This presentation will cover the IBM WebSphere Application Server Feature Pack for Web 2.0 connectivity.

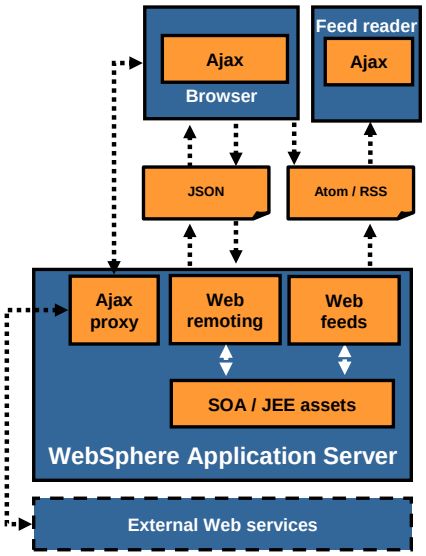
Agenda

- JSON4J
- Ajax proxy
- Summary

The presentation will touch on JSON4J and the Ajax proxy.

Web 2.0 to SOA connectivity overview

- For enabling connectivity from Ajax clients and mashup to external Web services, internal SOA services, and other JEE assets.
- Extends enterprise data to customers and partners through Web feeds.

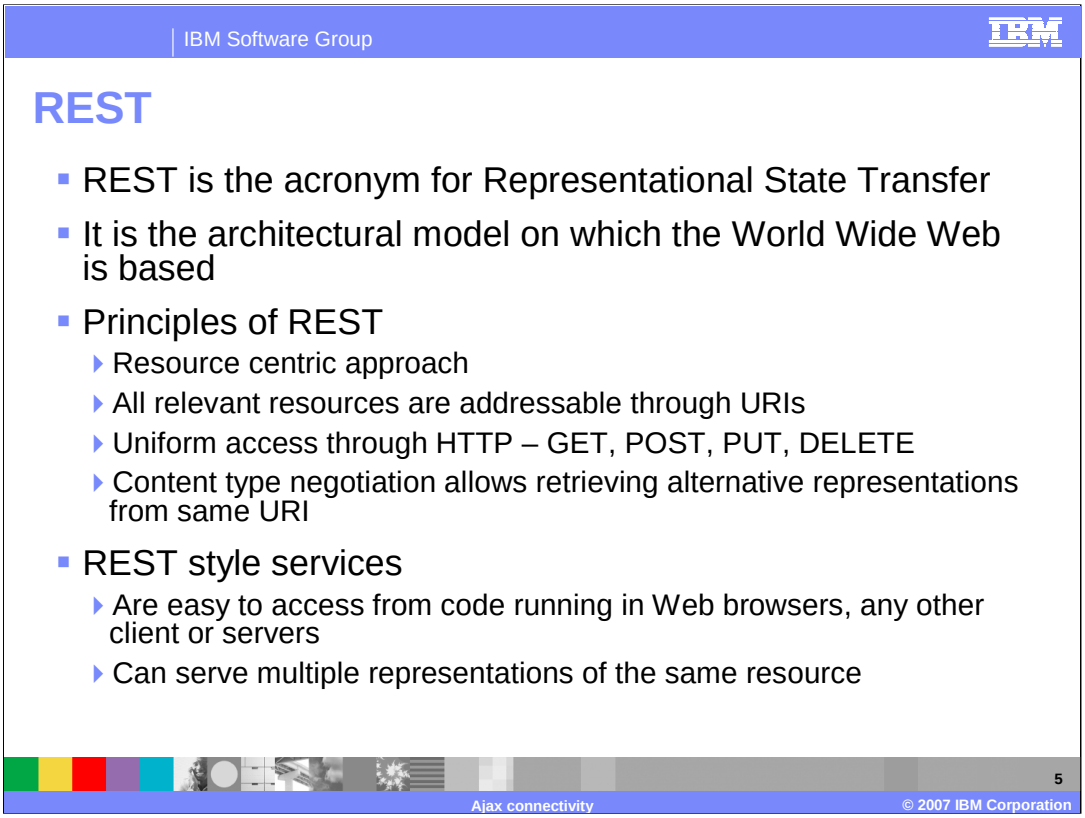


The Web 2.0 to SOA connectivity is for enabling connectivity from Ajax clients to external Web services, internal SOA services, and other JEE assets. It allows for exposing enterprise data over the Web.

