MySQL Mart Tutorial

The Open Source SOA Platform

04/26/06 Version 1.0

MySQL Mart Tutorial

This document is a brief introduction to MySQL integrated with LogicBlaze FUSE with emphasis on the MySQL Mart Example. MySQL is an open source relational database management system using the Structured Query Language (SQL).

Rev.1

Contents

1. MySQL in LogicBlaze FUSE

LogicBlaze FUSE is distributed with one of two databases: one version is bundled with the Apache Derby database. The other version is bundled with a connector to the MySQL database.

A MySQL connector is pre-installed for you as part of the LogicBlaze FUSE distribution when you run the LogicBlaze FUSE for MySQL automatic installer for your platform. This connector allows you to connect to the MySQL 5.x Database.

> You must download and install MySQL 5.x separately as it is not bundled in this installation. See the Additional Resources section at the end of this document for the link to the MySQL Web site.

2. Overview of the MySQL Mart Demonstration

The MySQL Mart demonstration is an example of an inventory system for a grocery store. You can view the inventory of the store directly, let a supplier check the stock of an item they inventory, let a partner monitor inventory, and send transactions (random or user specified) to change inventory. The example demonstrates LogicBlaze FUSE using MySQL as the database server. The demo also shows the use of the following technologies: Apache ServiceMix, Apache ActiveMQ and Ajax.

The demo was implemented using two approaches : (1) the **Ajax Demo** shows the use of Ajax and a ServiceMix JDBC component, and (2) the **Non-Ajax Demo** shows the use of a `servlet/jsp` front-end interface. Both approaches were implemented to illustrate the differences between them. As you will see if you run both of them, the Ajax Demo is a much more efficient implementation.

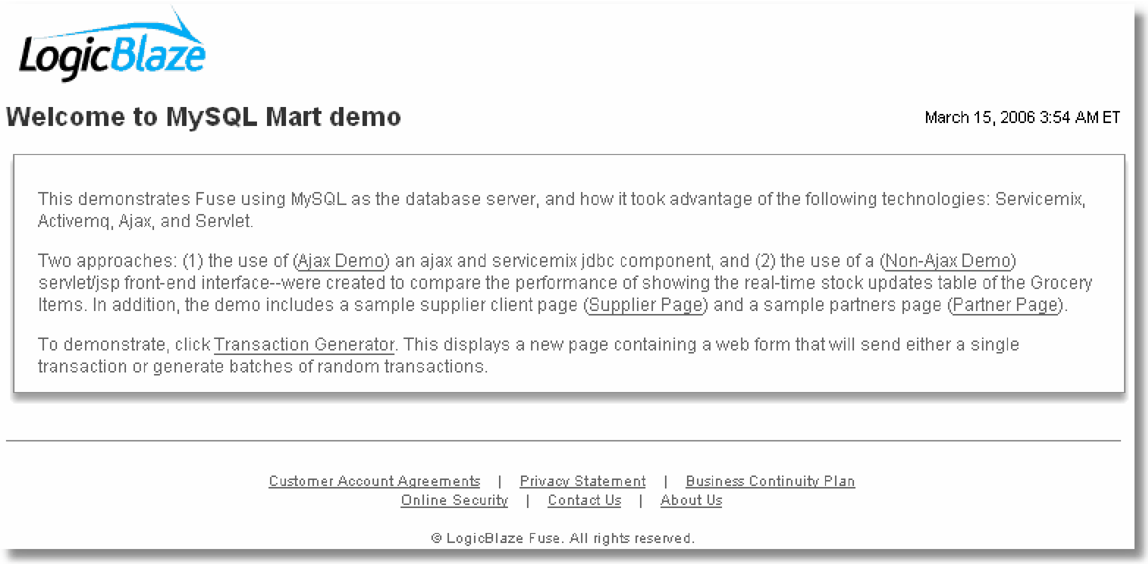
3. Ajax Implementation

This section shows the details of the first (and preferred) approach using Ajax.

3.1. Running the Demonstration

To run the Ajax demo perform the following steps:

1. Go to `http://localhost:8080/mysql-mart/`



LogicBlaze

Welcome to MySQL Mart demo

March 15, 2006 3:54 AM ET

This demonstrates Fuse using MySQL as the database server, and how it took advantage of the following technologies: Servicemix, Activemq, Ajax, and Servlet.

