Performance Improvement with UI Caching

Overview

Joget has been thoroughly profiled and optimized to ensure that there is minimal overhead at the platform level.

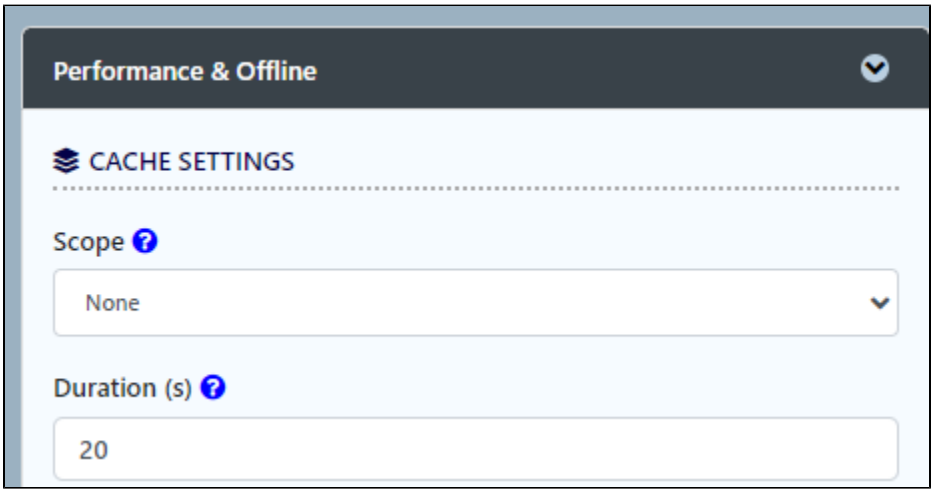
However, in enterprise apps that rely on dynamic data, each page request would require many database queries. Database calls are slow not just in query execution, but especially in network I/O. In most cases, database calls are the main performance bottlenecks that cause lengthy page response times and limit scalability.

Caching is the storing of data in memory to reduce the need for database calls. When applied correctly, caching can greatly improve app performance and scalability.

How UI Caching Works

Caching is now available for all UI pages.

Requiring just a couple of settings, any UI menu and page can be easily cached to eliminate bottlenecks and reduce server-side processing.

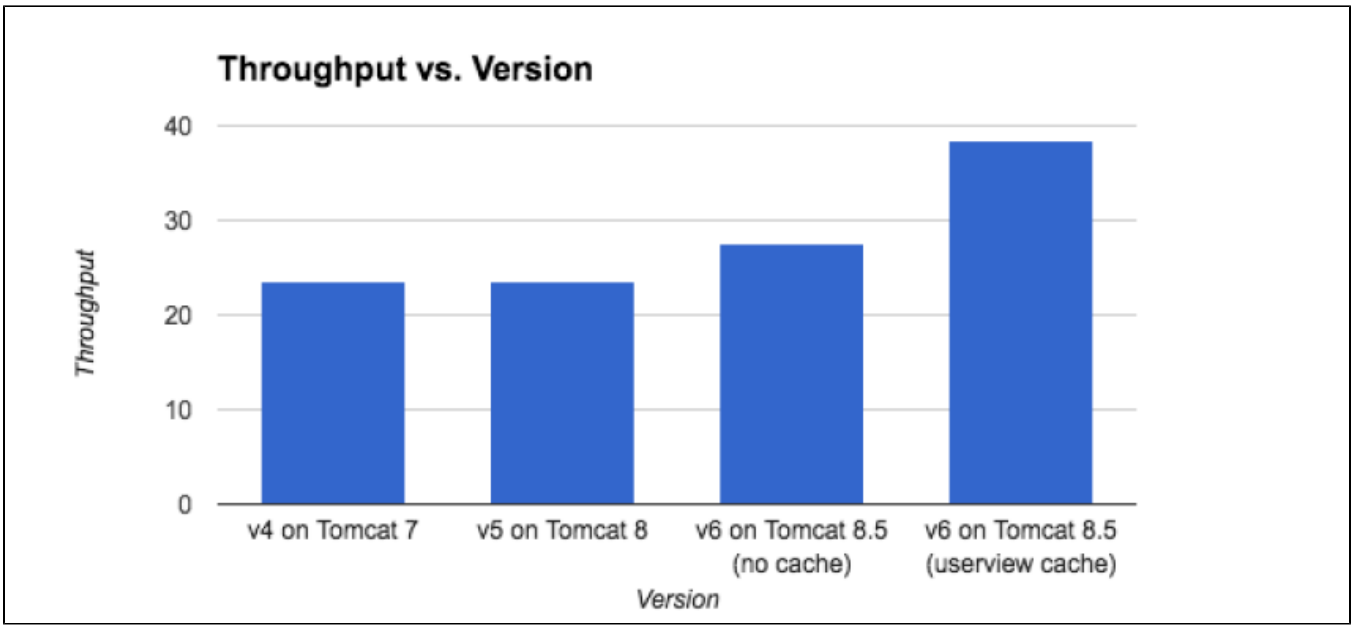


In every UI Menu property page, there is a Performance tab which contains the cache settings.

Scope	**None**: No caching is enabled. This is the default setting. **Application**: The UI page content is cached at the application level. This means that the same content is visible to all users. This scope is suitable for readonly content that is meant to be shared by everyone e.g. public HTML content, general forms, etc. **User**: The UI page content is cached per user. This means that there is a copy of personalized content stored for each user login. This scope is suitable for a user's private content where there is heavy processing required to display, and is acceptable for the content be slightly delayed e.g. custom datalist inbox, personalized forms, etc. Note that content is cached only for GET requests. Whenever a POST request is received, the corresponding cache entry is cleared to reflect the latest update.
Duration (s)	The duration in seconds to cache the content.

It is important to note that **NOT all pages can or should be cached**, depending on the actual processing time and data or privacy requirement for each page.

However when configured appropriately, app performance can improve significantly.
In a mixed use case test app (a user logs in, views a datalist inbox, starts a process, and logs out), the throughput (average requests per second) for a cached version in a load test shows an improvement of almost 30% to 60% compared to previous uncached versions.



More Resources

The following are additional resources for performance optimization:

- Performance Analyzer
- Database Connection Monitoring and Leak Detection
- Deployment Best Practices