**Microservices with JSON Web Token (JWT), Header, Spring Boot,Caching(Hazelcast)**

**Prerequisite**

* JDK 1.7+
* Maven 3+
* Hazelcast-3.6(or above)-refer Hazelcast-Installation documentation

**Stack**

* Java
* Single Sign On
* JSON Web Token
* Spring Boot
* AngularJS

**Projects**

Create the below services (projects) in maven

1. OnlineShoppingCart
2. AuthService
3. CatalogueService
4. ShippingService
5. OrderService
6. **OnlineShoppingCart Service**

**Project Structure**

├── src

│   └── main

│   ├── java

│   │   └── com

│   │   └── techm

│   │   └── shoppingcart

│ │   └── main

│ │      └──OnlineShoppingCart.java

│ │

│ └──resources

│ │

│ ├── static(AngularJS)

│   └*──* application.properties

└── pom.xml

**Pom.xml**

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.techm.shoppingcart</groupId>  <artifactId>OnlineShoppingCart</artifactId>  <version>0.0.1-SNAPSHOT</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.1.RELEASE</version>  </parent>  <properties>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jersey</artifactId>  </dependency>  <!-- angular dependencies -->  <dependency>  <groupId>org.webjars</groupId>  <artifactId>angularjs</artifactId>  <version>1.4.9</version>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>io.jsonwebtoken</groupId>  <artifactId>jjwt</artifactId>  <version>0.6.0</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.json/json -->  <dependency>  <groupId>org.json</groupId>  <artifactId>json</artifactId>  <version>20160810</version>  </dependency>  <dependency>  <groupId>org.webjars</groupId>  <artifactId>bootstrap</artifactId>  <version>3.3.6</version>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast</artifactId>  <version>3.8</version>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast-client</artifactId>  <version>3.8</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <version>1.2.5.RELEASE</version>  <executions>  <execution>  <goals>  <goal>repackage</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build>  </project> |

|  |
| --- |
| **OnlineShoppingCart.java** |
| **Path 🡪 src/main/java/com/techm/shoppingcart/main/OnlineShoppingCart.java** |
| package com.techm.shoppingcart.main;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.annotation.ComponentScan;  @SpringBootApplication  @ComponentScan("com.techm.shoppingcart.\*")  public class OnlineShoppingCart {  public static void main(String[] args) {  SpringApplication.run(OnlineShoppingCart.class, args);  final Logger LOGGER = LoggerFactory.getLogger(OnlineShoppingCart.class);  String customlogger ="ADMS Loger::::";  LOGGER.info(customlogger+"Online Shopping Cart Service Started....");  }  } |

**Static(AngularJS)**

├── src

│   └── main

│   ├── java

│   │   └── com

│   │   └── techm

│   │   └── shoppingcart

│ │   └── main

│ │      └──OnlineShoppingCart.java

│ │

│ └──resources

│ │

│ ├── static(AngularJS)

