

***Leverage J2EE when building
Enterprise Web 2.0 Applications***

*Technical Brief
April 2007*

Copyright © 2007 by Nexaweb Technologies, Inc. All rights reserved.

Nexaweb is a registered trademark and Internet Messaging Bus is a trademark of Nexaweb Technologies, Inc. All other marks are property of their respective companies.

No part of this publication may be reproduced in any form or by any means, or stored in a database or retrieval system, without the prior written consent of Nexaweb Technologies, Inc.

The information contained in this document is subject to change without notice. Nexaweb Technologies, Inc. assumes no responsibility for any errors that may appear.

The software/microcode described in this document is furnished under a license, and may be used or copied only in accordance with the terms of such license. All information, subsystem hardware, and applications described in this document were developed by Nexaweb Technologies, Inc. Nexaweb Technologies, Inc. retains all applicable copyrights therein.

Nexaweb Technologies, Inc. makes no expressed or implied warranties in this document relating to the use or operation of the products described herein. NEXAWEB TECHNOLOGIES, INC. EXPRESSLY DISCLAIMS THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. In no event shall Nexaweb Technologies, Inc. be liable for any indirect, special, inconsequential, or incidental damages arising out of or associated with any aspect of this publication, even if advised of the possibility of such damages.

Overview

J2EE has been a staple of enterprise development for a long time. Whether you consider it bloated or difficult to work with, many enterprises have entrusted their mission-critical applications to the platform. There are many reasons for this - standards, vendor adoption, the number of frameworks, etc. Regardless of the reason, Nexaweb works seamlessly with the J2EE environment.

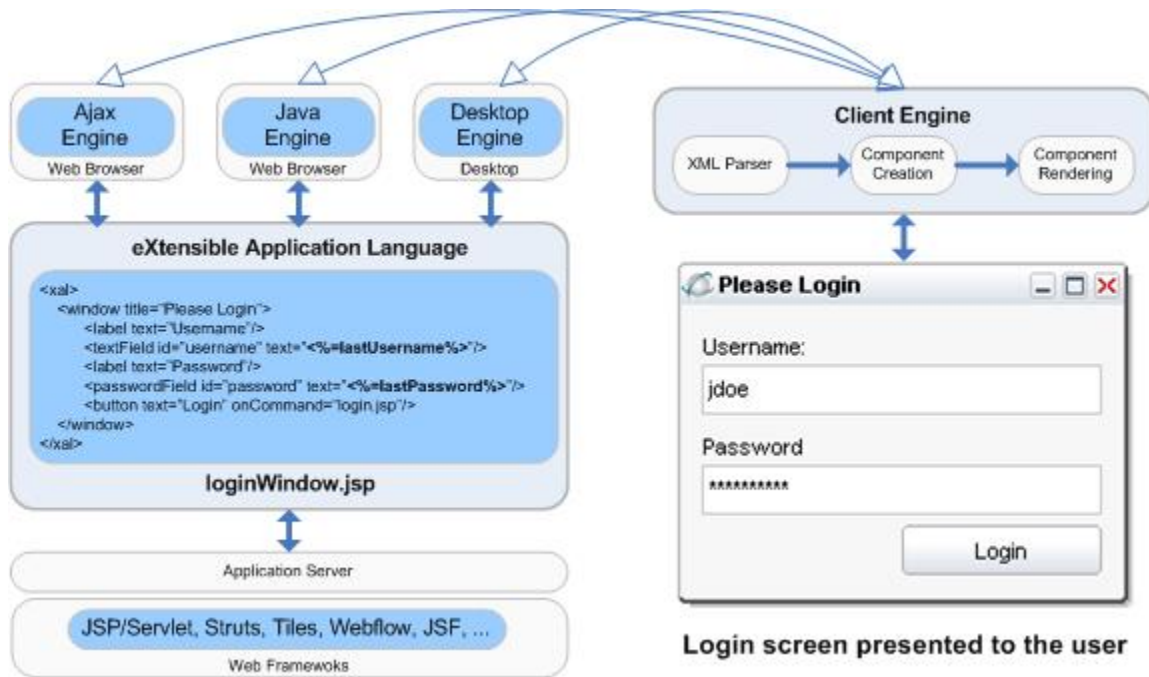
Nexaweb's Platform can be completely integrated into any J2EE infrastructure. Out-of-the-box, the Nexaweb Platform snaps into any J2EE 2.3 Servlet Container by deploying a set of resources and API's that developers can use to build next generation applications or migrate existing applications. The Nexaweb Platform has been tested and deployed on all of today's most popular J2EE application servers including Tomcat, JBoss, Websphere, Weblogic, etc.