Section

JSON4J

This section covers JSON4J



REST, Representational State Transfer, is the architectural model on which the World Wide Web is based. REST is a resource centric approach where all relevant resources are addressable through URIs. REST allows uniform access through HTTP – GET, POST, PUT, DELETE where the content type negotiation allows retrieving alternative representations from same URI. REST style services are easy to access from code running in Web browsers; any other client or servers and can serve multiple representations of the same resource.

JSON

- The vast majority of the clients of a restful service will be written in JavaScript™
- In recognition of that, JSON (JavaScript object notation) allows for rapid exchange of JavaScript objects, but also in a simple, human-readable format
- JSON consumes a little less bandwidth than XML and works well with all browsers
- JSON is built up from a collection of name-value pairs and ordered lists of values

```
{
        "customer" : {
                "name" : "Jane Doe",
                "company" : "Acme Enterprises”
        }
}
```


JSON, or JavaScript Object Notation, allows for rapid exchange of JavaScript objects, in a simple, human-readable format. JSON consumes a little less bandwidth than XML and works well with all browsers. JSON is built up from a collection of name-value pairs and ordered lists of values. JSON4J library is an implementation of JSON for use within Java environments.

JSON4J

- JSON4J library is an implementation of JSON for use within Java environments
- JSON4J provides a fast transform for XML->JSON conversion
- Reasons to use JSON
  - Dealing with XML at the browser has some challenges
    - XML must be parsed, translated to a DOM tree or JavaScript object
    - Requires additional JavaScript DOM objects
    - Often requires additional JavaScript to hide browser specific complexity
    - Impacts client side performance

JSON4J provides fast transformation from XML to JSON. Dealing with XML at the browser has some challenges: the XML must be parsed and translated to a DOM tree or complex object that can be readily manipulated by JavaScript. It also requires additional JavaScript DOM objects to be present and often additional JavaScript is needed to hide the browser specific complexity. Dealing with XML at the client impacts client side performance.

Section

Ajax proxy

This section covers the Ajax proxy.

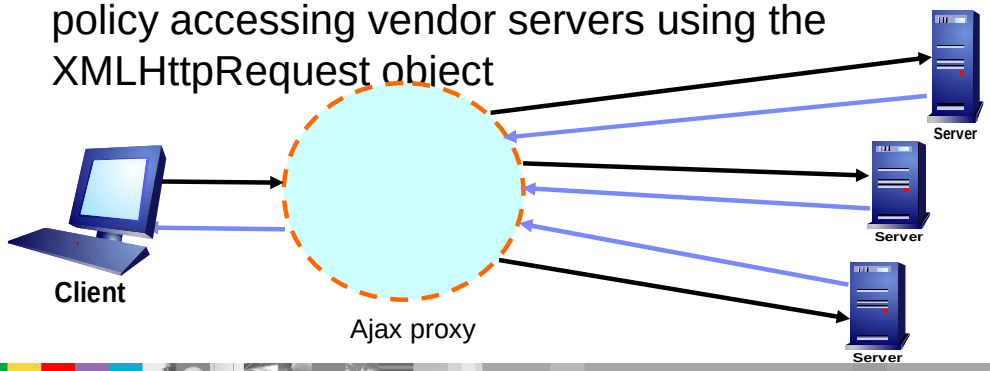
Overview

- Ajax based Web applications sometimes want to do Ajax requests to servers different from the server that served the HTML document
- This is restricted by today's XMLHttpRequest and IFrame implementations
  - Requests are only allowed to the server that served the current document
  - This is to prevent malicious Ajax code served on one site from using the browser as basis for attacking other servers

Ajax based Web applications sometimes want to make Ajax requests to servers other than the server that served the HTML document. For example, your portlet could be served from www.mycompany.com but your Ajax application tries to load a feed from cnn.com. The ability to do Ajax requests to servers other than the server that served the HTML document is restricted by today's XMLHttpRequest and IFrame implementations. For security, requests are only allowed to the server that served the current document. This prevents malicious code served on one site from using a browser as basis for attacking other servers.

Ajax proxy

- An Ajax proxy is an application level proxy server that mediates HTTP requests and responses between Web browsers and servers
- Allows Web browsers to bypass the *same-origin* policy accessing vendor servers using the XMLHttpRequest object



An Ajax proxy is an application-level proxy server that mediates HTTP requests and responses between Web browsers and servers. Ajax proxies allow Web browsers to bypass the same-origin policy and therefore to access vendor servers using XMLHttpRequest. To realize this bypassing, you can choose from two approaches.

