Lab 10. Tomcat Upgrade

Technological changes and innovations happen at a very rapid pace. In order to accommodate the current technology requirements and serve the user with the latest technology, an upgrade of the systems is needed. The new version of the systems comes with the latest features and bug fixes, making them more stable and reliable.

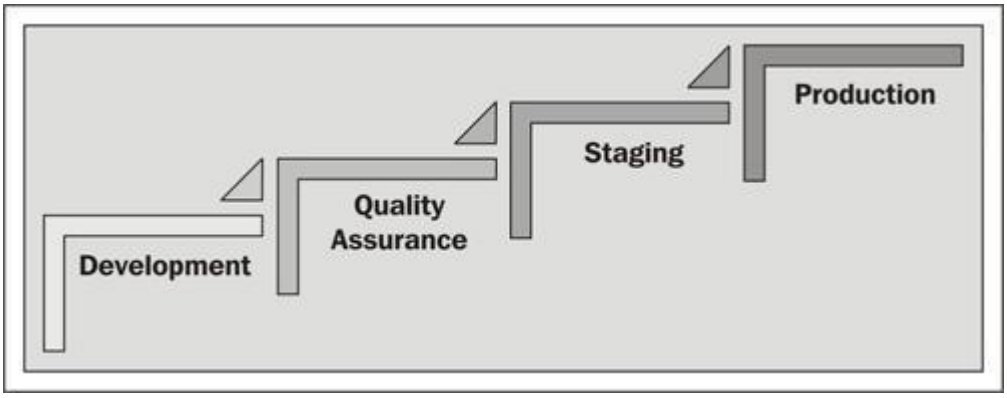
In this lab, we will discuss:

- The life cycle of the upgrade process
- Best practices followed by the IT industry
- How to upgrade Tomcat 6 to Tomcat 8

Every organization follows their process to upgrade the servers, based on the criticality of the system. Normally, evaluation of a product is done by the Technical Architect. Based on the application's criticality, the architect defines the architecture, which needs to be followed to upgrade the application. Upgrades in production can be done only after a successful upgrade on the development server(s).

Different types of environment

There are basically four types of environments for any system in an IT industry, based on their architecture. The following figure shows the different environments created in most industries:



Development environment

It can be defined as the combination of software and hardware, which is required for a team to build the code and deploy it. In simple words, it is a complete package required to build the code and deploy it.

The following points describe why we need a development environment and its advantages:

- Consolidation: For example, by looking at the infrastructure needs of the development environment as a whole, you might find that you only need a single web/application server to deploy the application.
- Estimation of resources: Resources are required to support the development activities and test the development environment, before it is made available on any production infrastructure in support of a business project.

Quality Assurance environment

This environment is mainly used for the integration of the development module and followed by the functional testing undertaken by the Quality Assurance team. If the QA team finds any issues with the functionality of the

application, they notify the developers to resolve the issue.

Staging environment

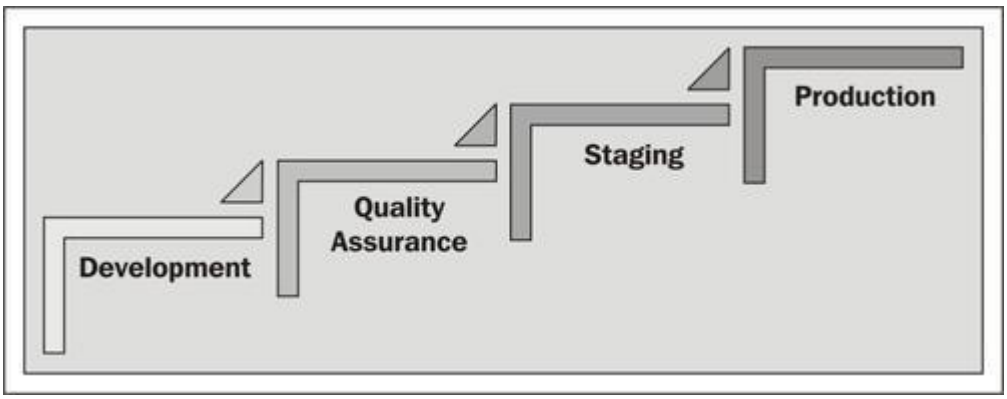
This environment is a replica of the production environment. It is mainly used for performance testing to simulate the real-time issues during the user load.