│ │ │

│ │ ├──app

│ │ │ └──app.js

│ │ ├──controller

│ │ │ ├──homeController.js

│ │ │ ├──loginController.js

│ │ │ ├──orderController.js

│ │ │ ├──shippingController.js

│ │ │ └──SuccessController.js

│ │ ├──images

│ │ │ └──image.jpg

│ │ ├──lib

│ │ │   ├──angular-cookies.js

│ │  │ ├──angular-cookies.min.js

│ │ │ ├──angular-route.min.js

│ │ │ ├──angular.min.js

│ │ │ ├──bootstrap.min.css

│ │ │ ├──bootstrap.min.js

│ │ │ ├──jquery-1.10.2.min.js

│ │ │ └──jquery.dataTables.css

│ │ │

│ │ ├── META-INF

│ │ │ └──MANIFEST.MF

│ │ │

│ │ ├──Service

│ │ │ ├──loginService.js

│ │  │ └──successService.js

│ │ ├──views

│ │ │   ├──home.html

│ │  │ ├──login.html

│ │ │ ├──order.html

│ │ │ ├──shipping.html

│ │ │ └──success.html

│ │ │

│ │ └──index.html

│ │

│ └*──* application.properties

│

└── pom.xml

|  |
| --- |
| **app.js** |
| **Path 🡪 src/main/resources/static/app/app.js** |
| **var** microservice = angular.module('microservice', ['ngRoute', 'ngCookies']);  microservice.config(['$routeProvider', **function**($routeProvider) {    $routeProvider  .when('/', {  templateUrl : 'views/home.html',  controller : 'homeController'  })    .when('/home', {  templateUrl : 'views/home.html',  controller : 'homeController'  })  .when('/login', {  templateUrl : 'views/login.html',  controller : "loginController"  })    .when('/success', {  templateUrl : 'views/success.html',  controller : "successController"  })  .when('/order', {  templateUrl : 'views/order.html',  controller : "orderController"  })    .when('/shipping', {  templateUrl : 'views/shipping.html',  controller : "shippingController"  })    .when('/logout', {  templateUrl : 'views/home.html',  controller : "homeController"  })  .otherwise({  redirectTo : '/'  });      }  ]); |
| **homeController.js** |
| **Path 🡪 src/main/resources/static/controller/ homeController.js** |
| microservice.controller('homeController', **function**($scope, $route,$location) {    $scope.cancel = **function**() {  $location.path("/");  $route.reload();  window.location.reload();    };    $scope.logout = **function**() {  $location.path("/");  $route.reload();  window.location.reload();    };  }); |

|  |
| --- |
| **loginController.js** |
| **Path 🡪 src/main/resources/static/controller/ loginController.js** |
| microservice.controller('loginController',**function**($scope, $location, $route,loginService,$cookieStore,$rootScope){  $scope.error = **false**;  $rootScope.isLogin = **false**;  $rootScope.isLogout=**false**;  $scope.success=**function**(data){  $scope.msg=data;  }  $scope.login = **function**(){  loginService.generateJwtToken($scope.username, $scope.password, **function**(data){  **if** (data != **null**) {  $cookieStore.put('cookiejwtToken',data);  loginService.setJwttoken(data);  }  loginService.authenticate($scope.username,$scope.password,**function**(data) {  **if**($cookieStore.get('cookiejwtToken') !=**null**){    $scope.isLogin = **true**;  $scope.isLogout=**true**;  $scope.afterlogin="loggedin";  $location.path("/success");  window.location.reload(); }});  });  };  $scope.logout = **function**() {  $cookieStore.remove('cookiejwtToken');    $location.path("/logout");  window.location.reload();  };    }); |
| **orderController.js** |
| **Path 🡪 src/main/resources/static/controller/ orderController.js** |
| microservice.controller('orderController', **function**($scope, $location,$route, loginService,$cookieStore,successService,$rootScope) {  $rootScope.isLogin = **true**;  $rootScope.isLogout=**true**;  $scope.showOrder = **function**() {  **var** jwtcookie =$cookieStore.get('cookiejwtToken');  **if** (jwtcookie != **null**) {  $scope.msg=jwtcookie;  }  **else**{  $location.path("/logout");  $route.reload();  window.location.reload();  }  }  $scope.cancel = **function**() {  $location.path("/");  $route.reload();  window.location.reload();  };  }); |

|  |
| --- |
| **shippingController.js** |
| **Path 🡪 src/main/resources/static/controller/ shippingController.js** |
| microservice.controller('shippingController', **function**($scope, $location,$route, loginService,$cookieStore,successService,$rootScope) {  $rootScope.isLogin = **true**;  $rootScope.isLogout=**true**;  $scope.showShipping = **function**() {  **var** jwtcookie =$cookieStore.get('cookiejwtToken');  **if** (jwtcookie != **null**) {  $scope.msg=jwtcookie;  }  **else**{  $location.path("/logout");  $route.reload();  window.location.reload();  }  }  $scope.cancel = **function**() {  $location.path("/");  $route.reload();  window.location.reload();  };    }); |

