* [Skip to primary navigation](https://j2eereference.com/singleton-in-clustered-environment/" \l "genesis-nav-primary)
* [Skip to content](https://j2eereference.com/singleton-in-clustered-environment/#genesis-content)
* [Skip to primary sidebar](https://j2eereference.com/singleton-in-clustered-environment/#genesis-sidebar-primary)
* [Skip to footer](https://j2eereference.com/singleton-in-clustered-environment/#genesis-footer-widgets)
* [**CORE JAVA**](https://j2eereference.com/category/core-java/)

* [**DESIGN PATTERNS**](https://j2eereference.com/category/design-patterns/)

* [**JSP**](https://j2eereference.com/category/jsp/)

* [**SERVLETS**](https://j2eereference.com/category/servlets/)

* [**BUILDING TOOLS**](https://j2eereference.com/category/building-tools/)

* [**JQUERY**](https://j2eereference.com/category/jquery/)

* [**SPRING**](https://j2eereference.com/category/spring/)

* [**HIBERNATE**](https://j2eereference.com/category/hibernate/)

* [**MONGO DB**](https://j2eereference.com/category/mongo-db/)

* [**MORE**](https://j2eereference.com/singleton-in-clustered-environment/)

[**J2EE REFERENCE**](https://j2eereference.com/)

**Header Right**

**Main navigation**

* [**HOME**](https://j2eereference.com/)

* [**ABOUT US**](https://j2eereference.com/about/)

* [**JAVA LEARNING CENTERS**](https://j2eereference.com/singleton-in-clustered-environment/)

* [**CONTACT US**](https://j2eereference.com/contact-us/)

**Singleton in Clustered Environment**

MARCH 27, 2014 BY J2EEREFERENCE [1 COMMENT](https://j2eereference.com/singleton-in-clustered-environment/#comments)

**What is Singleton Design Pattern?**

Singleton design pattern comes into picture when we have to control the number of object instances that can be created per Class per JVM. It controls the number of instances that can be created per class. This pattern is especially used. say for example , when you want a Configuration Object that should be shared through out the entire application.  You don’t need to create many objects by reading the same configuration data. So, Singleton pattern helps you to maintain a single Configuration object for the entire application. Thus, it helps a lot in performance wise.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | public class Singleton {       private static Singleton singleton = null;       /\* A private Constructor to prevent access from      \* outside this class      \*/     private Singleton(){ }       /\* Static 'instance' method \*/     public static Singleton getInstance() {        if (singleInstance == null) {           synchronized (Singleton.class) {              if (singleInstance == null) {              singleInstance = new Singleton();              }           }         }       return singleInstance;    }  } |

In the above example, I have used double locking mechanism ( which shouldn’t be used prior to J2SE 5.0 since there are subtle bugs with statement reordering ) in the **getInstance()**method to prevent creation of duplication of **Singleton** instances since **if(singleinstance == null)**is not thread safe. The best way is to create the Singleton objects during deployment of the application either in a servlet configured to load on startup or using contextListeners. We have just coverd the basics of Singleton to startup with

Singleton design pattern can only make restrictions within a JVM in creation of the object. Thus, they work well till the point you have a single JVM. What about the situation where the same application is deployed across several JVM instances in a cluster. When you have a non-ditributed local singleton object in the application, you will end up having singleton object in all the nodes (JVMs)  of the cluster. The architecture of the cluster nodes  also affects the way of managing the singleton instance in it. Before we go into JVM clustering, let us first find out why do we need to go for a cluster in short.

Clustering of application servers (JVMs) gives us these advantages in running the server application. 1) High Availability 2) High Performance 3) High Fault Tolerance. Clustering involves the two important steps as follows

1 ) Load Balancing  
2) Fail-over

**What is Load Balancing?**As the name states, there will be a load balancer which sits between the callers and callees and distribute the load across different servers doing the same function. This is done to achieve high availability and high performance. The example for an load balancer can be F5 load balancer, webserver or just a servlet.

**What is Fail-over?**Sometimes , when the caller makes a lot of requests to a particular server successively and the server is not able to handle them, it fails in between the requests. The Fail over system should detect this failure and redirect the requests to another available server instance.

**How  to manage the Singleton instance in cluster?**

I’d suggest some of the ways which come handy to manage singleton instance in the cluster.

