Session Management Features

Apache Shiro Session Management Features

Sessions are buckets of data that your users carry with them for a period of time when using your application. Sessions have traditionally been exclusive to web or EJB environments. No more! Shiro enables sessions for any application environment. Further, Shiro offers a host of other great features to help you manage sessions.

Features

- **POJO/J2SE based (IoC friendly)** - Everything in Shiro (including all aspects of Sessions and Session Management) is interface-based and implemented with POJOs. This allows you to easily configure all session components with any JavaBeans-compatible configuration format, like JSON, YAML, Spring XML or similar mechanisms. You can also easily extend Shiro's components or write your own as necessary to fully customize session management functionality.

- **Session Storage** - Because Shiro's Session objects are POJO-based, session data can be easily stored in any number of data sources. This allows you to customize exactly where your application's session data resides - for example, the file system, an enterprise cache, a relational database, or proprietary data store.

- **Easy and Powerful Clustering** - Shiro's sessions can be easily clustered using any of the readily-available networked caching products, like Ehcache, Coherence, GigaSpaces, et. al. This means you can configure session clustering for Shiro once and only once, and no matter what web container you deploy to, your sessions will be clustered the same way. No need for container-specific configuration!

- **Heterogeneous Client Access** - Unlike EJB or Web sessions, Shiro sessions can be 'shared' across various client technologies. For example, a desktop application could 'see' and 'share' the same physical session used by the same user in a server-side web application. We are unaware of any framework other than Shiro that can support this.

- **Event listeners** - Event listeners allow you to listen to lifecycle events during a session's lifetime. You can listen for these events and react to them for custom application behavior - for example, updating a user record when their session expires.

- **Host address retention** – Shiro Sessions retain the IP address of the host from where the session was initiated. This allows you to determine where the user is located and react accordingly (mostly useful in intranet environments where IP association is deterministic).

- **Inactivity/expiration support** – Sessions expire due to inactivity as expected, but they can be prolonged via a `touch()` method to keep them 'alive' if desired. This is useful in Rich Internet Application (RIA) environments where the user might be using a desktop application, but may not be regularly communicating with the server, but the server session should not expire.

- **Transparent web use** - Shiro's web support implements the `HttpSession` interface and all of it's associated APIs. This means you can use Shiro sessions in existing web applications and you don't need to change any of your existing web code.

- **Can be used for SSO** - Because Shiro session's are POJO based, they are easily stored in any data source, and they can be 'shared' across applications if needed. This can be used to provide a simple sign-on experience since the shared session can retain authentication state.

Get Started in 10 Minutes with Shiro

Try out Shiro for yourself with our 10 Minute Tutorial. And if you have any questions about Shiro, please check out our community forum or user mailing list for answers from the community.