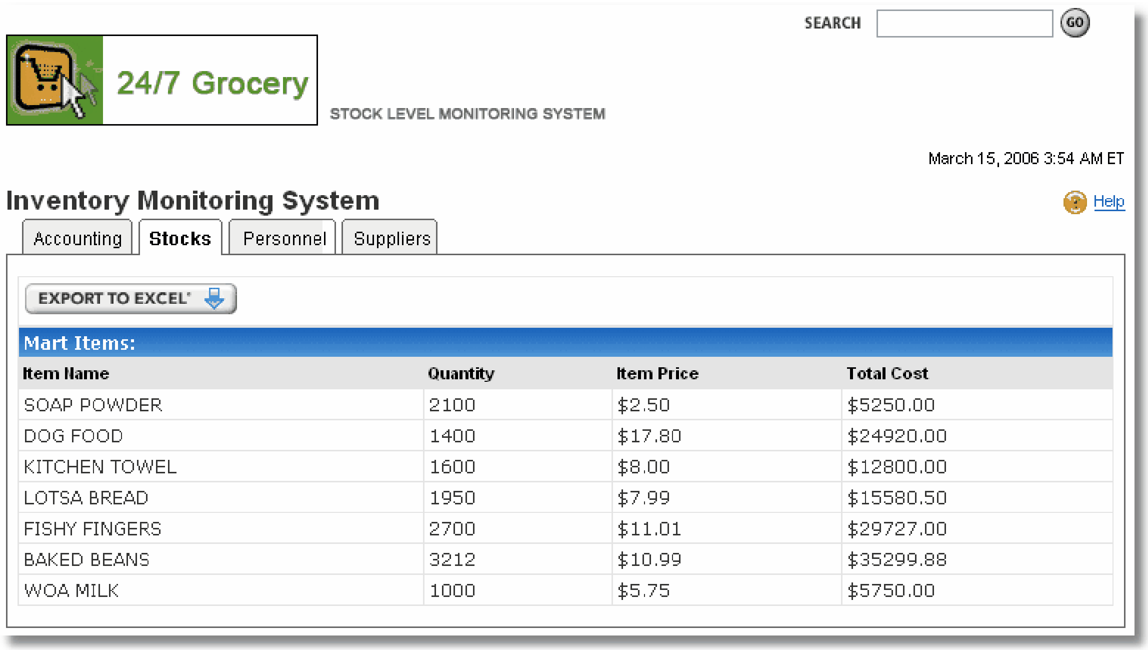
Two approaches: (1) the use of (Ajax Demo) an ajax and servicemix jdbc component, and (2) the use of a (Non-Ajax Demo) servlet/jsp front-end interface--were created to compare the performance of showing the real-time stock updates table of the Grocery Items. In addition, the demo includes a sample supplier client page (Supplier Page) and a sample partners page (Partner Page).

To demonstrate, click Transaction Generator. This displays a new page containing a web form that will send either a single transaction or generate batches of random transactions.

Customer Account Agreements | Privacy Statement | Business Continuity Plan
Online Security | Contact Us | About Us

© LogicBlaze Fuse. All rights reserved.

2. Click on the “Ajax Demo” link. This will show you the inventory of the “24/7 Grocery” store.



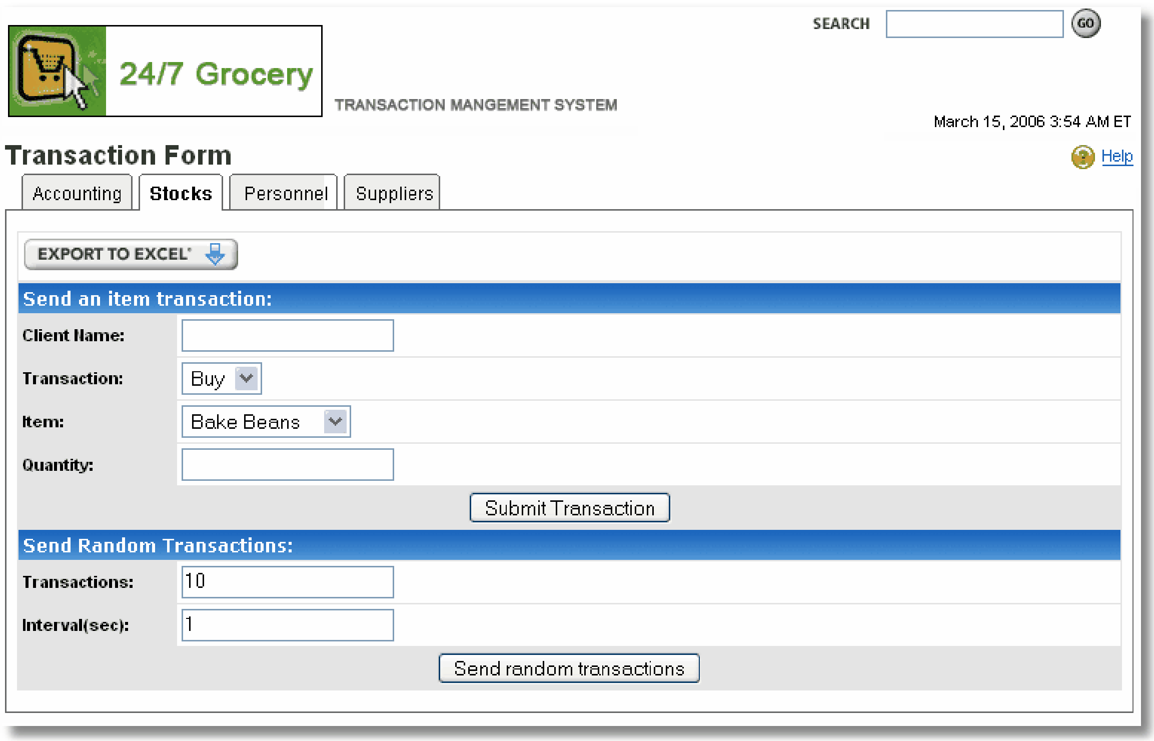
March 15, 2006 3:54 AM ET

Inventory Monitoring System

Mart Items:

Item Name	Quantity	Item Price	Total Cost
SOAP POWDER	2100	$2.50	$5250.00
DOG FOOD	1400	$17.80	$24920.00
KITCHEN TOWEL	1600	$8.00	$12800.00
LOTSA BREAD	1950	$7.99	$15580.50
FISHY FINGERS	2700	$11.01	$29727.00
BAKED BEANS	3212	$10.99	$35299.88
WOA MILK	1000	$5.75	$5750.00

3. Click the link to open the "Transaction Generator". You may chose to specify a values and submit a single transaction or send 10 random transactions. Please watch your "Inventory Monitoring System" window to see the results of this action.



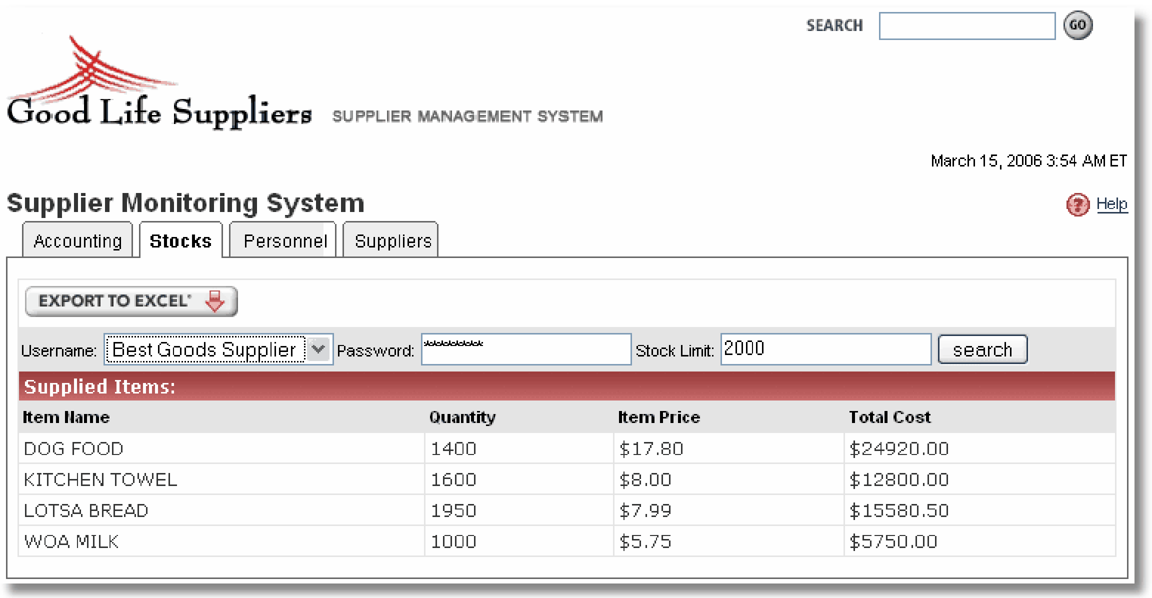
Send an item transaction:

Client Name:	
Transaction:	Buy
Item:	Bake Beans
Quantity:	

Send Random Transactions:

Transactions:	10
Interval(sec):	1

4. Click the link to open the "Supplier Page". Enter a value in the `Stock Limit` field. Click "Search" to see what items have gone below the limit.

