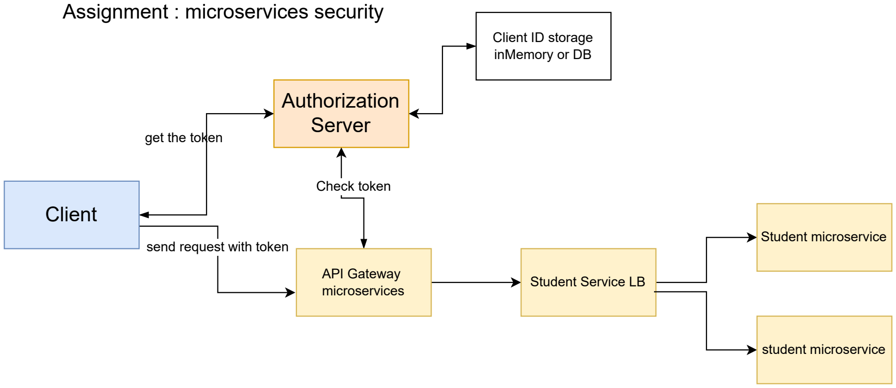
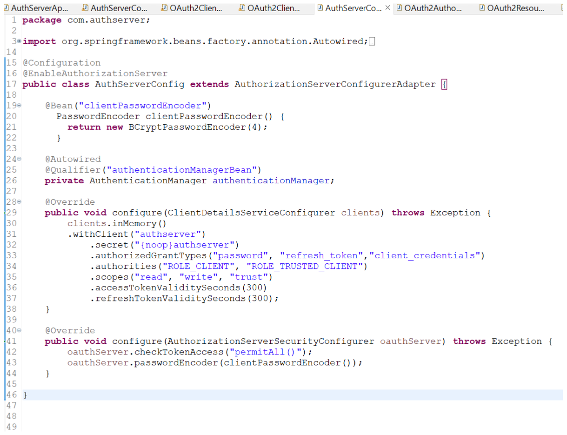
Assignment : microservices security



Authorization Server:

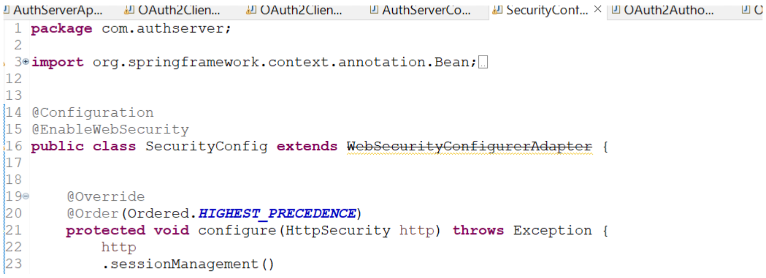
AuthorizationServerConfig



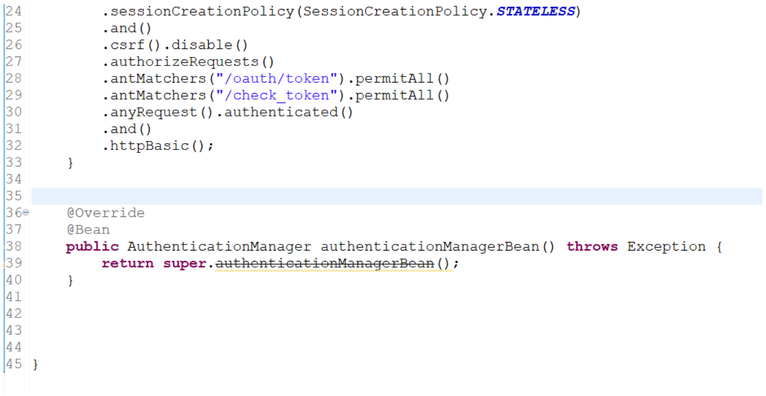
```java
package com.authserver;

import org.springframework.beans.factory.annotation.Autowired;

@Configuration
@EnableAuthorizationServer
public class AuthServerConfig extends AuthorizationServerConfigurerAdapter {

    @Bean("clientPasswordEncoder")
      PasswordEncoder clientPasswordEncoder() {
        return new BCryptPasswordEncoder(4);
      }

    @Autowired
    @Qualifier("authenticationManagerBean")
    private AuthenticationManager authenticationManager;

    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients.inMemory()
        .withClient("authserver")
            .secret("{noop}authserver")
            .authorizedGrantTypes("password", "refresh_token","client_credentials")
            .authorities("ROLE_CLIENT", "ROLE_TRUSTED_CLIENT")
            .scopes("read", "write", "trust")
            .accessTokenValiditySeconds(300)
            .refreshTokenValiditySeconds(300);
    }

    @Override
    public void configure(AuthorizationServerSecurityConfigurer oauthServer) throws Exception {
        oauthServer.checkTokenAccess("permitAll()");
        oauthServer.passwordEncoder(clientPasswordEncoder());
    }

}
```

