Precisamos agora realizar a configuração para armazenarmos a sessão do usuário no Redis, para isso precisaremos adicionar algumas dependências no projeto, vá até o arquivo **pom.xml** e logo antes de fechar a tag **dependencies**coloque as dependências abaixo:

```
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-data-redis</artifactId>
    <version>1.0.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.6.2</version>
</dependency>
```

COPIAR CÓDIGO

Feito isso, vá até o pacote **configuration** e crie uma classe chamada **RedisConfiguration** , o primeiro ponto que devemos estabelecer é justamente a comunicação com o Redis, para isso, vamos precisar de um objeto do tipo **JedisConnectionFactory**

```
@Bean
public JedisConnectionFactory jedisConnectionFactory() {
    JedisPoolConfig pool = new JedisPoolConfig();
    JedisConnectionFactory factory = new JedisConnectionFactory(pool)
    factory.setHostName("localhost");
    factory.setPort(6379);
    return factory;
}
```

COPIAR CÓDIGO

Na sequência devemos informar como será feito o acesso dos dados que serão armazenados no Redis, vamos precisar para isso um objeto do tipo **RedisTemplate**:

```
@Bean
```

```
@Bean
public RedisTemplate<String, Object> redisTemplate() {
    RedisTemplate<String, Object> template = new RedisTemplate<String
    template.setConnectionFactory(jedisConnectionFactory());
    template.setKeySerializer(new StringRedisSerializer());
    template.setValueSerializer(new GenericToStringSerializer<Object>
    return template;
}
```

COPIAR CÓDIGO

Por fim, como iremos levar nossa aplicação para a Amazon e teremos um ambiente seguro, devemos especificar que nossa aplicação não deverá tomar nenhum tipo de ação caso as informações armazenadas no Redis sejam manipuladas.

```
@Bean
public ConfigureRedisAction configureRedisAction() {
    return ConfigureRedisAction.NO_OP;
}
```

COPIAR CÓDIGO

Para finalizar, devemos especificar que a sessão deverá ser armazenada no Redis, anote a classe com `@EnableRedisHttpSession` e faça a classe RedisConfiguration herdar de `AbstractHttpSessionApplicationInitializer`. A última etapa será dizer que essa classe RedisConfiguration irá expor os Beans que configuramos através da anotação `@Configuration`:

```
@EnableRedisHttpSession
@Configuration
public class RedisConfiguration extends AbstractHttpSessionApplicatio
```

COPIAR CÓDIGO

Feito isso, vamos inicializar o Redis:

Windows:

Vá até a pasta C:\Arquivos de programas\Redis e clique em **redis-server.exe**

Linux:

Vá até o terminal e coloque:

```
sudo systemctl start redis
```

COPIAR CÓDIGO

MAC:

Vá até o terminal e coloque:

```
brew services start redis
```

COPIAR CÓDIGO

Inicialize os dois Tomcat, acesse o primeiro Tomcat, configurado na porta 8080 e adicione um livro no carrinho de compras. Depois vá para segundo Tomcat, configurado na port 8081 e veja se o livro está presente no carrinho de compras. Posteriormete confirme se a sessão do usuário foi armazenada no Redis[1]

[1] Se estiver no Windows vá até a pasta C:\Arquivos de programas\Redis e clique em **redis-cli** e coloque **keys ***, se estiver no Linux ou no Mac digite no terminal **redis-cli** e coloque **keys ***