1. **QUESTION**

[RESPECTABLE]

Proof of Concept to demonstrate sharing session between different applications. Expectation is to have scalable session management where there is addition of new applications.

**Source Code GIT Repository:**

**Web Browser Link:**

[**https://github.com/Ajay-Kumar-Aspiring-Minds-Round-2/SharingSession**](https://github.com/Ajay-Kumar-Aspiring-Minds-Round-2/SharingSession)

**GIT Clone Link:**

[**https://github.com/Ajay-Kumar-Aspiring-Minds-Round-2/SharingSession.git**](https://github.com/Ajay-Kumar-Aspiring-Minds-Round-2/SharingSession.git)

**Technology Stack:**

****

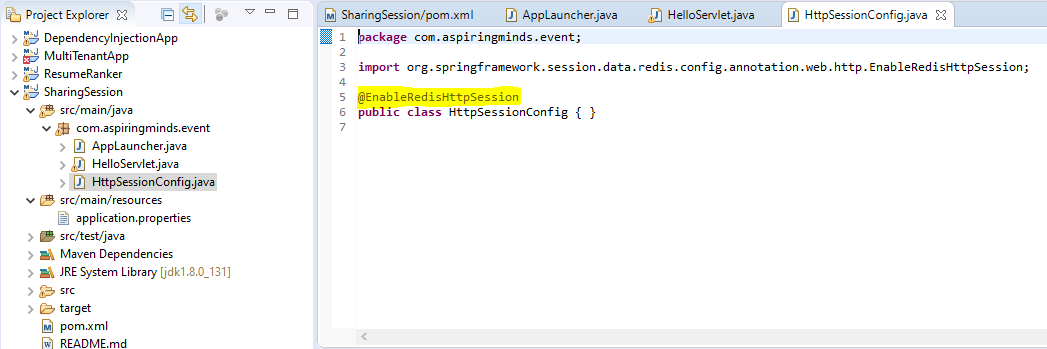


**Key Points & Configurations:**

* Spring Session provides an API and implementations for managing a user’s session information.
* Spring Session’s below modules are used in the sample Sharing Session App.
  + **Spring Session Core** - provides core Spring Session functionalities and APIs
  + **Spring Session Data Redis** - provides SessionRepository and ReactiveSessionRepository implementation backed by Redis and configuration support.
* Spring Session uses **Redis** to back a web application’s **HttpSession** when using Spring Boot.
* The user’s information is stored in **Redis** rather than Tomcat’s **HttpSession** implementation.

**Spring Configuration**

* We can create a Spring configuration which is responsible for creating a Servlet Filter that replaces the HttpSession implementation with an implementation backed by Spring Session.
* Add the below Spring Configuration which is shown in the screenshot:

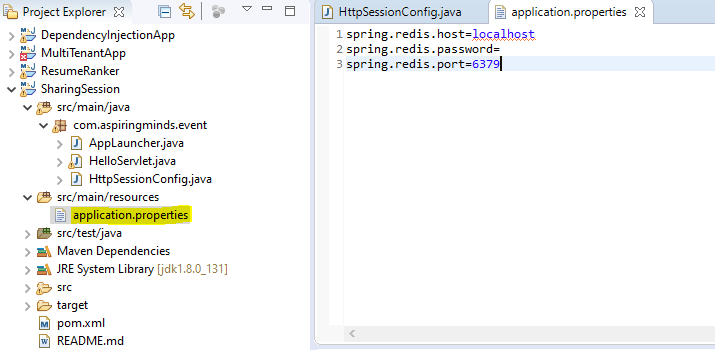


* The **@EnableRedisHttpSession** annotation creates a Spring Bean with the name of **springSessionRepositoryFilter** that implements Filter. The filter is what is in charge of replacing the **HttpSession** implementation to be backed by Spring Session. In this instance **Spring Session is backed by Redis**.

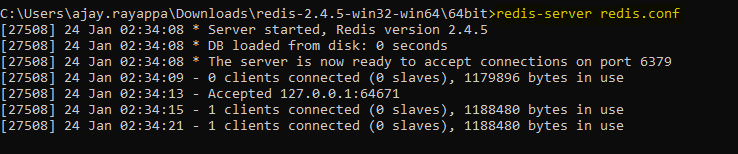
**Configuring the Redis Connection**

* Spring Boot automatically creates a **RedisConnectionFactory** that connects **Spring Session** to a **Redis Server** on **localhost** on **port 6379** (default port).