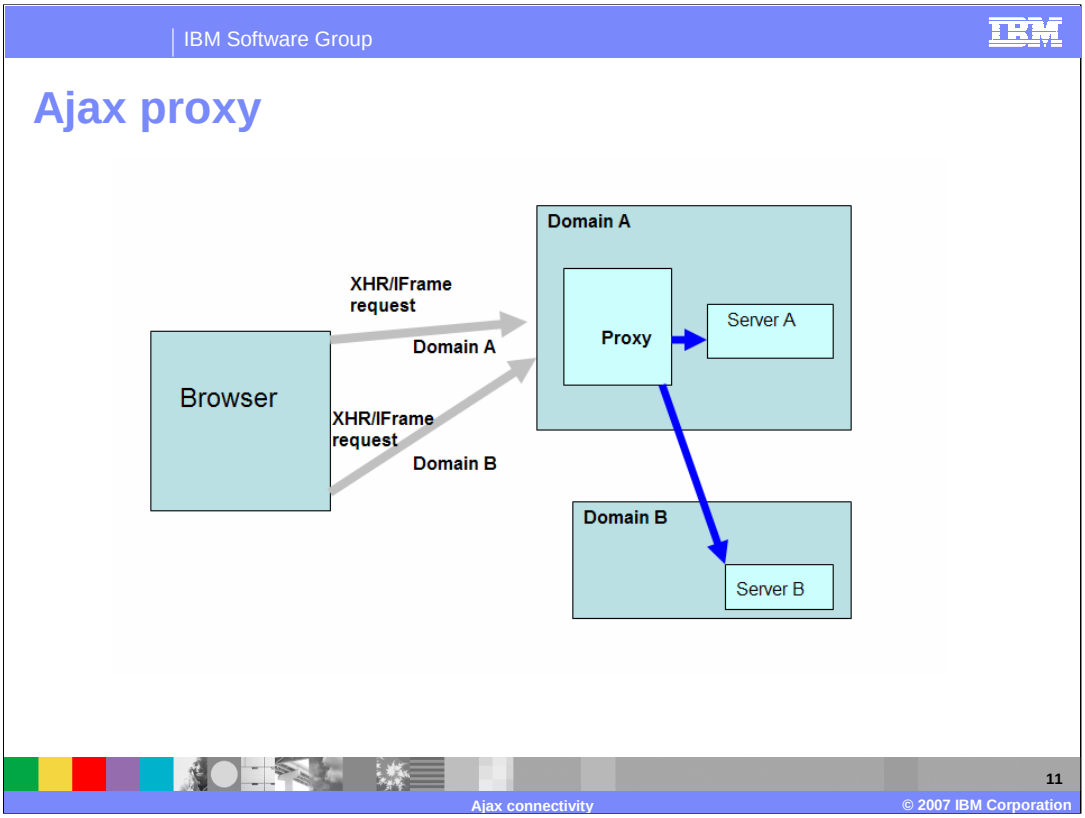
The first approach is that the client-side Web application is aware of the vendor URL and passes it as a request parameter in the HTTP request to the Ajax proxy. The proxy then forwards the request to some www.remoteservice.com. Note that you can hide the use of a proxy server in the implementation of the Ajax library used by the Web application developer. From the Web application developer's point of view, it might appear that there is no same-origin policy at all.

The second approach is that the client-side Web application is not aware of the vendor URL, and it tries to access resources on the Ajax proxy server through HTTP. By a predefined encoding rule, the Ajax proxy translates the requested URL into a vendor server URL and retrieves contents on behalf of the client. In this case, it looks to the Web application developer like the application is communicating with the proxy server directly.

In a network environment, a proxy is positioned between the requesting client and the server. It accepts requests from the client and passes them onto the server. It accepts the request for the server and passes the result onto the client.

The proxy provided with the IBM WebSphere Application Server Feature Pack for Web 2.0 is a reverse proxy installed near one or more servers. Connections coming through the reverse proxy are forwarded to the requested server. From the client's perspective, it appears that the requests are originating from the same server even though the reverse proxy might forward requests to different Web servers.