|  |
| --- |
| **successController.js** |
| **Path 🡪 src/main/resources/static/controller/ successController.js** |
| microservice.controller('successController',**function**($scope,$location,$route, loginService,$cookieStore,successService,$rootScope) {    $rootScope.isLogin = **true**;  $rootScope.isLogout=**true**;    $scope.logout=**function**(){    $location.path("/logout");  $route.reload();  window.location.reload();  };  $scope.showSuccess = **function**() {  **var** jwtcookie =$cookieStore.get('cookiejwtToken');  **if** (jwtcookie != **null**) {      $scope.msg=jwtcookie;      }  **else**{  $location.path("/logout");  $route.reload();  window.location.reload();    }  }  $scope.cancel = **function**() {  $location.path("/");  $route.reload();  window.location.reload();  };        $scope.order = **function**() {  **var** jwtcookie =$cookieStore.get('cookiejwtToken');  **if** (jwtcookie != **null**) {  successService.saveOrder(jwtcookie, **function**(data) {  **if**(data == **null** || data === "" ){    $location.path("/login");  $scope.error = **true**;  }  **else** **if**((data != **null**)){  **var** tokenData=data.split('!')[0];  alert("inside order controller : "+data);  alert("inside order controller : "+tokenData);  **if**(tokenData==="else"){    $location.path("/login");  $scope.error = **true**;  }  **else** **if**(tokenData==="if"){  $location.path("/order");  window.location.reload();  }    }    });  }  }    $scope.shipping = **function**() {    **var** jwtcookie =$cookieStore.get('cookiejwtToken');  **if** (jwtcookie != **null**) {    successService.saveShipping(jwtcookie, **function**(data) {    **if**(data == **null** || data === "" ){    $location.path("/login");  $scope.error = **true**;  }  **else** **if**((data != **null**)){  **var** tokenData=data.split('!')[0];  alert("inside shipping controller : "+data);  alert("inside shipping controller : "+tokenData);  **if**(tokenData==="else"){    $location.path("/login");  $scope.error = **true**;  }  **else** **if**(tokenData==="if"){  $location.path("/shipping");  window.location.reload();  }    } });  }  }  }); |
| **loginService.js** |
| **Path 🡪 src/main/resources/static/Service/ loginService.js** |
| microservice.service("loginService", **function**($http,$location) {  **var** responsePromise =**null**;  **var** jwttoken = {};  **this**.setJwttoken = **function**(Jwttokendata) {  jwttoken = Jwttokendata;  }  **this**.getJwttoken = **function**() {  **return** jwttoken;  }  **var** isLogin ={}  **this**.setIsUserLogin =**function**(isUserloggedin){  isLogin = **this**.isUserloggedin;  }    **this**.getIsUserLogin =**function**(){  **return** isLogin;  }    **this**.authenticate = **function**(username, password, callback) {  **var** user = {  "username" : username,  "password" : password  };    **var** responsePromise = $http({  url : "http://172.18.33.34:8022/catalogueService/login",  method : "POST",  data : user,  headers : {  'Content-Type' : 'application/json',  'jwtToken' : jwttoken  }  });  responsePromise.success(**function**(data, status, headers, config) {  alert("Inside CatalogueService with jwt token::: " + data);  callback(data);  });  responsePromise.error(**function**(data, status, headers, config) {  alert("AJAX failed! because no webservice is attached yet");  });  }  **this**.generateJwtToken = **function**(username, password, callback) {  **var** user = {  "username" : username,  "password" : password  };  **if**((username=="admin" || username=="test" || username =="user")&&(password=="password")){    responsePromise = $http({  url : "http://172.18.33.34:8001/loginService/generateToken",  method : "POST",  data : user,  headers : {  'Content-Type' : 'application/json'  }  });  }  **else**{    alert("UserName or Password is wrong");  }  responsePromise.success(**function**(data, status, headers, config) {  alert("Inside loginService jwtToken generation Sucess::: " + data);  callback(data);    });  responsePromise.error(**function**(data, status, headers, config) {  alert("AJAX failed! because no webservice is attached yet");  });  }  }); |
| **successService.js** |
| **Path 🡪 src/main/resources/static/Service/ successService.js** |
| microservice.service("successService", **function**($http, $location, loginService) {    **this**.saveOrder = **function**(jwttoken,callback) {    **var** responsePromise = $http({  url : "http://172.18.33.34:8088/orders/createOrder",  method : "POST",    headers : {  'Content-Type' : 'application/json',  "jwttoken" : jwttoken  }  });  responsePromise.success(**function**(data, status, headers, config) {  alert("Inside Order service with jwt token");    callback(data);  });  responsePromise.error(**function**(data, status, headers, config) {  alert("AJAX failed! because no webservice is attached yet");  });  }          **this**.saveShipping = **function**(jwttoken,callback) {    **var** responsePromise = $http({  url : "http://172.18.33.34:8084/shipping/newShipping",  method : "POST",    headers : {  'Content-Type' : 'application/json',  "jwttoken" : jwttoken  }  });  responsePromise.success(**function**(data, status, headers, config) {  alert("Inside Shipping service with jwt token");  callback(data);  });  responsePromise.error(**function**(data, status, headers, config) {  alert("AJAX failed! because no webservice is attached yet");  });  }      }); |

|  |
| --- |
| **home.html** |
| **Path 🡪 src/main/resources/static/views/ home.html** |
| <br>  <br><br>  <div class=*"container"*>  <div class=*"row"*>  <h2><b><marquee behavior=*"slide"* direction=*"up"*>Microservices</marquee></h2></b>  <h3><b><marquee behavior=*"slide"* direction=*"up"*>JsonWebToken in Headers</marquee></h3></b>  <div>  <img ng-src=*"images/secure-UI-jwt.jpg"* width=*"1200"* height=*"500"* />  </div>  </div>  </div> |

|  |
| --- |
| **order.html** |
| **Path 🡪 src/main/resources/static/views/ order.html** |
| <br>  <br>  <hr>  <div >  <div class=*"container"*><center><div><h2>Welcome !!!</h2></div>  <br>  <br>  <br>  <br>  <br>  <br>  <div><h2><b><marquee behavior=*"slide"* direction=*"right"*>You are now inside the order Service</marquee></h2></b></div>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <font color= *"red"*><b><div>JWT Token</div></b></font>  <div ng-init =*"showOrder()"*>{{msg}}</div>  </center>  </div><br>  </div>  <br>  <br>  <br>  <br> |

