



ACHIEVE 2.0 System Architecture

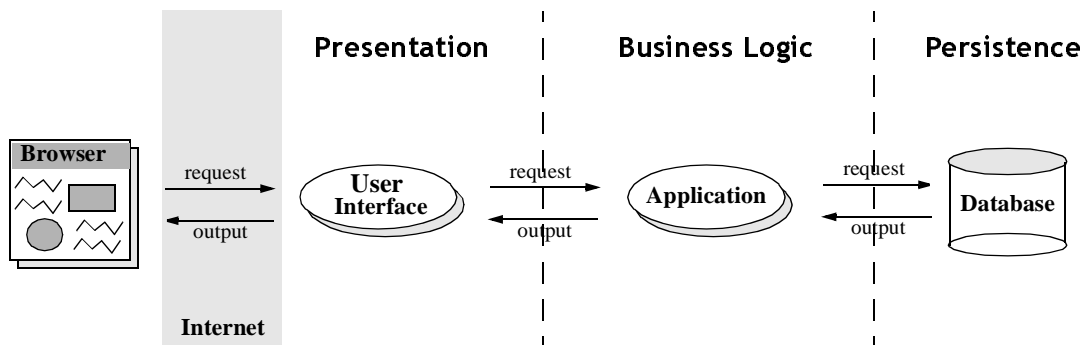
Introduction

ACHIEVE 2.0

ACHIEVE captures and delivers student performance indicators to parents and educators on demand. To provide this level of service, ACHIEVE is deployed using the Application Service Provider (ASP) model; it is online, web-based, and remote hosted.

ACHIEVE employs a standard 3-Tier Architecture:

- **Presentation Tier:** Processes requests and presents results that are rendered on the browser.
- **Business Logic Tier:** Provides an interface to the database and performs work-flow tasks.
- **Persistence Tier:** Stores data and maintains data integrity.



BENEFITS

By utilizing an ASP model and 3-tier architecture, ACHIEVE provides the following benefits from both customer and development standpoints:

- **24x7 Availability:** Teachers can log on to ACHIEVE at their convenience, 24 hours a day, 7 days a week.
- **Transparent Distribution:** New features and updates can occur on a regular basis, which are seamless to the user. There are no software upgrades that schools must install.
- **Abstraction:** A 3-tier architecture allows for easy maintenance and implementation of new features by isolating and encapsulating application functionality.
- **Scalability:** ACHIEVE is designed such that it can serve thousands of concurrent users.

Presentation Tier

SUMMARY

The Presentation Tier is the User-interface (UI), or front-end of the application. It is a thin client, created using Java Server Pages (JSPs) that are compiled at run-time into HttpServlets. The HttpServlets process user requests and output the resultant data as HTML text, which is sent through the Internet to the user's browser.

ARCHITECTURE: FROM JSP TO HTTP SERVLET

When the user performs some action in ACHIEVE, clicking a link or button, the request for a specific JSP file is sent to the web server. A JSP file is made up of HTML code that describes how to render the information in the browser, and Java code that makes requests to the Business Logic Tier. ACHIEVE is made up of hundreds of JSP files that do everything from log users on to ACHIEVE, to save application data, to display customized console screens.

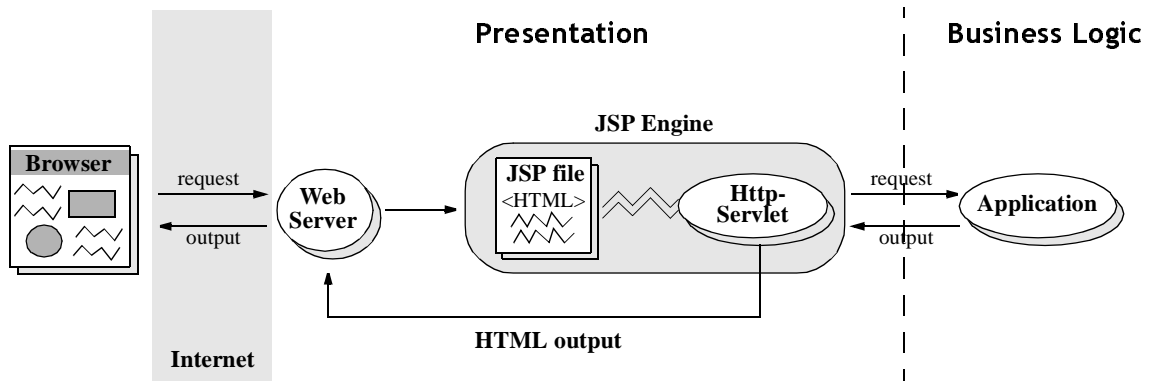
The web server, which handles incoming requests to ACHIEVE, forwards the request for the JSP file to a JSP Engine that parses the file into java code and compiles it into an HttpServlet. JSP files are parsed the first time they are requested, and thereafter when the JSP file changes.

The compiled HttpServlet is a Java class that processes the user's request. It makes calls to the Business Logic Tier and when the requested data is returned or the request processed, the HttpServlet generates HTML text to present the results on the user's browser. The resultant pages are comprised solely of HTML and Java Script, thereby leaving the client suitably thin.

BENEFITS

JSP files offer flexibility to the user and the development team:

- **Dynamic Content:** HttpServlets allow personalized data to be displayed depending on the request and the user. For example, personalized data is displayed on the teacher console for each user.