The proxy can be used to broker client requests from multiple Domains while using Ajax. JavaScript sandboxing rules prevent network requests to servers other than where the JavaScript originated from. As an example, if the JavaScript application originated from Domain A and attempts to use an XMLHttpRequest to Domain B, the browser will prevent the Domain B request. The proxy can be used to broker the request which provides the appearance to the client that the request came from the same server from which the JavaScript originated.

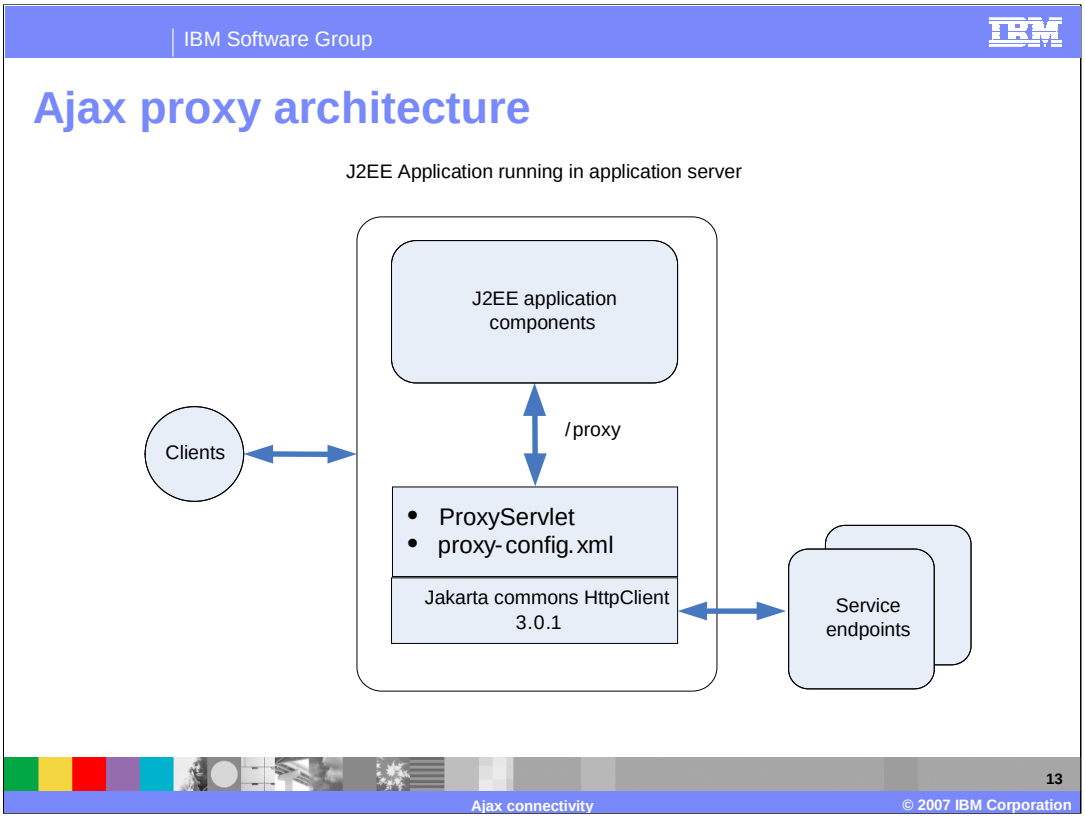


Ajax communication methods include XMLHttpRequest and IFrame requests. This image abbreviates XMLHttpRequest as XHR. These methods allow the browser to send HTTP requests to a server at any time with or without a user action. One limitation of an IFrame or XMLHttpRequest request is the restriction to make a request to a server other than the one that served the original HTML page. This limitation is sometimes known as a same-domain limitation or same-origin policy and exists as a security measure to prevent hacker script injection by redirecting the page to an untrusted server. However, an Ajax-based Web application might need to make a request to a server that is different than the server that served the main HTML page. Client-side methods exist to work around the same-domain limitation, but these methods have drawbacks. The recommended solution to the same-domain limitation is to use a proxy server to forward the request to a server on a different domain. An example of proxy server use is shown in the diagram. In this diagram, a proxy server is forwarding requests to a server in domain B.

