ENHYDRA
JAVA
APPLICATION
SERVER
ARCHITECTURE

FOR
OPEN
SOURCE

http://www.enhydra.org

OVERVIEW

This paper is written for the technical audience tasked with performing an in-depth review of the Enhydra Java Application Server, Enhydra, focusing on its genesis, architecture, functionality, and planned evolution. The original design center of Enhydra captures the motivations behind the guiding philosophy of the architecture. Following a functional breakdown of Enhydra, an example is used as a walk through of the typical execution flow of an Enhydra application. The paper concludes with a roadmap discussion to address the bridge from the existing 2.0 implementation to the proposed enhancements of version 2.1.

ENHYDRA JAVA APPLICATION SERVER GENESIS

The design strategy and roadmap for the Enhydra Java Application Server was driven by a specific design center representing the intended Lutris Technologies Consulting audience. Lutris Consulting required an N-Tier Web application development and run-time environment that would meet the needs of the widest range of customer requirements possible. The Enhydra Java Application Server needed to support the value proposition that it establishes an application environment based on the same standards that have made the Internet the success that it is today. Furthermore, Enhydra must take into consideration strategic, emerging standards that promise to make it the best possible platform for a long term deployment strategy capable of supporting additional services, minimizing the need for throw-away work.

CONSULTANTS, IT AND DEVELOPERS

The marketing requirements for the Enhydra Java Application Server focused on a development community characterized by consultants, including Lutris Consulting, the IT environment and product developers. The attributes listed below address the special needs of this community.

An Extensible Foundation

- Ensure general applicability to the widest range of application scenarios.
- Allow customers and consultants to build custom-instances of standard functionality (e.g., special version of a session manager service).
- Provide basic functionality that may be extended, adapted or enhanced over time.
- Support client/server strategies from 2 tier to N-tier application design.
- Support development that allows moderate to low-level access to underlying server technology. (Critical to ensuring a robust server application.)

Support for Best-of-Breed Tools

- Minimize negative side effects (training, flexibility, choice, limited hiring pool) of a proprietary application development environment.
- Give HTML designers and development engineers the ability to choose the best of off-the-shelf, 3rd party tools.
- Maximize existing knowledge of user interface designers and their preferred tool sets.
- Reduce training requirements.

Internet Standards for Minimizing Costs and Maximizing Flexibility

- Maximize support of de-facto, defined and emerging Internet standards to achieve important benefits.
- Maximize the customer's ability to maintain and manage Enhydra applications.
- Maximize the customer's ability to recruit and attract development staff from the largest pool possible, including consulting firms or college new hires.
- Maximize the customer's ability to transfer knowledge from project to project.
- Maximize the customer's ability to train staff on non-proprietary concepts of the Internet.
- Maximize the ability to build applications that scale from small to significant size using structured programming languages.
- Support appropriate weight client strategies.
- Support customer's ability to choose widest range of hardware, database and 3rd party components in general.

Ready to Go Development

- Provide demo applications for quick modification for customer demonstration and training.
- Support development and runtime execution in non-network environment, including Windows 95 laptop.
- Minimize time required for project ramp-up by addressing issues including source code organization, versioning and management.
- Support team development.

DERIVED BENEFITS

- In reality, not every design consideration is the result of an explicit goal. There are recognized benefits that have resulted from design decisions, such as the use of Java, the have gone beyond stated goals. Lutris has found Java delivers additional value, including.
- Inherent organization of code as enforced by the Java Development Kit environment
- Enhanced documentation through comments and use of JavaDoc mechanism
- Introduction of fewer bugs. Code tends to run after first compile.
- Although developed with the Sun JDK, Enhydra applications can use any third party JDK, such as Symantec Café or Microsoft Visual J++.

ENHYDRA COMPONENT OVERVIEW

Enhydra Java Application Server 2.0 consists of the Enhydra Java Application Server Runtime Environment (known hereon as simple Enhydra) and the Enhydra Java Application Server Development Kit (Enhydra/Dev). These products allow developers to construct and execute multi-tiered applications. Multi-tiered applications are now possible because of the standard interfaces in Java and other Web technologies. Using traditional mainframe or client/server technology developers can only construct single or two-tiered applications. With the advent of Web technologies and standards, it is possible to partition applications into multiple tiers.

Multi-tiered Applications

Most Web-based applications consist of three fundamental types of components:

Presentation Objects contain the logic that presents information to an external source and obtains input from that source. The presentation logic generally provides menus of options to allow the user to navigate through the different parts of the application, and it manipulates the input and output fields on the display device. Frequently the presentation component also performs a limited amount of input data validation.

