



FITECH LABORATORIES

---

xTier™ 2.2.1 - Examples Guide

Copyright © Fitech Laboratories Inc.  
300 Montgomery Street • Suite 621  
San Francisco, CA 94104  
Phone 415.371.8234 • Fax 415.371.8237

# Table of Contents

Introduction .....	2
Compiling and Running .....	3
Ant Build File .....	3
Compiling and Running the Examples.....	3
Example Configuration .....	4
JTA Transaction (Tx) Service Example.....	4
Cache Service Example .....	4
Email Service Example .....	4
xTier™ Cache Service Example.....	5
xTier™ Grid Service Example.....	6
xTier™ Sequence Pattern Matching Example .....	8

## Introduction

xTier™ 2.2.1 comes with a number of well documented examples. All the examples can be found in the `%XTIER_ROOT%/examples` directory. After installing xTier™ the developer should take some time to browse through the examples shipped with the product in order to get a better feel on where and how to use xTier™ services in their applications. All of the xTier™ service examples are well commented and self explanatory, however, some services may warrant special attention due to the interesting features they demonstrate or certain configuration parameters that the developer may need to adjust.

All of the sample code for the individual services can be found in the `com.fitechlabs.xtier.examples.services` package. Each individual service example is further divided into a package respectively. There are several other packages that are included as part of the sample code.

The package `com.fitechlabs.xtier.examples.utils` contains the `Utils` class, which has several different utility methods that are used in the service examples. Of particular interest is the `startXtier()` method, which is responsible for starting the xTier™ Kernel using the micro-kernel.

The package `com.fitechlabs.xtier.examples.jboss3` contains the micro-kernel for JBoss 3.x.

The final package, `com.fitechlabs.xtier.examples.microkernel`, contains the standard (in-process) micro-kernel implementation which is used to start the xTier™ Kernel.

## Compiling and Running

All the examples are located under the `%XTIER_ROOT%/examples` directory. The code samples can all be individually executed as they all have `main()` methods. Please note that the cache and transaction examples require the presence of an Oracle database. Several of the services require configuration for the specific environment that they are running in. Any configuration that needs to occur is noted in the following sections.

### Ant Build File

Included with the xTier™ 2.2.1 examples is a build file for use with Ant. The target to compile the application is `compile` which relies on `init` (for initialization) and `clean` (to clean all previously compiled class files). Additionally, the examples can be run using Ant. To obtain a list of the specific targets that run the individual examples use the `list` target. (I.e. type `ant list` from the command line.)

### Compiling and Running the Examples

xTier™ can be easily compiled from with an IDE, or alternatively, they can be compiled and executed from the command line utilizing the provided Ant `build.xml` file.

In order to use the build file for compiling or executing the xTier™ 2.2.1 examples, a properly installed and configured version of Ant must be present on the development machine.

Please note, several of the examples require configuration in order to execute. The following section provides the information required to

configure these example. (Note, any code changes that are required within the examples are clearly marked in their respective **.java** files.)

To run these examples, please ensure that Ant is accessible from the path, and change to the **%XTIER\_ROOT%\examples\java** directory.

To compile the applications: **ant compile**

To obtain the list of examples to execute: **ant list**

To run a given example: **ant <service.name>**

for example, **ant service.info**

## Example Configuration

### JTA Transaction (Tx) Service Example

The transaction service example requires database connectivity. In order to be able to run this example, the user will need to change the value of the **JDBC\_URL** constant in the **TxServiceExample** class to reflect the database connectivity parameters of the local environment. Note that the underlying database *must support* XA transactions.

### Cache Service Example

This example, just like the transaction service example, also requires database connectivity and, in order to run it, the user will need to change value of the **JDBC\_URL** constant in the **CacheDbManager** class to reflect the database connectivity settings of the local environment.

### Email Service Example

The email service example demonstrates how to use the simple API provided by the xTier™ email service to send email messages in both the Java and .NET environments. In order to run this example the user will need to make several changes. First, in **xtier\_kernel.xml** the attribute value for **smtp-host** should be changed to an appropriate SMTP server name. Second, within the **EmailExample** class, the email address must be changed to a valid email address.

## xTier™ Cache Service Example

The xTier™ Cache provides a high-performance *non-replicable* cache. Simply stated, non-replicable cache does not replicate cache elements across the cluster when an object is created or updated but rather sends a small invalidation message to all of the nodes. For a detailed description of this, see the Javadoc for the Cache service.