|  |
| --- |
| **login.html** |
| **Path 🡪 src/main/resources/static/views/ login.html** |
| <br><div><form>  <div class=*"form-group"*>  <label class=*"col-sm-2"* for=*"usr"*>Name:</label>  <div class=*"col-xs-4"*>  <input type=*"text"* id=*"uname"*ng-model=*"username"* placeholder=*"Enter username"* required>  </div>  </div>  <div class=*"form-group"*>  <label class=*"col-sm-2"* for=*"pwd"*>Password:</label>  <div class=*"col-xs-4"*>  <input type=*"password"* id=*"password"*ng-model=*"password"* placeholder=*"\*\*\*\*\*\*\*\*\*"* required> <br>  </div>  </div><div class=*"control-label"*>  <label class=*"col-sm-12"*></label>  <div class=*"col-xs-4"*>  <button type=*"submit"* class=*"btn btn-success"*data-ng-click=*"login()"* ng-disabled=*"myForm.$invalid"*>Login</button>  </div></div>  </form>  </div> |

|  |
| --- |
| **shipping.html** |
| **Path 🡪 src/main/resources/static/views/ shipping.html** |
| <br>  <br>  <hr>  <div >  <div class=*"container"*><center><div><h2>Welcome !!!</h2></div>  <br>  <br>  <br>  <br>  <br>  <br>  <div><h2><b><marquee behavior=*"slide"* direction=*"right"*>You are now inside the shipping Service</marquee></h2></b></div>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <font color= *"red"*><b><div>JWT Token</div></b></font>  <div ng-init =*"showShipping()"*>{{msg}}</div>  </center>  </div><br>  </div>  <br><br><br><br> |

|  |
| --- |
| **success.html** |
| **Path 🡪 src/main/resources/static/views/ success.html** |
| <br>  <br>  <hr>  <div >  <div class=*"container"*>  <center><div><b><h2><marquee behavior=*"alternate"*>Welcome!!! Successfully Logged In!!!</marquee></h2></b></div>  <br>  <br>  <button type=*"button"* data-ng-click=*"order()"*>Order</button><br><br>  <button type=*"button"* data-ng-click=*"shipping()"*>Shipping</button>  <br>  <br>  <font color= *"red"*><b><div>JWT Token</div></b></font>  <div ng-init =*"showSuccess()"*>{{msg}}</div>  </center>  </div>  <br>  </div>  <br>  <br>  <br>  <br> |

|  |
| --- |
| **index.html** |
| **Path 🡪 src/main/resources/static / index.html** |
| <!DOCTYPE html>  <html data-ng-app=*"microservice"*>  <head>  <title>JWT In Header</title>  <meta charset=*"utf-8"*>  <meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>  <link rel=*"stylesheet"*href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"*>  <script src=*"https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"*></script>  <script src=*"lib/bootstrap.min.js"*></script>  <script src=*"lib/angular.min.js"*></script>  <script src=*"lib/angular-route.min.js"*></script>  <script src=*"lib/angular-cookies.min.js"*></script>  <script src=*"lib/angular-cookies.js"*></script>  <script src=*"app/app.js"*></script>  <script src=*"controller/homeController.js"*></script>  <script src=*"controller/loginController.js"*></script>  <script src=*"controller/successController.js"*></script>  <script src=*"controller/orderController.js"*></script>  <script src=*"controller/shippingController.js"*></script>  <script src=*"service/loginService.js"*></script>  <script src=*"service/successService.js"*></script>  <style>  /\* Remove the navbar's default rounded borders and increase the bottom margin \*/  *.navbar* {  margin-bottom: *-20px*;  border-radius: *0*;  }  /\* Remove the jumbotron's default bottom margin \*/  *.jumbotron* {  margin-bottom: *0*;  }  /\* Add a gray background color and some padding to the footer \*/  **footer** {  background-color: *#f2f2f2*;  padding: *5px*;  }  *#logo* {  text-align: *center*;  }  *#footer* {  background-color: *#555*;  color: *white*;  /\* padding: 15px; \*/  }  *#logo* {  text-align: *center*;  }  **html,** **body** {  height: *100%*;  width: *100%*;  }  *.row.content* {  background-image: *url("images1.jpg")*;  background-repeat: *no-repeat*;  height: *648px*;  }  </style>  </head>  <body ng-controller=*"loginController"*>  <nav class=*"navbar navbar-inverse navbar-fixed-top"*>  <div class=*"container-fluid"*>  <div class=*"navbar-header"*>  <button type=*"button"* class=*"navbar-toggle"* data-toggle=*"collapse"*  data-target=*"#myNavbar"*>  <span class=*"icon-bar"*></span> <span class=*"icon-bar"*></span> <span  class=*"icon-bar"*></span>  </button>  <a class=*"navbar-brand"* href=*"#"*><b>JWT In Header</b></a>  </div>  <div class=*"collapse navbar-collapse"* id=*"myNavbar"*>  <ul class=*"nav navbar-nav navbar-right"*>    <li><a href=*"#login"* data-ng-show=*"!isLogin"*><span class=*"glyphicon glyphicon-log-in"*> Login </span></a></li>    <li><a href=*"#logout"* data-ng-click = *"logout()"* data-ng-show=*"isLogout"*><span class=*"glyphicon glyphicon-log-out"*> Logout </span></a></li> </ul>  </div>  </div>  </nav>  <br>  <div ng-view></div>  </body>  </html> |