Business Objects contain the application logic that governs the business function and process. Business objects are invoked either by a presentation component when a user requests an option, or by another business function. The business functions generally perform some type of data manipulation.

Data Objects contain the logic that interfaces with a data storage system, such as database systems or hierarchical file systems, or with some other type of external data source such as a data feed or an external application system. Business objects invoke Data Objects to save persistent state.

ENHYDRA JAVA APPLICATION SERVER RUNTIME ENVIRONMENT

Enhydra combines a runtime environment for Java servlets, a manager for running Enhydra applications, connectors to common database products, a wide range of connectivity options for Web servers, and administration and debugging tools.

What is a Servlet?

A natural question here might be "What is a servlet?" A servlet enhances an HTTP server by enabling request/response services. When a client sends a request to the Web server, the Web server can send the request information to a servlet and have the servlet construct the response that the server sends back to the client. In addition, a servlet can be loaded automatically when the Web server starts or it can be loaded the first time a client requests its services. After loading, a servlet continues to run, waiting for additional client requests.

Why is this significant? Simply put, it allows you to leverage your Web server to take advantage of servlets--a Java class that dynamically extends your server's function.

Servlets provide enormous advantages:

- Servlets run in the context of the Enhydra server process. This saves the time and effort to create a new CGI process to serve an incoming request. Running servlets in the Enhydra environment allows them to be dynamically managed and monitored.
- Servlets are written in Java, a stable, secure language.
- Servlets let you create complicated, high-performance, cross-platform Web applications without having to create multiple versions of them. You can use Enhydra with your current Web server to take advantage of Java's "write-once, use everywhere" capabilities.

Servlets are not specific to Enhydra. The Servlet API is a standard extension to the Java Platform and a number of companies have implemented the Servlet API.

Servlets and Applications

In Multi-tiered applications (counting the browsing device as tier one), Servlets generally serve in the second tier to implement presentation logic. Because of the static nature of html interfaces used by Web browsers, it is impossible for dynamic displays to be implemented using standard HTML. The Enhydra Java Application Server introduces the notion of presentation objects to solve this problem. Presentation objects allow information to change dynamically on a Web page. An Enhydra Application is a servlet that contains presentation objects.

Note that in complex Multi-tier applications, the servlet and thus Enhydra application may communicate with additional tiers where the business logic is located. In distributed systems this logic may be spread across the enterprise as Enterprise Java Beans (EJBs) or CORBA objects. In smaller scale solutions or ones in which the HTML browser is the only client, all logic, presentation, business or otherwise, may be encoded directly into the Servlet. In these cases the Servlet constitues the entire middle tier.

Connectivity Options

Because of the large number of Web servers deployed in the enterprise, Enhydra provides a number of options for connecting to Web servers. Enhydra connects to Web servers by WAI, RMI, or CGI-relay. Enhydra can function as a standalone Web server directly accepting HTTP requests and offering the fundamental Web server functionality (e.g. GET, POST, MIME, HTTP-BASIC-AUTH, etc).

Database Connectivity

Enhydra uses JDBC to connect to relational databases. JDBC provides a level of abstraction so the actual database engine (e.g. Oracle, Informix) will be invisible to the application developer. Enhydra currently supports JDBC level 3 and 4 drivers with Informix, Oracle, Sybase, MSQL, or other standard implementations of JDBC.

Administration

Enhydra provides a graphical tool for configuring the applications, Servlets, the channels (methods of connection) they use and their security. The administration tool also displays statistics kept by Enhydra to help administrators understand the current state of Enhydra applications.

ENHYDRA JAVA APPLICATION SERVER DEVELOPMENT ENVIRONMENT

The Enhydra Development Environment supports essential aspects of application development such as, source control, source tree layout, and a full debugging environment. Enhydra DE also provides the Enhydra Jolt language for building dynamic HTML pages with embedded Java and the Enhydra Jolt compiler for creating compiled presentation objects. Enhydra Jolt eliminates the need for using less scalable and maintainable scripting languages. The highly extensible nature of the Enhydra Java Application Server gives the developer the ability to augment or replace functional components, such as session management, to the most esoteric customer legacy requirement. A development console gives the developer the ability capture debug information, such as URL requests and responses and Java traces.

Enhydra Jolt

Enhydra Jolt is a technology[1] that enhances the static presentation capabilities of HTML. Its design supports the systematic embedding of Java functionality within an HTML page. Enhydra Jolt files, like HTML files, are simple ASCII files; Enhydra Jolt files distinguish themselves from standard HTML files by their "jhtml" extension.

