

macromedia<sup>®</sup>  
**JRUN<sup>™</sup>4**

***White Paper***  
***May, 2002***

**Christophe Coenraets**  
**Technical Evangelist**

## Introduction

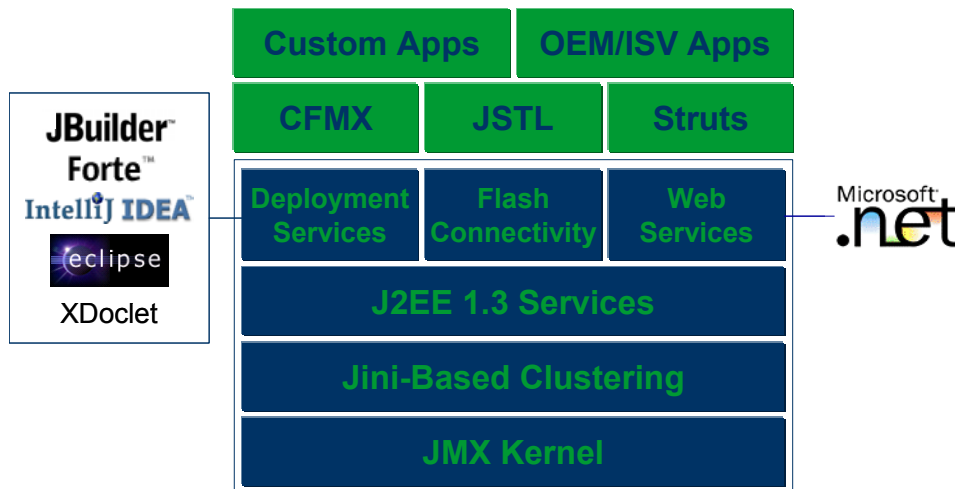
With JRun 4, Macromedia introduces the most innovative J2EE compliant application server in the industry and the most significant release in JRun history. JRun 4 groundbreaking use of Java technologies—like JMX (Java Management eXtensions) for standard-based server management and customization, and Jini for state-of-the-art server clustering—has already been praised by leading analysts and industry experts.

We developed JRun 4 around a simple vision and some core principles: provide corporate developers, OEMs, and ISVs with an open, innovative, and straightforward J2EE platform, free of legacy and proprietary technology, and loaded with performance and productivity features. In other words, provide you with 100% of the features you need, and 0% of what you should stay away from.

As the industry, pressed by poor usability and incomplete e-business transactions statistics, is reclaiming the right for a richer client experience, JRun 4, with its built-in Macromedia Flash connectivity, is the only application server that can natively dress up a J2EE application in a high impact, highly interactive user interface featuring rich controls, drag-and-drop, and all the rich features end-users have been deprived of. Macromedia Flash Remoting was one of the hottest technologies demonstrated at JavaOne 2002, where Macromedia had been chosen by Sun as a Platinum sponsor.

You don't have to choose between productivity and portability. Macromedia is uniquely positioned to provide the best of the two worlds competing for the Internet applications market: JRun 4 brings ease-of-use, productivity, and rich user experience to the open standard and cross platform community.

## JRun 4 Architecture Overview



This section provides a high level overview of the JRun 4 architecture. The rest of this document covers each architectural component in more details.

- **JMX Kernel** – JRun 4 is built on an innovative JMX service-based architecture. All the JRun 4 features are implemented as JMX services (MBeans) plugged into the JRun kernel. This architecture provides an easy to manage, extensible, and customizable foundation for deploying J2EE applications.
- **Jini-Based Clustering** – JRun 4 groundbreaking Jini-based clustering is implemented at the kernel level to allow any service plugged into the kernel to be clustered.
- **J2EE 1.3 Services** – JRun 4 is certified J2EE 1.3 compliant. Like all JRun features, J2EE features (for example, the EJB container and the Web Container) are implemented as JMX-services plugged into the JRun kernel.
- **Deployment Services** – JRun 4 sets a new standard in J2EE application deployment by providing a unique set of deployment services. For example, automatic deployment and hot modification completely eliminate the painful process of repackaging, redeploying, and restarting the server each time a change is made to the application. Deployment services also provide tight integration with leading commercial and open source IDEs.
- **Flash Services** – Built-in Macromedia Flash services make it easy for developers to provide a J2EE application with a highly interactive Flash user interface.
- **Web Services** – JRun 4 provides built-in support for web services, SOAP, WSDL, and UDDI for interoperability with heterogeneous systems including Microsoft .NET. With JRun 4, developers can publish and consume web services without having to deal with the low-level details of their underlying APIs.
- **Frameworks** – In order to make it easier for developers, OEMs, and ISVs to build J2EE applications JRun provides easy integration with all the leading frameworks and tag libraries.

