



xTier™ Java and .NET Middleware Components Overview

Built by developers for developers xTier™ Java and .NET Middleware Components provides enterprise application developers with reusable infrastructure-level "building blocks" which extend and enhance the J2EE and .NET specifications.

Key Benefits:

- 22 high-value services
- Service Oriented Design
- Cross-Paradigm Development
- 2-Way Non-Intrusive Integration
- Mix-and-Match (use only the services needed, and nothing more)

Today's Information Systems

A careful analysis of today's enterprise information systems reveals a number of very different approaches undertaken by companies to build and maintain their information systems. Smaller companies with simpler requirements tend to be more agile and composed in their choice of technology for information systems, often using a combination of off the shelf products and custom build solutions. Typically, these companies rigidly standardize on a particular technology stack or/and product(s) for their in-house development.

On the other hand, larger companies with more complex business processes require more sophisticated information systems to support these processes. Further, consolidation of operations, M&A activity, as well as departmental IT decisions often results in the IT infrastructure being significantly fragmented as it has been pieced using different technologies, architectures and products – there are literally dozens upon dozens of different technologies, architectures, internal and off the shelf products. These companies usually have significant legacy IT assets that still support mission critical business processes yet in many cases are holding back active technology migration and advancement.

To better understand today's information systems we will narrow our analyses by concentrating on information systems (or their major parts) that have been actively developed in the past 2-4 years. We can assume that these relatively new systems were built to address new business requirements or to migrate the existing information systems and therefore technological choices that were made would define technological direction within which these systems will evolve during their lifespan. Focusing on this type of system it is possible to firmly identify three major types of information systems:

- Predominantly J2EE systems
- Predominantly .NET systems
- Custom & heterogeneous Java- and .NET-based systems

Note that very few systems can be categorically put under a certain type or be defined as based primarily on one technology or/and product. While most of the systems still have a mix of technologies, architectures and products each of them would have a certain “center of gravity” that will clearly identify the system according to these three types. (It is interesting to note that custom Java systems, belonging to the third type above, usually use many technologies from the J2EE stack - they are just not built around a specific J2EE compliant container.)

Information System Anatomy

Throughout the analyses of information systems performed by Fitech Laboratories Inc. it was found that all these systems have remarkably different internal structures yet most of them exhibit one common characteristic - namely the fact that on the infrastructure level most of these systems lacked certain pre-built functionality that was recreated across all of these projects, regardless of what technologies or/and products are used.

Technologies such as J2EE and .NET, and products associated with them, are used primarily to get out-of-the-box infrastructure level functionality – the functionality that is not business specific yet in most cases is absolutely essential for the project success. Indeed, business applications developers genuinely want to concentrate on business functionality while using pre-built low-level infrastructure logic. Yet a significant amount of system services on different structural levels are not provided by either J2EE or .NET. For example, even relatively simple things like static configuration management, internationalization, object pooling, or even logging are not addressed by either J2EE or .NET in any sufficient way (or not at all). On the other side of the spectrum, more complex services such as distributed caching, process workflow, grid computing or high performance marshalling/de-marshalling are not addressed by J2EE and .NET in any way. Furthermore, in cross-paradigm development where services have to be tightly integrated across heterogeneous environments and provide easy interoperability, infrastructure level services are almost non-existent.

In these situations different companies chose to go different routes: some develop all of the necessary missing pieces in house, with others picking and mixing open source and commercial products piece by piece and trying to “glue” them together. Both of these approaches, however, have fundamental problems.