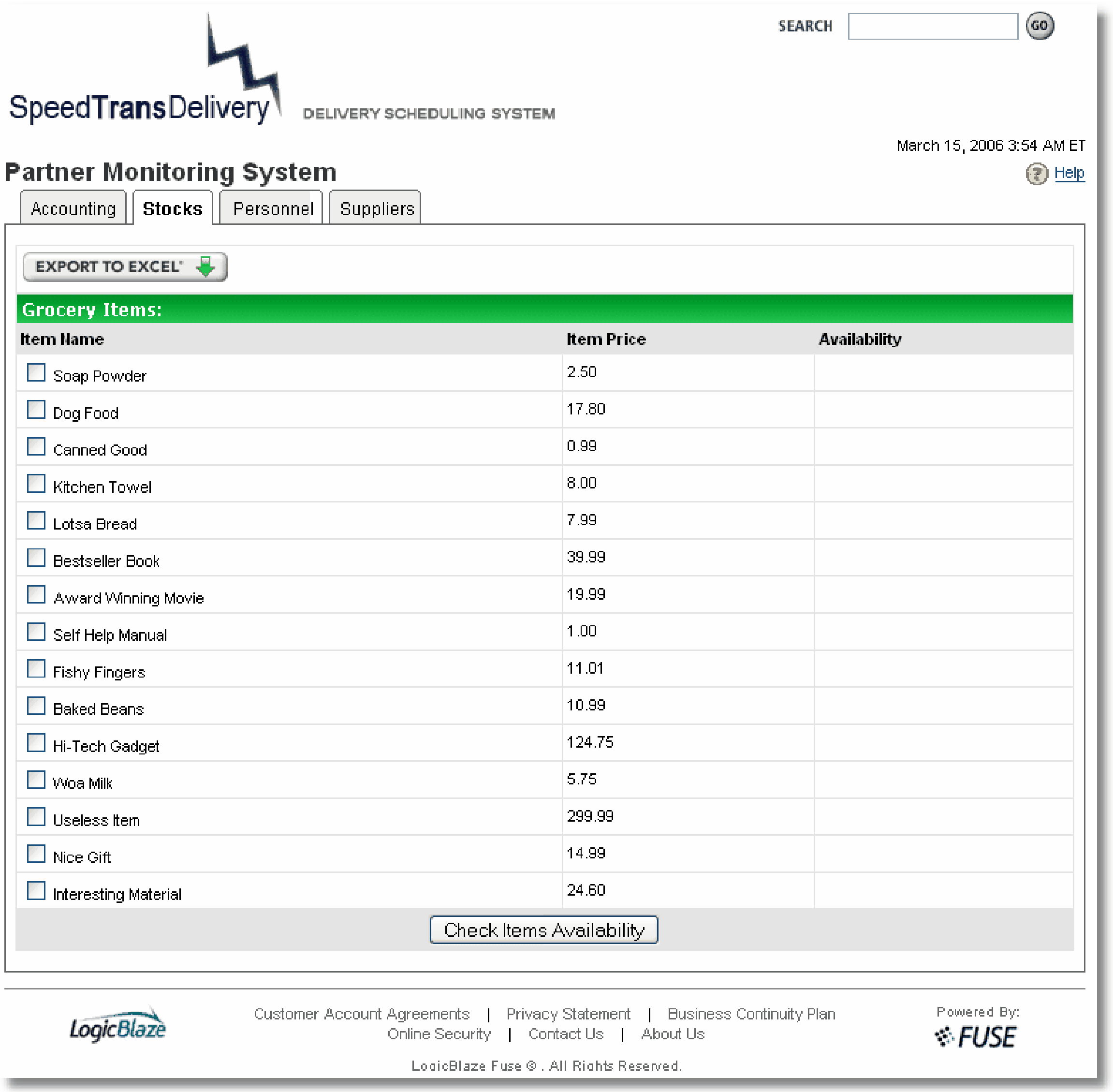


Username: Best Goods Supplier Password: ******** Stock Limit: 2000

Supplied Items:

Item Name	Quantity	Item Price	Total Cost
DOG FOOD	1400	$17.80	$24920.00
KITCHEN TOWEL	1600	$8.00	$12800.00
LOTSA BREAD	1950	$7.99	$15580.50
WOA MILK	1000	$5.75	$5750.00

5. Click the link to open the “Partner Page”. Select as many check boxes as you wish in “Grocery Items” column and click “Check Items Availability” to view the availability of each individual grocery item.


SEARCH
GO
SpeedTransDelivery
DELIVERY SCHEDULING SYSTEM
March 15, 2006 3:54 AM ET
Help
Accounting
Stocks
Personnel
Suppliers
EXPORT TO EXCEL
Check Items Availability
LogicBlaze
Customer Account Agreements | Privacy Statement | Business Continuity Plan
Online Security | Contact Us | About Us
Powered By: FUSE
LogicBlaze Fuse ©. All Rights Reserved.

Partner Monitoring System

Grocery Items:

Item Name	Item Price	Availability
☐ Soap Powder	2.50	
☐ Dog Food	17.80	
☐ Canned Good	0.99	
☐ Kitchen Towel	8.00	
☐ Lotsa Bread	7.99	
☐ Bestseller Book	39.99	
☐ Award Winning Movie	19.99	
☐ Self Help Manual	1.00	
☐ Fishy Fingers	11.01	
☐ Baked Beans	10.99	
☐ Hi-Tech Gadget	124.75	
☐ Woa Milk	5.75	
☐ Useless Item	299.99	
☐ Nice Gift	14.99	
☐ Interesting Material	24.60	

3.2. How the MySQL Mart Demo Works

The diagrams below is representative of the architecture we have used to demonstrate the use of MySQL + Ajax. The first diagram is a complete view of the demonstration. For simplicity the main diagram has been broken into many diagrams (shown later) representing each of the individual systems.

Because the `MySQL Database`, `JDBC Component`, `Inventory View Component`, and `Inventory Transport Component` are the same in each example we will discuss them separately first.