Within a Enhydra Jolt file, Java code supplies dynamic content, while a non-programmer can easily edit existing static content with any best-of-breed Web-authoring tool. The Enhydra Jolt syntax allows for the integration of Java with HTML, or the modular separation of HTML templates and Java libraries into separate files.

There are two categories within the Enhydra Jolt syntax: <JOLT> tags and Enhydra Jolt Fields. The <JOLT> tags, with their various attributes, isolate Java sections and conditionally insert static HTML content. Enhydra Jolt Fields support the ability to access automatically decoded URL arguments, or to embed values directly from a Java object. Enhydra Jolt files are the preferred building blocks of Presentation Objects. The *Enhydra Joltc* compiler compiles Presentation Objects. These Presentation Objects execute in the Enhydra application server that may exist standalone or connected to any Web Server.

Why Enhydra Jolt?

Simply, Enhydra Jolt extends HTML to include dynamic display information. It does this by embedding Java in the HTML code. The Java code embedded in the HTML executes on the server and not in the context of the Web browser. This alleviates the performance and security concerns of Java executing in Web browsers. With Enhydra Jolt, it is also possible to have blocks of regular HTML code invoked as a Java method or from a <JOLT CALL> tag. From a development perspective, graphic designers can use the design tools of choice and let server developers supply the code for the dynamic display information. This division of labor allows each person to focus on their appropriate discipline and increases productivity.

Enhydra Jolt was defined and identified with the goal of making Java the language of choice for server-side dynamic HTML. This reduces to one the number of languages used to build the server portion of an application. The result is that Enhydra applications can leverage the full use of a structured language, from presentation through the business and data logic.

[1] By Lutris Technologies Inc., sponsor of the Enhydra initiative.

Enhydra Jolt Portability

Enhydra Jolt and the generation of presentation objects make it possible to use the Enhydra Jolt extensions in the design of any thin client application on any third party Java application server. Developers can use Enhydra Jolt for structured design of dynamic HTML on any server capable of loading and executing Java classes.

Enhydra Jolt Compiler

The Enhydra Jolt compiler using standard Java tools for its implementation. Specifically, the Enhydra Jolt compiler is generated by JJTree (a parse tree preprocessor for JavaCC) and JavaCC (the Java Compiler Compiler) from the Enhydra Jolt grammar.

The Enhydra Jolt compiler takes a file that contains either Enhydra Jolt Tags or Enhydra Jolt Fields and generates a Java class file. The following are the steps the Enhydra Jolt compiler takes to complete its task.

1) Enhydra Jolt HTML file parsing (.jhtml -> internal parse tree) The Enhydra Jolt Parser reads the .jhtml file and builds a parse tree representation of the file.

2) Java source generation (internal parse tree -> . java) Enhydra classes generate the code that traverse the parse tree and output Java source code for each of the tree nodes.

3) Java class compilation (.java -> .class) The Java source file is then compiled into a Java class file by JavaC (the Java Compiler).

Application Development Environment

Often times, there is a great deal of energy spent on defining the development environment for any software engineering product. Through Lutris' consulting experience, we have developed a simple, but powerful development environment for constructing Enhydra applications. This development environment encourages a methodology that Lutris has found to be highly successful for rapidly building and deploying Web-based applications. However, the application development environment is not required to use the technology provided by Enhydra. The interfaces to the Enhydra components can fit into any development environment. Enhydra applications essentially implement the standard servlet interface, so any Java IDE can be used in their development.

The Enhydra/Dev environment consists of the following parts:

- A layout for a source tree that easily lends it self for building a deploying Enhydra applications.
- An Unix-like portable environment for executing makefiles and providing basic tools for software development.
- A source version control tool for group development. The tool is CVS and is in widespread use.
- **Enhydra Application Wizard** for jump starting application development by producing the skeleton files in the source tree. Once these files are in place, a developer can use CVS to check the file out of the source tree and code the appropriate logic.
- **Enhydra Jolt Compiler** for quickly building Presentation Objects.
- **Enhydra XMLC Compiler** for building structured Object-Oriented presentation objects for HTML or XML applications. *This technology is in the final stages of development and will be released shortly.*

- **Enhydra Data Object Design Studio (DODS)**. A GUI for designing and automatic construction of Enhydra Data Objects.

- Example Applications. Many engineers learn a programming environment by viewing example applications.

Development Console

The Enhydra development console allows an individual to monitor a running Enhydra application or servlet. It acts as a systematic viewing window through which a developer can dissect all application transactions.

The development console uses an Enhydra filter for its implementation. This filter intercepts and records the servlet, servlet request and servlet response API calls. Using a browser interface, the developer can then query this transaction history.

The console conveniently divides the information into the following categories:

- **Request Information**. This breaks up the http request into component headers, decodes URL arguments and POST data, and displays other http information such as cookies.

