

BLUESTONE SOFTWARE'S EJB TECHNOLOGY



Table of contents

1. Introduction.....	3
2. EJB integration and the Bluestone application server	4
3. EJB use.....	8
4. EJB Persistence	11
5. TP Monitor.....	15
6. Distribution	18
7. Load Balancing and Failover.....	18
8. Modular EJB server.....	20
9. Total-e-Business (T-e-B) Framework	21
10. Conclusion.....	22
11. EJB: Additional Information	23

Table of figures

Figure 1: EJB/Universal Business Server integration: general architecture.....	6
Figure 2: Bluestone EJBs within the framework of a Web application.	9
Figure 3: EJBs can be used by XML applications.....	10
Figure 4: Direct access to EJBs from Java applications.....	10
Figure 5: Mapping between EJBs in memory and the database.	11
Figure 6: Going through an intermediary XML representation.	12
Figure 7: Universal Business Server's EJB persistence service.	12
Figure 8: The EJB Environment Builder enables configuration	15
Figure 9: State synchronization according to the outcome of the transaction.....	16
Figure 10: Modular Transaction manager.	17
Figure 11: EJBs can be accessed locally or remotely via RMI.	18
Figure 12: General architecture of load balancing and failover services.....	19
Figure 13: The smart stub routes the EJB call to the Load Balance Broker.....	20

1. Introduction

As part of its strategy to support the Java 2 Enterprise Edition (J2EE) platform, Bluestone Software's infrastructure integrates an implementation of EJB 1.1 technology.

This document addresses the following questions:

- ◆ What are the characteristics of Bluestone Software's EJB technology?
- ◆ How does this technology integrate with the rest of Bluestone's infrastructure (Universal Business Server, XML Server and Total-e-Business)?
- ◆ How and when is this technology used?

Objects: the building block of tomorrow's enterprise system

With the basic Java model, enterprise entities or business processes are modeled after the object concept. This concept is based on two conceptual and technical pillars:

- The Abstract Data Type (ADT) theory,
- Dynamic binding.

ADT enables implementation of new data types that are adapted to each particular situation and thus facilitates high-level programming.

Dynamic binding technology makes it possible to organize information systems in modules (the objects) that represent abstract data types that communicate dynamically between each another by sending messages.

Communication by sending messages differs from a simple procedure call of non-object languages:

- Each message is sent to a specific object.
- The code executed by the object receiving the message is selected automatically and dynamically at execution according to the message name and the ADT associated with the object. In the case of a simple procedure call, the code is selected statically at the link time.

This model enables high-level, modular programming, high reutilization of objects and development using frameworks. This development approach consists of building an object structure, identifying generic processes and making a host infrastructure for objects that are application-oriented.

This approach makes it possible to implement flexible and extensibles applications.

2. EJB integration and the Bluestone application server

The EJB implementation described in this document is found within Bluestone Software's application server: the Universal Business Server (UBS). This application server provides a mature environment for the implementation of enterprise applications and is particularly well-suited to Web and XML applications.

With regard to its EJB technology, Bluestone has set several goals:

- ◆ Their implementation of EJB is found within the UBS application server and relies on its services to offer such advanced capacities as load balancing, fault tolerance, security and configurable quality of service.
- ◆ The UBS incorporates not only an EJB container, but also a group of J2EE platform frameworks (servlets, JSP, JDBC, JMS, etc.). Therefore, any EJB that uses J2EE services can be deployed in the UBS.
- ◆ In addition to J2EE, Bluestone provides developers with numerous unique services (XML, pre-built e-commerce components, development tools, heterogeneous data source integration modules, etc.). EJB technology is completely integrated with Bluestone's frameworks and can use, or be used by, these different services.
- ◆ Moreover, it is possible to access all of these services, including J2EE, without using EJBs. EJBs are integrated with Bluestone's frameworks, but you can also take advantage of these frameworks and the associated tools without using EJBs.

It is important to understand the difference between use of an isolated EJB server/container and use of a technology such as the one Bluestone proposes, which integrates EJBs within a rich runtime and development environment. Beyond functional contributions, such a solution enables implementation of numerous optimizations and unification with regard to transversal services such as load balancing and fault tolerance. For example, the same infrastructure is used for load balancing EJBs as well as servlets.

Enterprise JavaBeans (EJB): from objects to components

Today, Java application servers integrate new functionality in order to go even further in the development of enterprise systems. Market players such as Sun, BEA, IBM and Bluestone, are pushing the Java 2 Enterprise Edition (J2EE), which not only brings together Servlets and JDBC, but such technologies as Java Server Pages (JSP) and JavaMail. In this case, EJBs play an extremely important role: based on other J2EE elements, they create an enterprise component model.

In brief, the goal of EJB technology is to surpass the basic Java object model by integrating new functionality important for enterprise systems. Most notably, EJBs bring functionality in terms of:

- Automatic management of object life cycle (instantiation, destruction, and activation).
- Object security: who can use which object and how?
- Object persistence: how are objects stored on a long-term basis?
- Transaction on the objects: how should one proceed with using ACID transactions including Rollback possibilities while working with objects and not relational database tables?
- Distribution: how can remote applications access objects?
- Scalability: by implementing various technologies.

These EJBs, or « super-objects », are called components.

The EJB Specification is targeted to developers of application servers or development environments. It describes how the different elements function and interact. The goal of this specification is to make the code of components and of the programs using these components portable across different servers. The developers of EJB-compatible applications must therefore conform to some very strict rules.

From a developer's point of view, an EJB is presented as a group of files that brings together:

- Java Classes,
- Java Interfaces,
- Deployment information,
- Metadata.

To best understand this paper, it is important to know specific EJB concepts and vocabulary. It is particularly helpful to specify the notions of Session Bean and Entity Bean, which describe the EJB types:

- **Session Beans** represent objects whose access supports security, transaction and distribution models. The Session Bean framework integrates the idea of context management, which enables external conversational interactions to be established in the case of multiple-client access. Session Beans implement « business logic »,
- **Entity Beans** also integrate security, transaction and distribution frameworks, but unlike Session Beans, they also support persistence. EJBs are used to implement what are commonly called « business objects »; they are designed to allow business entity modeling.