|  |
| --- |
| **application.properties** |
| **Path 🡪 src/main/resources/application.properties** |
| server.port = 8000  server.host =172.18.33.34 |

1. **Auth Service**

**Project Structure**

├── src

│   └── main

│   ├── java

│   │   └── com

│   │   └── techm

│   │   └── auth

│ │   ├── jwt

│ │   │   ├── GenerateJwtToken.java

│ │   │   └── JwtUtil.java

│ │   │

│ │ └── main

│ └──resources └──AuthService.java

│   └── application.properties

└── pom.xml

**Pom.xml**

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>AuthService</groupId>  <artifactId>AuthService</artifactId>  <version>0.0.1-SNAPSHOT</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.1.RELEASE</version>  </parent>  <properties>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jersey</artifactId>  </dependency>  <!-- angular dependencies -->  <dependency>  <groupId>org.webjars</groupId>  <artifactId>angularjs</artifactId>  <version>1.4.9</version>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>io.jsonwebtoken</groupId>  <artifactId>jjwt</artifactId>  <version>0.6.0</version>  </dependency>  <dependency>  <groupId>org.json</groupId>  <artifactId>json</artifactId>  <version>20160810</version>  </dependency>  <dependency>  <groupId>org.webjars</groupId>  <artifactId>bootstrap</artifactId>  <version>3.3.6</version>  <scope>runtime</scope></dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast</artifactId>  <version>3.8</version>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast-client</artifactId>  <version>3.8</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <version>1.2.5.RELEASE</version>  <executions>  <execution>  <goals>  <goal>repackage</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins></build>  </project> |

**Steps involved in this approach**

1. Create JWT Token for combining both username and password.

2.Include this generated token in each and every request headers.

|  |
| --- |
| **GenerateJwtToken.java** |
| **Path 🡪 src/main/java/com/techm/auth/jwt/ GenerateJwtToken.java** |
| package com.techm.auth.jwt;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.ws.rs.Consumes;  import javax.ws.rs.Produces;  import javax.ws.rs.core.MediaType;  import org.json.JSONObject;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.stereotype.Component;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.RestController;  import com.hazelcast.client.HazelcastClient;  import com.hazelcast.client.config.ClientConfig;  import com.hazelcast.core.HazelcastInstance;  import com.hazelcast.core.IMap;  @RestController  @RequestMapping(value="/loginService")  @Component  @CrossOrigin  public class GenerateJwtToken {    private static final String signingKey = "signingKey";  String token = null ;  private static final Logger LOGGER = LoggerFactory.getLogger(GenerateJwtToken.class);  String customlogger ="ADMS Loger::::";    @RequestMapping(value="/generateToken",method = RequestMethod.POST)  @Produces(MediaType.APPLICATION\_JSON)  @Consumes(MediaType.APPLICATION\_JSON)  @CrossOrigin  public String generateToken(HttpServletResponse httpServletResponse,@RequestBody String userCredentials,HttpServletRequest httpServletRequest) {    String ipAddress = httpServletRequest.getRemoteAddr();  System.out.println("IP Address: "+ipAddress);    //Hazelcast instance  ClientConfig clientConfig = new ClientConfig();  clientConfig.getGroupConfig().setName("dev").setPassword("dev-pass");  clientConfig.getNetworkConfig().addAddress("172.18.33.34:5701", "172.18.33.34:5702");        HazelcastInstance client = HazelcastClient.newHazelcastClient(clientConfig);      IMap<String, String> jwttoken = client.getMap("jwttoken");      jwttoken.evictAll();      LOGGER.info(customlogger+"GenerateJwtToken Started");  LOGGER.info(customlogger+"Loggedin User Credentials ::" + userCredentials);  JSONObject userCredjson = null;  String username = null;  String password = null;  try {  if (userCredentials != null)  userCredjson = new JSONObject(userCredentials.toString());  if (userCredjson != null) {  username = userCredjson.optString("username");  password = userCredjson.optString("password");  LOGGER.info(customlogger+"Entered username ::"+username+"\t Entered password ::"+password);  if(username !=null)  token = JwtUtil.generateToken(signingKey, username,password);  LOGGER.info(customlogger+"Generated JwtToken::: " +token);  System.out.println("success");  String value = new StringBuilder(ipAddress).append("-").append(username).toString();  System.out.println(value);  jwttoken.put(value,token);  }  } catch (Exception e) {  LOGGER.error(customlogger+"Exception Occured in generateToken() method");  e.printStackTrace();  }  return token;  }      @RequestMapping(value= "/test")  public String getmessage(){    return "sucess";  }    } |