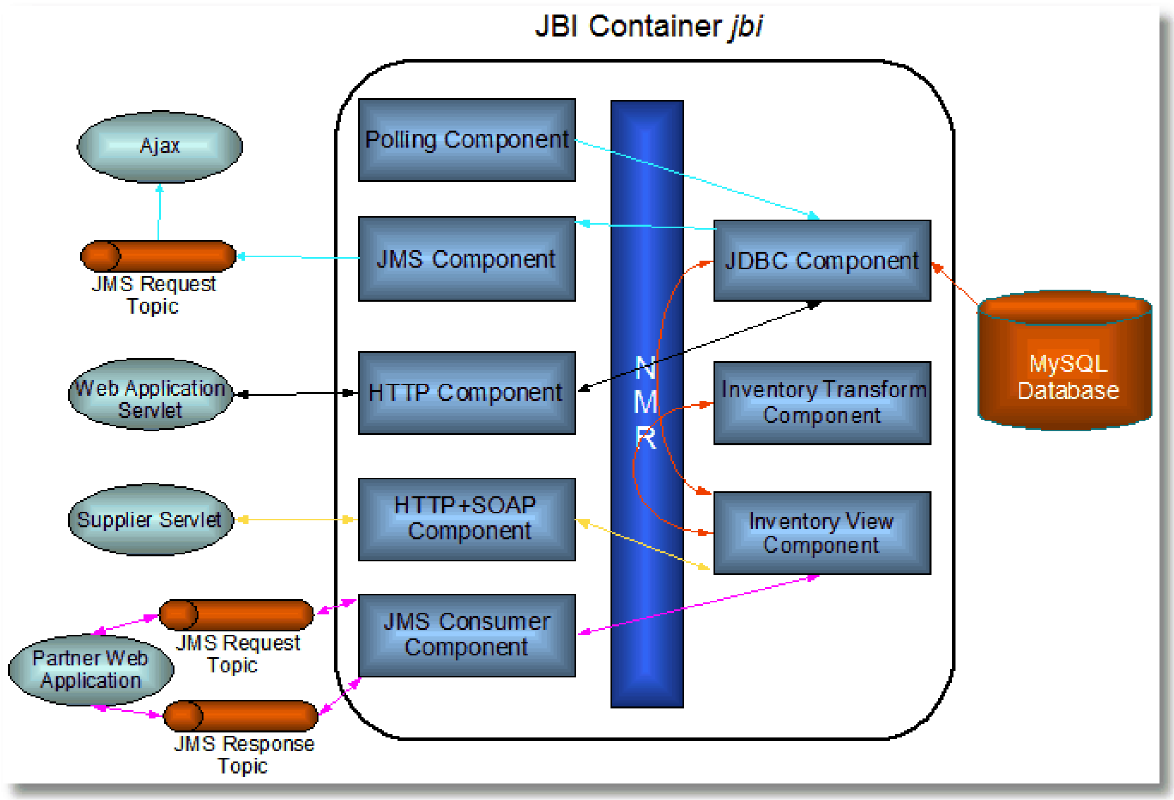


Figure 1: All Systems View

- **The MySQL Database** is responsible for maintaining information. We have two tables that are maintained by the database. The database name is `inventorydb` and the tables are named `transactions` and `items`. All but the Transaction Web Application System accesses the Items table. The Transaction Web Application System accesses the Transactions table.

  The tables and database are created automatically at the start of the example. The file `dbcreator.sql` contains all the SQL statements for the creation and initialization of the databases and tables.

- **The JDBC Component** is a lightweight service engine[1] component. This is the only component that can access the database directly. It communicates with the database using JDBC messages. It is important to note that this component does not modify messages - it just processes them for the next destination. It sends and receives normalized messages when communicating with components that wish to access the database. The normalized message consists of XML wrapped around a SQL statement.

```
<sql> sql statement </sql>
```

- **The Inventory View Component** is a chained component and therefore serves only to route messages. It makes no changes to the messages itself. We need this routing component because of the nature of messaging in JBI. The numbered list below takes you step by step through the flow of the message:
    1. The `Inventory View Component` receives the message from the a source such as the `HTTP+SOAP Component`. This message is a JBI Message containing information pertaining to fields of the table.
    2. The message is routed on to the `Inventory Transform Component`.
    3. The `Inventory Transform Component` modifies the message and sends it back to the `Inventory View Component`.
    4. The `Inventory View Component` sends the newly transformed message on to the `JDBC Component`.
    5. After the `JDBC Component` has processed the message it sends it back to the the `Inventory View Component`.
    6. Lastly the `Inventory View Component` sends the message back to the original sender (for example, `HTTP+SOAP Component`).
- **The Inventory Transform Component** takes in a JBI message with information about the MySQL tables and transforms the information into SQL statements. The Inventory Transform Component receives the message from the Inventory View Component, transforms the message into SQL statements, then sends the transformed message back to the Inventory View Component. This is the last change to the message before it goes on to the JDBC Component.