## J2EE 1.3 Compatible

JRun 4 was one of the first application servers to become certified *J2EE 1.3 compatible*, demonstrating Macromedia's commitment to J2EE. This is your guarantee that any J2EE 1.3 compliant application will run on JRun. As a certified server, JRun 4 provides a complete and compliant implementation of all the latest J2EE APIs, including:

- JavaServer Pages (JSP) 1.2
- Servlets 2.3
- Enterprise JavaBeans (EJB) 2.0
- Java Message Service (JMS) 1.0.2
- Java Transaction API (JTA) 1.0
- Java Authentication and Authorization Service (JAAS) 1.0
- J2EE Connector Architecture (JCA) 1.0
- Java API for XML Parsing (JAXP) 1.1
- JavaMail 1.2



JRun 4 was also among the first application servers to participate in Sun's J2EE 1.3 deploymentathon. Refer to the following URL for more information about Sun's deploymentathon:

<http://developer.java.sun.com/developer/technicalArticles/J2EE/deploymentathon3/#jslist13>

JRun supports deployment of applications packaged for the reference implementation (such as the deploymentathon's petstore) with no modification to the deployment descriptors.

## EJB 2.0

JRun 4 EJB container has been entirely re-architected to provide superior and native support for the new EJB 2.0 features including local interfaces, message-driven beans, the new container managed persistence (CMP) 2.0, and EJB-QL. JRun 4 also provides many developer productivity features discussed in this document (such as the Enterprise Deployment Wizard and the XDoclet integration) to facilitate the development and deployment of Enterprise JavaBeans 2.0.

## JMS

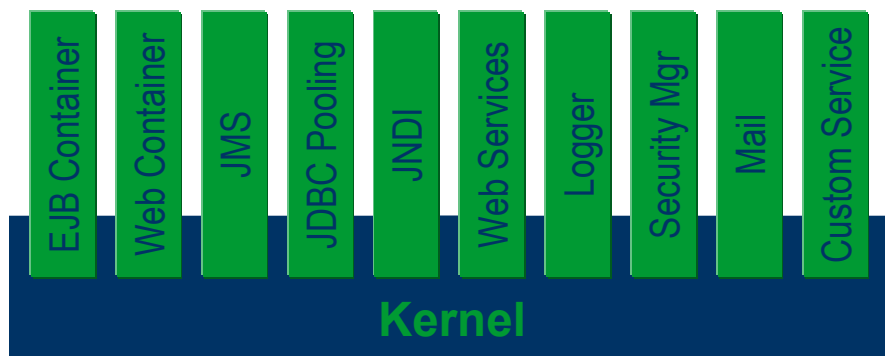
JRun 4 includes a durable and transactable JMS 1.02b compliant provider that supports both point-to-point and publish/subscribe synchronous and asynchronous messaging. JRun also provides support for external JMS providers, such as SonicMQ.

## JMX Service-Based Architecture

JRun 4 is built on an innovative JMX service-based architecture. JMX (Java Management eXtensions) is an emerging Java standard for management and customization. More information on JMX can be found at:

<http://java.sun.com/products/JavaManagement/index.html>.

All the JRun features (EJB container, web container, logging, etc.) are implemented as JMX services (called MBeans) plugged into the JRun kernel. These services can be managed by the JMX-enabled JRun Management Console (JMC), or other JMX-enabled management tools.



Some of the key benefits of the JRun JMX service-based architecture include:

- **High application availability** – Services are independent from each other and can be restarted individually.
- **Extensibility** – Administrators, advanced developers, or OEMs can create their own custom services (MBeans) and plug them into the JRun kernel.
- **Customization** – Unneeded services can be unplugged to avoid the overhead of unused functionality.
- **Modularity** – This flexible architecture allows JRun to quickly adapt to future changes in the J2EE specification, and to support different versions or implementations of an API by unplugging a service and replacing it with another one.

All the JRun services can take advantage of the clustering capabilities built into the JRun kernel.

