

7.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>

GIT Clone Link:

<https://github.com/Ajay-Kumar-Aspiring-Minds-Round-2/SharingSession.git>

Technology Stack:

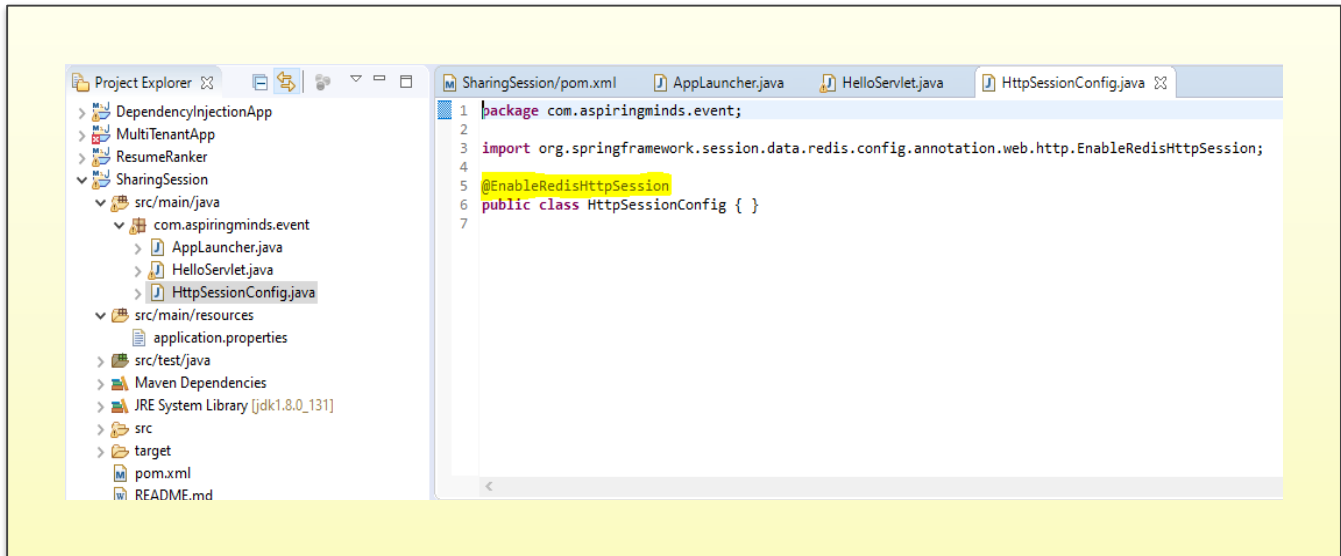


Key Points & Configurations:

- 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:

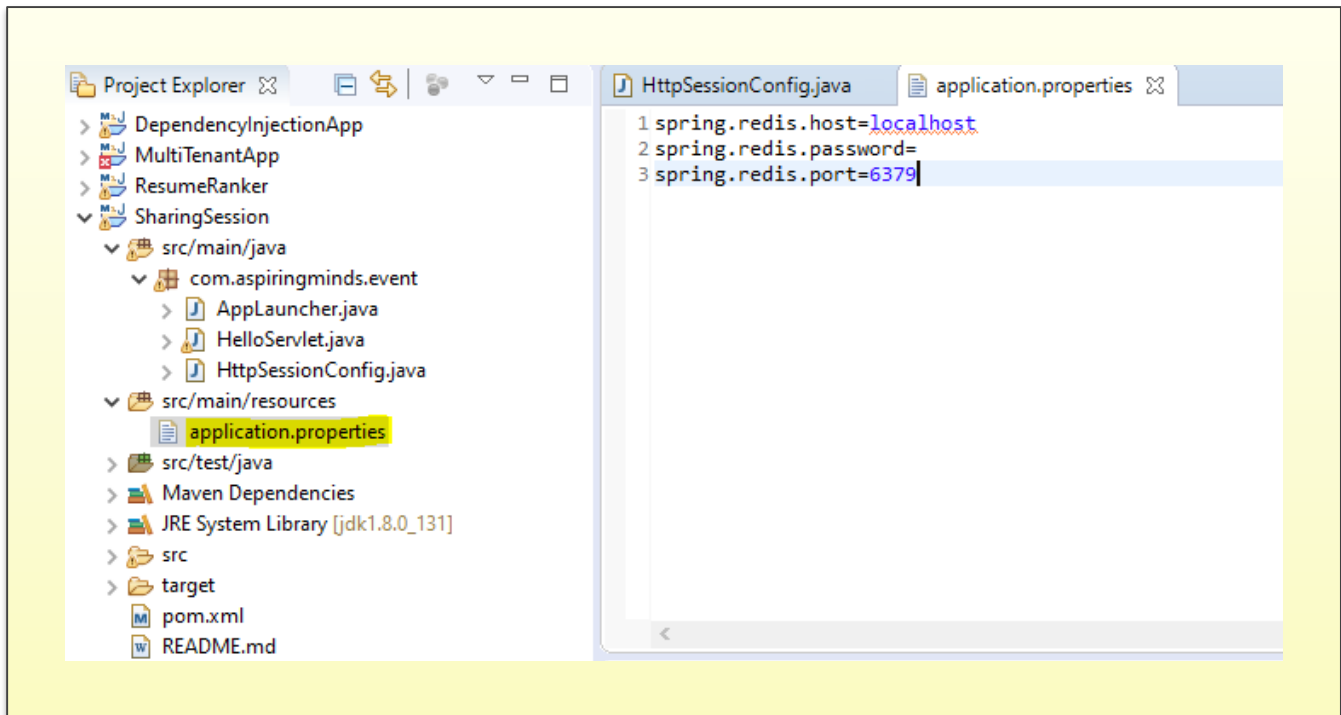


- The **@EnableRedisHttpSession** annotation creates a Spring Bean with the name of **springSessionRepositoryFilter** that implements Filter. The filter is what is in charge of replacing the **HttpSession** implementation to be backed by Spring Session. In this instance **Spring Session is backed by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

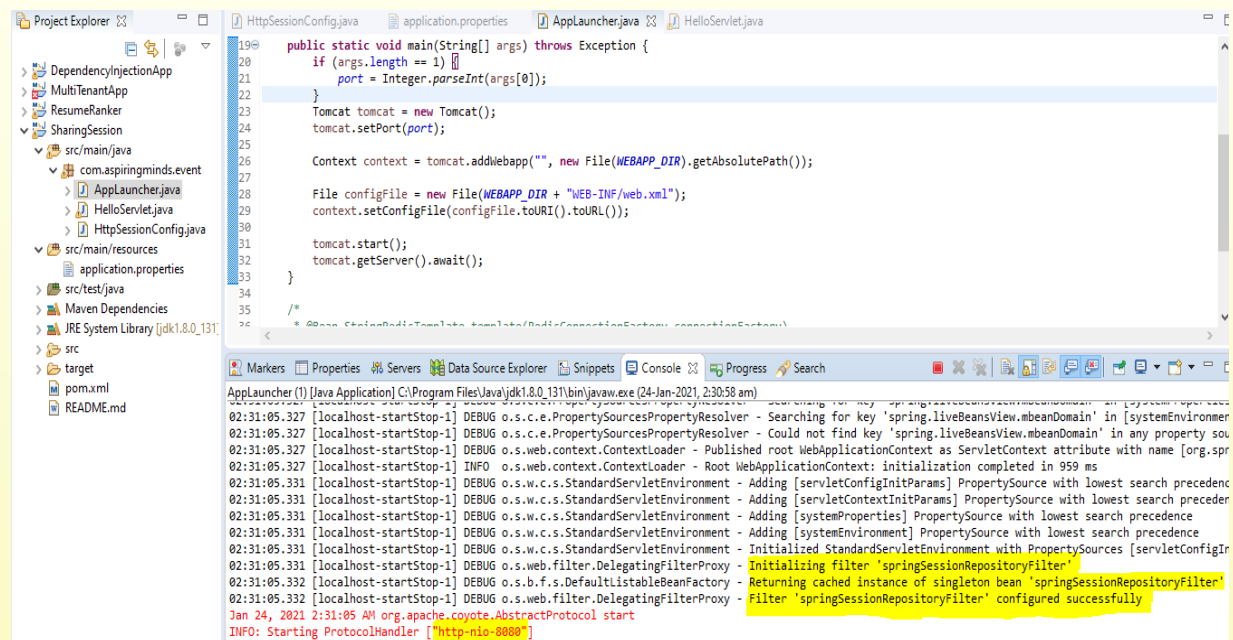


- Spring Session uses **Redis** to back a web application's **HttpSession** when using Spring Boot.
- The user's information is stored in Redis rather than Tomcat's **HttpSession** implementation.
- Instead of using **Tomcat's HttpSession**, we are actually persisting the values in Redis. Spring Session 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**

```
C:\Users\ajay.rayappa\Downloads\redis-2.4.5-win32-win64\64bit>redis-server redis.conf
[27508] 24 Jan 02:34:08 * Server started, Redis version 2.4.5
[27508] 24 Jan 02:34:08 * DB loaded from disk: 0 seconds
[27508] 24 Jan 02:34:08 * The server is now ready to accept connections on port 6379
[27508] 24 Jan 02:34:09 - 0 clients connected (0 slaves), 1179896 bytes in use
[27508] 24 Jan 02:34:13 - Accepted 127.0.0.1:64671
[27508] 24 Jan 02:34:15 - 1 clients connected (0 slaves), 1188480 bytes in use
[27508] 24 Jan 02:34:21 - 1 clients connected (0 slaves), 1188480 bytes in use
```