1 A lightweight component is a POJO that cannot accept service unit deployments. A lightweight component is deployed on a standard JBI component called `servicemix-lwcontainer`. For more information on this please see: http://www.servicemix.org/What+is+a+lightweight+component.

The Polling System

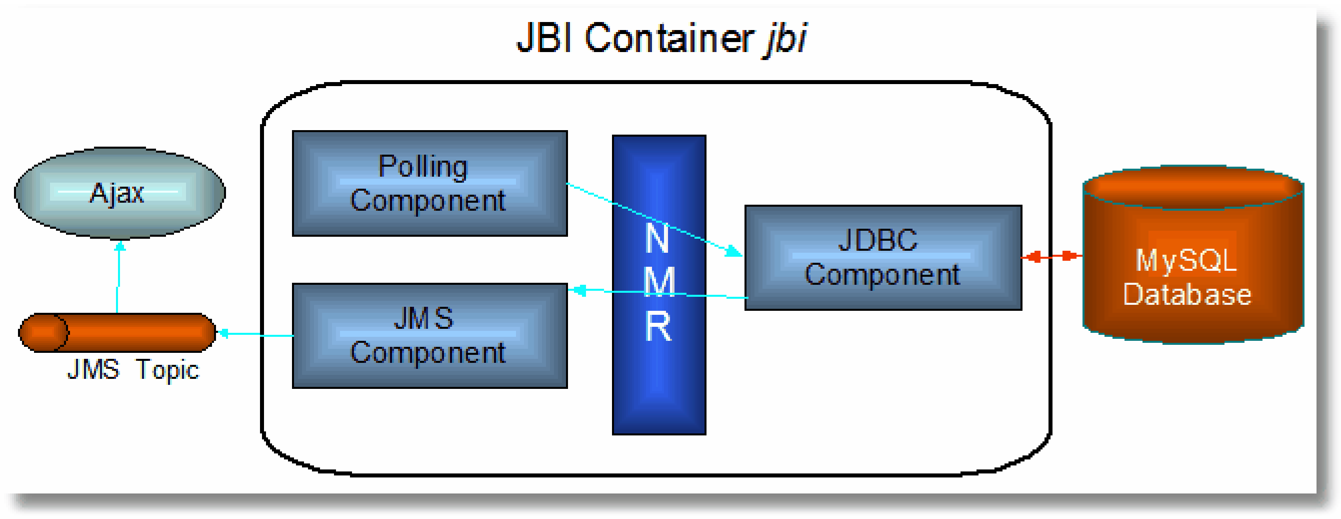


Figure 2: The Polling System

- **The Polling Component** is a service engine used to check and see if the values of the stock have been updated. To do this we check the delta (change) of the time stamps every few seconds. The check consists of a normalized message to the JDBC Component. Inside the normalized message is a SQL statement in the form of a string that gets compared to the previous value.

```
message.setContent(createSqlQuerySource(
 "SELECT item_name, curr_stock, item_price " +
 FROM items WHERE last_update > '" + lastUpdate.toString() + "'"));
```

  The polling component is an example of communicating with the MySQL database from within the JBI container. This is a one way communication. The results of the action are sent on to another location. In this example it is the JMS Client.

- **The JMS Component** receives JBI messages from the JDBC Component as a result of the polling. The messages are the results of the database being polled by the polling component.
- **The JMS Topic** receives JMS messages from the JMS Client. The Topic takes the JMS message and makes it accessible to the Ajax layer to acquire via polling.
- **The Ajax Layer** is used to communicate the data back to the screen in real time without refreshing.