Finally, the EJB container is the EJB execution infrastructure. Each provider of an EJB environment implements its own container.

➤ General Architecture

The diagram below, which represents the relationships between Bluestone's implementation of EJB and the rest of its infrastructure, gives an idea of the general approach taken and points out this technology's prominent features:

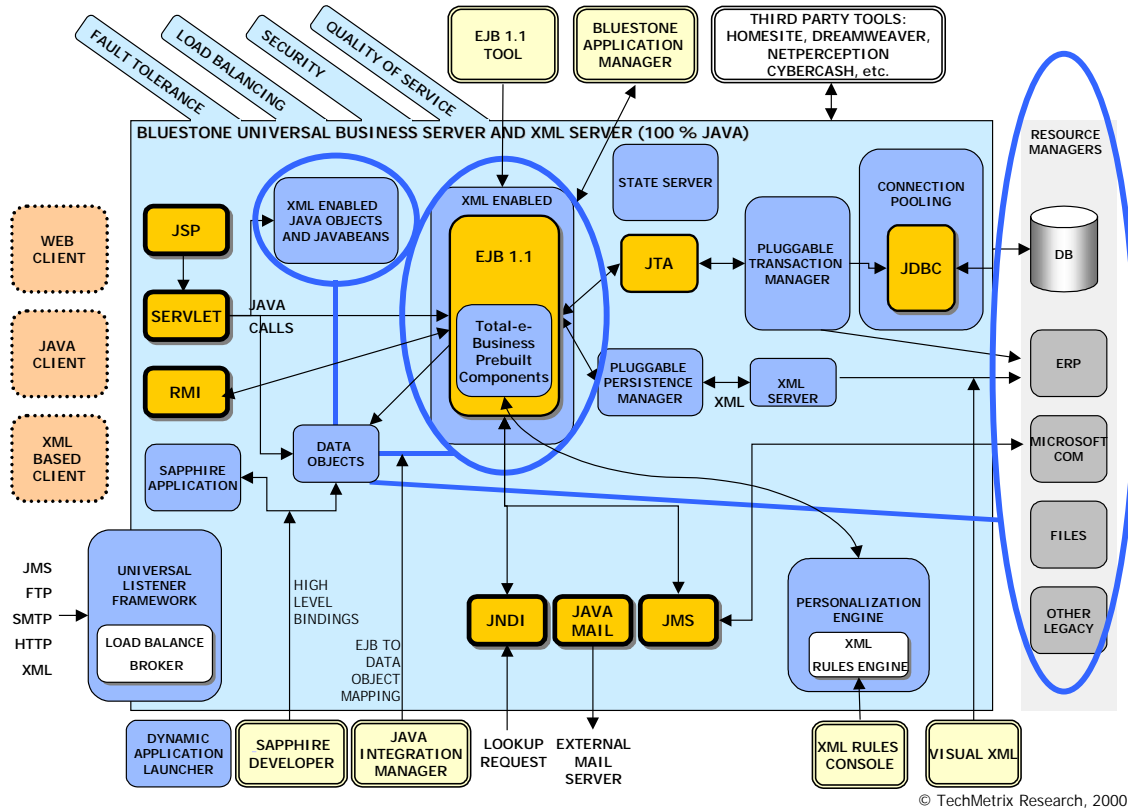


Figure 1: EJB/Universal Business Server integration: general architecture

Right away we notice, in addition to an implementation of EJB 1.1, the presence of J2EE frameworks (JSP, servlets, RMI, JNDI, JavaMail, JMS, etc.). Generally speaking, Bluestone implemented these frameworks relying largely on technologies developed over the course of the past few years for its Universal Business Server (the application server component of Sapphire/Web).

Configuration and quality of service

Configuring quality of service makes it possible to offer different levels of service, content or performance according to predefined user profiles. For example, it is possible to define user groups and automatically route their requests to different servers that offer, for example, different performance levels (more or less powerful CPU).

Besides J2EE frameworks, Bluestone also offers a group of services and APIs such as the XML Server, the Personalization Engine, the Universal Listener Framework, etc. These services work in conjunction with EJBs and J2EE. All of these elements, as well as the deployed application code, benefit equally from load-balancing, fault tolerance, security infrastructure and a configurable quality of service.

External data sources or applications can be accessed in a unified manner thanks to an abstraction layer created by the Software Integration Modules (SIMs) that make up a sub-group of the Data Objects represented in the diagram. The Enterprise Application Integration (EAI) capacities are completed by broad and sophisticated support for XML. It is worth noting in particular the presence of the Universal Listener Framework (ULF), which enables implementation of XML workflow on numerous standard protocols (HTTP, FTP, SMTP etc.).

Generally speaking, application logic is implemented as EJBs, JavaBeans and standard Java objects, depending on application requirements. It should also be noted that several graphical tools enable implementation of this application logic and its links with the user interface and external resources (Sapphire Developer, Java Integration Manager, Visual-XML, EJB 1.1 Tool). Administration of EJBs, as with other elements, is handled by a monitoring tool called BAM (Bluestone Application Manager).

Finally, there are also high-level vertical frameworks that can be used with or from EJBs:

- ◆ The Total-e-Business components, which are proposed as EJBs¹ and JSPs, provide the functionality necessary for implementation of an e-commerce site.
- ◆ The Personalization Engine enables definition and use of Web application personalization rules according to the users and their profiles. Up until now only specific products such as BroadVision or ATG Personalization Server furnished this type of functionality.

Enterprise Application Integration (EAI)

Enterprise Application Integration (EAI) makes it possible to share information within a business environment that includes all the players interacting with a company.

EAI solutions offer a framework providing methods and tools to achieve interoperability and information synchronization across multiple applications: mainframe, ERP, packaged systems, DBMS, and in-house applications. XML tends to be the standard format of all exchanged information.

¹ There is also a version in the form of JavaBeans.