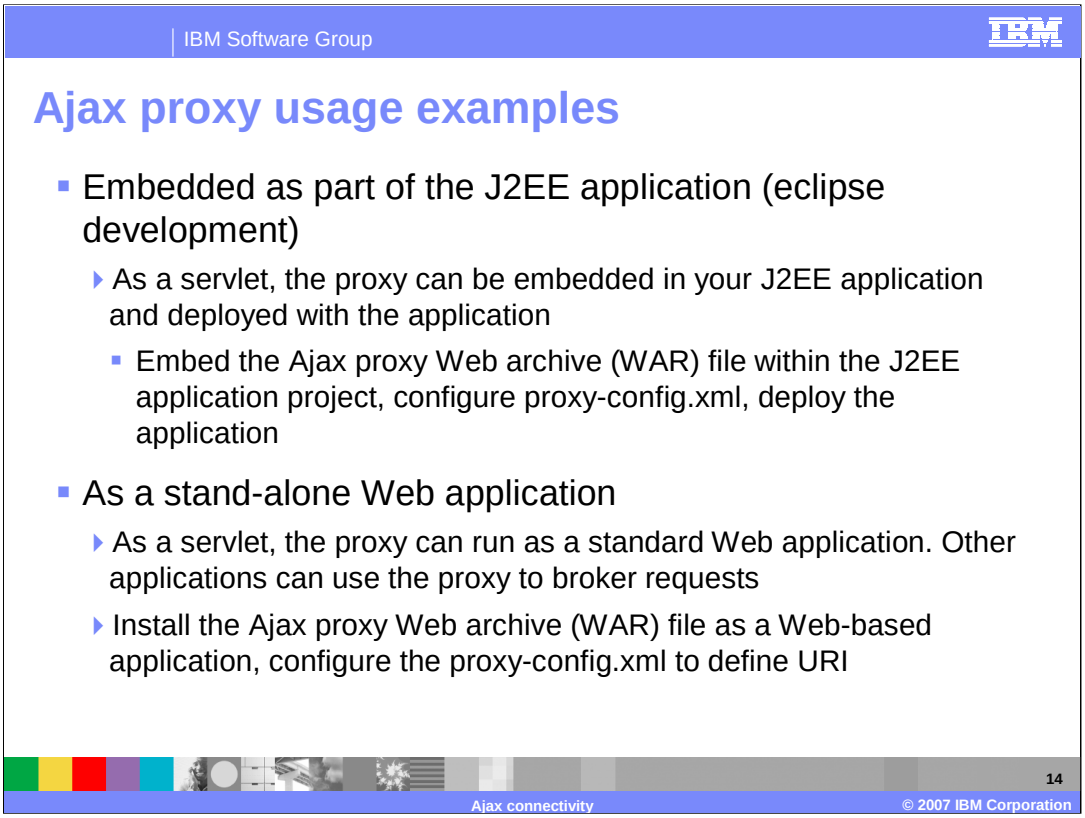
Ajax proxy goals

- A lightweight proxy to enable browser based access to cross-site services in a Web 2.0 fashion
- Ajax proxy can run embedded as a Web archive (WAR) within another J2EE application or stand-alone
- Uses J2EE application level security for proxy access control
- Support for white-listing policies for filtering on criteria of incoming requests such as: cookies, MIME types, HTTP headers, HTTP verbs like GET, POST, and PUT

An Ajax proxy enables browser-based access to cross-site services. It can run embedded within another J2EE application or stand-alone with the emphasis on ease of use. In addition, it uses J2EE application level security for proxy access control and does not require WebSphere runtime or configuration integration since its configuration is static. Ajax proxy also has support for white-listing policies for filtering criteria of incoming requests such as cookies and mime types.



This diagram shows the general architecture of Ajax proxy. Different end points aggregate to the same domain making it look like everything is coming from one place. Ajax proxy is customized with the application and it runs within the Web container therefore performance is determined by the tuning of the Web container as opposed to the proxy. The difference between this Ajax proxy and the other WebSphere Application Server proxies like the Web messaging channels is that it is a reverse proxy meaning that it is an application level proxy; it must be embedded within an application. Other proxies are lower level proxies.



There are some examples of Ajax proxy usage. The Ajax proxy can be applied in many ways. Here are just two scenarios which will give you an idea on how you might use the proxy function.

As a servlet, the proxy can be embedded in your J2EE application and deployed with the application. Embedding the proxy with the application allows the proxy to be deployed with the application in a ready-to-run configuration. You might choose to embed the Ajax proxy in your application if you are building an application that combines the content from one or more service endpoints in an Ajax based application.