- Started the first Tomcat instance on port 8080



The screenshot shows an IDE with the `AppLauncher.java` file open. The code defines a `main` method that starts a Tomcat server on a specified port (default 8080) and initializes a Spring web application context. The console output shows the startup logs, including the initialization of the `WebApplicationContext` and the configuration of the `springSessionRepositoryFilter`.

```
public static void main(String[] args) throws Exception {
    if (args.length == 1) {
        port = Integer.parseInt(args[0]);
    }
    Tomcat tomcat = new Tomcat();
    tomcat.setPort(port);

    Context context = tomcat.addWebapp("", new File(WEBAAPP_DIR).getAbsolutePath());

    File configFile = new File(WEBAAPP_DIR + "WEB-INF/web.xml");
    context.setConfigFile(configFile.toURI().toURL());

    tomcat.start();
    tomcat.getServer().await();
}
```

Console Output:

```
AppLauncher (1) [Java Application] C:\Program Files\Java\jdk1.8.0_131\bin\javaw.exe (24-Jan-2021, 2:30:58 am)
02:31:05.327 [localhost-startStop-1] DEBUG o.s.c.e.PropertySourcesPropertyResolver - Searching for key 'spring.liveBeansView.mbeanDomain' in [systemEnvironment]
02:31:05.327 [localhost-startStop-1] DEBUG o.s.c.e.PropertySourcesPropertyResolver - Could not find key 'spring.liveBeansView.mbeanDomain' in any property source
02:31:05.327 [localhost-startStop-1] INFO o.s.web.context.ContextLoader - Published root WebApplicationContext as ServletContext attribute with name [org.springframework.web.context.request.context]
02:31:05.327 [localhost-startStop-1] INFO o.s.web.context.ContextLoader - Root WebApplicationContext: initialization completed in 959 ms
02:31:05.331 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [servletConfigInitParams] PropertySource with lowest search precedence
02:31:05.331 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [servletContextInitParams] PropertySource with lowest search precedence
02:31:05.331 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [systemProperties] PropertySource with lowest search precedence
02:31:05.331 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [systemEnvironment] PropertySource with lowest search precedence
02:31:05.331 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Initialized StandardServletEnvironment with PropertySources [servletConfigInitParams, servletContextInitParams, systemProperties, systemEnvironment]
02:31:05.331 [localhost-startStop-1] DEBUG o.s.web.filter.DelegatingFilterProxy - Initializing filter 'springSessionRepositoryFilter'
02:31:05.332 [localhost-startStop-1] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Returning cached instance of singleton bean 'springSessionRepositoryFilter'
02:31:05.332 [localhost-startStop-1] DEBUG o.s.web.filter.DelegatingFilterProxy - Filter 'springSessionRepositoryFilter' configured successfully
Jan 24, 2021 2:31:05 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8080"]
```

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

```
D:\Ajay_Kumar_R\AspiringMinds>curl http://localhost:8080
Hello World!
D:\Ajay_Kumar_R\AspiringMinds>
```

- **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".

```
D:\Ajay_Kumar_R\AspiringMinds>curl http://localhost:8080
Hello World!
D:\Ajay_Kumar_R\AspiringMinds>curl -i -d "name=Ajay Kumar" http://localhost:8080
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: SESSION=49ca213c-fd7a-463d-bb32-3989d5eb63e3; Path=/; HttpOnly
Content-Length: 0
Date: Sat, 23 Jan 2021 18:50:08 GMT
```

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

```
C:\Users\ajay.rayappa\Downloads\redis-2.4.5-win32-win64\64bit\redis-cli.exe
redis 127.0.0.1:6379> KEYS *
1) "spring:session:expirations:1611429660000"
2) "spring:session:sessions:49ca213c-fd7a-463d-bb32-3989d5eb63e3"
redis 127.0.0.1:6379>
```