- **Servlet API Trace**. Each call to a servlet API is recorded together with the response. This information is presented with elapsed execution time.

- **Response Information**. This breaks up the http response into headers set by the application, http cookies that are set and displays the MIME data type and decoded data.

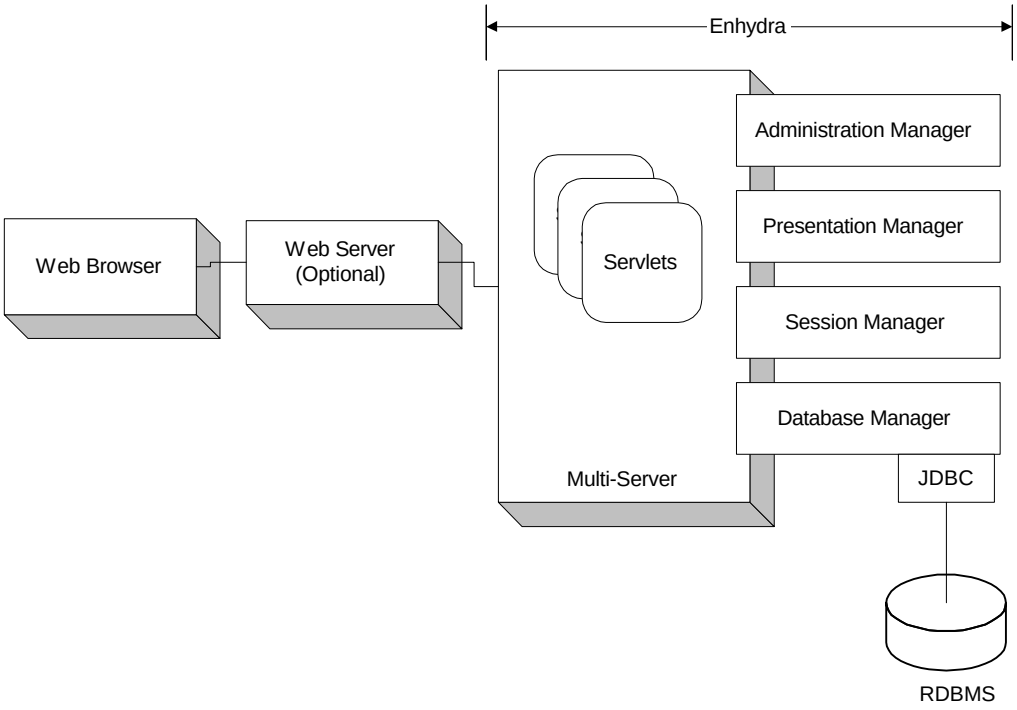
- **Session Information**. This displays the content of the Session object both before and after the request is processed. HTML renderable information is displayed in a HTML list, the names of other object types are simply displayed.

ENHYDRA 2.0 ARCHITECTURE

OVERVIEW

Enhydra consists of a loosely coupled set of components that work in harmony to allow multi-tiered applications to execute in a high-performance environment. All of the components have a well-defined, public set of interfaces and standard implementations. This means alternate implementations and extensions to the components are possible, without effecting existing applications. This loosely coupled model allows consultants and developers to modify the behavior of the Enhydra Java Application Server in order to met customer or product needs.

The following figure illustrates the high-level architecture for Enhydra.



ARCHITECTURE COMPONENTS

The following sections describe each of the major components for the Enhydra Java Application Server.