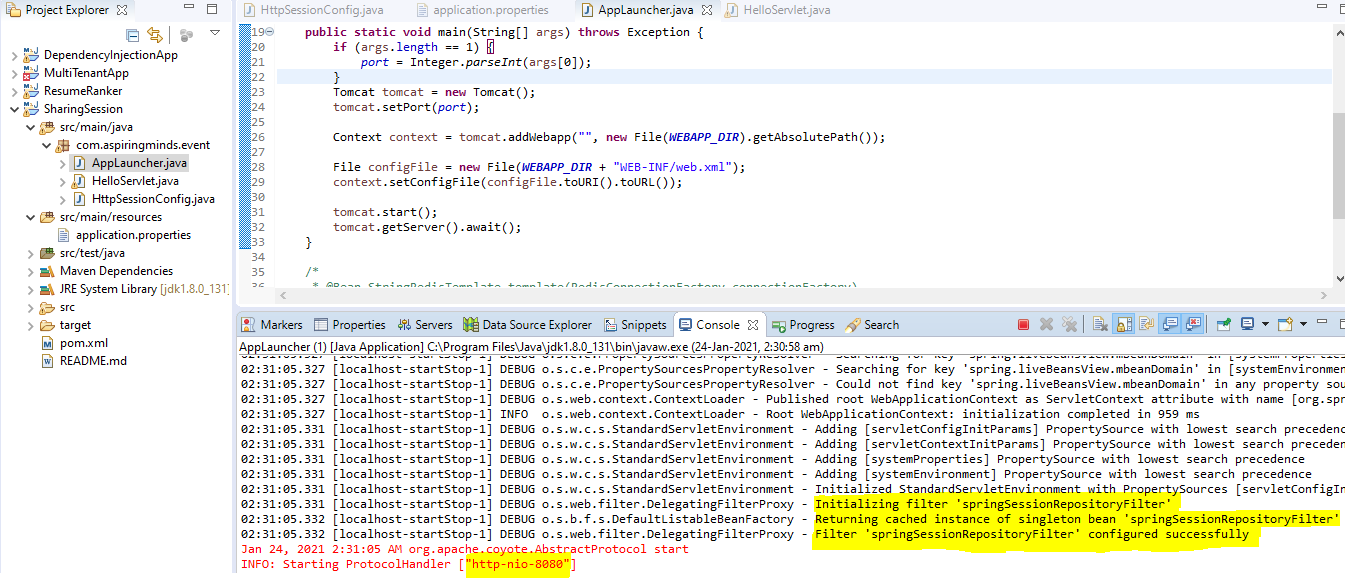
We can add the below code snippet under **application.properties** file.



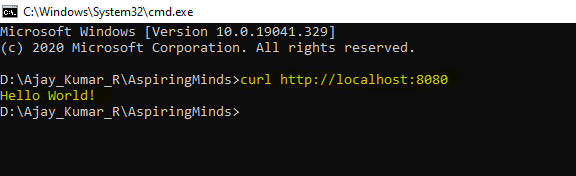
* Spring Session uses **Redis** to back a web application’s **HttpSession** when using Spring Boot.
* The user’s information is stored in Redis rather than Tomcat’s **HttpSession** implementation.
* Instead of using **Tomcat’s HttpSession**, we are actually persisting the values in Redis. Spring Session replaces the HttpSession with an implementation that is backed by Redis.
* When a new HttpSession is created, Spring Session creates a cookie named SESSION in your browser that contains the id of your session.
* App consists of an Embedded Tomcat that has a single **HelloServlet** servlet. When issuing a **GET** request, the servlet will respond with either the default **Hello World!** or if the name session attribute has been set with Hello **[name]**. The name session attribute can be changed by issuing a POST request with a name parameter.
* Start Redis by executing the **redis-server** command - **redis-server redis.conf**



* **Started the first Tomcat instance on port 8080**

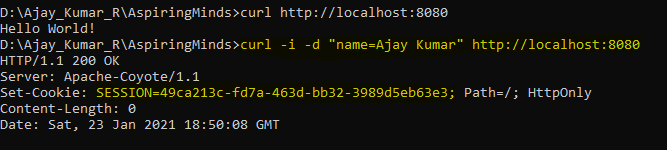


* Try issuing a **GET** request, for the first time servlet should respond with the default value from the HelloServlet - **Hello World!**