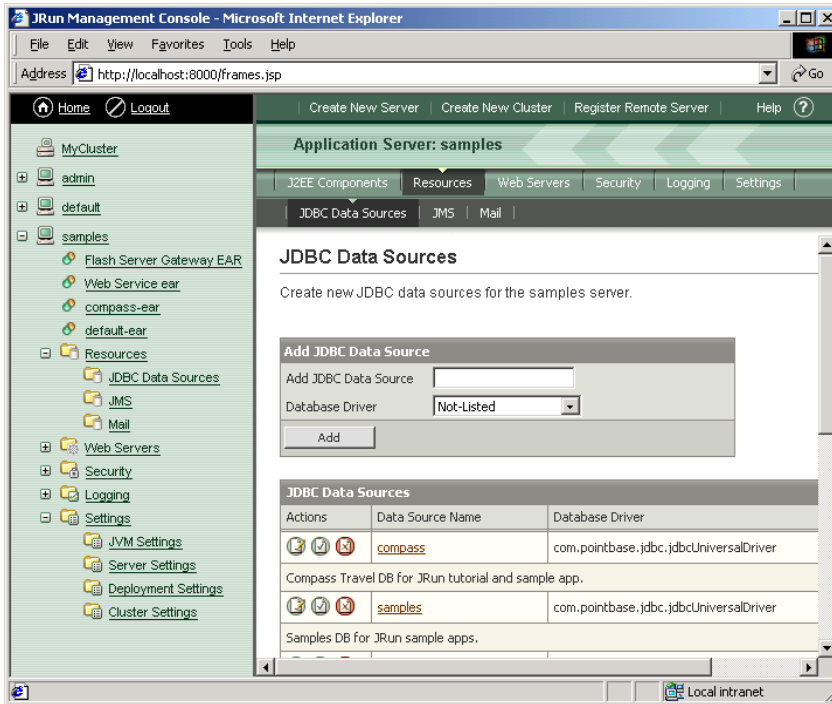
Adding your own custom service to JRun is easy: you simply add an entry to the `jrun.mlets` file. For example:

```
<mlet code="NotifyService"
      archive="NotifyService.jar"
      name=":mbean=NotifyService">
  <arg type="java.lang.String" value="admin@macromedia.com">
</mlet>
```

## JRun Management Console

The redesigned, JMX-enabled JRun Management Console (JMC) provides an easy-to-use, intuitive graphical user interface to manage your local and remote JRun servers.

Using the JMC, you easily create servers, define clusters, manage applications, and implement JAAS-based security.



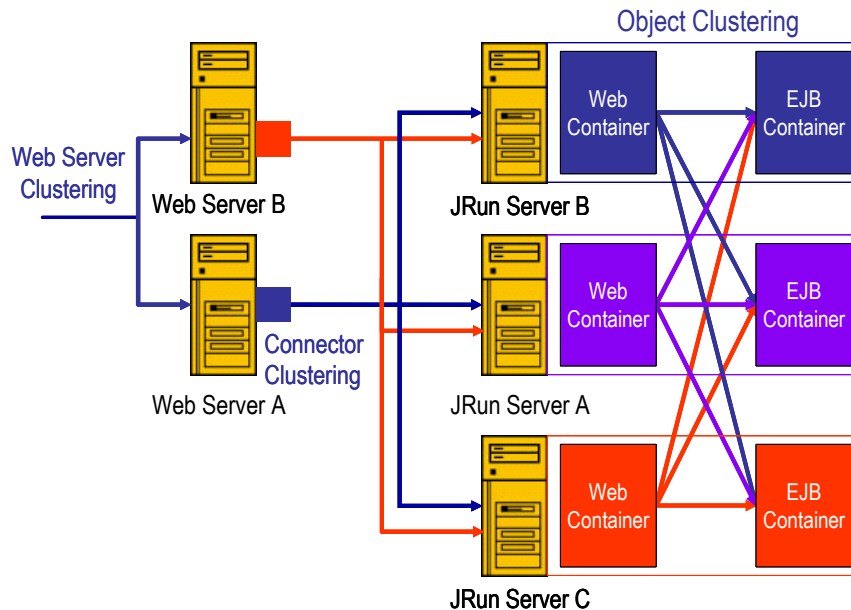
# Enterprise-Class Clustering

## Three-Level Clustering

JRun 4 comes with a new state-of-the-art clustering architecture that has already been praised by leading analysts and industry experts.