Enhydra Multiserver

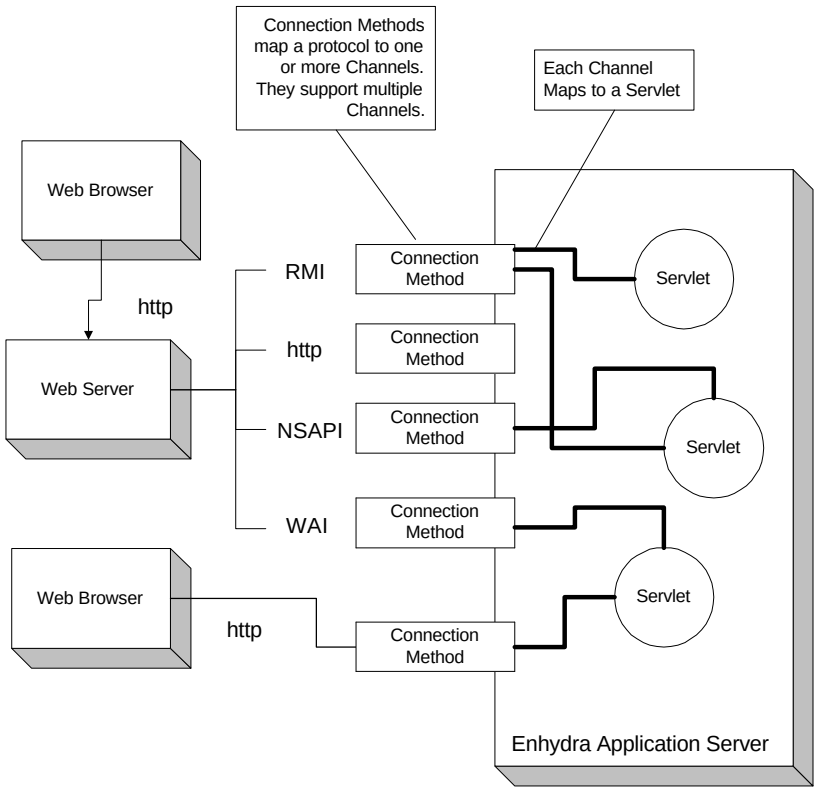
The Enhydra Multiserver manages the lifecycle and connectivity for servlets, including Enhydra Applications and third-party servlets. It is also responsible for managing the server configuration file, which defines the initial state of Enhydra.

Enhydra also provides a custom class loader with the Enhydra Multiserver. A single instance of an Enhydra class loader starts each servlet in the Multiserver. This one-to-one correspondence of servlets and class loaders allows new servlets and applications to download and execute without having to stop the entire server. Simply shutting down the class loader for a corresponding applet and then restarting it will cause a different version of the servlet to become available.

Connectivity

Enhydra can connect to Web servers via a number of protocols or directly to a Web browser via HTTP. Because many servlets are designed to be reusable components, the Multiserver allows any number of connections to any number of servlets. The Enhydra Multiserver also supports a number of connectivity options (WAI, RMI, CGI-Relay, and HTTP). This means a single servlet can be connected to multiple Web servers, using different protocols. This functionality greatly increases the reusability of servlets executing in the Enhydra environment. In addition, by using WAI or RMI it is possible to run distributed applications. The following figure shows an example configuration of connections and servlets.

Connection Methods and Channels



Connection Methods and Channels are the components that provide connectivity to servlets.

A ConnectionMethod is how requests get into the server from the outside world. ConnectionMethods are where the networking for Enhydra occurs. Each ConnectionMethod maintains an ordered list of Channels. Channels act as a logic port for the servlet to which it is connected. Each channel has an ID used to identify and refer to the channel. Each channel can be enabled or disabled. When disabled, connections are refused, and not passed on to any servlets. When enabled, connections are allowed to the destination servlet. Each channel uses a list of TransactionFilters for debugging and logging.

When a ConnectionMethod is started, the initialize() method will be called before any other methods. The servlet IDs for the channels are only valid in the ServletManager object provided to initialize(). Filter IDs are only valid in the FilterManager passed in to initialize().

The destroy() method will be called when the ConnectionMethod terminates. Any external network exposure must be eliminated in an implementation of destroy(). For example, any ServerSockets are closed, or any WAI registrations should be unregistered. After destroy() is called, no further calls can be made to the ConnectionMethod object.

Filters

Filters log and debug requests and responses to and from servlets. Filters wrap themselves around servlet requests and responses. That is, they take in an object and return a new object that is used in place of the original object. In almost all circumstances, the new objects should pass all calls through to the original object. Normally the design filters is such that they do not modify the behavior of the servlet. If a filter did not pass a call through to the original object, it would break the behavior of the servlet. Normally filters simply take note of the call, perhaps logging it to a file, perhaps printing a message to an OutputStream, then pass the call on.

The fact that filters do not change the behavior of the Servlet means a single servlet may have multiple filters associated with it. In addition, the multiple filters will be unaware of each other. If developer produces a filter that is not transparent, an administrator must be very careful of the order filters are composed. An example of a non-transparent filter is one that logs the calls to a servlet, but does not pass any calls on to the actual Servlet. Clearly this filter would need to be the first filter applied, so it is wrapped closest to the real servlet. Any filters between this hypothetical filter and the servlet would never be called.

When filters are being displayed to the user, toString() will be called on them, and the resulting string will be shown to the user. Therefore it is recommended that filters override the toString() function from the Object class with a method that returns a meaningful string.

Presentation Manager

The presentation manager handles the loading and execution of presentation objects in the context of an Enhydra application. The presentation manager transforms the name of the presentation object into a URL, uses the specified class loader to load the presentation object, and then executes the presentation object by executing its run() method. There is one instance of a presentation manager per Enhydra application. The presentation manager can also cache presentation objects in memory and any associated files that are part of the application. It uses the Enhydra class loader to get these other files.