Production environment

This can be defined as the real environment that is available to the users to perform their operations in the application. For example, banking sites, where you perform your money transactions can be called a production environment.

Different types of environment

There are basically four types of environments for any system in an IT industry, based on their architecture. The following figure shows the different environments created in most industries:



Development environment

It can be defined as the combination of software and hardware, which is required for a team to build the code and deploy it. In simple words, it is a complete package required to build the code and deploy it.

The following points describe why we need a development environment and its advantages:

- Consolidation: For example, by looking at the infrastructure needs of the development environment as a whole, you might find that you only need a single web/application server to deploy the application.
- Estimation of resources: Resources are required to support the development activities and test the development environment, before it is made available on any production infrastructure in support of a business project.

Quality Assurance environment

This environment is mainly used for the integration of the development module and followed by the functional testing undertaken by the Quality Assurance team. If the QA team finds any issues with the functionality of the application, they notify the developers to resolve the issue.

Staging environment

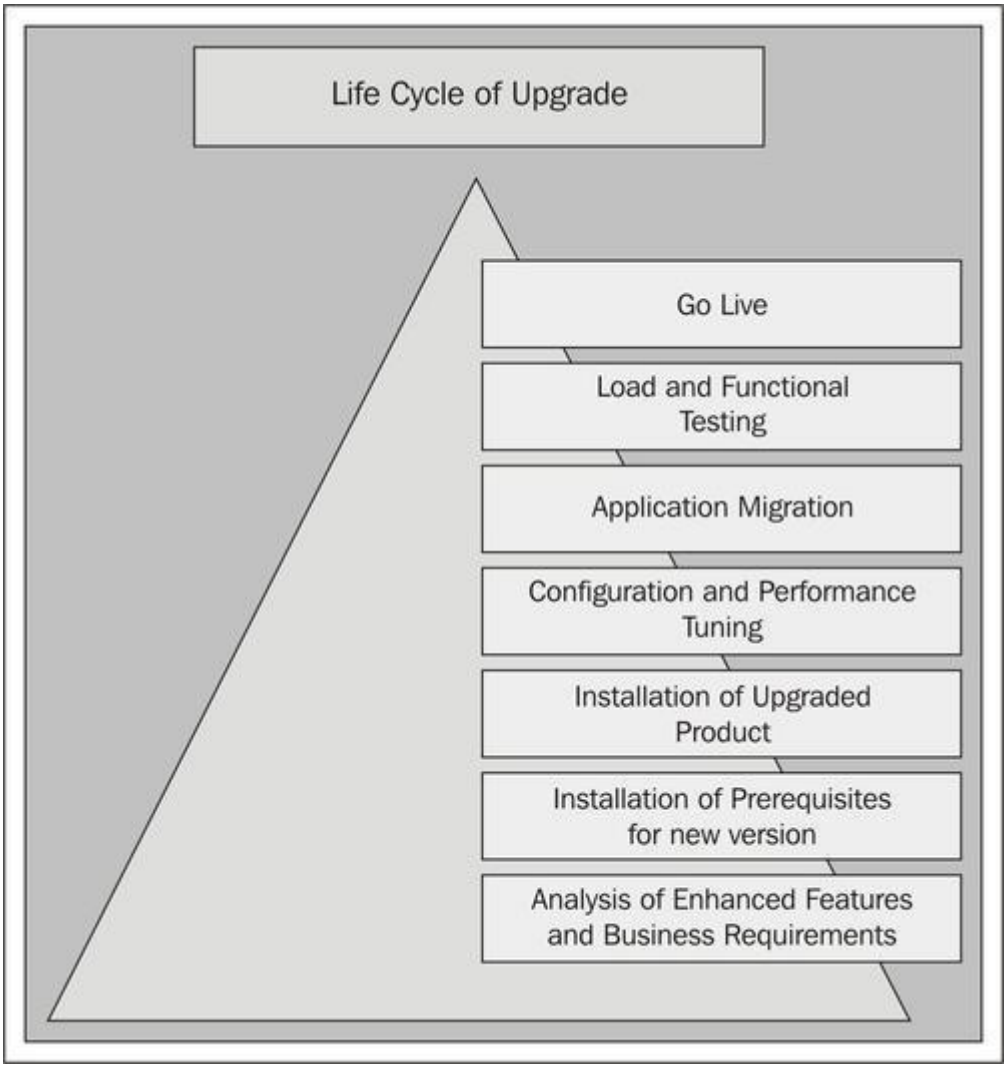
This environment is a replica of the production environment. It is mainly used for performance testing to simulate the real-time issues during the user load.