"JRun leverages the dynamic nature of Jini network technology to provide powerful J2EE clustering technology that is very easy for developers to employ," said Jim Waldo, distinguished engineer, Sun Microsystems, Inc. "Using Jini technology, JRun has created an elegant, peer-based mechanism for services such as Enterprise JavaBeans to discover and collaborate with one another. I am pleased to see Macromedia taking advantage of the adaptive qualities of Jini technology, enabling JRun to provide enhanced productive, performance, and flexibility for developers."

In JRun 4, clustering is implemented at three levels to offer the maximum level of reliability, scalability, and performance:



- Web Server Connector Clustering

A JRun connector is the piece of software that allows a web server (such as Netscape NES, Microsoft IIS, or Apache) to natively connect to JRun. In JRun 4, connectors provide load balancing and failover of HTTP requests sent by the web server to a cluster of JRun servers. File or JDBC-based session persistence, as well as in-memory session replication ensure that state information is never lost.

- Object Clustering

JRun 4 groundbreaking Jini-based object clustering enables load balancing and automatic failover of requests sent by clients (for example, JSPs, Servlets or regular Java clients) to server objects such as Enterprise JavaBeans, JMS queues and topics, JNDI trees, or other services. The object's state (for example the state of a stateful EJB) is automatically replicated in an optimized manner to provide the highest level of reliability while maintaining performance.

At server startup, clusterable services use Jini to register themselves with each server. The use of the Jini to power JRun 4 clustering architecture is generating a lot of interest and excitement among industry experts. James Gosling highlighted this innovative feature of JRun 4 in the opening keynote at JavaOne 2002.

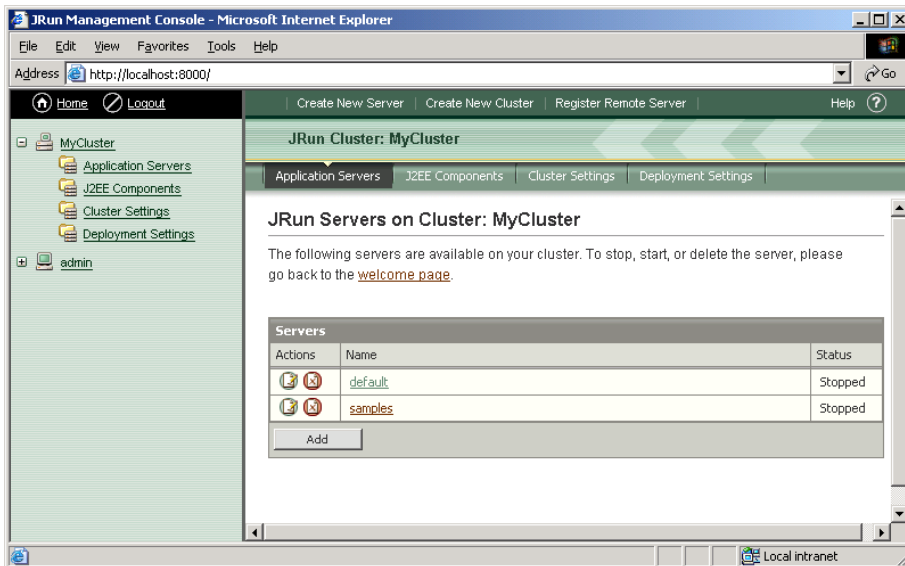


- **Web Server Clustering**

Web server clustering provides load balancing and failover of HTTP requests sent by browsers to a cluster of web servers. In JRun 4 web server clustering can be provided by Macromedia ClusterCats, a free alternative to hardware solutions.

## **Point-and-Click Cluster Administration**

Installing and managing a cluster is easy. Using the JRun Management Console (JMC), you define and administer a cluster in a point-and-click environment.



JRun also simplifies the deployment of applications in a clustered environment. The JRun Management Console (JMC) provides a one-step deployment option to all the servers in a cluster.

## Built-in Web Services Support

Web services allow incompatible systems to interoperate. The web services architecture is based on three standards:

- The Simple Object Access Protocol (SOAP), a lightweight XML-based protocol for sending messages and invoking methods on remote objects.
- The Web Services Description Language (WSDL), an XML-based language for describing a web services.
- The Universal Description, Discovery, and Integration (UDDI) API, a SOAP-based API for publishing and discovering web services.

JRun 4 provides built-in support for web services, SOAP, WSDL, and UDDI. With JRun 4, developers can publish and consume web services without having to deal with the low-level details of their underlying APIs. Using JRun 4, developers can also use web services to integrate their J2EE applications with Microsoft .NET.