- 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].

```
D:\Ajay_Kumar_R\AspiringMinds>curl http://localhost:8080
Hello World!
D:\Ajay_Kumar_R\AspiringMinds>curl -i -d "name=Ajay Kumar" http://localhost:8080
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: SESSION=49ca213c-fd7a-463d-bb32-3989d5eb63e3; Path=/; HttpOnly
Content-Length: 0
Date: Sat, 23 Jan 2021 18:50:08 GMT

D:\Ajay_Kumar_R\AspiringMinds>curl -H "Cookie: SESSION=49ca213c-fd7a-463d-bb32-3989d5eb63e3" localhost:8080
Hello Ajay Kumar!
D:\Ajay_Kumar_R\AspiringMinds>
```

- Session Replication to a Second Server**
Start the second Tomcat instance on port 8081

```
public static void main(String[] args) throws Exception {
    if (args.length == 1) {
        port = Integer.parseInt(args[0]);
    }
    Tomcat tomcat = new Tomcat();
    tomcat.setPort(port);

    Context context = tomcat.addWebapp("", new File(WEBAPP_DIR).getAbsolutePath());

    File configFile = new File(WEBAPP_DIR + "WEB-INF/web.xml");
    context.setConfigFile(configFile.toURI().toURL());

    tomcat.start();
    tomcat.getServer().await();
}
```

```
AppLauncher (2) [Java Application] C:\Program Files\Java\jdk1.8.0_131\bin\javaw.exe (24-Jan-2021, 2:58:06 am)
02:58:09.608 [localhost-startStop-1] DEBUG o.s.c.e.PropertySourcesPropertyResolver - Searching for key 'spring.liveBeansView.mbeanDomain' in [systemProperties]
02:58:09.608 [localhost-startStop-1] DEBUG o.s.c.e.PropertySourcesPropertyResolver - Searching for key 'spring.liveBeansView.mbeanDomain' in [systemEnvironment]
02:58:09.608 [localhost-startStop-1] DEBUG o.s.c.e.PropertySourcesPropertyResolver - Could not find key 'spring.liveBeansView.mbeanDomain' in any property source
02:58:09.608 [localhost-startStop-1] INFO o.s.w.c.s.StandardServletEnvironment - Published root WebApplicationContext as ServletContext attribute with name [org.springframework.web.context.support.AnnotationConfigWebApplicationContext]
02:58:09.610 [localhost-startStop-1] INFO o.s.w.c.s.StandardServletEnvironment - Root WebApplicationContext: initialization completed in 571 ms
02:58:09.611 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [servletConfigInitParams] PropertySource with lowest search precedence
02:58:09.611 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [servletContextInitParams] PropertySource with lowest search precedence
02:58:09.611 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [systemProperties] PropertySource with lowest search precedence
02:58:09.611 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Adding [systemEnvironment] PropertySource with lowest search precedence
02:58:09.611 [localhost-startStop-1] DEBUG o.s.w.c.s.StandardServletEnvironment - Initializing StandardServletEnvironment with PropertySources [servletConfigInitParams, servletContextInitParams, systemProperties, systemEnvironment]
02:58:09.612 [localhost-startStop-1] DEBUG o.s.web.filter.DelegatingFilterProxy - Initializing filter 'springSessionRepositoryFilter'
02:58:09.612 [localhost-startStop-1] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Returning cached instance of singleton bean 'springSessionRepositoryFilter'
02:58:09.612 [localhost-startStop-1] DEBUG o.s.web.filter.DelegatingFilterProxy - Filter 'springSessionRepositoryFilter' configured successfully
Jan 24, 2021 2:58:09 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8081"]
```

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

```
D:\Ajay_Kumar_R\AspiringMinds>curl http://localhost:8080
Hello World!
D:\Ajay_Kumar_R\AspiringMinds>curl -i -d "name=Ajay Kumar" http://localhost:8080
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: SESSION=49ca213c-fd7a-463d-bb32-3989d5eb63e3; Path=/; HttpOnly
Content-Length: 0
Date: Sat, 23 Jan 2021 18:50:08 GMT

D:\Ajay_Kumar_R\AspiringMinds>curl -H "Cookie: SESSION=49ca213c-fd7a-463d-bb32-3989d5eb63e3" localhost:8080
Hello Ajay Kumar!
D:\Ajay_Kumar_R\AspiringMinds>curl -H "Cookie: SESSION=49ca213c-fd7a-463d-bb32-3989d5eb63e3" localhost:8081
Hello Ajay Kumar!
D:\Ajay_Kumar_R\AspiringMinds>
```