The Transaction Web Application System

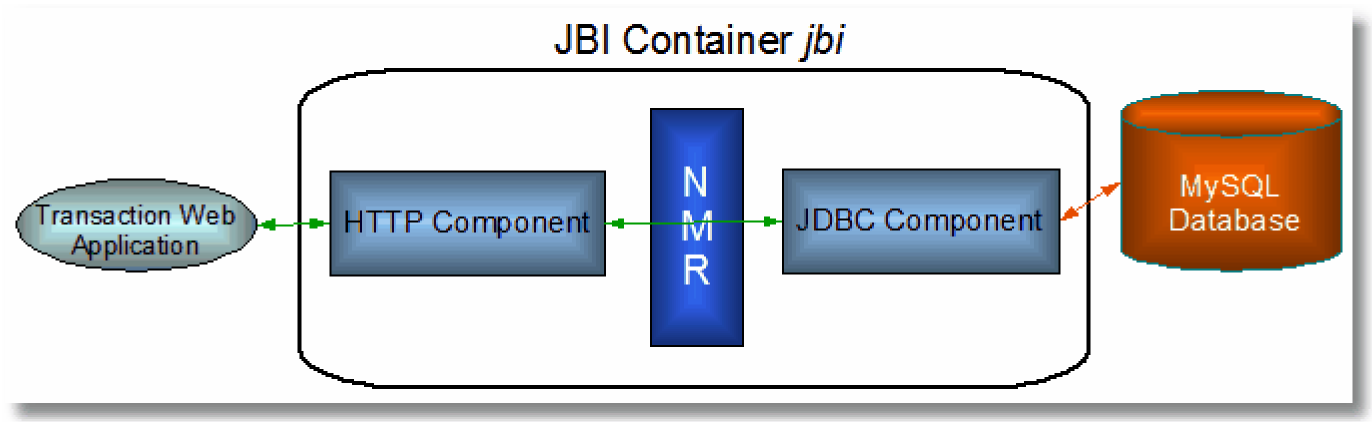


Figure 3: The Transaction Web Application System

- **The Transaction Web Application** adds new transactions to the transactions table. This is an example of an application that sits outside of the JBI container. It sends a message to the HTTP `Component` in the form of HTTP requests.
- **The HTTP Component** is a binding component used to communicate with applications that are external to the JBI environment. In this case, the `HTTP Component` accepts HTTP requests from the Transaction Web Application. The `HTTP Component` takes the HTTP message and normalizes it. The normalized message is then routed via the Normalized Message Router (NMR) to the `JDBC Component`.

The Supplier System

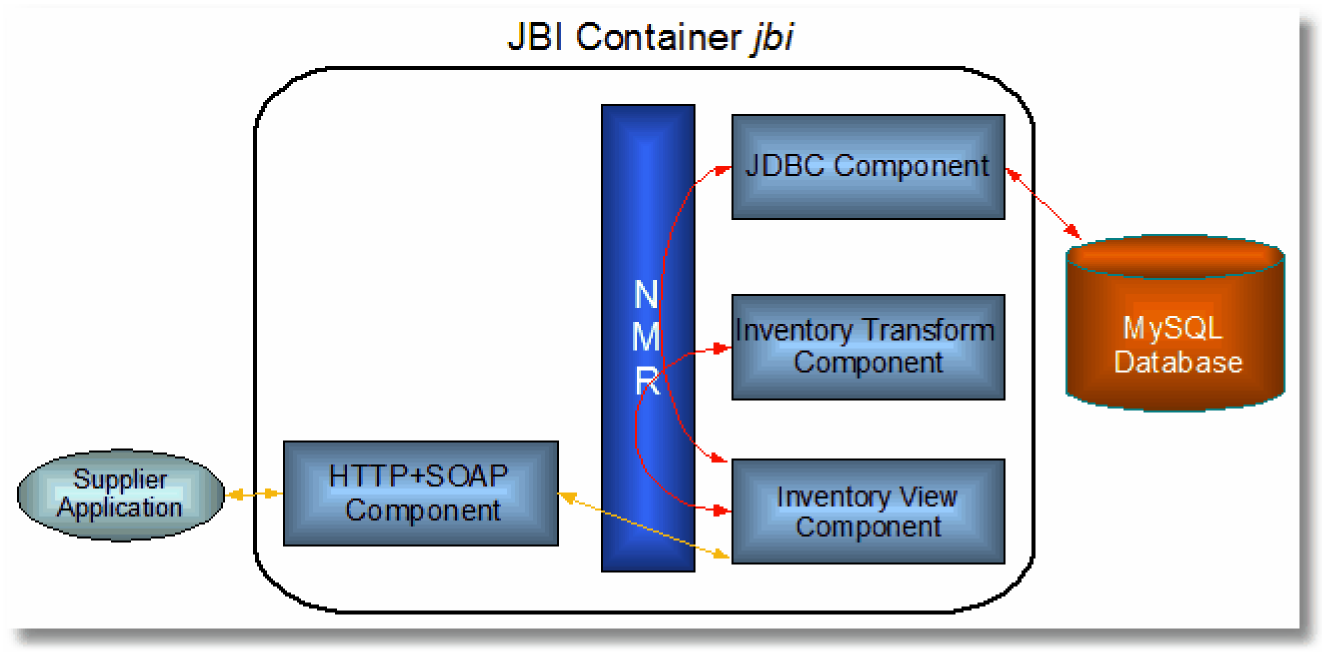


Figure 4: The Supplier System

- **The Supplier Application** is representative of a supplier store. This application serves as a way for the Supplier to query the grocery application for the current amounts in the grocery supply. The supplier only needs to know how to message in SOAP to be able to do this. The message (query) is sent to the `HTTP+SOAP Component`. When the result is determined the Supplier Application will receive a reply from the `HTTP+SOAP Component` with the results of the query.
- The `HTTP+SOAP Component` is a binding component similar to the `HTTP Component` (discussed above) with the obvious distinction of being able to accept Simple Object Access Protocol (SOAP) requests. The SOAP message is contained in an envelope that is sent to the `HTTP Component`.
- The `SOAP component` strips the SOAP envelope and extracts the fields and places them into a JBI message to send to the `Inventory View Component`. The `Inventory View Component` then returns the Message to the `HTTP+SOAP component` so that the results can be repackaged and sent back to the Supplier Application.