## Publishing Web Services

JRun does not force developers to use a specific tool, learn a new component model, or use a new programming paradigm. With JRun 4, developers do not *write* web services: they keep writing Java classes, JavaBeans or Enterprise JavaBeans, and then *expose* these components as web services.

The simplest way to expose a Java class as a web service in JRun is to change its extension from .java to .jws. A java class with the extension .jws is automatically compiled, and exposed as a web service.

For finer control, developers can also expose regular Java classes, JavaBeans and Enterprise JavaBeans as web services using a deployment descriptor. The deployment descriptor allows developers to specify which methods to expose, type mapping information, and security requirements.

Exposing a web service using a deployment descriptor is simple. For example, the following deployment descriptor declares a web service called *LoanService* that exposes the *calculate* and *apply* methods of a Java class called *Loan*.

```
<service name="LoanService" provider="java:RPC">
  <parameter name="methodName" value="calculate apply"/>
  <parameter name="className" value="samples.Loan"/>
</service>
```

Once a web service is deployed, JRun automatically generates its WSDL upon request. The Web service consumer simply opens an HTTP connection with a URL including the name of the server, the name of the web service, and the WSDL parameter. For example:

<http://www.thejrunserver.com/services/LoanService?WSDL>

## Consuming Web Services

JRun provides different options to consume web services.

- Proxy-based invocation – JRun 4 includes a **wsdl2java** tool that generates Java proxies based on the WSDL description of a web service. Using the generated proxies, you only need three lines of code in a client application to invoke a web service method. For example:

```
LoanServiceLocator locator = new LoanServiceLocator();
Loan loan = locator.getLoan();
double payment = loan.calculate(20000, 36);
```

- Dynamic invocation – The JRun 4 web services client API also allows developers to dynamically invoke a web service without a proxy.
- Tag-based invocation – If the consumer is a JSP, the web service can also be invoked using JRun 4 web services tag library. For example:

```
<web:invoke
  url="http://www.theserver.com/services/LoanService"
  resultType="double"
  operation="calculate"
  result="monthlyPayment"
  scope="page">
  <web:param name="amount" value="20000"/>
  <web:param name="months" value="36"/>
</web:invoke>
```

## Deployment Services

### Automatic Deployment

JRun 4 sets a new standard in J2EE application deployment. With JRun, deploying an application is as simple as copying an EAR, JAR, WAR, or RAR file to the server directory. JRun automatically senses the new file, deploys it, and makes it available in the target server: no need to restart the server. JRun 4 also recognizes unpackaged applications or components. This feature is particularly helpful during development. For example, you don't have to repackage an EJB in a JAR file after each modification.

In addition to automatic deployment, JRun also supports manual deployment using the JRun Management Console (JMC).

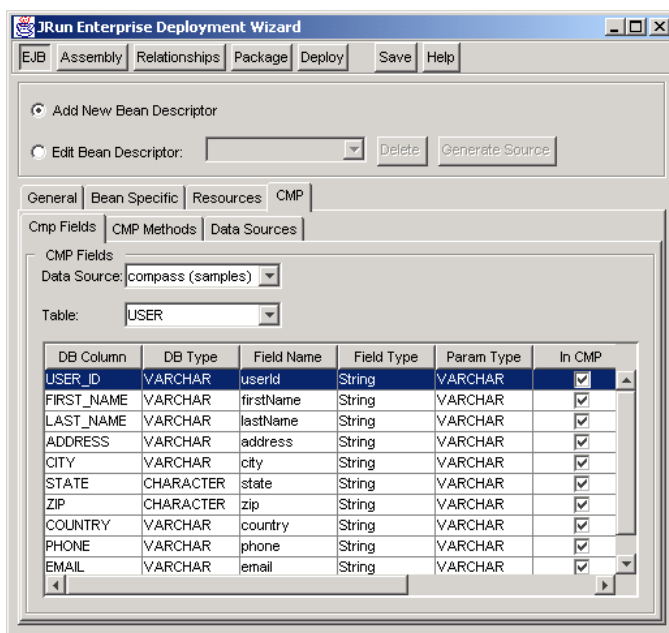
### Hot Modification

During development, developers can hot modify any application component (Servlets, JavaServer Pages, Enterprise JavaBeans, Java Classes, Tag libraries, deployment descriptors). Changes take effect immediately: No need to restart the server, ever.