SecurityConfig

to restrict only to get the token and verify token uris only

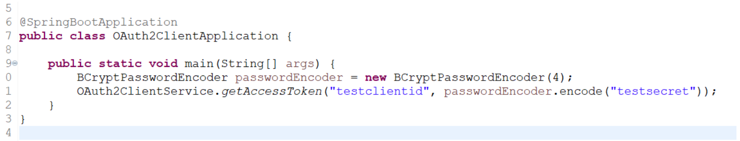


```java
package com.authserver;

import org.springframework.context.annotation.Bean;


@Configuration
@EnableWebSecurity
public class SecurityConfig extends ~~WebSecurityConfigurerAdapter~~ {


    @Override
    @Order(Ordered.HIGHEST_PRECEDENCE)
    protected void configure(HttpSecurity http) throws Exception {
        http
        .sessionManagement()
```



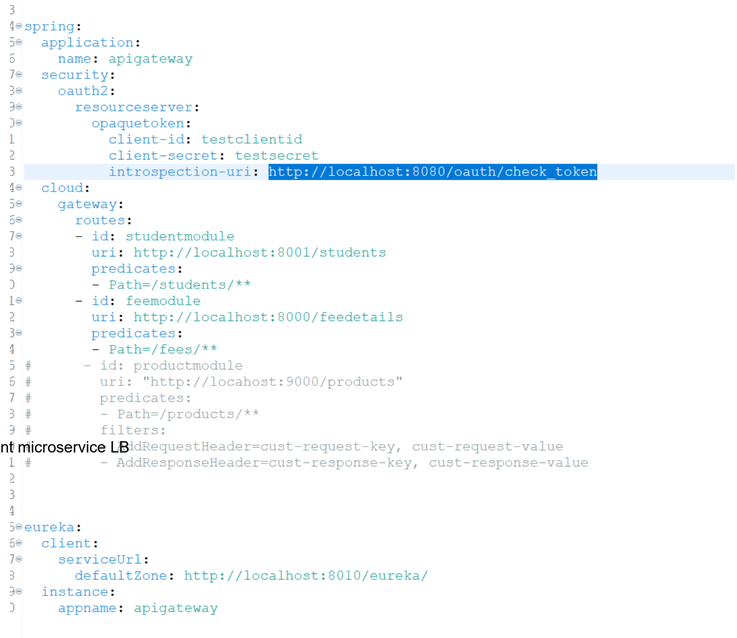
```java
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        .and()
        .csrf().disable()
        .authorizeRequests()
        .antMatchers("/oauth/token").permitAll()
        .antMatchers("/check_token").permitAll()
        .anyRequest().authenticated()
        .and()
        .httpBasic();
    }


    @Override
    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.~~authenticationManagerBean~~();
    }




}
```

Java client application to get the token from authorization server



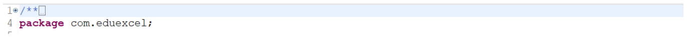
```java

@SpringBootApplication
public class OAuth2ClientApplication {

    public static void main(String[] args) {
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder(4);
        OAuth2ClientService.getAccessToken("testclientid", passwordEncoder.encode("testsecret"));
    }
}