3. EJB use

As components, EJBs are used in a modular manner. They communicate with the other system elements. Within a Java platform such as the one Bluestone proposes, the general paradigm that enables interoperability between the different components is message sending (or method invocation) between objects. It is in this way that EJBs can be used with Web applications, servlets for example, within the framework of distributed Java applications, and with a Java client and RMI communication.

EJBs enable implementation of application logic. We strongly recommend isolating EJB code from any presentation logic. This way, application logic can be used from different presentation models.

Generally speaking, you should use EJBs with discernment and only if you need to use the particular services they propose (persistence, transaction management, etc.). In any other case, you should consider using the JavaBean model or the standard Java object model. Bluestone's Universal Business Server enables use of these different models in a relatively unified manner.

➤ How are EJBs used within the framework of an HTML Web application?

As with all applications, the user interface management layer must be distinguished from the application process layer. For the Web user interface, Bluestone offers developers three possibilities: JSPs, servlets and Sapphire applications.

Method	Productivity	Graphical Tools	Interaction with EJBs	Standard J2EE
Servlet	Low	No	Explicit method call	Yes
JSP	Average	No	Explicit method call and JSP tags	Yes
App. Sapphire	High	Sapphire Developer	High-level bindings	No

In the case of a Sapphire application, there is a complete, high-level tool that links HTML interface elements and application components or external data sources. To do this, Bluestone offers an abstract layer, Data Objects, which makes it possible to bring together these components around a common ResultSet-type representation.

Data Objects can thus represent EJB objects, Java objects, JavaBeans or any external data source that can be accessed from a Software Integration Module.

Moreover, it is possible to use Data Objects from servlets and JSPs.

Graphical tools that enable implementation of JSP/EJB or servlet/EJB interactions are currently being developed.

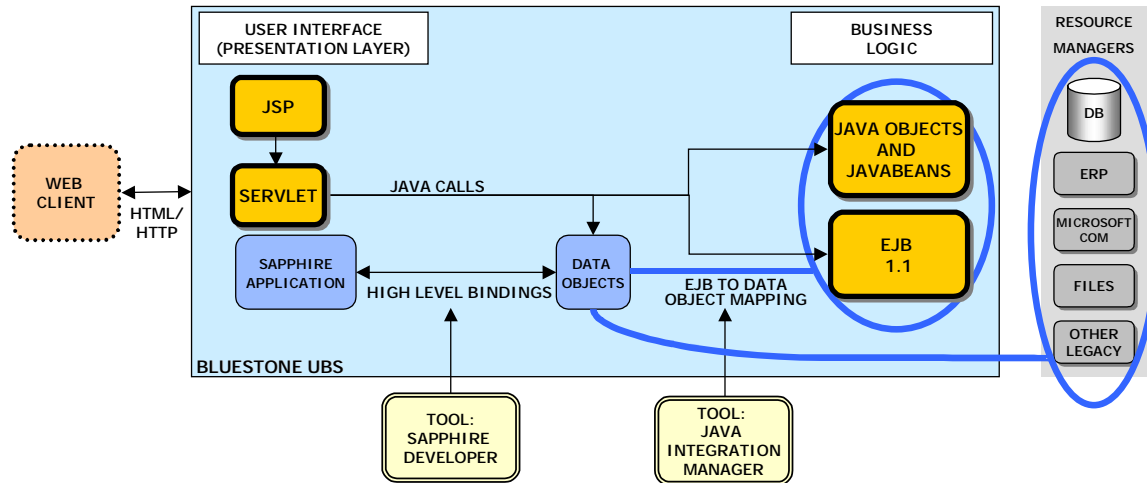


Figure 2: Bluestone EJBs within the framework of a Web application.

➤ How are EJBs used within the framework of an XML application?

An XML application receives, processes and possibly transmits XML documents. EJBs intervene again to model business logic and participate in processing.

The receipt and transmission of XML documents is handled by Bluestone's XML infrastructure. The Universal Listener Framework (ULF) is one of this technology's key elements. Capable of interacting with numerous communication protocols and conceived as an extensible framework, ULF guarantees total openness to diverse elements of an enterprise information system and its partners.

Once an XML document has been received it is processed thanks to Bluestone's Document Handlers. The Document Handlers resemble servlets, but are adapted to XML processing. They are written in Java by application developers. In order to carry out processing, Document Handlers can use all the services of Bluestone's infrastructure, and in particular EJBs via direct call.

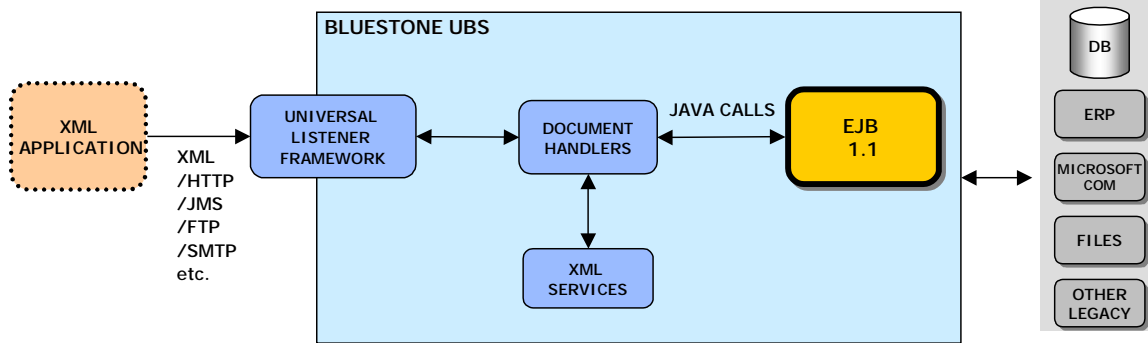


Figure 3: EJBs can be used by XML applications

➡ How are EJBs used within the framework of a Java-client application?

EJBs offer a remote method call model. A Java-client application uses RMI to communicate with the EJBs deployed in the UBS.