- **Performance & Scalability:** HttpServlets can efficiently handle many concurrent user requests, therefore providing improved performance and scaling to handle increased user traffic as opposed to other technologies.
- **Easy Maintenance & Development:** Remote-hosting allows changes to be implemented on the fly while appearing seamless to the user, and because changes to JSP files can be made in a live environment, the entire application doesn't have to be re-deployed for minor updates. JSP files can also be created and maintained fairly easily by HTML developers rather than Java developers who would otherwise have to write HttpServlets directly.

VENDOR TOOLS

| | |
|-----------------|--|
| WebLogic | Application that parses and compiles JSP files into HttpServlets. |
| Apache | Web server that receives incoming requests from and sends data through the Internet. |

Business Logic Tier

SUMMARY

The Business Logic Tier is the application layer. It consists of Enterprise Java Beans (EJBs) that query the database, perform tasks, and communicate with HttpServlets.

ARCHITECTURE: ENTERPRISE JAVA BEANS

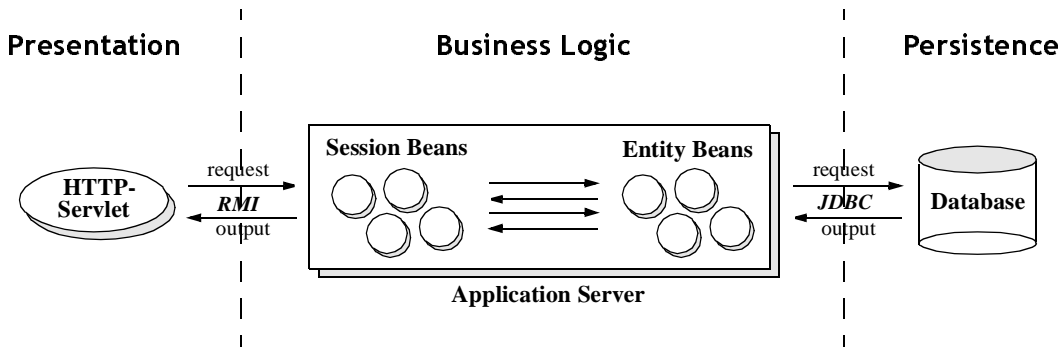
The Business Logic Tier consists of EJBs (distributed Java objects) that are built by and run on an application server. The application server and EJBs act as a go-between for the HttpServlets and the database.

ACHIEVE utilizes two types of EJBs: Entity Beans to retrieve and update data in the database and Session Beans to manipulate data. More specifically:

- **Entity Beans** are software representations of the data model. When a query is made to the database, an instance of an Entity Bean is created that replicates the data as a Java object. Entity Beans and the database are always in sync; operations such as read, create, update, and delete are immediately propagated from the Entity Bean to the database.
- **Session Beans** receive requests from the HttpServlets and perform workflow tasks that manipulate one or more Entity Beans.

For example, the user requests a student record. The HttpServlet passes the request to the appropriate Session Bean, which instantiates an Entity Bean to retrieve the student information, processes the returned information, and sends it back to the HttpServlet.

Session Beans also perform more complicated tasks, such as enrolling a student in a class. In this case, several Entity Beans are required to supply the relevant class and student information. The Session Bean, once it has the information, performs the task of associating the student with the class.



BENEFITS

EJBs and the application server provide numerous benefits:

- **Performance & Scalability:** Because Entity Beans reside in memory for a time, the information is cached, which minimizes the number of requests the database must process.

The application server is multi-threaded to make efficient use of memory.

- **Transaction Management:** TBD.

VENDOR TOOLS

| | |
|-----------------|--|
| Weblogic | Application server which builds, runs, and manages EJBs. It handles communication between EJBs, HttpServlets and the database. |
|-----------------|--|

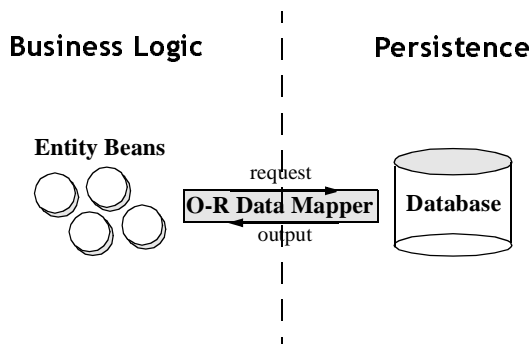
Persistence Tier

SUMMARY

The Persistence Tier is a Relational Database Management System (RDBMS) that stores and maintains the integrity of application data. It passes data to and receives data from Entity Beans through an Object to Relational (O-R) Data Mapper.

ARCHITECTURE: RELATIONAL DATABASE MANAGEMENT SYSTEM

ACHIEVE uses an Oracle RDBMS to persist application data. An O-R Data Mapper maps database tables to their corresponding Entity Beans in order to translate requests from the Entity Beans to the database.



BENEFITS

- **Abstraction:** The O-R Data Mapper adds a layer of abstraction between the database and Entity Beans so database changes have a minimal effect on the application.

VENDOR TOOLS

| | |
|----------------|--|
| Oracle | RDBMS to store data. |
| TopLink | O-R Data Mapper that maps database tables to Java objects. |

Conclusion

BENEFITS OF ARCHITECTURE

- **Abstraction, abstraction, and more abstraction:** The O-R Data Mapper provides a layer of abstraction between the database and the EJBs and the EJBs provide another layer of abstraction between the database and the JSP files, so modifications to the back-end of the application remain relatively isolated.
- **Performance & Scalability:** Because HttpServlets and Entity Beans are created on demand, and Entity Beans are cached, the system is not performing unnecessary operations and requests leverage from previously performed operations.

The WebLogic application server provides resource management and load balancing to further increase performance.