Without Lambda

1. **Lambda-Based Configuration:**
    - The first block of code (commented out) demonstrates configuring the HttpSecurity object using lambdas.
    - The lambda expressions simplify the code by allowing concise, in-line customization.
    - Code like http.csrf(csrfCustomizer -> csrfCustomizer.disable()); and http.authorizeHttpRequests(request -> request.anyRequest().authenticated()); show how easy it is to express configurations using lambda expressions.
2. **Without Lambda Configuration:**
    - The second block of code (also commented out) shows an equivalent configuration without using lambdas.
    - A Customizer implementation is explicitly created and passed to methods like http.csrf() and http.authorizeHttpRequests().
    - This approach is more verbose and may be preferable for those who need clearer readability or are working in environments without lambda support.
3. **Functional Configuration (Lambda-based):**
    - The last block of code is an active configuration using lambda expressions for better readability and conciseness.
    - Each configuration aspect (e.g., disabling CSRF, configuring HTTP basic authentication, and session management) is compactly expressed.

SecurityConfig.java

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

//	@Bean
//	public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
//		http.csrf(customizer->customizer.disable());
//		http.authorizeHttpRequests(request->request.anyRequest().authenticated());
////		http.formLogin(Customizer.withDefaults());
//		http.httpBasic(Customizer.withDefaults());
//		http.sessionManagement(session->session.sessionCreationPolicy(SessionCreationPolicy.STATELESS));
//
//		return http.build();
//	}



//without lambda
//	@Bean
//	public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
//
//		Customizer<CsrfConfigurer<HttpSecurity>> custCsrf= new Customizer<CsrfConfigurer<HttpSecurity>>() {
//			@Override
//			public void customize(CsrfConfigurer<HttpSecurity> configure) {
//			 configure.disable();
//			}
//		};
//		http.csrf(custCsrf);
//
//
	Customizer<AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizationManagerRequestMatcherRegistry> custHttp= new Customizer<AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizationManagerRequestMatcherRegistry>() {
//			@Override
//			public void customize(AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizationManagerRequestMatcherRegistry request) {
//				request.anyRequest().authenticated();
//			}
//		};
//		http.authorizeHttpRequests(custHttp);
//
//		return http.build();
//	}
```

```
        @Bean
        public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
                http.csrf(customizer->customizer.disable())
                  .authorizeHttpRequests(request->request.anyRequest().authenticated())
                  .httpBasic(Customizer.withDefaults())
                  .sessionManagement(session- >
                          session.sessionCreationPolicy(SessionCreationPolicy.STATELESS));

            return http.build();
        }

}
```