

# Optimizing Joget Platform

Here are some steps to optimize your Joget platform:

- Important Articles:
  - [How to Solve Your Joget Enterprise App Performance Problems](#)
  - [Joget Development Server Data Cleaning](#)
  - [Joget Performance Optimization and Scalability Tips](#)
- Environment:
  - Check the JVM heap and permgen memory usage. Is the system out of memory? If so, it's likely there's a memory leak somewhere. You can use VisualVM to check on memory usage and do a heap dump if required. [Tuning Tomcat Performance](#) .
  - Perform a thread dump when the system is slow. This gives a snapshot of what is happening at a specific time so you can see what classes and methods are running. With this, you can identify what is hogging system resources. You can use VisualVM for this too.
  - Check on system resources e.g. CPU, RAM, swap usage. Are they taken up by Java, the DB, or other processes?
  - Narrow down the issue. Is it slow all the time for every request, or just for some specific requests? Is it slow immediately even after a restart, or over some time?
  - Upgrade the server hardware with additional DRAM and increase your heap space allocation.
  - Use the [Performance Analyzer](#) to identify which part in Userview taken a long time to process.
  - Use the [Application Performance Management](#) to analyze.
- Programming & Database:
  - Review your custom plugins/BeanShells to ensure all resources are properly closed and released.
  - Are you running the database on a separate server from the Joget application server? [Reference](#)
  - Do check the database slow logs for a clue to slow queries.
  - Add an index to all your foreign keys in your database tables that are frequently accessed in datalist or JDBC. See <https://dzone.com/articles/how-to-optimize-mysql-queries-for-speed-and-perfor>
  - Upgrade to the latest database version, as it will come with new security fixes and faster performance.
- App Design:
  - Do consider placing the "daily use" & "setup/administration" menus into two separate userviews to make the datalist compact and fast. Add a link in each userview to jump from userview1 to userview2 and vice versa.
  - Remove/hide all menu counts where possible. Menu counts in other datalist in the same userview affect performance as it needs to perform a count in other datalist too to populate the menus.
  - Use [Sync LDAP User Directory Manager](#) to improve performance retrieving user information in your app.
  - Use the free [AJAX Select Box Plugin](#) plugin from Joget Marketplace if you have options fields with large datasets
  - Use [Performance Improvement with Userview Caching](#)
  - Use [Form Options Caching](#)
  - In your development or staging servers, delete unneeded or unused old versions of the app (that are unpublished). Just maintain 2 copies of the app, the current published version, and one older version.
  - Read [Datalist Performance Considerations](#)
- Resource Heavy Plugins - if the following plugins are not used for analysis at all, remove them for better performance:
  - [Form Data Audit Trail](#)
  - [Process Data Collector](#)
- Git:
  - Joget v7.0.6 and higher has option in JAVA\_OPT parameter to disable or turn off the Git. This reduces the processing overhead on Joget, by adding `-Dgit.disabled=true` to your Joget startup script.
- Consider implementing a [Joget Server Clustering](#) using the Joget Large Enterprise Edition (LEE) in a clustered environment for scalability and redundancy.