Nexaweb's seamless integration with J2EE is the result of Nexaweb's commitment to standards. This technical brief covers each component of the Nexaweb Platform to provide a detailed explanation of how you and your development team can leverage your J2EE framework to build Enterprise Web 2.0 application with Nexaweb. .

Client-side Rendering

There are many architectural decisions that Nexaweb has made to make integration with a J2EE architecture possible, but none more important than Nexaweb's client architecture. Nexaweb's Universal Client Framework (UCF) is a set of three rendering engines implemented in two runtimes (Ajax and Java). All of Nexaweb's clients work in the same way a Web browser works - they retrieve a file from the server and based on the Markup in the file, render the application by creating all the necessary components to display the application. Any additional resources (images, style sheets, data, business logic, etc.) which are necessary to complete the display are requested from the server and loaded into memory. Client-side rendering is a distinct advantage for Nexaweb and its customers for three reasons.

1. **Use of any web-framework** – Client-side rendering allows developers to use any web framework that can output text in the XML format of -all frameworks can do this. Other products that rely on server-side rendering such as JSF or Flex need to compile or down convert the markup on the server before sending it to the client. This means that in order to take advantage of the features of the product you have to work only within the framework. There are many web frameworks that have been created for J2EE (Struts, JSP/Servlets, JSTL, Tiles, Spring Webflow, JSF) and each one has features and benefits that make it a good choice for developers. Nexaweb's architecture allows you to choose the feature from each that is best for you and use them within the Nexaweb Platform.
2. **Leverage client-side processing** – Leveraging an end-user device's processing power to display the application means that server load is decreased and that each server can support more end users.
3. **Client-side Library** – Nexaweb's UCF not only renders the client but gives developers a robust library to handle events, receive real-time data and update application state. The client-side library will handle the synchronization of data and client-side state with the server automatically. This gives the Platform the ability to perform one-way binding of data to user interface elements. With Nexaweb, if you change the data the UI will change automatically even if the data was changed on the server.



Nexaweb Client Architecture and Workflow

Server-side Rendering Comparison:

The Nexaweb architecture is different from most products and frameworks in the marketplace today. Nexaweb's emphasis on giving development organizations a choice as to how they can implement their functionality is a result of an understanding that today's Web must be more agile than a Web 1.0 framework. Frameworks such as JSF and jMaki convert the XML into HTML and JavaScript on the server using a Model-View-Controller pattern and provide a limited client-side engine. Commercial offerings such as Adobe Flex and OpenLaszlo compile XML on the server into a binary format called SWF. The need to compile the XML means that developers are restricted on how the XML is created dynamically. This approach would mean that developers couldn't use XSLT, JSP, or any other dynamic mechanism to create their UI structure.

Nexaweb's client-side rendering doesn't have these same limitations and drawbacks as discussed in the previous section. Developers can utilize their existing skills to create Nexaweb Applications. Client-side rendering also doesn't mean a lack of server-side functionality. The Nexaweb Platform offers a robust set of functions that developers can take advantage of to create applications including:

- **Messaging** – Provided with the Nexaweb Platform is a bi-directional communication bus between the client and server.
- **Clustering** – For large scale mission critical applications developers can use Nexaweb's clustering mechanism to ensure that no message is ever lost.
- **Data Services** – Developers can take advantage of Nexaweb's Data Services layer to access Web Services and databases declaratively.
- **Monitoring** – Nexaweb provides built-in application monitoring that an administrator can use to check the health of their application and diagnose any problems.

Stateful Client

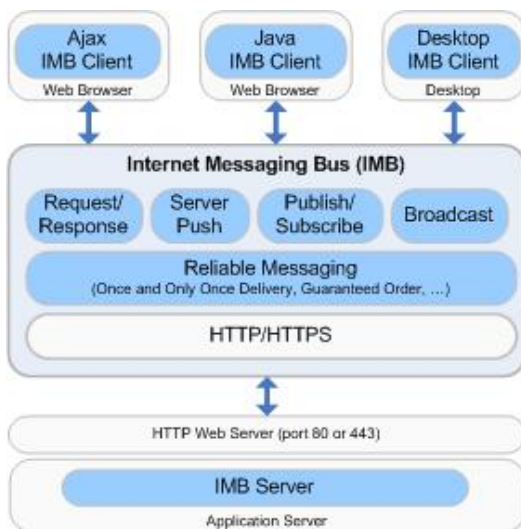
Unlike traditional HTML and Web Browser based applications, Nexaweb applications are stateful. When a browser changes pages, the newly loaded file contains the full state of the application for display. Nexaweb clients are modified incrementally, meaning the initial page that is loaded is the state of the application until it is changed via several different methods such as:

- **Client-side API's** – Each UCF engine stores the state of the user interface in a XML Dom structure that can be modified programmatically. Using client-side DOM API's developers can retrieve elements and modify their structure or attributes and any changes that are made will be automatically reflected in the user interface. For example, change the "text" attribute of a button component and the button's label on screen will change Remove all the "row" elements from a table and the table will appear cleared.
- **Server-side APIs** – Applications can be configured to synchronize the state of the user interface with the server. This makes it possible to modify the user interface by change the UI DOM on the server. The server-side API's are exactly the same as the client and any changes will be automatically synchronized with the client. The reverse is also true.
- **Declarative UI Modifications** – Nexaweb provides a declarative mechanism to modify the user interface using a syntax called xModify. Contained with-in xModify is a set of instructions that tell the client how the UI is to be modified. Developers create server-side event handlers (JSP/Servlets, Struts Handlers, ...) as they would in an HTML applications but instead of responding with the complete application state, the response is a set of xModify statements that change the client to achieve the new state.

Declarative Modifications

xModify is an important part of the Nexaweb architecture, it allows developers to process a client request on the server and respond with declarative instructions on how to change the application state declaratively. Any DOM modification that can be performed using programmatic APIs can be performed using modification instructions ([append](#), [clone](#), [create-document](#), [insert-after](#), [insert-at](#), [insert-before](#), [remove](#), [attribute](#), [remove-element](#), [replace-children](#), [replace](#), [set-attribute](#)).

Internet Messaging Bus™



Nexaweb's Internet Messaging Bus™ (IMB™) is the communications stack that the Nexaweb client uses to communicate with the server. Built on top of HTTP, the IMB makes standard GET/POST requests to the server. Developers can use client-side APIs, form submittal functionality, or directly invoke requests by placing the URL to the server resource as the handler of an event. Whatever the means used to initiate the request, the server-side side resource specified will be executed to process the event as it would be in an HTML application. Take for example a login screen; eventually the username and password will need to be checked against some credential repository on the server. To do this in a Nexaweb application, developers must only use the "RequestService" or the Form Managed Client Object to invoke the login.jsp resource.

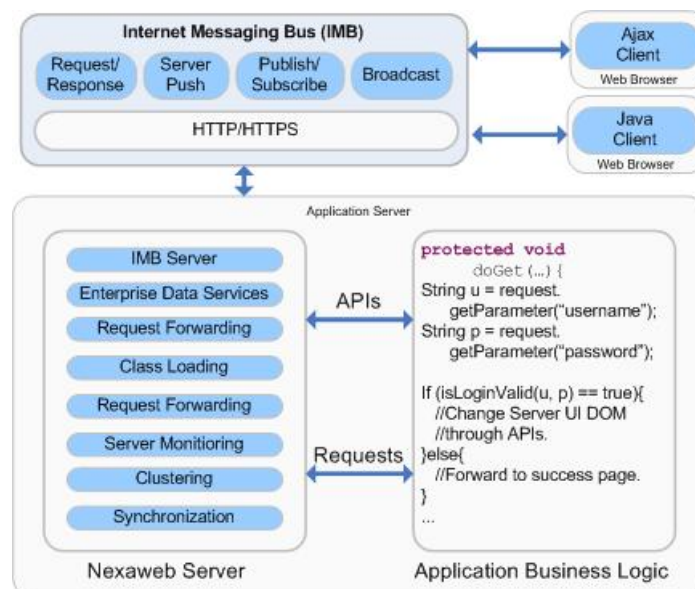
Inside the JSP page the username and password values can be retrieved using the `request.getParameter("username")` methods.

J2EE Integration

Nexaweb has several J2EE resources that must be deployed to take advantage of the full functionality of the Platform. Applications are not required to deploy the Nexaweb Enterprise Data Services (EDS) in order to utilize the Nexaweb Universal Client Framework. However, for applications that need real-time messaging, declarative binding to Web services and databases, clustering and application monitoring, Nexaweb's EDS are required. Deployment of Nexaweb's Enterprise Data Services are handled just like any other J2EE framework deployment. Nexaweb's J2EE resources must be added to the "web.xml" file and EDS Java libraries need to be placed in the Web application. Nexaweb makes this process easy for developers using two tools:

1. **Nexaweb Studio** – Automatically deploys Nexaweb into J2EE projects during the project creation process. Nexaweb Studio will also manage the process of upgrading the Nexaweb Platform and deploy the application to standard J2EE server through its integration of Eclipse's Web Tools Project.
2. **"Nexawebify"** – Is an ant task that allows developers to integrate Nexaweb into applications at build time. The Nexawebify task handles the injection of web.xml tags needed to run Nexaweb as well as the deployment of all Nexaweb resources in their correct locations. Using Nexawebify will make it easy to upgrade Nexaweb and reduce the complexity of adding Nexaweb to your application.