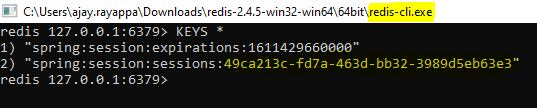


* **Now change the Session state.**

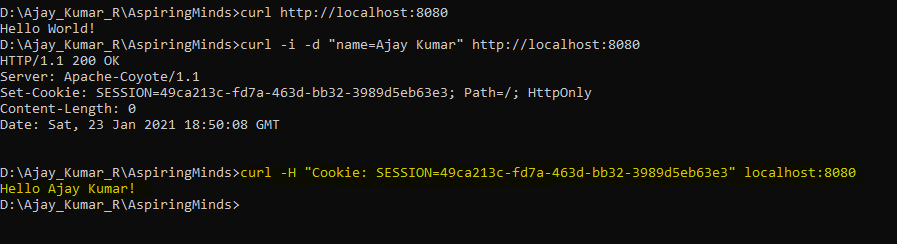
Change the value of the session attribute name by issuing a **POST** request with the **name** set as a parameter: "name=Ajay Kumar".



* Verify whether the session key values are added to the Redis Server instead of HttpSession. Session key values are persisted to redis server successfully.

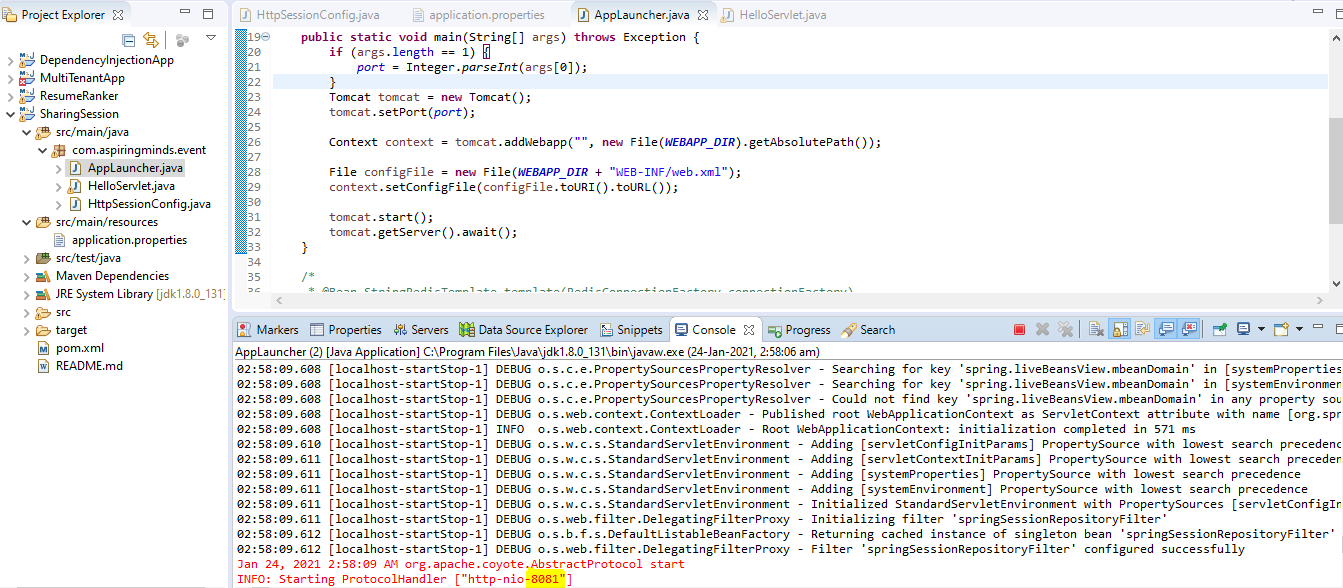


* As session state is changed or modified, verify that the session attribute has been changed by issuing a GET request. Yes, instead of default Hello World! previously persisted session value is returned in the session attribute **[name]**.



* **Session Replication to a Second Server**

Start the second Tomcat instance on port 8081



* Verify that the session attribute is available by issuing a GET request to the other Tomcat (8081) instance of the application.