### Tight IDE Integration

JRun 4 *Enterprise Deployment Wizard* streamlines the process of developing and deploying Enterprise JavaBeans. Using its Swing-based graphical user interface, you can create any type of EJB from scratch, or edit the deployment descriptor of existing EJBs, package them into JAR files, and deploy them to JRun. With the Enterprise Deployment Wizard new O/R mapping capabilities, you can generate and deploy a fully functioning Entity bean in minutes.

The Enterprise Deployment Wizard runs as a standalone tool, or as a plug-in on top of Borland JBuilder, Sun Forte, or IntelliJ IDEA.



## XDoclet Integration

XDoclet is a popular open source utility that can greatly simplify the development of J2EE components (such as Enterprise JavaBeans, Servlets, and custom tags) by allowing you to maintain your component meta-data using Javadoc-style tags in a single source file.

For example, to create an EJB, you traditionally have to maintain four different files: the home interface, the remote interface, the implementation class, and the deployment descriptor. Using XDoclet, you only maintain a single source file (the implementation class) that you annotate with special JavaDoc tags indicating how the auxiliary files should be built.

XDoclet has been tightly integrated into JRun 4. JRun senses modifications to the source file and automatically builds the auxiliary files (such as interfaces and deployment descriptors).

The following example shows the implementation class of a simple stateless session bean with the appropriate XDoclet annotations. When you modify and change this file, JRun's XDoclet service automatically:

- Generates the home and remote interfaces
- Compiles the implementation class, the home interface, and the remote interface
- Generates the deployment descriptor
- Deploys the EJB

```
/**
 * @ejb:bean    type="Stateless"
 *              name="Loan"
 *              jndi-name="ejb/Loan"
 * @ejb:transaction type="Required"
 */

public class LoanBean implements SessionBean {

    /**
     * @ejb:create-method
     */
    public void ejbCreate() {}

    public void ejbActivate() {}

    public void ejbPassivate() {}

    public void ejbRemove() {}

    public void setSessionContext(SessionContext context) {}

    /**
     * @ejb:interface-method
     */
    public double calculate(double amount, int months) {
        return (amount / months);
    }
}
```

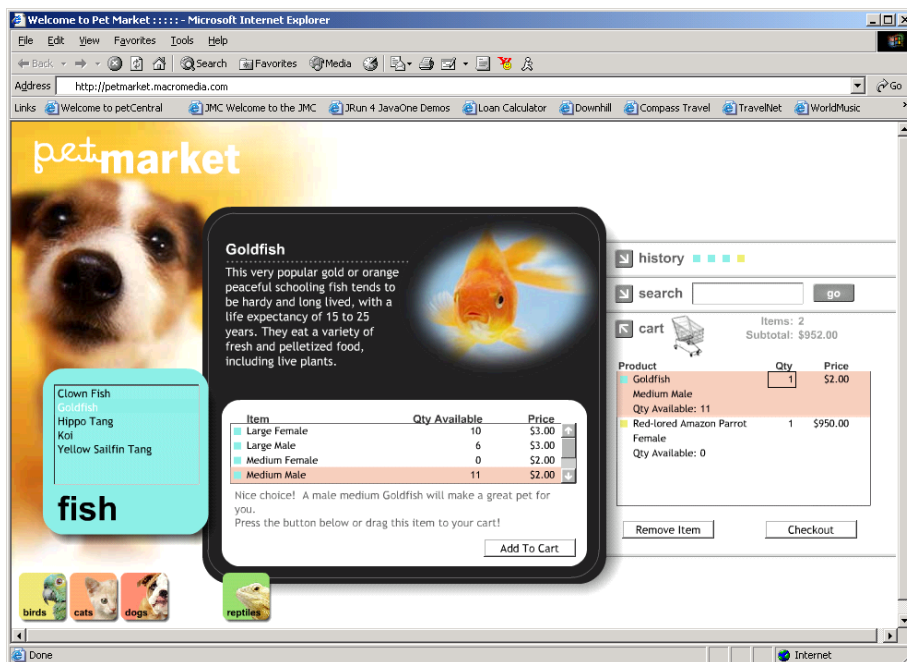


## Macromedia Flash MX Integration

As the range and complexity of applications being built on the Internet has grown, the limitations of HTML as a client programming environment have become increasingly apparent. As a result, developers are turning to rich client environments such as the Macromedia Flash Player to provide a higher level of interactivity and a better overall experience to end users.