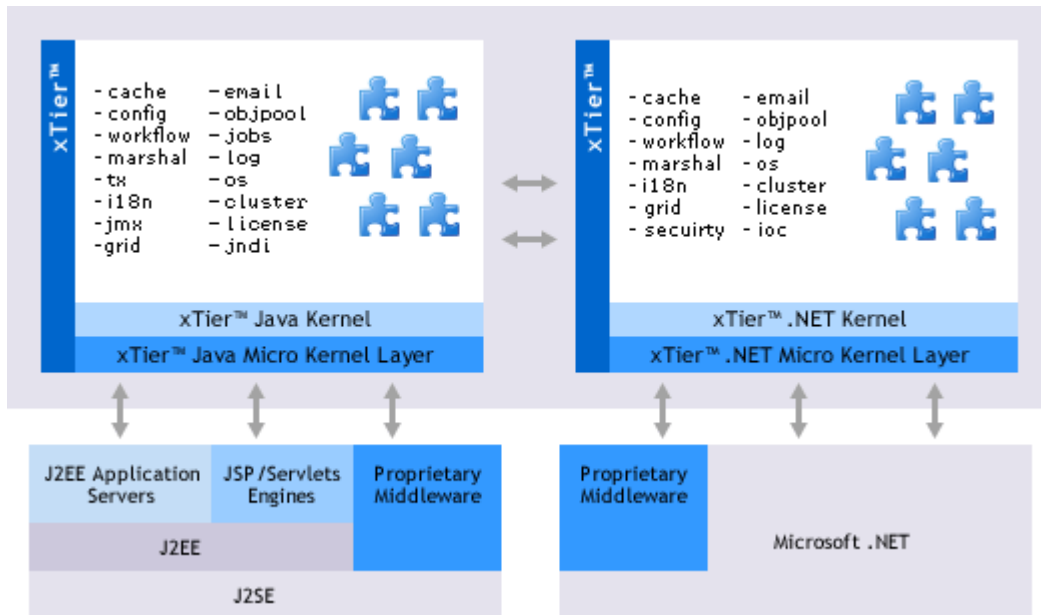
In the first case the company ventures into system software development which was never an intent at the beginning of the project and because of it these pieces are usually done hastily without much consideration toward the ramifications on the project moving forward, and as a result experience an expensive maintenance burden and accumulate a significant “technical debt”. Further it is often the case that “wheels get reinvented” time after time throughout the company because the “hastily” built module or service would only fit the specific problem it was built for.

In the second scenario mixing many products (open source and commercial) leads to a comprehensive integration effort (which in and of itself can have similar costs as in-house development) and in many situations it damages the conceptual integrity of the project. In the end, projects designed in such a way have a tendency to become coupled to too many external pieces and become unacceptably dependant, (in many cases such dependencies are realized only too late when, for example, open source project ceased to exist, change their course or became unsupported), leading to costly delays and reworks and delivering a potentially fatal blow to a system that is overly dependant on external open source software.

xTier™ - Product Definition

xTier™ Java and .NET Middleware Components provides a loosely-coupled set of pre-built services, for both Java/J2EE and .NET. These services extend the utility of J2EE and .NET by providing reusable infrastructure-level "building blocks" which are not part of the J2EE and .NET specifications and are often created and recreated for many development projects. The services provided by xTier™ are implemented utilizing the principle of Service Oriented Design (SOD), which results in an integrated, yet loosely-coupled collection of services. This frees the developer to "mix and match" the components that are best suited to a given development project rather than being forced, by the product, to use all of the services all of the time. Together with xTier's™ clean and intuitive API and short learning curve xTier™ quickly increases developer productivity while at the same time reduces the time to delivery and overall cost of the development project.

The following diagram provides an overview of the architecture of xTier™, the services that it provides and its relationship to the different hosting environments.



From its inception, xTier™ was designed and developed to provide clear and unambiguous additional value to J2EE and/or .NET and as such is meant to coexist within any J2EE or .NET infrastructure. It is important to note that xTier™ does not provide substitute services for those specified by J2EE and .NET but rather provides many additional services that extend and augment those currently provided in the J2EE and .NET specifications. Furthermore, xTier™ can be used within the client, business logic, and/or back-end tiers. In fact, xTier™ makes no assumption nor imposes any particular design constraint on whether it is running on the client or server tier.

Placing absolutely no inherent restrictions on how business applications use xTier's™ services developers can use one or two services, mix them with legacy services or use all of them in any combination. Business applications can continue to

use any J2EE or .NET native services without any interruption while slowly introducing services from xTier™, thus leaving other functionality untouched and removing the need for expensive "rip & replace" integration.

xTier™ has three key elements broadly defining its product positioning:

- Service Oriented Design
- Cross-Paradigm Development
- 2-Way Non-Intrusive Integration

Integration is the "Achilles Heel" of middleware products. By its very nature middleware assumes complex system-level functionality and thus integration is inevitable. Yet the high cost of the integration, usually associated with middleware, is one of the biggest barriers in successful adoption of new middleware technology.

xTier™ supports a unique 2-way non-intrusive integration that provides an innovative way for middleware integration. xTier™ provides a clean way for an application to be integrated into xTier™ (run within it) or for xTier™ itself to be integrated into an external hosting environment – thus a 2-way integration.