Production environment

This can be defined as the real environment that is available to the users to perform their operations in the application. For example, banking sites, where you perform your money transactions can be called a production environment.

Life cycle of the upgrade

In this topic, we will discuss the various steps performed during the upgrade. The life cycle consists of the end-to-end processes involved in the upgrade. Normally, upgrades are initiated from the development environment followed by the QA/stage/production environment. The following screenshot shows the basic sequence of steps followed in the upgrade for any system in the IT industry:



1. Analysis of Enhanced Features and Business Requirements: This step plays a very crucial role in the upgrade process. In this process, the standing committee (technical architect, business owner, and functional owner) decides which features are essential for the new version and how they are useful in supporting the business requirement.

2. Installation of Prerequisites for new version: By following the previous process, the infrastructure team makes sure that the entire software is available for the installation and their dependencies are also present.
3. Installation of Upgraded Product: In this process, the installation of a new version of products will be done by the infrastructure engineer.
4. Configuration and Performance Tuning: Once the installation is complete, it's time to do the configuration and performance tuning for the new product.
5. Application Migration: After the configuration is done for Tomcat 8, the upgrade is still incomplete. Now, the most tedious part of the migration starts here, that is, application migration from the current environment to the new system.

```
### Note


Before migrating the application from the current environment to the
new environment, you need to confirm whether the new version is
supported by the application or not. If not, confirm what are the
workarounds provided by the vendors.
```

6. Load and Functional Testing: After the application migration is complete, you have to perform the load and functional testing on the upgraded system, in order to make sure that the application is working as excepted, and also that the new features, which are embedded through the upgrade process, are working according to the business requirement.
7. Go live for the new environment.

Tomcat upgrade from 6 to 7

Until now, we have discussed various theoretical processes of the upgrade. Now it's time to have some fun, which every administrator wants in his/her career.

In this topic, we will discuss the most waited topic of the book, that is the Tomcat upgrade. It's always a wish for the web administrator to perform the upgrade from its previous major version to the new version. It also changes the perception of the administrator from day-to-day maintenance issues to the architecture-level integration. If you are involved in the upgrade activity for the product, then you are the first person in the organization who is working on that product, which gives you better visibility among other people. But before performing the upgrade, let's discuss what new features/updates are offered by Tomcat 7 as compared to Tomcat 6. Following are the features:

- Servlet 3.0
- Asynchronous support
- Dynamic configuration
- Extended Servlet API
- Simpler, faster, more developer-friendly
- Simplified embedding
- Improved logging
- System improvement

- No more memory leaks
- Security improvement

Now we know what are the advantages of using Tomcat 8, let's start the upgrade from Tomcat 6 to Tomcat 8. In order to initiate the upgrade process, the first thing that comes to mind is, on which hardware do we have to perform the upgrade? There are basically two ways to perform the upgrade:

- On the same system where Tomcat 6 is already running: This approach is used when we have high-end servers, which have enough RAM and CPU to handle the load generated by the new version.
- On a separate system: In today's IT infrastructure, this approach is very common due to increasing trends of virtualization. Here, low-end servers are created, which can only handle the load for the new version of Tomcat 8.

Note

The major advantage of this approach is that you can run your current operation in parallel during the upgrade, and there is no impact to the current environment.

We will take the second approach for the upgrade since this is the most common approach used in any IT industry.

Prerequisites for Tomcat 8

By default, Tomcat 6 runs on JDK 1.5 and Tomcat 8 requires JDK 1.6, so the major prerequisite for the Tomcat 8 upgrade is the installation of JDK 1.6. In *Installation of Tomcat 8*, we have discussed the detailed steps of the Java installation. Hence, we will move on to the next installation step.

In case you have to install Tomcat 8 on the same system where Tomcat 6 is running, then you must be thinking how is it possible to set the two different `JAVA_HOME` or `Path?` In that case, you have to install Tomcat with a different user and set the `JAVA_HOME` in the user profile. Also, the same user should have sudo access to run the Tomcat service.