1) **Usage of Java RMI** – Registering the Singleton instance in the rmi registry in a single node and provide stub in the clustered JNDI tree to make it available across all the nodes in the cluster. All the other nodes should be the RMI clients accessing this central node. The main disadvantage is the single point of failure. we can have a fail-over node for central node which also has the Singleton instance which needs to register itself to the cluster JNDI tree when there is no singleton stub registered by the central node.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | import javax.naming.\*;  import java.rmi.\*;    public class RMISingletonWrapper {  private static RMISingletonWrapper instance = null;  private static String SINGLETON\_JNDI\_NAME = "RMISingleton";    public static RMISingletonWrapper getInstance() {  return instance;  }    // All methods in delegate the method call to the actual  // Singleton that lives on the clustered JNDI tree.  public void delegate() {  try {  RMISingleton singleton = getRMISingleton();  singleton.delegate();  } catch (Exception e) {  // Could try and recover  e.printStackTrace();  }  }    // Locate the true Singleton object in the cluster.  private RMISingleton getRMISingleton() {  RMISingleton rmiSingleton = null;  try {  Context jndiContext = new InitialContext();  Object obj = jndiContext.lookup(SINGLETON\_JNDI\_NAME);  rmiSingleton = (RMISingleton)PortableRemoteObject.narrow(  obj,  Class.forName("examples.singleton.rmi.RMISingleton"));  } catch (Exception e) {  // Could try and recover  e.printStackTrace();  }  return rmiSingleton;  }  } |

2) **Distributed Singleton Caches** – As the name states, the singleton instance objects are present in all the nodes of the cluster as data caches. Since a Singleton instance exists on each container any update to the cached data by one singleton will not be synced to other singletons that exist on other containers.This issue can be resolved by the use of the Java Messaging API to send update messages between Containers. In this approach if an update is made to the cache on one Container a message is published to a JMS Topic. Each Container has a listener that subscribes to that topic and updates its Singleton cache based on the messages it receives. This approach is still difficult as you have to make sure that the updates received on each container are handled in a synchronous fashion. JMS messages also take time to process so the caches may spend some time out of sync.

In the following simplistic implementation of a distributed Cache a CacheManager Singleton holds a Map of cached items. Items to be cached are placed in a CachItem object which implements the ICacheItem interface. The CacheManager does not make any attempt to remove old items from the Cache based on any criteria like “Last Accessed Time”. *[SwarmCache](http://swarmcache.sourceforge.net/)* and *JBoss’s [TreeCache](http://docs.jboss.org/jbosscache/1.4.0/TreeCache/en/html_single/)* are well-suited examples of Distributed caches in cluster



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98 | import javax.jms.\*;    public class CacheManager implements MessageListener {  public static CacheManager instance = null;  public static Map cache = new HashMap();    private TopicConnectionFactory topicConnectionFactory;  private TopicConnection topicConnection;  private TopicSession topicSession;  private Topic topic;  private TopicSubscriber topicSubscriber;  private TopicPublisher topicPublisher;    private final static String CONNECTION\_FACTORY\_JNDI\_NAME =  "ConnectionFactory";  private final static String TOPIC\_NAME = "TopicName";    public static void initInstance() {  instance = new CacheManager();  }    public static CacheManager getInstance() {  return instance;  }    public synchronized void addCacheItem(ICacheItem cacheItem) {  CacheMessage cacheMessage = new CacheMessage();  cache.put(cacheItem.getId(), cacheItem.getData());  cacheMessage.setMessageType(CacheMessage.ADD);  cacheMessage.setCacheItem(cacheItem);  sendMessage(cacheMessage);  }    public synchronized void modifyCacheItem(ICacheItem cacheItem) {  CacheMessage cacheMessage = new CacheMessage();  cache.put(cacheItem.getId(), cacheItem.getData());  cacheMessage.setMessageType(CacheMessage.MODIFY);  cacheMessage.setCacheItem(cacheItem);  sendMessage(cacheMessage);  }    public ICacheItem getCacheItem(String key) {  return (ICacheItem)cache.get(key);  }    private CacheManager() {  try {  InitialContext context = new InitialContext();  topicConnectionFactory = (TopicConnectionFactory)  context.lookup(CONNECTION\_FACTORY\_JNDI\_NAME);  topicConnection = topicConnectionFactory.createTopicConnection();  topicSession = topicConnection.createTopicSession(  false, Session.AUTO\_ACKNOWLEDGE);  topic = (Topic) context.lookup(TOPIC\_NAME);  topicSubscriber = topicSession.createSubscriber(topic);  topicSubscriber.setMessageListener(this);  topicPublisher = topicSession.createPublisher(topic);  topicConnection.start();  } catch (Exception e) {  e.printStackTrace();  }  }    public void onMessage(Message message) {  try {  if (message instanceof ObjectMessage) {  ObjectMessage om = (ObjectMessage)message;  CacheMessage cacheMessage = (CacheMessage)om.getObject();  ICacheItem item = cacheMessage.getCacheItem();  interpretCacheMessage(cacheMessage);  }  } catch (JMSException jmse) {  jmse.printStackTrace();  }  }    private void interpretCacheMessage(CacheMessage cacheMessage) {  ICacheItem cacheItem = cacheMessage.getCacheItem();  if (cacheMessage.getMessageType()==CacheMessage.ADD) {  synchronized (this) {  cache.put(cacheItem.getId(), cacheItem.getData());  }  } else if (cacheMessage.getMessageType()==CacheMessage.MODIFY) {  synchronized (this) {  cache.put(cacheItem.getId(), cacheItem.getData());  }  }  }    private void sendMessage(CacheMessage cacheMessage) {  try {  Message message = topicSession.createObjectMessage(cacheMessage);  topicPublisher.publish(message);  } catch (Exception e) {  e.printStackTrace();  }  }  } |