The presentation manager is contained within an instance of httpPresentationServlet class. This design allows a servlet to have multiple presentation objects managed by the presentation manager. The presentation manager also manages the resources necessary to execute the presentation objects.

The presentation manager also provides the key with which the session manager uses to locate a session. The key either is a cookie or generated by URL rewriting.

Session Manager

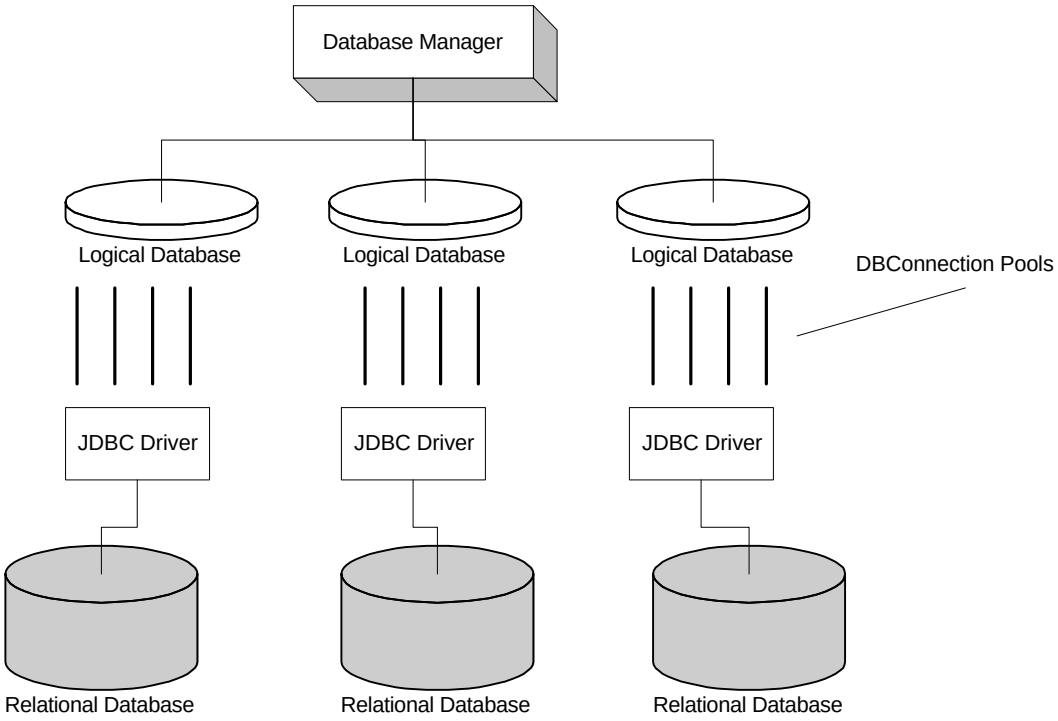
The session manager and session objects provide a flexible, lightweight mechanism that enables stateful programming on the Web. Enhydra provides a general implementation of session management that serves as a basis for more sophisticated state models. A session is a series of requests from the same user that occur during a time-period. This transaction model for sessions has many benefits over the single-hit model. It can maintain state and user identity across multiple page requests. It can also construct a complex overview of user behavior that goes beyond reporting of user hits.

Session management gives servlets and other server-side applications the ability to keep state about a user as the user moves through the application. Enhydra maintains user state by creating a Session object for each user. These Session objects are stored and maintained on the server. When a user first makes a request to an application, the user the session manager assigns a new Session object and a unique session ID. The session ID matches the user with the Session object in subsequent requests. The Session object is then passed as part of the request to the servlets that handle the request. Servlets can add information to Session objects or read information from them. After the user has been idle for more than a certain period, the user's session becomes invalid and the Session Manager destroys the corresponding Session object.

Database Manager

The database is the component that manages a pool of connections across any number of logical databases. Logical databases hide the nuances across different JDBC and database implementation and contain a set of connections. These connections are shared across thread boundaries. The connections are responsible for maintaining a connection to a JDBC driver and the state of a database connection, including the current statement(s) and result set(s) that are in progress. The following shows the overall model of the database manager and relationship to actual databases.

Queries and Transactions



The classes DBQuery and DBTransaction implement the query and transactional semantics for an application. DBQuery allocates connections from the database connection pool, ensures the integrity of those connections, and manages result sets after a query executes. To implement transaction semantics, an application creates instances of the DBTransaction. The application invokes methods to insert, delete, or modify objects in the scope of the transaction. The transaction completes by either a commit or a rollback method invocation on the instance of the DBTransaction class.