We use JJWT to generate/parse JWT Token.

|  |
| --- |
| **JwtUtil.java** |
| **Path 🡪 src/main/java/com/techm/auth/jwt/ JwtUtil.java** |
| package com.techm.auth.jwt;  import io.jsonwebtoken.JwtBuilder;  import io.jsonwebtoken.Jwts;  import io.jsonwebtoken.SignatureAlgorithm;  import java.util.Date;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  public class JwtUtil {  public static String generateToken(String signingKey, String username, String password) {    final Logger LOGGER = LoggerFactory.getLogger(JwtUtil.class);  String customlogger ="ADMS Loger::::";  LOGGER.info(customlogger+"JwtUtil Started.");    long nowMillis = System.currentTimeMillis();    JwtBuilder builder = Jwts.builder()  .setIssuedAt(new Date(nowMillis))  .setSubject(String.valueOf(username))  .setIssuer(signingKey)    .claim("password", password)  .signWith(SignatureAlgorithm.HS256, signingKey);      return builder.compact();  }  } |

.setIssuedAt(new Date(nowMillis)) : time of issuing the token

.setSubject(String.valueOf(username)) : Sets the JWT Claims [sub](https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-25#section-4.1.2) (subject)

.setIssuer(signingKey) : Sets the JWT Claims [iss](https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-25#section-4.1.1) (issuer) value.

.claim("password", password) : Sets a custom JWT Claims parameter value

.signWith(SignatureAlgorithm.HS256,

signingKey) : Signs the constructed JWT using the specified algorithm with the specified key, producing a JWS.

|  |
| --- |
| **AuthService.java** |
| **Path 🡪 src/main/java/com/techm/auth/main/AuthService.java** |
| package com.techm.auth.main;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.annotation.ComponentScan;  @SpringBootApplication  @ComponentScan("com.techm.auth.\*")  public class AuthService {  public static void main(String[] args) {  SpringApplication.run(AuthService.class, args);  final Logger LOGGER = LoggerFactory.getLogger(AuthService.class);  String customlogger ="ADMS Loger::::";  LOGGER.info(customlogger+"Auth Service Started....");  }  } |

|  |
| --- |
| **application.properties** |
| **Path 🡪 src/main/resources/application.properties** |
| server.port = 8000  server.host =172.18.33.34 |

1. **Catalogue Service**

**Project Structure**

├──src

│   └──main

│   ├──java

│   │   └──com

│   │   └──techm

│   │   └──cadt

│ │   ├──catalogue

│ │ │ └─rest

│ │   │   ├──CatalogueServiceController.java

│ │ │ │

│ │   │   └──JwtUtil.java

│ │ └── start

│ │ ├──ApplicationConfig.java

│ │ └──CatalogueService.java

│ └──resources

│   └── application.properties

└── pom.xml

**Source**

**Pom.xml**

|  |
| --- |
| <projectxmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>Catalogue</groupId>  <artifactId>CatalogueService</artifactId>  <version>1.0</version>  <packaging>jar</packaging>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.1.RELEASE</version>  </parent>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jersey</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>com.h2database</groupId>  <artifactId>h2</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.json</groupId>  <artifactId>json</artifactId>  <version>20090211</version>  </dependency>  <dependency>  <groupId>io.jsonwebtoken</groupId>  <artifactId>jjwt</artifactId>  <version>0.6.0</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast</artifactId>  <version>3.8</version>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast-client</artifactId>  <version>3.8</version>  </dependency>    </dependencies>  </project> |