The most interesting aspect of xTier's™ two-way integration capabilities is the way xTier™ integrates into the other hosting environments such as J2EE application servers, a .NET environment and custom Java and/or .NET infrastructures. xTier™ uses a micro kernel based design to achieve this type of integration in a simple, powerful and cost-effective way. The micro kernel provides a thin functional layer between the hosting environment and the xTier™ system (kernel and services) allowing the xTier™ kernel and services to operate fully independently from the external environment. This design allows xTier to easily integrate into application server, simple JSP engine, in-house proprietary middleware or simply a J2SE environment and provide the same set of features and functionality.

The micro kernel removes dependencies between the xTier™ kernel and the specific underlying software infrastructure. This, among other advantages, facilitates deployment and re-deployment of an application developed using xTier™ on different underlying infrastructures with only the requirement being a micro kernel change, as the application code that has been developed with xTier™ remains mainly unchanged.¹ xTier™ 2.0 ships with pre-defined micro kernels for JBoss 3.x, JBoss 4, WebLogic 7.x and WebLogic 8.x

Service Oriented Design

xTier™ provides more than 20 pre-built system services implemented using Service Oriented Design (SOD). As its core design principle xTier™ avoids any empty "architectural framework" code or any other lock-in functionality – making each service as cohesive as possible and providing uncompromised conceptual integrity throughout the entire product.

xTier™ provides very clean packaging: one service per package (or namespace²), one XML configuration per service, and the only exposed APIs consist of interfaces³ allowing the end user to code 100% against interfaces. All services share the rigid structure in the way they are accessed and configuration, and have uniform usage pattern. Developers are not

¹ This assumes that application code makes no use of any environment specific features.

² Namespaces are used in .NET languages such as Visual Basic.NET or C#.

³ Except for helper, adaptor and exceptions classes that must have an implementation.

limited to any specific set of services or to any predefined way to use them – they can use one service, two or all of them in any way and in any combination⁴.

SOA/WS and xTier

When examining xTier™ it is easy to confuse xTier's™ Service Oriented Design (SOD) with a Web Services' Service Oriented Architecture (SOA). xTier's™ SOM does not imply that xTier™ is based on a Web Services implementation. Instead, the similarities between xTier™ and Web Services have to do with the architectural principles that both xTier™ and SOA implementations share, that being SOD. When considering technologies using a SOD approach, the notion of a "service" requires that software components be abstracted such that a given set of operations becomes a discrete and autonomous collection of functionality with well defined lifecycle management.

Web Services typically bundle specific business assets or processes as "discrete and autonomous collections of functionality" - an inventory "service" or an order inquiry "service", for example. Using the same underlying notion, xTier™ bundles infrastructure-level functionality into discrete "services" for utilization as the basis for enterprise development projects as in, for example, the xTier™ distributed cache "service" or the xTier™ grid computing "service". xTier™ and Web Service-based SOA are very complimentary technologies since they apply to different areas of software development and yet share many fundamental concepts.

Cross-Paradigm Development

xTier™ is the first software middleware that goes beyond the basic cross-platform interoperability. Additionally to high performance cross-platform data interchange, xTier™ provides an identical set of services for both the Java and .NET platforms where each service's implementation is highly optimized for each particular environment while sharing identical APIs, configuration and usage patterns. That enables clients with heterogeneous environments to break away from having different development paradigms in heterogeneous development environments without sacrificing performance, increasing integration cost or acquired development knowledge and skills.

Conclusion

By providing more than 20 loosely-coupled, pre-built middleware services for both Java/J2EE and .NET platforms and extending the utility of J2EE and .NET, xTier™ delivers a no-nonsense, simple to use product that every enterprise developer can instantly appreciate. From its design inception through the first line of code to the current version "xTier™ was developed by developers for developers."

⁴ Note that certain xTier™ services use other xTier™ services.

Fitech Laboratories Inc.

Corporate Headquarters

300 Montgomery St., Suite 621
San Francisco, CA 94104
USA

Phone: 1-415-371-8234
Fax: 1-415-371-8237

East Coast Sales Office

330 Madison Ave., 9th floor
New York, NY 10017
USA

Phone: 1-646-495-5076

Fitech Laboratories Japan

Toranomon40 MT Bldg. 3F
5-13-1 Toranomon Minato-ku
Tokyo 105-0001, Japan

Phone: +81-3-5402-7711
Web: www.fitechlabs.co.jp