadehttp.dll?jadepetstoredemo&_jadereferenceclass=
header&_jadereferencedocument=header&_session=f14f8c337be99d41&_
gotopage=signon">
Sign In</a> <br>
<a href=""></a> &nbsp; <a
href=""></a> </td>
</tr>

<tr>
<td colspan="2">
<hr noshade size="1">
</td>
</tr>
</tbody>
</table>
</form>
</td>
</tr>

<tr>
<td valign="top" width="20%">
<table cellspacing="0" cellpadding="1" width="100%" border="0">
<tbody>
<tr>
<td class="petstore_title" align="middle" bgcolor="#336666"><font
color="#ffffff">Pets</font> </td>
</tr>

<tr>
<td bgcolor="#336666">
<table cellspacing="1" cellpadding="5" width="100%" border="0">
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<tr>
<td class="petstore" bgcolor="#ffffff"><a href=
"http://cnwcrsla/jade/jadehttp.dll?JadePetStoreDemo&_jadeReferenceClass=
Header&_jadeReferenceDocument=Header&_session=f14f8c337be99d41&_
goToPage=Product&_categoryId=Birds">
Birds</a></td>
</tr>

<tr>
<td class="petstore" bgcolor="#ffffff"><a href=
"http://cnwcrsla/jade/jadehttp.dll?JadePetStoreDemo&_jadeReferenceClass=
Header&_jadeReferenceDocument=Header&_session=f14f8c337be99d41&_
goToPage=Product&_categoryId=Cats">
Cats</a></td>
</tr>

<tr>
<td class="petstore" bgcolor="#ffffff"><a href=
"http://cnwcrsla/jade/jadehttp.dll?JadePetStoreDemo&_jadeReferenceClass=
Header&_jadeReferenceDocument=Header&_session=f14f8c337be99d41&_
goToPage=Product&_categoryId=Dogs">
Dogs</a></td>
</tr>

<tr>
<td class="petstore" bgcolor="#ffffff"><a href=
"http://cnwcrsla/jade/jadehttp.dll?JadePetStoreDemo&_jadeReferenceClass=
Header&_jadeReferenceDocument=Header&_session=f14f8c337be99d41&_
goToPage=Product&_categoryId=Fish">
Fish</a></td>
</tr>

```

```

<tr>
<td class="petstore" bgcolor="#ffffff"><a href=
"http://cnwcrsla/jade/jadehttp.dll?JadePetStoreDemo&_jadeReferenceClass=
Header&_jadeReferenceDocument=Header&_session=f14f8c337be99d41&_
goToPage=Product&_categoryId=Reptiles">
Reptiles</a></td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
<td valign="top" width="60%">
<p class="petstore_title">Products for the Reptiles Category</p>

<table cellspacing="0" cellpadding="1" width="100%" border="0">
<tbody>
<tr>
<td bgcolor="#808080">
<table cellspacing="0" cellpadding="2" width="100%" bgcolor=
"#ffffff" border="0">
<tbody>
<tr>
<td class="petstore_listing"><a href=
"http://cnwcrsla/jade/jadehttp.dll?JadePetStoreDemo&_jadeReferenceClass=
Product&_jadeReferenceDocument=Product&_session=f14f8c337be99d41&_
goToPage=Items&_productId=RP-LI-02">
Iguana</a><br>
Friendly green friend</td>
</tr>

<tr>
<td class="petstore_listing"><a href=
"http://cnwcrsla/jade/jadehttp.dll?JadePetStoreDemo&_jadeReferenceClass=
Product&_jadeReferenceDocument=Product&_session=f14f8c337be99d41&_
goToPage=Items&_productId=RP-SN-01">
Rattlesnake</a><br>
Doubles as a watch dog</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>

<div align="right"><br class="petstore_listing">
<br>
</div>
</td>
<td valign="top"></td>
</tr>

<tr>
<td colspan="2"></td>
</tr>

<tr>
<td colspan="2">
<table cellspacing="0" cellpadding="0" width="100%" bgcolor=
"#ffffff">
<tbody>
<tr>
<td>
<hr noshade size="1">
</td>
</tr>
</tbody>
</table>

<tr>
<td class="petstore_footer" align="middle">The Jade Pet Store Demo
is a fictional sample application from the J2EE BluePrints. For
more information, visit the J2EE BluePrints Web site at <a href=
"http://java.sun.com/j2ee/blueprints/"><code>

```

```

http://java.sun.com/j2ee/blueprints/
```

Note the following:

- JADE has replaced JADE_TAGS with the corresponding property values
- Additional fields have been added to the document, which are internal to JADE. These are listed in the following table:

Field Name	Description
_ts	Timestamp information
_referenceClass	The class related to the HTML document
_referenceDocument	The HTML document from which the page was created
_session	Session key
_goToPage	On hyperlinks, specifies the document to be returned

Appendix I: Comparison

Advantage & Disadvantages

What are the benefits of using HTML Documents instead of dynamically generated Web pages?

- HTML is hand crafted or created using an external Web page design tool. This can usually result in a page that looks closer to what is expected, across multiple browsers on multiple platforms
- Page generation is much faster as usually only a small subset of the page needs to be generated
- Scalability is improved because there is no reliance on transient objects
- Greater flexibility on the page contents as any HTML and scripting can be used

The disadvantages are:

- HTML knowledge is essential
- Understanding of the browser request messages is required
- You are dealing directly with two technologies (JADE & HTML) and hence the coding requirements are more complex

Performance Comparison

A performance comparison was made between the dynamic generation and HTML documents using the Microsoft Application Centre Test (ACT) tool. The applications were written to provide similar functionality.

The results of this test are shown in the following table.

Test type	Dynamic	HTML Documents
Simultaneous browser connections:	1	1
Warm up time (secs):	0	0
Test duration:	00:00:05:00	00:00:05:00
Test iterations:	339	602
Summary		
Total number of requests:	3,748	7,832
Total number of connections:	3,744	7,832
Average requests per second:	12.49	26.11
Average time to first byte (msecs):	229.08	34.89
Average time to last byte (msecs):	249.11	36.02
Average time to last byte per iteration (msecs):	2,754.15	468.68
Number of unique requests made in test:	7	7
Number of unique response codes:	1	1
Errors Counts		
HTTP:	0	0
DNS:	0	0
Socket:	0	0
Additional Network Statistics		
Average bandwidth (bytes/sec):	169,232.99	249,358.10
Number of bytes sent (bytes):	1,749,517	3,821,514
Number of bytes received (bytes):	49,020,380	70,985,916
Average rate of sent bytes (bytes/sec):	5,831.72	12,738.38
Average rate of received bytes (bytes/sec):	163,401.27	236,619.72
Number of connection errors:	0	0
Number of send errors:	0	0
Number of receive errors:	0	0
Number of timeout errors:	0	0
Response Code: 200 - The request completed successfully.		
Count:		3,748
Percent (%):		100.00

The table shows that using the HTML documents feature, the application can process about twice the number of requests that the dynamic generation can.

Further, as mentioned previously, the HTML documents option scales better than the dynamic generation option. The number of objects in the database did not have any effect on the throughput. In both cases, the throughput remained constant.

Running a J2EE application that was downloaded from the Web, it is worth noting that the performance of this system was poor (59 requests in 5 minutes). The J2EE application used the Cloudscape database. No code changes were made to this application.

A similar .NET application had three times more throughput than JADE with a small database, but the throughput decreased quite significantly as the number of objects in the database were increased. The .NET application used the freely available MSDE database (which is identical to SQL Server 2000 in performance but limited to five concurrent workloads). Again no code changes were made to the application.