**Steps involved in this approach**

1. In the target micro service once you reach the request headers data decode the jwt token.

2. Compare the Logged in user details against decoded JWT token values.

3. On successful validation of these credentials provide access to resource.

|  |
| --- |
| **CatalogueServiceController.java** |
| **Path 🡪 src/main/java/com/tech/cadt/catalogue/rest/ CatalogueServiceController.java** |
| package com.tech.cadt.catalogue.rest;  import java.util.ArrayList;  import java.util.List;  import java.util.Set;  import javax.ws.rs.Consumes;  import javax.ws.rs.Produces;  import javax.ws.rs.core.MediaType;  import org.json.JSONException;  import org.json.JSONObject;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.boot.web.support.SpringBootServletInitializer;  import org.springframework.cache.annotation.Cacheable;  import org.springframework.http.HttpHeaders;  import org.springframework.stereotype.Component;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestHeader;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.ResponseBody;  import org.springframework.web.bind.annotation.RestController;  import com.hazelcast.client.HazelcastClient;  import com.hazelcast.client.config.ClientConfig;  import com.hazelcast.config.Config;  import com.hazelcast.config.XmlConfigBuilder;  import com.hazelcast.core.Hazelcast;  import com.hazelcast.core.HazelcastInstance;  import com.hazelcast.core.IMap;  import com.tech.cadt.start.CatalogueService;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  @RestController  @RequestMapping("/catalogueService")  @Component  @CrossOrigin  public class CatalogueServiceController extends SpringBootServletInitializer {  private static final Logger LOGGER = LoggerFactory.getLogger(CatalogueServiceController.class);  String customlogger ="ADMS Loger::::";  private static final String signingKey = "signingKey";    @Produces(MediaType.APPLICATION\_JSON)  @Consumes(MediaType.APPLICATION\_JSON)  @RequestMapping(value = "/login", method = RequestMethod.POST)  @CrossOrigin  @ResponseBody  public String authenticateCustomer(HttpServletResponse httpServletResponse, @RequestHeader HttpHeaders headers,@RequestBody String userCredentials,HttpServletRequest httpServletRequest) {      String ipAddressCatalogue = httpServletRequest.getRemoteAddr();  System.out.println("IP Address: "+ipAddressCatalogue);    ClientConfig clientConfig = new ClientConfig();    clientConfig.getGroupConfig().setName("dev").setPassword("dev-pass");  clientConfig.getNetworkConfig().addAddress("172.18.33.34:5701", "172.18.33.34:5702");    HazelcastInstance client = HazelcastClient.newHazelcastClient(clientConfig);  IMap map = client.getMap( "jwttoken" );        System.out.println( "Token" + map.values() );  System.out.println("key"+map.keySet().toString());    String authvalue= map.keySet().toString();  String result = authvalue.replaceAll("[\\[\\]]","");  System.out.println("authvalue--"+result);          LOGGER.info(customlogger+"userCredentials::" + userCredentials);  JSONObject userCredjson = null;  CatalogueService customerObj = null;  String username = null;  CatalogueService customerresultObj = null ;  boolean validation =false;  LOGGER.info(customlogger+"Header Object in Customer Service::::::"+headers);  List<String> token = headers.get("jwtToken");  LOGGER.info("JWT token:::::"+token.get(0));  String listObj = token.get(0);  String jwtUsername = JwtUtil.getSubject(httpServletResponse, listObj, signingKey);  String jwtPassword = JwtUtil.getPassword(httpServletResponse, listObj, signingKey);  LOGGER.info(customlogger+"Jwt User credentials after Parsing:::::");  LOGGER.info(customlogger+"username::"+jwtUsername);  LOGGER.info(customlogger+"password::"+jwtPassword);  System.out.println("token"+token);    String valueCatalogue = new StringBuilder(ipAddressCatalogue).append("-").append(jwtUsername).toString();      System.out.println("valueCatalogue----"+valueCatalogue);    try {    if (jwtUsername != null) {    if ((token.equals(map.values())) && (result.equals(valueCatalogue)) ) {  validation = true;    LOGGER.info("JWT Token Validataion Sucessful in Catalogue Service.");  return "if!"+token;  }  else if(result.equals(null)||result.equals("")){    System.out.println("inside result null");  return result;  }  else  {      LOGGER.info("JWT Token Validataion failed in Catalogue Service.");  validation = false;  return "else!"+token;  }  } else {  validation = false;  return result;  }      } catch (Exception e) {    }    return result;      }    } |