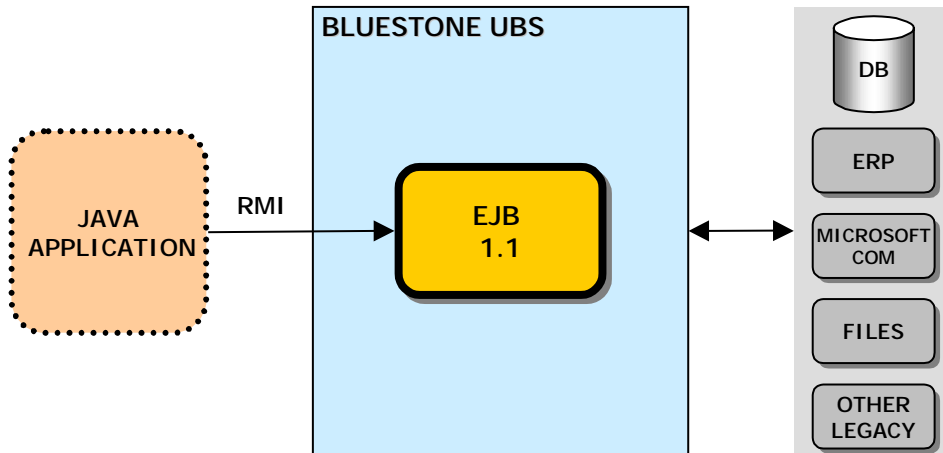


Figure 4: Direct access to EJBs from Java applications

As we will see in the section dedicated to load balancing and fault tolerance, it is also possible to capitalize on these services during use of Java-client applications. In this case, the architecture is a little more complex (see section 7).

4. EJB Persistence

The EJB persistence service applies to Entity Beans. The objective is to provide these objects with a reliable means of storage that integrates with a company's existing system.

At runtime, EJBs are in memory, as are all other Java objects. Entity Beans include a state and represent the entities handled by the applications.

Traditionally, enterprise entities are modeled and stored in the database. EJB technology recognizes this situation and offers the possibility of establishing a mapping between database tables and EJBs in memory. This mapping enables specification of how the EJB instance variables are represented in a database. For example, you can specify that the "name" field of Customer class EJBs map to the "NAME" column in the "CUSTOMER" table. In this way, at runtime the EJB server can store EJBs in the database, or conversely it can use the information in the database and store the EJBs in memory.

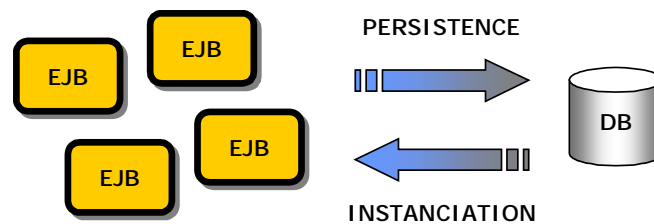


Figure 5: Mapping between EJBs in memory and the database.

With EJBs, mapping is done differently according to the EJB container used (in practice, according to the application server). This means that in order to import an EJB in a different container, it is necessary to go through this development phase again.

The problem of persistence is complex and many research projects are being carried out on this subject. One of the important points to remember is that at runtime objects (as it happens, EJBs) are not isolated entities but referenced mutually. Therefore, the problem of persistence does not concern isolated objects, but complex object graphs. There are many questions to answer: How can an object graph in memory be projected on a disk and vice versa? How can synchronization problems between the graph on disk and the graph in memory be resolved? How do you go about loading in memory only the parts of the graph being used at a given moment? How can the graph be saved in a relational database (object/relational mapping technologies), etc.

Bluestone's technology offers elements that facilitate use of such a mapping.

Bluestone supports both persistence types defined by the EJB specification.

➤ Container Managed Persistence (CMP)

In CMP mode, the Bluestone server handles persistence automatically. In particular, the server generates the SQL commands destined for the database.

In order to do so, Bluestone relies on XML technology through integration of the UBS and its XML server. The principle is to go through a XML representation of the EJB graph, then to map this representation to the database. This approach relies on the tried-and-true capacities of the XML Server. This server can map any XML document to a database.

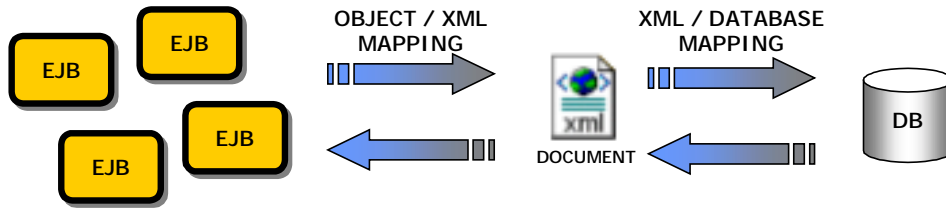


Figure 6: Going through an intermediary XML representation.

Even better, thanks to Software Integration Modules (SIMs) and other UBS elements, Bluestone's XML server is capable of communicating not only with databases, but also with other types of external resources (ERP, mainframes, etc.).

Visual-XML, a graphical tool, enables specification of the information required to map an XML document to a database².

As seen in the diagram below, the underlying implementation is modular:

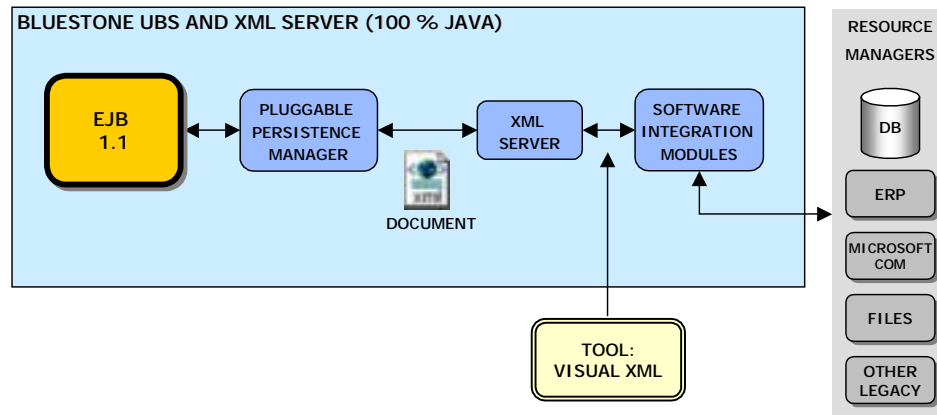


Figure 7: Universal Business Server's EJB persistence service.