Modify your application.xml and geronimo-application.xml (if using WebSphere Application Server Community Edition 2.0) to include the Ajax proxy Servlet. Normally, if you already have an enterprise EAR file created, you can associate the Ajax proxy for IBM WebSphere WAR file with your EAR project and Eclipse will take care of the rest.

Modify the proxy-config.xml to define URI context paths, URLs, and policies that the proxy will support.

Build your application with Ajax proxy servlet and deploy it to either WebSphere Application Server V6.1 or WebSphere Application Server Community Edition 2.0

As a servlet, the proxy can run as a standard Web application. Other applications can use the proxy to broker requests. To do this, modify the proxy-config.xml to define URI context paths and polices that the proxy will support. Then deploy the Ajax proxy for IBM WebSphere WAR file to your application server.

proxy-config.xml

- Defines the **policy** by which URI requests are allowed to pass through the proxy
- Defines how **context paths** from the client will be mapped to the URI on a sever
- The proxy-config.xml file can be modified by an editor and needs to be in location on the CLASSPATH which the proxy Servlet can locate
- Changes to the proxy-config.xml file are not dynamic

The proxy-config.xml file defines the policy by which URI requests are allowed to pass through the proxy and how context paths from the client will be mapped to the URI on a server.

This file can be modified in an editor and needs to be in a location on the CLASSPATH that the proxy Servlet can locate. Changes to the proxy-config.xml file are not dynamic therefore the servlet needs to be restarted for the changes to be recognized.

proxy-config.xml example

geronimo-web.xml | proxy-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<proxy-rules
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:proxy="http://www.ibm.com/xmlns/prod/websphere/portal/v6.0.1/proxy-config">
    <proxy:mapping contextpath="/http/*" />
    <proxy:mapping contextpath="/rss/money_latest.rss" url="http://rss.cnn.com" />
    <proxy:mapping contextpath="/intl/*" url="http://www.google.com" />

    <proxy:policy url="*" acf="none">
        <proxy:actions>
            <proxy:method>GET</proxy:method>
        </proxy:actions>
        <proxy:cookies>
            <proxy:cookie>ABCD</proxy:cookie>
        </proxy:cookies>