3 ) **Available Vendor Products ( Singleton in Cluster )**

i) *Terracotta / Oracle Coherance* – A concept of in-memory object replication, which provides a singleton view across all JVMs. Please check out their documentations respectively

ii) *JGroups* – This tool is a toolkit for reliable messaging. It can be used to create clusters whose nodes can send messages to each other. It helps us in managing the Singleton instance by making use of the messaging feature across nodes.It allows to form a group and Applications (JVM’s) can participate and JGroups will send messages to everyone in the group so that they can be in sync.

The main features include

* Cluster creation and deletion. Cluster nodes can be spread across LANs or WANs
* Joining and leaving of clusters
* Membership detection and notification about joined/left/crashed cluster nodes
* Detection and removal of crashed nodes
* Sending and receiving of node-to-cluster messages (point-to-multipoint)
* Sending and receiving of node-to-node messages (point-to-point)

i) *JBoss – HASingleton Service*

A clustered singleton service is deployed on multiple nodes in a cluster but runs on only one of the nodes. The node running the singleton service is typically called the master node. When the master fails, another master is selected from the remaining nodes and the service is restarted on the new master. Please check the [link](http://www.onjava.com/pub/a/onjava/2003/08/20/jboss_clustering.html) to learn about JBoss clustering in detail

iv) *WebLogic – Singleton Service*

Weblogic has the concept of Singleton Service – where only instance runs within the cluster and all clients will look up to the same instance. Please check the [link](http://docs.oracle.com/cd/E12840_01/wls/docs103/cluster/service_migration.html#wp1051458) to learn about Weblogic Singleton Service

v) *Websphere – ObjectGrid*

WebSphere supports the concept of singleton across cluster in the WebSphere XD version of the application servers. Please check this [link](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCYQFjAA&url=https%3A%2F%2Fpublib.boulder.ibm.com%2Finfocenter%2Fieduasst%2Fv1r1m0%2Ftopic%2Fcom.ibm.iea.wxd_v6%2Fwxd%2F6.0.1z%2FObjectGrid%2FXD601z_ObjectGrid_Overview%2Fplayer.html%3Fdmuid%3D20061231125612655207&ei=0kYwU8qNEsq4rAf19IHADw&usg=AFQjCNEykrU8sBSy-ADT7IMt6UJFiX_9CA&bvm=bv.62922401,d.bmk) to learn more on Object Grid

**Related Posts**

* [INTERRUPTING A THREAD IN JAVA](https://j2eereference.com/interrupting-thread-java/)
* [WHAT IS THREADLOCAL IN JAVA ?](https://j2eereference.com/what-is-threadlocal-in-java/)
* [ARRAYLIST CUSTOM IMPLEMENTATION](https://j2eereference.com/arraylist-custom-implementation/)
* [DIFFERENCE BETWEEN VOLATILE AND SYNCHRONIZED KEYWORD?](https://j2eereference.com/difference-volatile-synchronized-keyword/)
* [HOW TO WRITE THREAD SAFE CODE?](https://j2eereference.com/write-thread-safe-code/)
* [HOW TO AVOID DEADLOCK IN JAVA ?](https://j2eereference.com/avoid-deadlock-java/)
* [WHAT IS DEADLOCK IN JAVA?](https://j2eereference.com/what-is-deadlock-in-java/)
* [DIFFERENCE BETWEEN SERIALIZABLE AND EXTERNALIZABLE](https://j2eereference.com/difference-serializableand-externalizable/)
* [BINARY SEARCH TREE IMPLEMENTATION IN JAVA](https://j2eereference.com/binary-search-tree-implementation-java/)
* [HOW HASHSET INTERNALLY WORKS ?](https://j2eereference.com/hashset-in-java/)