² In its current version, Visual-XML does not allow mapping of an XML document on several databases.

When EJBs must be persisted or be instantiated, a specific module, the Persistence Manager is responsible for seeing through the mapping. In the current implementation, Bluestone provides a Persistence Manager that is based on the XML model described above. This Persistence Manager is able to:

- ◆ Generate an XML representation from an EJB graph.
- ◆ Generate or update an EJB graph from an XML representation.
- ◆ Communicate with the XML Server and with the underlying persistence engine.

The modular architecture, which is based on a high-level framework called "Persistence Toolkit," makes it possible to replace the XML Persistence Manager by a module that can handle persistence differently. For example, the Persistence Manager can work directly with a database via JDBC. With UBS, it is also possible to plug in third-party mapping tools; however, in this case the link to the XML Server is no longer available.

➔ Bean Managed Persistence (BMP)

In BMP mode, the EJB developer must implement persistence. To do this, certain well-defined methods must be used (`ejbStore`, `ejbLoad`, etc.) for each EJB class using this persistence mode. Database communication must be implemented manually by using JDBC, for example.

Bluestone provides specific aid to developers who wish to use this persistence mode. Indeed, the persistence code can be generated automatically by the tools. This code interacts with the Persistence Manager. Once this code has been generated, the developer is free to replace it with completely different code or to modify it so that it meets his/her specific needs.

Furthermore, Bluestone provides an API that enables specification of additional information about the processes carried out on EJBs. The EJB container uses this information to minimize and optimize database access.

CMP or BMP: What's the Right Choice?

In Bean Managed mode, the developer must implement the interaction code between his EJB and the persistence engine. This is a complex and intricate task. Thus, this mode mustn't be seen as a general development model, but more as a possibility to get at the low-level persistence mechanisms if necessary. In other words, in particular situations in which you must carry out specific actions or when you reach the limits of automatic persistence mode (CMP). It is not realistic to consider using this model for all EJBs due to the cumbersomeness and complexity of development. The Container Managed mode must thus be considered the general persistence model. It handles this functionality automatically.

➤ Support for fine-grained objects

EJB technology is not adapted to fine-grained objects (objects of which there is a large number with methods that are called frequently). For this type of object, the EJB specification recommends using a lighter model.

For fine-grained objects Bluestone proposes the use of the basic Java object model or JavaBeans.

Generally, EJBs and fine-grained objects interact with one another because it is not uncommon that an EJB be made up of even finer objects. This creates some difficulty because this implies that the services proposed on EJBs must also take into consideration these fine-grained objects that are not EJBs.

In response to this problem, Bluestone makes it possible to combine the EJB model with the basic Java object model, as well as with JavaBeans. Indeed, the persistence model based on the use of Bluestone's XML Server functions in the same way with EJB objects, Java objects, JavaBeans or an object graph that combines the three models. It offers an integration of the same platforms objects models. Most EJB servers don't provide this facility yet.

Object Granularity: Recommendations of the EJB Specification

The EJB 1.1 specification warns developers about using the EJB model for implementation of fine-grained objects. Notably, it indicates that the cost of method calls on an EJB is too high to support the frequent interactions that characterize fine-grained objects. Moreover, the specification defines the notion of dependent object and recommends implementing these objects by using the basic Java object model. To (over)simplify, dependent objects are components of an EJB that are not directly accessible from outside the EJB. Typically, they are fine-grained objects.

The EJB specification excludes fine-grained object management (relationships between objects, persistence, and so on) from its sphere of application. In addition to implementing the specification, each EJB container editor must therefore respond to this problem. Today, in fact, there are very few containers offering a solution to this problem.

5. TP Monitor

In a multi-level architecture such as Bluestone's, the majority of application code is executed on the server side. It is possible that thousands of users request the applications. The application server(s) are thus required to manage a multi-user mode. Operation coherence requires use of a transactional model that can group together all of the logical operations in units of atomic processes that are independent from one another.

In this area, Bluestone's EJB technology supports ACID transactions and enables:

- ◆ EJB use in a transactional context. The changes made to an EJB can be rolled back should the transaction fail, or validated on a long-term basis should the transaction succeed.
- ◆ Automatic opening and closing of transactions during a method call on EJBs and during the return of these methods.

Bluestone offers a tool that makes it possible to specify the transaction policy to use for each method of an EJB.

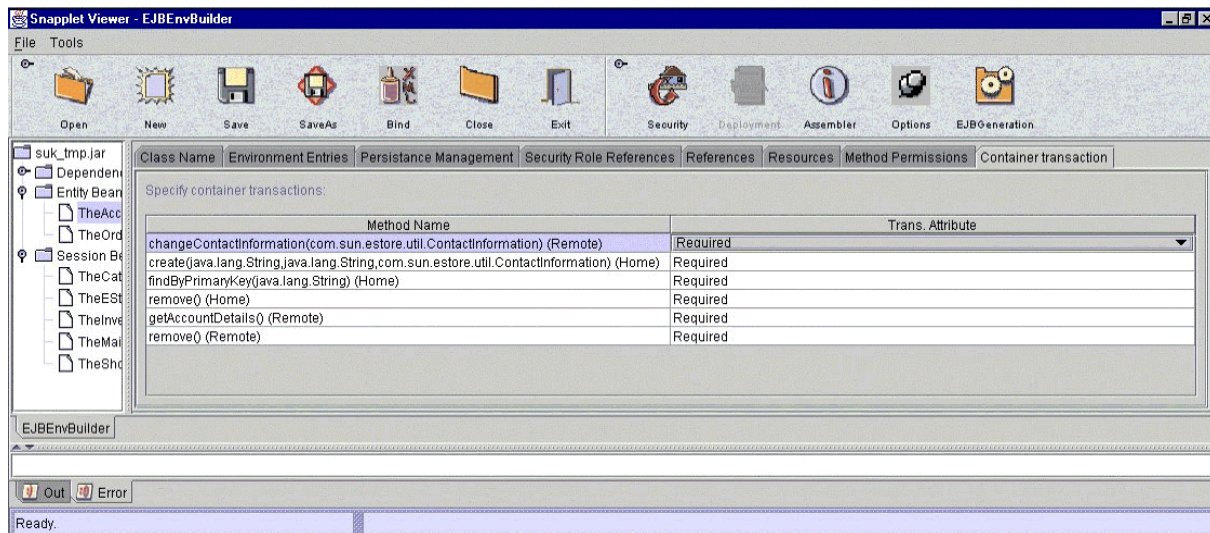


Figure 8: The EJB Environment Builder enables, among other things, configuration of transactional semantics.