    </proxy:policy>


</proxy-rules>
```


In this example, you can see the mapping contextpath defining the URLs from which the HTTP requests will be mapped. You can also see the policies defined.

Importance of Ajax proxy

- Can be configured to only support access to certain Web sites
- Content filtering from a vendor site
  - A proxy server guaranteeing the content type returned from vendor Web site matches a configured permissible content type is an example of content filtering
- Ajax proxy server can be used to convert response data to a format a Web application expects
  - Example of a proxy server converting data: A proxy server converting a Web service response from XML to JSON for consumption by a browser

Choosing to proxy Ajax requests instead of using a browser based or client-side proxy alternative can be beneficial. A proxy server can be configured to only support access to certain Web sites; whereas a browser-based solution does not have the ability to restrict cross-domain access on a per server basis. Another proxy server benefit is content filtering from a vendor site. A proxy server guaranteeing the content type returned from vendor Web site matches a configured permissible content type is an example of content filtering. Using the proxy server to convert response data to a format a Web application expects is also another benefit of the proxy. For example, proxy server can convert a Web service response from XML to JSON for consumption by a browser.

Ajax proxy limitations

- HTML rewrites
  - Will not rewrite the URL request in the body of the data returned in the HTTP response
- Example:
  - proxy mapping rule [**<proxy:mapping contextpath="/http/*" /> URL]** requests to the proxy with the context path **/http/www.myothersite.com** will redirect to **http://www.myothersite.com** and return the result to the browser.
  - Image tag <IMG href="/images/mypicture.jpg" /> in an HTML page returned by www.myothersite.com.

There are a few limitations of the Ajax proxy; specifically HTML rewrites and security.

With HTML rewrite, the proxy will not attempt to rewrite the URL request in the body of the data returned in the HTTP response. Take a look at the example shown here. The browser will issue a request for /images/mypicture.jpg to the proxy servlet which will respond with a 404. The proxy will not resolve that the image request is intended for www.myothersite.com.

Ajax proxy limitations (continued)

- Security
  - Does not provide filtering IP address
  - Example:
    - If proxy is configured for one application, another application can access the proxy

With Security, the proxy does not provide filtering IP addresses. As an example, even though the proxy may be configured to be used by your application, there is nothing to prevent another application from accessing the proxy. If you need to secure the proxy, then this should be done in the context of J2EE security. WebSphere Application Server documentation provides additional information on how to secure your J2EE application.

Ajax proxy logging