Installation of Tomcat 8 for the upgrade

Once you are done with the installation and configuration of JDK 1.6 on the system, it's time to install Tomcat 8 on the machine. Perform the following steps:

1. Download the latest stable version from the Tomcat official site, http://tomcat.apache.org/download-80.cgi. Once the download is complete, save it in the /opt location.
2. Unzip the Tomcat 8 source, that is apache-tomcat-8.5.61.zip, using the following command:

```
[root@localhost opt]# unzip apache-tomcat-8.5.61.zip
```

3. After you unzip apache-tomcat-8.5.61.zip, it will create a folder named apache-tomcat-8.5.61 in the opt directory.
4. Go to the bin directory of apache-tomcat-8.5.61 using the following command:

```
[root@localhost opt]# cd apache-tomcat-8.5.61/bin/
```

5. Run the following command. If you miss executing the following command, then Tomcat will not start at the time of starting the services. The reason is that, the package comes with read/write permissions but no execute permissions are given to the package. We have to manually update the permissions.

```
[root@localhost bin]# chmod 0755 *.sh
[root@localhost bin]# pwd
/opt/apache-tomcat-8.5.61/bin
```

```
### Note

`chmod 0755 file` is equivalent
to` u=rwx (4+2+1),go=rx (4+1 & 4+1)`. The` 0`
specifies no special modes.
```

6. Start the Tomcat services and validate the Tomcat setup.

```
### Note

If you are doing the installation on the same machine, you have to
change the default connector port for Tomcat 8, or you will receive
the` Port already in use` exception and the services will
not begin. To change the default connector port, you have to
edit` server.xml`.
```

Configuration of Tomcat 8

You must be pondering as to whether we are installing or upgrading Tomcat. After reading this section, you will understand the actual upgrade. We will discuss the various configurations needed to be done with reference to Tomcat 8. It should also perform the same functionality as Tomcat 6, with the integration of new features. Let's discuss the configuration in a sequential manner.

JVM configuration

JVM plays a vital role in the performance and maintenance of the J2EE container. It is very important that we should accommodate the old environment's JVM parameters and add the new enhanced features, which are introduced with the new Java SDK version, while upgrading from Tomcat 6 to Tomcat 8. We have to customize the environment, based on the application's requirement. There are many things we have to keep in mind, while configuring JVM. Some of them are mentioned as follows:

- How many applications are currently running on Tomcat 6
- The number of concurrent users
- The current configuration
- Upgrade to 64 bit from 32 bit

Let's do a comparison on the default memory allocation for Tomcat 6 and Tomcat 8. You will find that there are not many significant changes done with reference to memory allocation. But, in practice, when we upgrade the system, we define more JVM memory as compared to the previous version. The reason is that, it has to support the current application and also the enhanced features of the latest version.

```
using thread-local object allocation.
Mark Sweep Compact GC

Heap Configuration:
   MinHeapFreeRatio = 40
   MaxHeapFreeRatio = 70
   MaxHeapSize      = 268435456 (256.0MB)
   NewSize          = 1048576 (1.0MB)
   MaxNewSize       = 4294901760 (4095.9375MB)
   OldSize          = 4194304 (4.0MB)
   NewRatio         = 2
   SurvivorRatio    = 8
   PermSize         = 12582912 (12.0MB)
   MaxPermSize      = 67108864 (64.0MB)
```

The previous screenshot shows the memory structure for Tomcat 6 and the internal division of its components. The following screenshot shows the memory allocation for Tomcat 8. If you compare the systems, you will find that there are not many differences in the architecture of the memory scheme, but in a real-time production environment, this configuration varies from system to system.

```
Heap Configuration:
   MinHeapFreeRatio = 40
   MaxHeapFreeRatio = 70
   MaxHeapSize      = 134217728 (128.0MB)
   NewSize          = 1048576 (1.0MB)
   MaxNewSize       = 4294901760 (4095.9375MB)
   OldSize          = 4194304 (4.0MB)
   NewRatio         = 2
   SurvivorRatio    = 8
   PermSize         = 16777216 (16.0MB)
   MaxPermSize      = 67108864 (64.0MB)
```

Note

If you are upgrading the system from 32 bit to 64 bit, then the memory allocation will be 30 percent more than the current allocated memory on the system.