➔ How are object graph transactions managed?

During the course of a transaction, EJB code can call a transactional resource manager (a database using JDBC, for example). In this case, the transaction commit or rollback, as well as the isolation levels, is taken into consideration by the resource manager in a traditional manner.

But EJB technology goes even further. Indeed, EJBs can be considered objects having transactional semantics. Thus, modifications made to EJBs during a transaction must be saved in a reliable and durable manner (commit) or cancelled (rollback), depending on the situation.

We are going to look at the case of Entity Beans. The key idea used by Bluestone's EJB technology to offer transactional semantics is based on the following facts:

- ◆ The persistence model makes it possible to maintain a representation of EJB state in the database.
- ◆ It can also synchronize memory state with database state.

The fundamental principle is thus to consider the state in the database as the committed state. At the end of a transaction, you have two potentially different states of the graph of the EJB involved: one in the database and the other in memory. If the transaction is completed by a successful commit, the state in memory is projected to the database (via the persistence mechanism) and becomes the committed state. If the transaction fails (rollback), the state in memory is no longer considered valid and the state in the database will constitute the departure point for the next transaction.

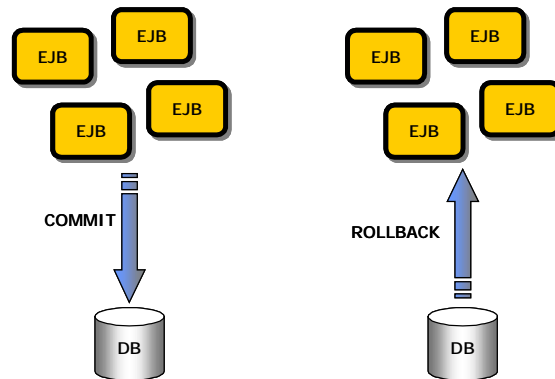


Figure 9: State synchronization according to the outcome of the transaction.

The EJB specification defines three commit-time options with regard to the way in which Entity Beans are synchronized with the persistence engine (EJB Specification 1.1, section 9.1.10). Bluestone's implementation uses option B: Given that there is an Entity Bean in memory, the container does not maintain exclusive access to the state of this EJB in the persistent store. This persistent state can therefore be modified independently of the state in memory. The container thus resynchronizes the state of the EJB in memory with the persistent state at the beginning of each transaction.

➤ Architecture

Bluestone's implementation of EJB transactions uses a modular architecture based on the JTA specification (Java Transaction API).

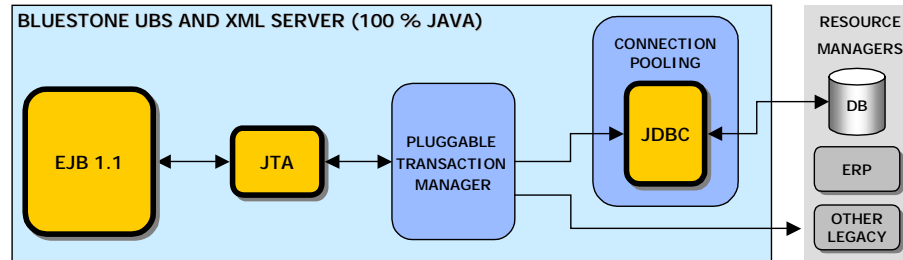


Figure 10: Modular Transaction manager.

With this architecture the transaction manager can be considered a module that communicates with the rest of the system via very specific interfaces. This guarantees openness to different transaction managers on the market.

➤ Other characteristics

Bluestone supports the notion of the isolation level, which enables specification of how concurrent transactions can interact.

It is possible for a Java-client program to manage opening and closing of transactions (begin, commit and rollback commands) thanks to the standard APIs of the `javax.transaction.UserTransaction` package.

In accordance with the EJB specification, nested transactions are not authorized.

For distributed transactions, Bluestone has chosen the XA standard. The implementation of XA and two-phase commit is present at Bluestone's infrastructure level. This requires the use of XA-compatible JDBC 2.0 drivers, such as is available for Oracle.

6. Distribution

EJB technology offers a distributed object communication model. Whether an EJB is remote or in the same memory space, it is accessed in the same manner at the programming level, via the RMI model.

Bluestone provides a specifically optimized RMI implementation. This implementation is capable of detecting whether communication is with a local or remote EJB, and will optimize call processing in the latter case.

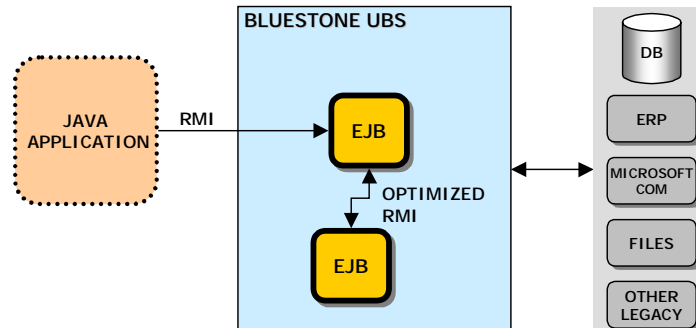


Figure 11 : EJBs can be accessed locally or remotely via RMI.