When using this service, there are two interfaces for the cache, the first, **com.fitechlabs.xtier.services.cache.Cache**, and the second **com.fitechlabs.xtier.services.cache.jcache.JCache** extends the **Map** interface. The main distinction between the two is that the Cache interface throws *checked exceptions*, while **JCache** throws *unchecked exceptions* as well as the fact that the xTier™ JCache conforms to the JCache specification.

An example demonstrating the xTier™ Cache Service can be found in the **%XTIER\_ROOT%/examples/java/com/fitechlabs/xtier/examples/services/cache** directory.

It is important to note that this example requires a properly configured Oracle database. The connection URL for the Oracle instance needs to be specified in the **CacheDbManager** class. (The instances that need to be updated are noted in the code with TODO: tags.) All configuration of the tables utilized by the cache are created and initialized by the example code. In addition to the database URLs you need to have the appropriate Oracle drivers in the classpath (JDBC drivers for Oracle can be found at **%XTIER\_ROOT%/lib/ext** folder).

## xTier™ Grid Service Example

The xTier™ 2.2.1 Grid service provides the developer with all of the tools to create an application that can run within a *computational grid* without writing all of the “plumbing” required to distribute the task, aggregate the results, and manage the cluster of computers. See the Javadoc for a discussion of computational grids.

xTier™ 2.2.1 ships with a grid enabled prime number factoring example located in the

`%XTIER_ROOT%\examples\java\com\fitechlabs\xtier\examples\services\grid` directory. The example consists of four classes:

**GridServiceExample.java** (the main example class), **PrimeFinder.java** (The implementation of the prime number factoring algorithm),

**PrimeGridTaskUnit.java** (the implementation of each individual grid task to be distributed over the grid), and **PrimeGridTaskUnitFactory.java**

(the implementation of the factory to create **PrimeGridTaskUnit** instances). The code examples are well documented and the Javadoc

discussing the API is extensive. As such, this manual will concentrate on the additional configuration parameters that need to be set and provide a description of the mechanics of running an xTier™ 2.2.1 Grid Service application.

The xTier™ Grid Service relies on the xTier™ Cluster Service to provide the clustering capabilities. In order to configure the Grid Service Example there are configuration parameters in several configuration files which must be updated to appropriately reflect the multiple machines that the application will be running on.

As such, there are several parameters that need to be changed in the configuration files. Launch the xTier™ 2.2.1 Management Console and open **xtier\_cluster.xml** (located in the **examples** directory off the xTier™ root). The first setting that needs to be changed is within the **<network**

... /> tag, specifically **mcast-ttl** attribute. This parameter represents the number of "hops" that are required to talk to the other nodes in the cluster. Initially this attribute is set to **mcast-ttl="0"** which facilitates execution *on the local machine only*. For a machine participating in the cluster local network the value should be set to **mcast-ttl="1"**. The number of hops for other configurations will vary with the network topology. A good way to check how many "hops" are required is to use a tool such as Trace Route (**tracert** on Windows).

The next section to be updated, is the **<seed node/>** tag. This section contains all of the nodes that this instance of xTier™ should try to contact when starting. The following section illustrates a configuration where one other node is to be contacted.

```
<join retries="2">
  <seed-nodes>
    <node addr="192.168.1.100" port="64002"/>
  </seed-nodes>
</join>
```

For every seed node to contact an appropriate **<node addr="" port="" />** should be created. When a node tries to join the cluster it will first send an IP-Multicast join-cluster advertisement. If no reply is received, it will attempt to contact each of the seed-nodes specified, and if there is still no reply, it will assume that it is first in the cluster. Please note, for an absolute guarantee of correct completion of the startup sequence (in that all nodes of the cluster are participating in the cluster), make sure to specify all possible cluster members.

When considering the topology of the nodes in your grid, choose one of the nodes to be the "driver" node, while the remaining nodes will be "standby" nodes. All of the nodes must have a running instance of xTier™ and have the class files for the implemented grid tasks. (Note, while there is a logical distinction amongst the nodes between a "driver" and the "standby" node(s), xTier™ itself does not make any distinction. That is to say, the distinction is completely arbitrary, as the xTier™ installed and running on each of the workers has no limitation.) For a detailed discussion of the relationship please refer to the Javadoc.

## xTier™ Sequence Pattern Matching Example

xTier™ 2.2.1 ships with comprehensive example demonstrating simulation involved search for sequence patterns match with given pattern and score threshold. Simulation is performed in a parallel fashion over the large database of existing patterns using grid computing and distributed caching capabilities provided by Fitech's xTier™.

Sequence Pattern Matching Example including fully commented source code, all necessary scripts, Ant build and full documentation is located at **%XTIER\_ROOT%\examples\match-grid** folder.