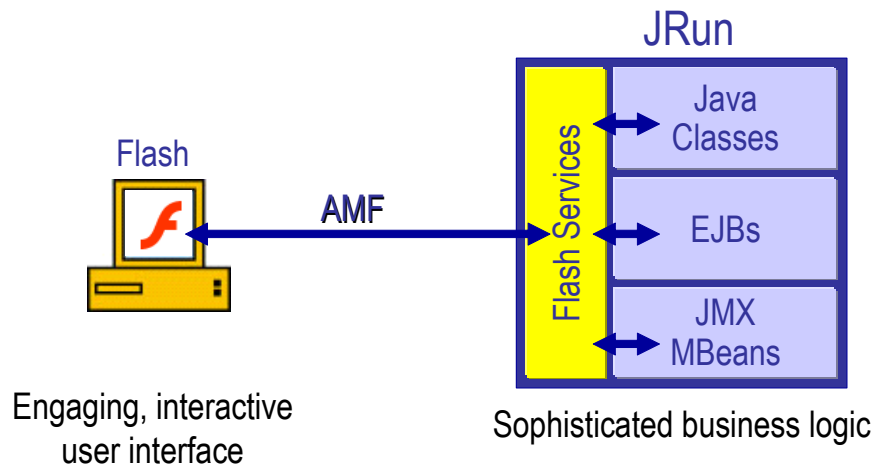
JRun 4 introduces powerful new capabilities for creating rich Internet applications that leverage the power of Macromedia Flash. With native connectivity to the Macromedia Flash Player and integrated debugging within the Macromedia Flash MX authoring environment, it's easy to provide J2EE applications with a highly interactive Macromedia Flash user interface.

The following screen shot shows a Flash user interface for Sun's petstore application. This interface provides a much better user experience. It is highly interactive, contained into one screen, and it supports zooming, and drag and drop.



## Macromedia Flash Remoting Service

The Macromedia Flash Remoting service provides a secure, high-performance connection between JRun and the Macromedia Flash player. A native part of the JRun environment, this service uses a high-performance binary message format called Action Message Format (AMF) that runs over HTTP to speed the transfer of data between client and server. Encoding and decoding are handled automatically by JRun and the Macromedia Flash client. Using the Flash Remoting Service, Flash user interfaces can natively invoke methods of regular Java Classes, Enterprise JavaBeans, and JMX MBeans deployed in JRun.



## Debugging Rich Applications with JRun and Flash MX

Macromedia Flash MX provides a powerful tool for debugging applications that use the Flash Remoting service, and this tool is tightly integrated with JRun. The NetConnect Debugger monitors calls between the Macromedia Flash Player, the Macromedia Flash application, and JRun – providing a unified view of application behavior from within the Flash MX authoring environment.

## Performance Tracing

JRun instrumentation feature allows you to identify and resolve performance bottlenecks in your code by tracing and timing each method invocation in your application. Instrumentation is supported for JSPs, Servlets, and EJBs.

## Web Server Integration

### Built-In Production Quality Web Server

JRun comes with its own integrated, production-quality Web Server.

### High performance Web Server Connectors

High performance and clustering-enabled Web server connectors also allow you to use JRun with all the major Web Servers including:

- Apache
- Microsoft IIS
- Netscape Enterprise Server
- IPlanet
- Zeus

## In-the-Box Database Connectivity

JRun supports any database for which a JDBC driver is available. In addition, JRun comes with type 4 JDBC drivers for all the major databases:

- Oracle
- Microsoft SQL Server
- Sybase
- DB2
- Informix

These drivers are for an unlimited number of connections and can be deployed with your application.

## OEM Toolkit

JRun OEM toolkit allows OEMs to customize the JRun installation and tailor JRun servers to their needs.

## Operating System Support

JRun runs on all the major operating systems, including:

- Windows 98/ME/NT/2000/XP
- Solaris 7, 8
- Red Hat Linux 6.2, 7.X
- IBM AIX 4.3, 5L v5.1
- HP-UX 11.0, 11i
- Compaq Tru64 Unix 5.1
- Turbolinux 6.5
- SuSE Linux 7.2, 7.3

## Java Virtual Machine (JVM) Support

JRun supports a wide variety of virtual machines:

- Sun Java Virtual Machine 1.3.1 or higher
- IBM Java Virtual Machine 1.3.1 or higher
- Appeal JRockit