In the future, Bluestone plans on offering an implementation of RMI over IIOP, which allows the EJB model to interoperate with the CORBA model while preserving, to a certain extent, RMI semantics³. This will allow a C++ application to communicate with Bluestone EJBs via CORBA, for example.

7. Load Balancing and Failover

In order to offer load balancing and failover, Bluestone relies on its existing infrastructure, broadening it to take EJBs into consideration. The general idea is to be able to deploy several server instances⁴ on several machines. Client requests are then automatically dispatched to the different instances by a module called the Load Balance Broker (LBB).

³ RMI over IIOP does not offer such fundamental services as distributed garbage collection.

⁴ An instance corresponds to an running process, in other words, a JVM. It is also worth noting that each instance can be multithreaded.

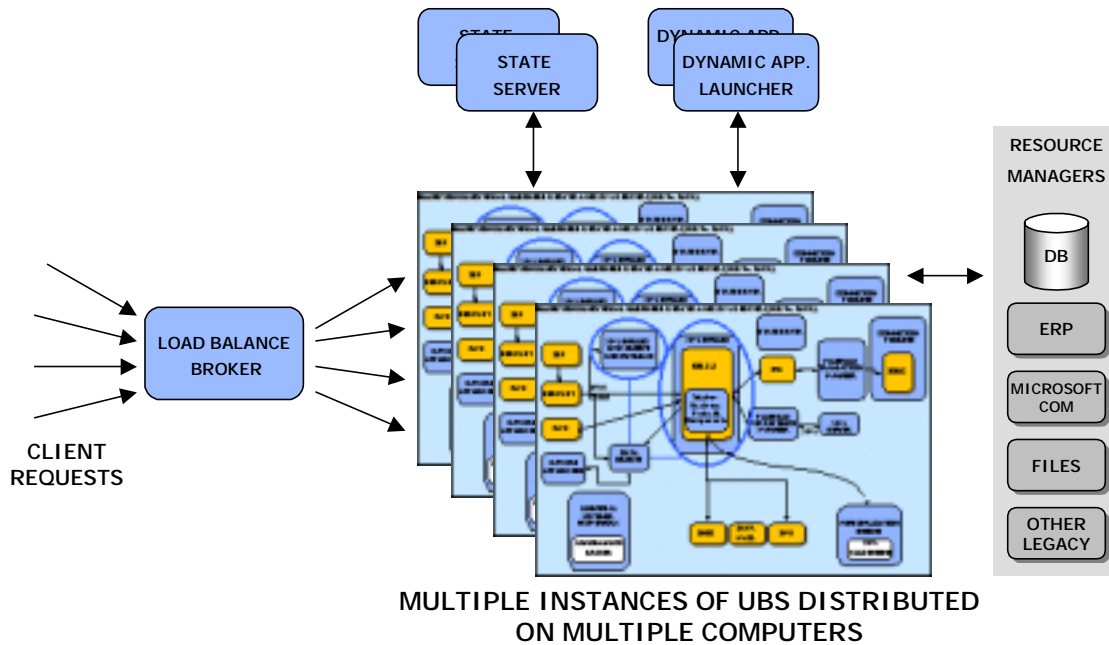


Figure 12: General architecture of load balancing and failover services.

In the event of instance failure (software or hardware crash, network problem and so on), the LBB is capable of detecting the particular instance's unavailability and reacting accordingly. Moreover, if one instance becomes available again, it will automatically be reintegrated with active instances.

The Dynamic Application Launcher is the module responsible for launching an instance after a crash. It can be deployed in several replicated copies so as to be fault tolerant.

The State Server, which can also be replicated, is used to save the context state of each user session⁵. These states are made persistent in the database⁶. Should an instance fail, the states can be reassigned to another instance. The UBS thus supports the notion of session-level failover.

Two different types of client requests can use EJBs:

- ◆ Requests addressed directly to an EJB via the Home or Remote Interface.
- ◆ Indirect requests destined to be treated beforehand by another framework (servlet, DocHandler XML, Sapphire application, etc.).

⁵ For this to work, the Java objects representing these states must be able to support serialization.

⁶ For Entity Beans, the State Server is not used because the state of the objects is already persisted in the database via the EJB persistence service.

In the case of direct requests, the client addresses a stub that represents the remote EJB. But instead of a traditional RMI stub routing the request directly to the EJB, there is a stub that is specifically generated by UBS, which routes the request toward the LBB. In this way, even the direct calls take advantage of load balancing and failover services.

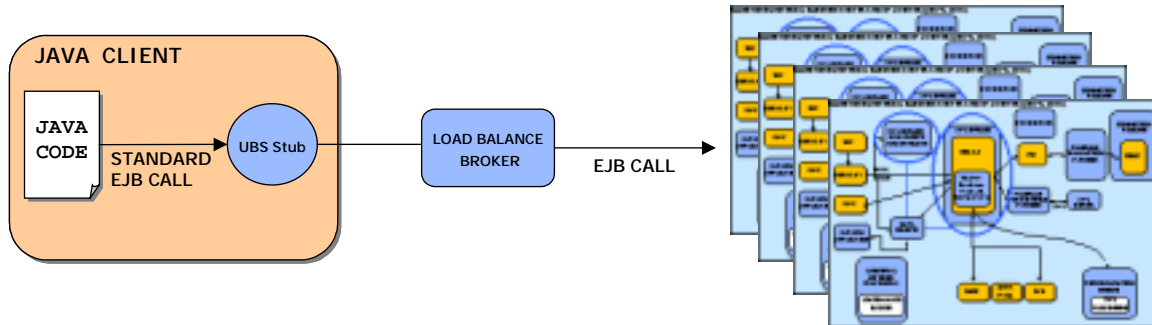


Figure 13: The smart stub routes the EJB call to the Load Balance Broker.

Also, it is worth noting that the LBB uses the notion of session affinity, which consists of routing client requests to the UBS instance that has the session state corresponding to this client stored in memory.

Should a UBS instance fail, the LBB can be configured to automatically retry launching the request rather than sending an error signal to the client. Note that the LBB itself can be replicated.