Once deployed, the developers can integrate Nexaweb functionality into their application using configuration files and API's specific to server development. Deployment of Nexaweb will not interfere with an existing application. This allows developers to migrate sections of an application over to the Nexaweb Platform incrementally or run a Nexaweb application and HTML application side-by-side.



Nexaweb Server Integration into a J2EE Deployment

Server-side APIs

Contained within the Nexaweb Server is a set of Java-based APIs that developers can take advantage of to interact with the services provided. Below is a list of methods on the "ServiceManager" class that allows developers to access server functionality:

- `getMessagingService()`
- `getPushConnectionManager()`
- `getSessionManager()`
- `getSharedStoreManager()`

For developers who prefer to interact with server-side components classes (Button, Window, TextField, etc.) as they would in JSF, Nexaweb provides a set of type-safe wrappers for DOM manipulation called Nexaweb Foundation Classes (NFC). These classes can be utilized to manipulate an application's presentation from the server. The Nexaweb Platform handles the synchronization of the changes with the client as it would if the developer manipulated the server-side UI DOM directly.

Migrating HTML Applications

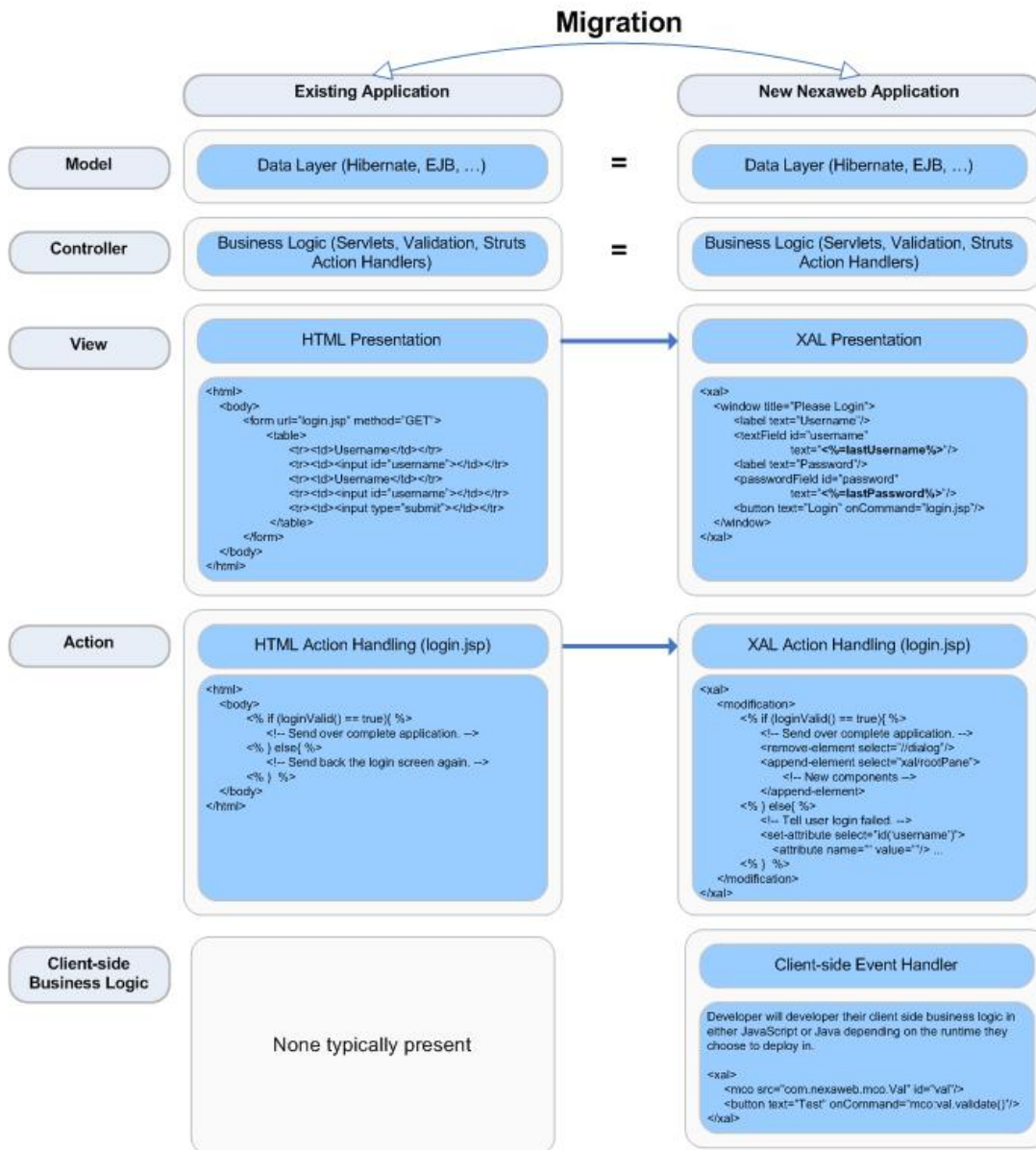
This section of the document outlines the steps for migrating an existing HTML application to Web 2.0 application with Nexaweb. The deployment of Nexaweb into your application is the first step in migrating your application. As stated in the J2EE integration section of this document, the process of migrating applications can occur incrementally. For a long time, migration of applications has been considered a long and painful process. Nexaweb makes this migration easier than other products for two reasons:

1. **Incremental Migration** – Applications often have certain areas of pain or poor usability. Allowing developers to run a Nexaweb application and standard J2EE web application together make it possible to incrementally update the portions of the application that need immediate attention. Nexaweb has helped many customers through this process, most notably Best Western who had a Web application that they deployed to 4000 sites around the world. Each site was connected to a central server over a V-Sat network that caused very long connection times. The long connection time was typically between 10-15 seconds. This connection time made the click and refresh Web paradigm extremely painful for the end-user. Best Western was able to update a section of their application that end-users need to work with throughout the day to Nexaweb which reduced the need for round trips through the use of client-side caching and other techniques. For more information on the Best Western and how you can do the same for your application, check out the case study online at: <http://www.nexaweb.com/bestwestern.html>
2. **Leverage current assets** – To completely change the architecture or scrap existing application assets is a non-starter for many development projects. Nexaweb, through its architecture and resources, makes it possible to leverage a good portion of existing application assets such as server-side business logic, data model, and client-side components. For many applications the data model and server-side business logic can remain untouched. Using Best Western as an example again: the application they migrated to Nexaweb leveraged their existing PL-SQL pages and J2EE resources, which drastically reduced the time needed to migrate the application.

Applications that are being migrated will need to change something and that thing is the output of their requests from HTML to the eXtensible Application Language (XAL). Typically this is the

Nexaweb's and J2EE

application developer's biggest effort in creating a Nexaweb Application. XAL provides a complete language for creating applications declaratively. Layout, Event Handling, Data Binding, and a complete set of widgets are included in the language. Most developers will spend their initial time learning XALs layout characteristics in order to get their application to look the way they want. Included in the language are several layout panes that can control components position and size.



The image above contains a diagram of the work involved in migrating a J2EE HTML application to a Nexaweb application. The picture shows how to handle the conversion of a simple login page to a Nexaweb login screen. Any "=" sign means that the assets are the same after migration and the arrow shows what assets need to be changed.

Design Considerations

- Where are going to handle the event? Nexaweb allows developers to choose client-side or server-side event handling. Developers should choose which events belong where to give the user the best possible experience.
- What Web framework should we use? Nexaweb's gives you the freedom to choose the framework that is best suits your team and that will meet the application's requirements. There are many Web frameworks that can make your application development simpler, but depending on your requirements and team skill sets, choosing the right framework will not only make development simpler, but also faster.
- What Nexaweb APIs are going to be needed on the server to make your application development easier? Nexaweb offers a complete set of server-side APIs to interact with the Nexaweb Platform. Depending on the needs of the application, developers can choose which to use in there development.
- Will you use the shared UI document and/or Type-safe Nexaweb Foundation classes to develop in a JSF-like manner? Nexaweb allows developers to use either a declarative programming model similar to HTML or an Object Oriented programming model. Depending on the skill sets of the team, either method may be used.
- If you are migrating an application, are you going to migrate the complete application or start with the problem areas? Depending on the approach you may need to install the Nexaweb Platform into your Web application, using either Nexaweb Studio or the Nexawebify ant task. Nexaweb will not interfere with the existing application in any way.
- Will you need to deploy a clustered solution? Applications that have 1,000's end-users or can't tolerate a single point of failure can use Nexaweb's clustering functionality to handle these requirements.

Conclusion

Whether you are trying to upgrade an existing application or starting from scratch, Nexaweb can be easily integrated into your normal J2EE application workflow. Nexaweb has been deployed in over 4000 enterprises using a J2EE architecture, each one of which requires the most reliable and robust infrastructure. To get started go to Nexaweb.com and click the "Download" button.