FILED UNDER: [CORE JAVA](https://j2eereference.com/category/core-java/), [DESIGN PATTERNS](https://j2eereference.com/category/design-patterns/)

**Reader Interactions**

**Comments**

1. **sp says**

[JULY 15, 2016 AT 9:20 AM](https://j2eereference.com/singleton-in-clustered-environment/#comment-1976)

Very informative. After searching and reading so many article, I found this is the one I was looking for.

[Reply](https://j2eereference.com/singleton-in-clustered-environment/?replytocom=1976#respond)

**Leave a Reply**

Top of Form

Bottom of Form

**Primary Sidebar**

Top of Form

Search this website

Bottom of Form

FOLLOW US ONLINE

* [View J2eereference-166104970118637’s profile on Facebook](https://www.facebook.com/J2eereference-166104970118637/)
* [View j2eereference’s profile on Twitter](https://twitter.com/j2eereference/)
* [View j2eereference’s profile on LinkedIn](https://www.linkedin.com/in/j2eereference/)

SUBSCRIBE BY EMAIL

Top of Form



Bottom of Form

RECENT POSTS

* [AngularJS MVC Architecture](https://j2eereference.com/angularjs-mvc-architecture/)
* [Command Design Pattern](https://j2eereference.com/command-design-pattern/)
* [Template Design Pattern](https://j2eereference.com/template-design-pattern/)
* [Callable and Future Interface](https://j2eereference.com/callable-and-future-interface/)
* [Spring dependency injection with Java configuration](https://j2eereference.com/spring-dependency-injection-with-java-configuration/)
* [What is Spring Boot?](https://j2eereference.com/what-is-spring-boot/)
* [Difference between Stack and Heap memory](https://j2eereference.com/difference-stack-heap-memory/)
* [What is ThreadMXBean in java?](https://j2eereference.com/what-is-threadmxbean-in-java/)
* [What is serialVersionUID](https://j2eereference.com/what-is-serialversionuid/)
* [What is exchanger in concurrency?](https://j2eereference.com/exchanger-in-concurrency/)
* [@import annotation in Spring](https://j2eereference.com/import-annotation-in-spring/)
* [Configuring Spring Lifecycle Callbacks with java](https://j2eereference.com/configuring-spring-lifecycle-callbacks-with-java/)
* [Configuring Spring Bean Scope with java](https://j2eereference.com/configuring-spring-bean-scope-with-java/)
* [UncaughtExceptionHandler in java](https://j2eereference.com/uncaughtexceptionhandler-in-java/)
* [Spring configuration using java](https://j2eereference.com/spring-configuration-using-java/)

**Footer**

[Core Java](https://j2eereference.com/category/core-java/)[Design Patterns](https://j2eereference.com/category/design-patterns/)  
[JSP](https://j2eereference.com/category/jsp/)

[Servlets](https://j2eereference.com/category/servlets/)[HTML](https://j2eereference.com/category/html/)  
[Building Tools](https://j2eereference.com/category/building-tools/)

[AJAX](https://j2eereference.com/category/ajax/)[SCJP](https://j2eereference.com/category/scjp/)  
[jQuery](https://j2eereference.com/category/jquery/)

[Testing](https://j2eereference.com/category/testing/)[Spring](https://j2eereference.com/category/spring/)  
[UML](https://j2eereference.com/category/uml/)

[Struts](https://j2eereference.com/category/struts/)[Java Centers](https://j2eereference.com/singleton-in-clustered-environment/)  
[Java Training](https://j2eereference.com/singleton-in-clustered-environment/)

[Home](https://j2eereference.com/)  
[About Us](https://j2eereference.com/about/)  
[Contact Us](https://j2eereference.com/contact-us/)

Copyright © j2eereference.com. All right reserved.   
[https://j2eereference.com/wp-content/uploads/2013/02/twitter.png](https://twitter.com/j2eereference) [https://j2eereference.com/wp-content/uploads/2013/02/linkdin.png](https://www.linkedin.com/in/j2eereference) [https://j2eereference.com/wp-content/uploads/2013/02/facebook.png](https://www.facebook.com/J2eereference-166104970118637/)