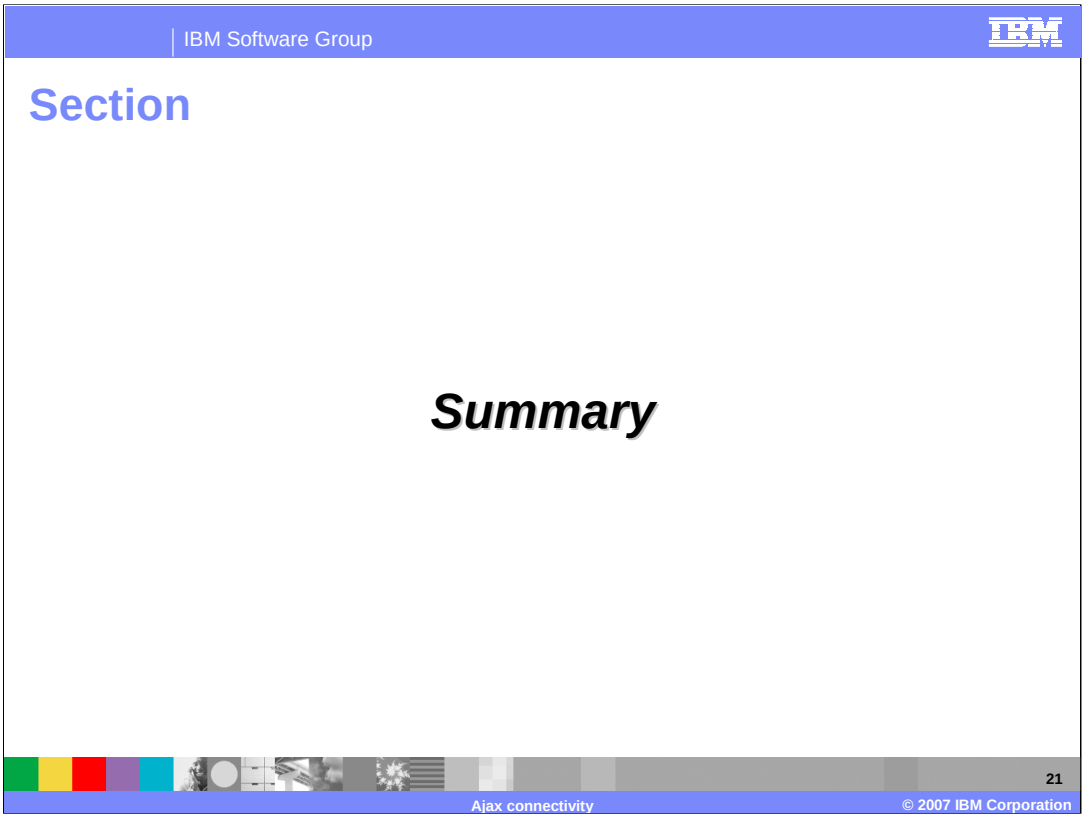
- Ajax proxy uses Jakarta commons logging
  - Logging framework that supports Java logging and log4j
- For WebSphere Application Server (V6.0, V6.1)
  - Supports trace and info logging
  - Logging can be set through the administrative console
- WebSphere Application Server Community Edition uses the log4j logging package
  - Modifying the log4j-server.Properties file
  - Log4j.category.com.ibm.ws.Ajaxproxy.servlet.proxyServlet=DEBUG

The Ajax proxy uses the Jakarta commons logging facility. The Jakarta commons logging provides a logging framework which can integrate other logging facilities such as log4j and java.util.logging. The logging configuration that you use will depend on the application server you are deploying the Ajax proxy to.

For WebSphere Application Server V6.0 and V6.1, the Ajax proxy integrates with the logging in WebSphere facility and supports INFO level and FINEST level logging. Select the com.ibm.ws.Ajaxproxy package under the Logging and Tracing panel. Select *finest* to enable the TRACE level logging. To view request and header information sent to and from the Ajax proxy, enable tracing for the com.ibm.ws.Ajaxproxy.servlet package. INFO level logging is enabled by default in WebSphere.

For other Application Servers such as WebSphere Community Edition that uses the log4j logging package, edit the log4j-server.properties file to add logging for the Ajax proxy servlet as is shown here.

There is a demonstration included in this education package to show how to install Ajax proxy into Eclipse. The demonstration also shows how to create an Ajax proxy project and how the proxy can be used with an example Web site.



This section is the summary.

Summary

- **Ajax connectivity** is for enabling connectivity from Ajax clients and mashups to external Web services, internal SOA services, and J2EE assets.
- **JSON4J** library allows for rapid exchange of JavaScript objects by implementing JSON for use within Java environments.
- **Ajax proxy** provides safe, reliable access to Internet based services and mashups from browser based Ajax applications

Ajax connectivity is for enabling connectivity from Ajax clients and mashups to external Web services, internal SOA services, and J2EE assets. The JSON4J library implements JSON for use in Java environments, and the Ajax proxy provides safe, reliable access to Internet based services and mashups from browser based Ajax applications.

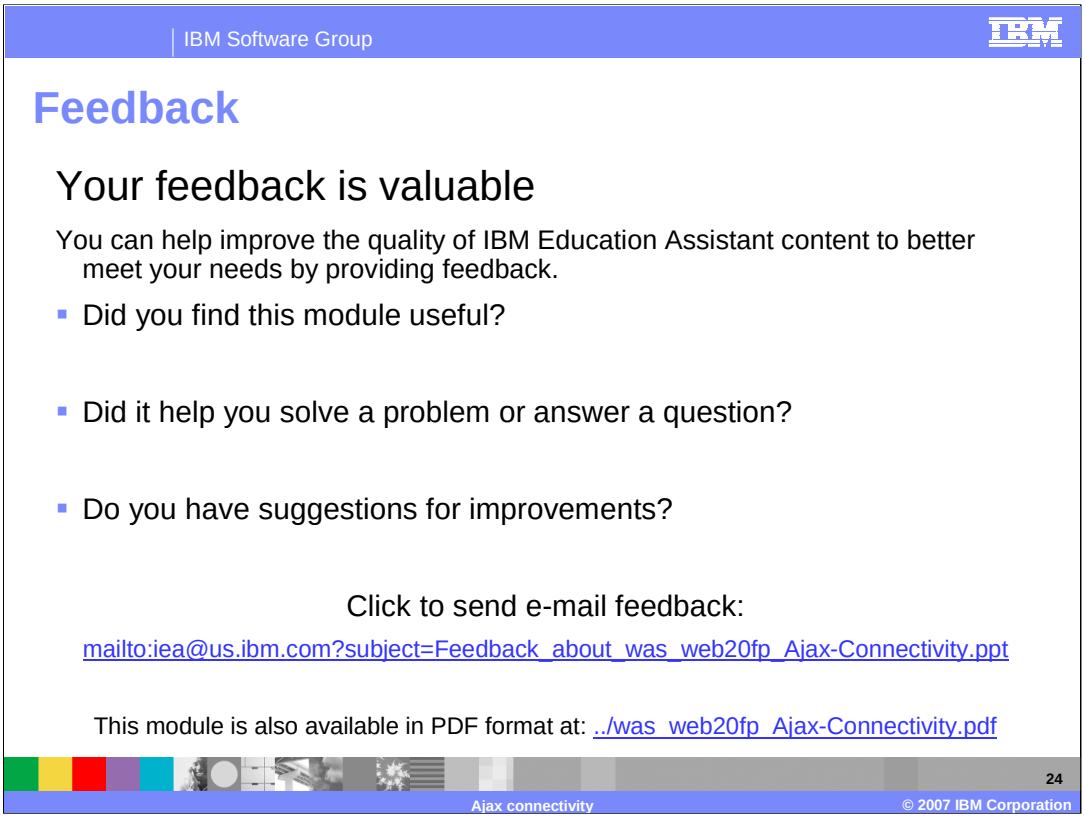
References

- Jakarta Commons Logging facility. The Jakarta Commons Logging provides a logging framework which can integrate other logging facilities such as log4j and java.util.logging.

  http://jakarta.apache.org/commons/logging/

- Information on REST: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- JSON: http://www.json.org/

These links are provided for your reference if you want more detailed information on any of these topics.



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

J2EE, Java, JavaScript, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.