While doing the upgrade, you have to enable the same configuration parameters with reference to the old version, also, you have to enable the entire customized configuration to the environment. In order to provide support for the application, it helps in maintaining the performance of the Tomcat server. Let's take an example of the Tomcat 6 configuration and implement it with reference to Tomcat 8, as shown in the following lines of code:

```
JAVA_OPTS="-Xms128m -Xmx512m -XX:MaxPermSize=256m -
Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000"
```

Now, if you want to do the same configuration with Tomcat 8, then you have to increase the configuration based on the availability of the resource and application requirement as shown in the following lines of code:

```
JAVA_OPTS="-Xms1024m Xmx1024m -XX:MaxPermSize=256m
-Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000"
```

Note

The memory leak issue is completely resolved in Tomcat 8. If we increase the JVM memory, then the application will utilize the memory more intensively. Also, this will not cause any issues with the system over a period of time, as compared to the previous version of Tomcat, where we have to recycle Tomcat at regular intervals.

Database connection settings

Database response is very critical in application performance tuning. An insignificant error will also have a big impact on the application. But, if you do the configuration correctly, it will lead to a miracle (we will see the performance results exceeding the benchmark) in the application's performance. Let us compare the datasource configuration between Tomcat 6 and Tomcat 8. The highlighted code in the following lines of code show the difference in configuration for Tomcat 6 and 7:

Tomcat 6:

```
<Resource name="jdbc/myoracle" auth="Container" type="javax.sql.DataSource"
driverClassName="oracle.jdbc.OracleDriver
"url="jdbc:oracle:thin:@192.168.0.1:1521:mysid"username="scott" password="tiger"
maxActive="20" maxIdle="10" maxWait="-1"/>
```

Tomcat 8:

```
<Resource name="jdbc/myoracle" auth="Container" type="javax.sql.DataSource"
driverClassName="oracle.jdbc. OracleDriver"
url="jdbc:oracle:thin:@192.168.2.1:1521:mysid" username="scott" password="tiger"
maxActive="50" maxIdle="10" maxWait="-1"/>
```

Tomcat 6 and 7 are connected to different DB servers, but they are connected to the same database using the same username and password. The reason is that, at the time of the upgrade, it's recommended to use the new instances of the database. The advantage of this approach is, if the database crashes, then the current application will have no impact and you can do the upgrade in parallel. Also, you can try different configuration parameters to improve the performance.

Note

If you are upgrading the environment, it's recommended to use the latest JDBC driver for connecting the database.

Application migration

Application migration is very difficult and a tricky task in the upgrade life cycle. We cannot deploy the application directly. Some of the applications are not compatible with the new version of Tomcat 8 and also with JDK 1.6. In such a situation, we have to build and compile the application in the new version. It is a tricky issue when it comes to application deployment. We cannot directly point the errors without running the application in either TRACE or DEBUG mode. If we analyze the application logs carefully, we will find exceptions such as `Path not found` and `Class not found`.

Note

It's always recommended to check the compatible matrix of the third-party application JAR for the new version of the application.

Following are the steps for application migration:

1. Application recompilation on the JDK supported for Tomcat 8.
2. Upgrade the third-party JAR to the latest version.
3. Deployment of the new application.
4. Testing of the newly deployed application.

Alias configuration

In the current environment, you might have configured many virtual hosts. But you cannot use the same URL for the application, as they point to the old environment. To resolve this issue, you have to configure the dummy URL in the new Tomcat 8 environment for testing purposes. This will help us perform the pre-go live task for the environment.

Following are the steps for Alias configuration:

- **Creation of dummy URL:** By creating the dummy URL, the administrator can check the basic functionality for the application.
- **User testing using dummy URL:** By performing user testing on the new environment, the functional team can verify the application's functionality.
- **Creating CNAME from the current production environment to the new dummy URL:** By creating the **Canonical Name(CNAME)**, we are pointing the old application URL to the new environment.
- **Configuration of the virtual host on Tomcat 8:** By creating the virtual host in Tomcat 8, we let Tomcat know where the application content will be redirected.

The previous processes are followed during the upgrade of Tomcat 6 to Tomcat 8.

Summary

In this lab, we have discussed the various strategies used in the upgrade of Tomcat 6 to Tomcat 8 and the various steps followed during the upgrade process such as the life cycle of the upgrade, the upgrade configuration of Tomcat 8, and the DataSource configuration.

In the next lab, we will discuss the various advanced-level configurations of Tomcat 8 and how they are used in the real-time IT industry such as virtual hosting, multiple instances of Tomcat 8, multiple application deployment, environment configuration, and so on.