The Partner Web Application System

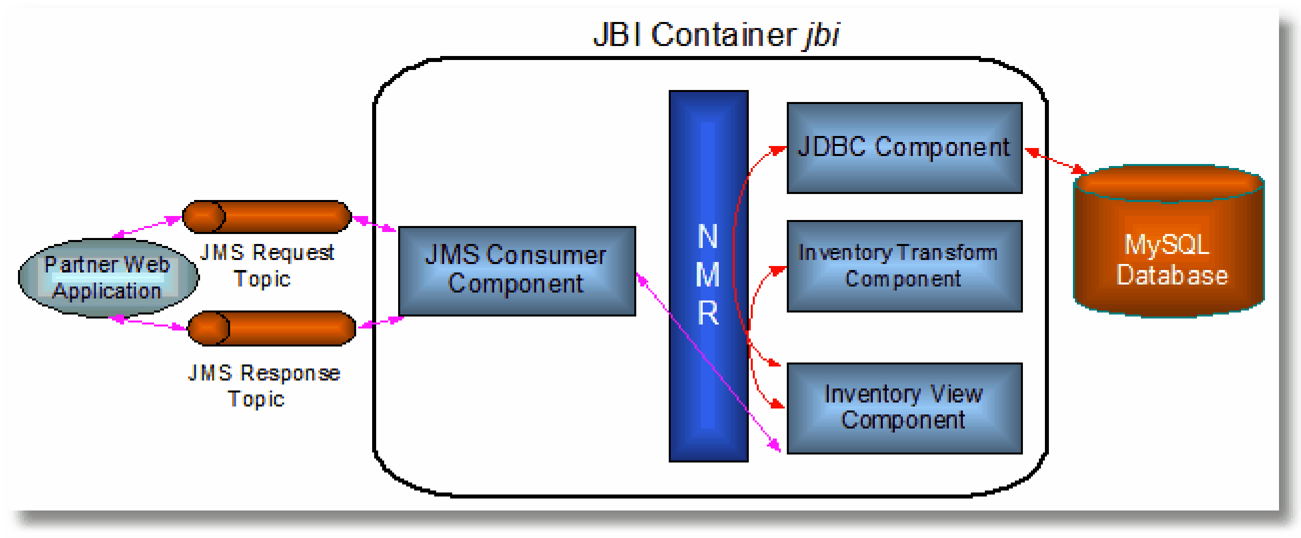


Figure 5: The Partner Web Application System

- **The Partner Web Application** checks for the availability of items in the store. It is representative of a partner company. It does this by posting a JMS request message to the JMS Topic.
- **The JMS Request Topic** is where the Partner Web Application places a request for checking availability. The `JMS Consumer Component` will access the request from the list location.
- **The JMS Response Topic** is where the results are placed by the `JMS Consumer Component`. The Partner Web Application will access the results of the query from here. The response is a simple message indicating if the item is in stock.
- **The JMS Consumer Component** provides a JMS request-response mechanism. It accesses requests that has been placed in a the `JMS Request Topic` then normalizes the information into a JBI message. This message just consists of the query that is to be executed. The `JMS Consumer Component` sends this message on to the `Inventory View Component`.
- **The JMS Consumer** receives the message back from the `Inventory View Component` in the form of a JBI message. The information is then repackaged into a JMS message where it is placed on the `JMS Response Topic`.

4. Non-Ajax Implementation

This section shows the details of the (non-preferred) approach without Ajax.

4.1. Running the Demonstration

To run the Non-Ajax demo perform the following steps:

1. Go to `http://localhost:8080/mysql-mart/`
2. Click the “Non-Ajax Demo” link. This will show you the inventory of the “24/7 Grocery” store.
3. Click the link to open the “Transaction Generator”. You may chose to specify a values and submit a single transaction or send 10 random transactions. Please watch your “Inventory Monitoring System” window to see the results of this action.
4. Click the link to open the “Supplier Page”. Enter a value in the “Stock Limit” field. Click “Search” to see what items have gone below the limit.
5. Click on the link to open the “Partner Page”. Select as many check boxes as you wish in “Grocery Items” column and click “Check Items Availability” to view the availability of each individual grocery item.

4.2. Details

The functionality of the Ajax and the non-Ajax Demonstrations are essentially the same, but the execution has two very distinct feels. The Ajax demonstration (the one we prefer) is cleaner in its feel when compared to the non-Ajax demonstration. Notice how the non-Ajax demonstration has to refresh the whole web page. The Ajax demonstration just refreshes the numbers.

5. Additional Resources

For more information about MySQL please see http://www.mysql.com/.

MySQL Documentation for 5.0 - http://dev.mysql.com/doc/refman/5.0/en/index.html

For more information on lightweight components please see http://www.servicemix.org/What+is+a+lightweight+component.

For more information on deploying lightweight components please see http://docs.codehaus.org/display/SM/Deploying+Lightweight+Components+Tutorial.