```

API Gateway microservice to accept the token and respond with student details

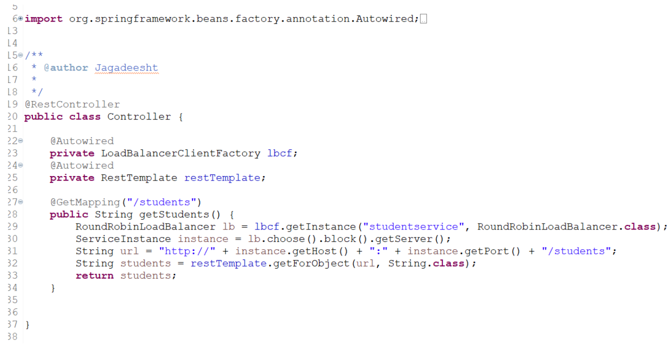


```yaml

spring:
  application:
    name: apigateway
  security:
    oauth2:
      resourceserver:
        opaquetoken:
          client-id: testclientid
          client-secret: testsecret
          introspection-uri: http://localhost:8080/oauth/check_token
  cloud:
    gateway:
      routes:
      - id: studentmodule
        uri: http://localhost:8001/students
        predicates:
        - Path=/students/**
      - id: feemodule
        uri: http://localhost:8000/feedetails
        predicates:
        - Path=/fees/**
#      - id: productmodule
#        uri: "http://locahost:9000/products"
#        predicates:
#        - Path=/products/**
#        filters:
#        - AddRequestHeader=cust-request-key, cust-request-value
#        - AddResponseHeader=cust-response-key, cust-response-value



eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8010/eureka/
  instance:
    appname: apigateway
```

Student microservice LB

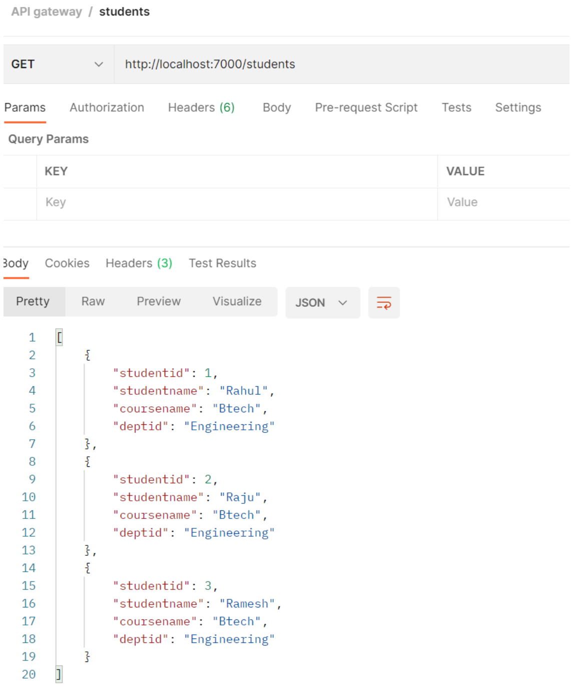


```java
/**
package com.eduexcel;
```



```java
import org.springframework.beans.factory.annotation.Autowired;


/**
 * @author Jagadeesht
 *
 */
@RestController
public class Controller {

    @Autowired
    private LoadBalancerClientFactory lbcf;
    @Autowired
    private RestTemplate restTemplate;

    @GetMapping("/students")
    public String getStudents() {
        RoundRobinLoadBalancer lb = lbcf.getInstance("studentservice", RoundRobinLoadBalancer.class);
        ServiceInstance instance = lb.choose().block().getServer();
        String url = "http://" + instance.getHost() + ":" + instance.getPort() + "/students";
        String students = restTemplate.getForObject(url, String.class);
        return students;
    }


}
```

TestIng



GET http://localhost:7000/students

Query Params

KEY	VALUE
Key	Value

```json
[
    {
        "studentid": 1,
        "studentname": "Rahul",
        "coursename": "Btech",
        "deptid": "Engineering"
    },
    {
        "studentid": 2,
        "studentname": "Raju",
        "coursename": "Btech",
        "deptid": "Engineering"
    },
    {
        "studentid": 3,
        "studentname": "Ramesh",
        "coursename": "Btech",
        "deptid": "Engineering"
    }
]
```