Administration Manager

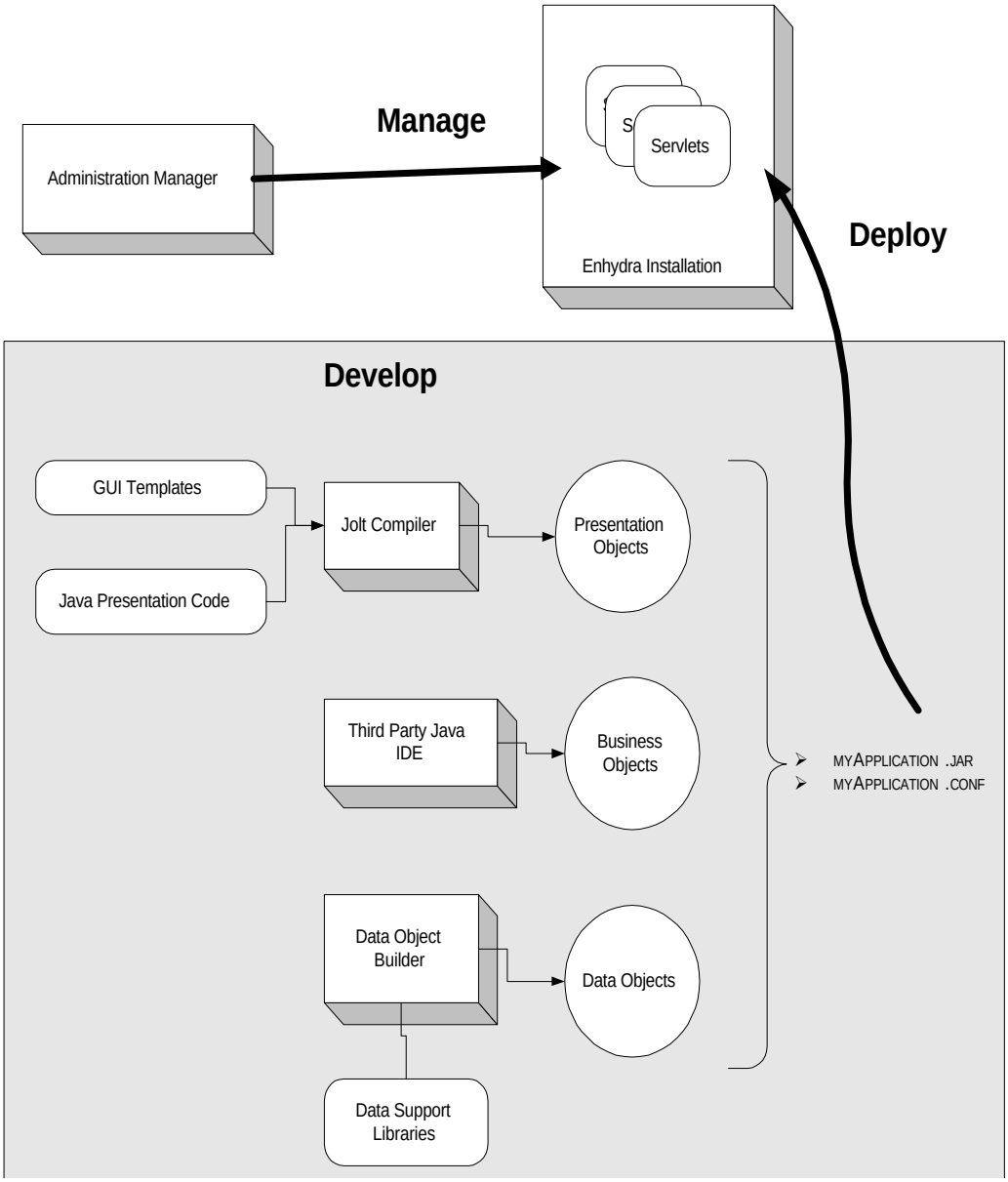
The administration manager is graphical tool that allows a system manager to configure and monitor an instance of Enhydra and associated applications. All configuration information for Enhydra and Enhydra applications is stored in configuration files. When Enhydra starts, it reads these configurations files and starts the server process and any specified applications. Once the instance of Enhydra is running, the administration manager is able to perform management operations. All management operations work in the same manner; the active state (including resource parameters) of Enhydra, an application, or servlet is changed and the change may be saved in the configuration file.

The following are the management operations performed by the administration manager.