8. Standalone EJB server

Although BlueStone's EJB server integrates with UBS, it also allows for a standalone mode in which the UBS does not intervene. In this mode, the EJB server functions within a JVM and does not require the presence of UBS (however, there must be an external JNDI service).

This standalone mode can be useful in several situations:

- ◆ During EJB development: Using the standalone mode prevents the developer from having to install and use UBS, which is a complex environment. The developer can simply install the EJB server on his workstation and then test and fine-tune his EJBs. Not only does this speed development, it also makes it easier.
- ◆ During deployment of EJBs requiring use of neither frameworks (XML, Servlet, etc.) nor UBS services (load balancing and so on). For example, when deploying EJBs in the framework of non-critical applications that are destined to a restricted number of users or when deployment must respect particular architectural constraints (implementation of an architecture that is not based on the notion of application server, use of a specific failover or load-balancing infrastructure, etc.).

In standalone mode, clients access EJBs directly via RMI calls. As the EJB server runs independently, it does not capitalize on UBS services, in particular load balancing, failover and scalability.

On the other hand, all EJB 1.1-required services are included. Most notably, the EJB server includes a security service. Authentication is based on use of a mechanism that allows for the transfer of client identification information in JRMP frames (protocol used by RMI). This information can then be processed by the security service, which provides EJB access authorization.

Moreover, the proposed EJB transactional service is identical to the one provided by UBS; it is based on a JTA implementation.

9. Total-e-Business (T-e-B) Framework

T-e-B is a complete framework enabling the implementation of an e-commerce website. Based on high-level, ready-to-use EJBs, it offers, thanks to its framework structure, very wide adaptation and configuration possibilities. T-e-B has found its place within Bluestone's UBS and XML Server. T-e-B combines the use of EJBs and other system components: JSP, XML, Personalization Services, etc.

T-e-B includes most notably about twenty Entity Beans and enables implementation of numerous services:

- ◆ Content management (creation, publication and maintenance).
- ◆ Source control.
- ◆ Workflow.
- ◆ Personalization.
- ◆ Click Stream Analysis.
- ◆ Recommendation Engine.
- ◆ E-commerce.
- ◆ Shopping Cart.
- ◆ Payment, credit card processing.
- ◆ Taxes.
- ◆ Shipping, interaction with UPS services.
- ◆ Etc.

10. Conclusion

Faced with the new challenges created by the Internet and the implementation of n-tier architectures, it is important to understand how Enterprise JavaBean technology can be used to the advantage of an application server such as the Universal Business Server.

In this document, we have shed some light on the key points of Bluestone Software's EJB technology. In particular, we will retain the fact that this technology is found within a complete application server with which it enjoys advanced integration:

- ◆ The EJB technology thus benefits from the transversal services offered by UBS such as load balancing, fault tolerance, security, administration, supervision, and configurable quality of service.
- ◆ It integrates with such other J2EE platform technologies as servlets, RMI, JNDI and JMS.
- ◆ It integrates with UBS-specific frameworks and is supported with graphical tools and wizards.
- ◆ It functions within the same system process as the other UBS modules (servlets, etc.), which makes for significant performance gains during communication with these modules.
- ◆ It enjoys openness to strategic EAI technologies such as Software Integration Modules (SIMs) and XML.
- ◆ It makes up the central high-level component model of the Total-e-Business framework.

11. EJB: Additional Information

EJB-interest mailing list archives	http://archives.java.sun.com/cgi-bin/wa?A0=ejb-interest
jGuru EJB FAQ	http://www.jguru.com/jguru/faq/faqpage.jsp?name=EJB
EJB Now	http://www.ejbnw.com/
EJB at SUN	http://java.sun.com/products/ejb/
Forum	http://forum.java.sun.com/forum
Comp.lang.java.bean	http://wwwzenger.informatik.tu-muenchen.de/persons/backscha/ejbsig/archive/lang_java_beans/threads.html
Practical techniques and guidelines	http://www.devx.com/upload/free/features/entdev/1999/07jul99/jg0799/jg0799.asp
EJB Special Interest Group	http://www.mgm-edv.de/ejbsig/ejbsig.html
EJB and Object Oriented development: A critical perspective	http://www.techmetrix.com/lab/trendmarkers/tmk1299/tmk1299-2.shtml

A white paper written by
TechMetrix Research

Authors

Philippe Mougín (pmougín@techmetrix.net)

Contributor

Jean-Christophe Cimetiere (jcc@techmetrix.com)

Translation and editing

Gina Faucher (gfaucher@techmetrix.net)

Publication date: April 2000

USA

TechMetrix Research
6 New England Executive
Park, Suite 400
Burlington, MA 01803

Tel.: +1 781-270-7486
Fax: +1 781-270-7487

<http://www.techmetrix.com>
info@techmetrix.com

**EUROPEAN
HEADQUARTERS**

TechMetrix Research
55/57 Rue Saint Roch
75001 PARIS
FRANCE

Tel.: + 33 1 44 55 40 00
Fax: + 33 1 44 55 40 01

<http://www.techmetrix.net>
info@techmetrix.net

SWITZERLAND

TechMetrix Research
World Trade Center
Avenue Gratta-Paille 2
CH-1000 LAUSANNE 30
PO Box 476

Tel.: +41 021 641 10 65
Fax : +41 021 641 13 10

<http://www.techmetrix.net>
info@techmetrix.net



TechMetrix Research
6 New England Executive Park, # 400
Burlington, MA 01803
Phone/Fax: 781-270-7486/87

Assessments and conclusions rendered by TechMetrix Research are proprietary. TechMetrix Research and/or TechMetrix Research analysts cannot be held liable for any damages directly or indirectly caused by decisions made using any TechMetrix Research material.

Names appearing in this document that are registered trademarks are not mentioned as being so, nor is the trademark symbol inserted with each mention of these registered trademarks. This document uses these trademarks for editorial purposes only. In no way does TechMetrix Research have the intention of infringing on any registered trademark mentioned in editorial.

© TechMetrix Research 2000 - www.techmetrix.com