- **Start/Stop** applications or servlets that are currently executing in an Enhydra installation.
- **Add/Remove** an application or servlet from an Enhydra installation.
- **Modify** the operational attributes for Enhydra, an application or servlet.
- **Check Status** on the active state of Enhydra, an application or servlet. This includes properties like the number of active applications, the default time for a session, and the size of the database connection pool.

ANATOMY OF AN APPLICATION

There are three stages in the life of an application: development of the application, deployment of the application, and the management of the application. With the Enhydra technology line, the lifecycle of an application is well-defined and easy to implement. The following figure illustrates the lifecycle of an Enhydra application.



DEVELOP

An Enhydra application has three tiers of logic: presentation, business, and data. Enhydra/Dev provides tools to help construct objects for these layers as well as integrating with best-of-breed tools for other aspects of application development.

The presentation tier typically consists of a number of HTML pages generated with any number of page layout tools. The HTML can embed JOLT tags or fields in the HTML. Since the Enhydra Jolt elements are valid HTML fields, page layout tools will not take exception to their presence. Enhydra/Dev also provides templates for illustrating for developers some models for mixing HTML and Enhydra Jolt.

The business tier typically consists entirely of Java code and encapsulates the actual business logic of the application. Other than our recommended source tree layout and group development tools, Enhydra/Dev provides no other functionality for this layer. It is expected that a Java IDE will be used to develop business objects.

The data tier encapsulates the persistent data needed by the business tier. The data needed by the business logic is usually stored in a relational database. Enhydra provides the database manager classes for retrieving and querying that data. Enhydra/Dev also provides a set of libraries and a tool built on that library to ease the mapping of relational data into objects.

DEPLOY

An Enhydra application consists of two files. A JAR file that contains all the objects that make up the application and any additional resources and a configuration file that contains information relevant to the startup and resource parameters necessary for the application. Placing these files in a well-known location to an instance of Enhydra, is the deployment model for servlets and Enhydra applications.

By using the Enhydra development tools and methodology, it is simple to generate the files that comprise an Enhydra application. It is not imperative that you use the tools supplied by Enhydra/Dev, but it makes the process of generating the deployment files very easy.

MANAGE

Once an application deploys to an installation of Enhydra, the administration manager then controls the state of that application. The administration manager adds the application into the Enhydra installation, modifies its resource parameters (e.g. database connection pool size), and controls the lifecycle of the servlet of Enhydra application.

ENHYDRA JAVA APPLICATION SERVER ROADMAP

The following presents the suggested list of functionality provisionally planned over the next couple of releases of the Enhydra Java Application Server.

ENHYDRA 2.1 MAINTENANCE RELEASE

- Introduction of the Enhydra XMLC compiler technology.
- Full support for the Servlet 2.0 interfaces.
- Refinement of Enhydra DODS (the Graphical Tool for creating Object to Database Mappings).

ENHYDRA 2.1

- Scalability enhancements. Permit clustered instances of Enhydra to communicate to each other. This also includes allowing sessions to migrate between Enhydra instances.
- Movement from Enhydra Business Objects to an Enterprise Java Beans (EJB) server framework.
- Database Query Enhancements to support generic queries.
- Application Deployment Scalability. Allow application to be deployed across clusters and have the entire application replicated.
- Improved Enhydra Application Wizard.
- SNMP Management. Instrument Enhydra using SNMPv1.
- Support for HTTP/S.
- Servlet Grouping. This allows a group of servlets to share session and allows for reuse of third party servlets.

CONCLUSIONS

Lutris Technologies, http://www.lutris.com, the creator of Enhydra, has used the technologies described in this paper in a number of successful, large-scale consulting engagements. Our experiences in rapidly building and deploying these applications have driven our technology. Lutris believes this is the best way to construct technology in the Web-based area. Our technology will continue to expand to embrace standard interfaces, such as Enterprise Java Beans, while taking a pragmatic approach to delivering software that our customers can leverage.

ABOUT LUTRIS TECHNOLOGIES

We are Lutris Technologies, a consulting services company specializing in the design and development of custom Web business solutions for medium to large corporate IT and MIS. Built from extensive industry experience in high-end object-oriented client/server development and distributed systems management, Lutris

consulting represents the expertise necessary to build quality mission critical applications for Fortune 100-style reliability and scalability.

Dynamic corporations are demanding highly adaptive solutions that support their ability to participate and excel in highly competitive markets. Lutris believes that the unique union of Internet standards, object technology and client/server design at last makes adaptive solutions supporting dynamic business processes available to all businesses, independent of their past platform purchase decisions.

Lutris offers a complete consulting service package, in the form of a virtual engineering organization, to deliver business solutions that combine the best of the Internet potential with the pragmatic design, development and deployment requirements of corporate MIS and IT.