|  |
| --- |
| **JwtUtil.java** |
| **Path 🡪 src/main/java/com/tech/cadt/catalogue/rest/ JwtUtil.java** |
| package com.tech.cadt.catalogue.rest;  import io.jsonwebtoken.Claims;  import io.jsonwebtoken.Jws;  import io.jsonwebtoken.Jwts;  import javax.servlet.http.HttpServletResponse;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  public class JwtUtil {    private static final Logger LOGGER = LoggerFactory.getLogger(JwtUtil.class);    public static String getSubject(HttpServletResponse httpServletResponse,String token, String signingKey) {    LOGGER.info("JWTUtil Service Started::::::");  LOGGER.info("Inside getSubject() method::::");  Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);  String username = claims.getBody().getSubject();  LOGGER.info("Subject :::"+username);  return username;  }  public static String getPassword(HttpServletResponse httpServletResponse,String token, String signingKey) {  LOGGER.info("JWTUtil Service Started::::::");  LOGGER.info("Inside getPassword() method::::");  Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);  Object password = claims.getBody().get("password");  LOGGER.info("password :::"+password);  return password.toString();  }  } |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **ApplicationConfig.java** | | **Path 🡪 src/main/java/com/tech/cadt/start/ApplicationConfig.java** |   package com.tech.cadt.start;  import javax.inject.Named;  import org.glassfish.jersey.server.ResourceConfig;  import org.springframework.context.annotation.Configuration;  import org.springframework.stereotype.Component;  @Configuration  @Component  public class ApplicationConfig {  @Named  static class JerseyConfig extends ResourceConfig {  public JerseyConfig() {  this.packages("com.tech.cadt.catalogue");  }}  } |

|  |
| --- |
| **CatalogueService.java** |
| **Path 🡪 src/main/java/com/tech/cadt/start/ CatalogueService.java** |
| package com.tech.cadt.start;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.annotation.ComponentScan;  import org.springframework.stereotype.Component;  @SpringBootApplication  @Component  @ComponentScan("com.tech.cadt.catalogue")  public class CatalogueService {  public static void main(String[] args) {  SpringApplication.run(CatalogueService.class, args);  final Logger LOGGER = LoggerFactory.getLogger(CatalogueService.class);  String customlogger ="ADMS Loger::::";  LOGGER.info(customlogger+"CustomerService Started");  }  } |

|  |
| --- |
| **application.properties** |
| **Path 🡪 src/main/resources/application.properties** |
| server.port = 8022  server.host =172.18.33.34 |

1. **ShippingService**

**Project Structure**

├──src

│   └──main

│   ├──java

│   │   └──com

│   │   └──techm

│   │   └──cadt

│ │   ├──shipping

│ │ │ └─rest

│ │   │   ├──ShippingServiceController.java

│ │   │   └──JwtUtil.java

│ │ └── start

│ │ ├──ApplicationConfig.java

│ │ └──ShippingService.java

│ └──resources

│   └── application.properties

└── pom.xml

**Source**

**Pom.xml**

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>ShippingService</groupId>  <artifactId>ShippingService</artifactId>  <version>1.0</version>    <packaging>jar</packaging>    <!-- <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.2.0.RELEASE</version>  </parent> -->  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.1.RELEASE</version>  </parent>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jersey</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>com.h2database</groupId>  <artifactId>h2</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.json</groupId>  <artifactId>json</artifactId>  <version>20090211</version>  </dependency>  <dependency>  <groupId>io.jsonwebtoken</groupId>  <artifactId>jjwt</artifactId>  <version>0.6.0</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast</artifactId>  <version>3.8</version>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast-client</artifactId>  <version>3.8</version>  </dependency>    </dependencies>  </project> |

**Steps involved in this approach**

1. In the target micro service once you reach the request headers data decode the jwt token.

2. Compare the Logged in user details against decoded JWT token values.

3. On successful validation of these credentials provide access to resource.

|  |
| --- |
| **ShippingServiceController.java** |
| **Path 🡪 src/main/java/com/tech/cadt/shipping/rest/ ShippingServiceController.java** |
| package com.tech.cadt.shipping.rest;  import java.util.List;  import javax.persistence.criteria.Order;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.ws.rs.Consumes;  import javax.ws.rs.GET;  import javax.ws.rs.Produces;  import javax.ws.rs.core.MediaType;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.HttpHeaders;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestHeader;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.ResponseBody;  import org.springframework.web.bind.annotation.RestController;  import org.springframework.web.client.RestTemplate;  import com.hazelcast.client.HazelcastClient;  import com.hazelcast.client.config.ClientConfig;  import com.hazelcast.core.HazelcastInstance;  import com.hazelcast.core.IMap;  @RestController  @RequestMapping("/shipping")  @CrossOrigin  public class ShippingServiceController {  private static final Logger LOGGER = LoggerFactory.getLogger(ShippingServiceController.class);  private static final String signingKey = "signingKey";    @Consumes(MediaType.APPLICATION\_JSON)  @RequestMapping(value="/newShipping", method = RequestMethod.POST)  @CrossOrigin  @ResponseBody  public String createnewShipping(HttpServletResponse httpServletResponse,@RequestHeader HttpHeaders headers,HttpServletRequest httpServletRequest) {  String ipAddressShipping = httpServletRequest.getRemoteAddr();  System.out.println("IP Address: "+ipAddressShipping);    ClientConfig clientConfig = new ClientConfig();    clientConfig.getGroupConfig().setName("dev").setPassword("dev-pass");  clientConfig.getNetworkConfig().addAddress("172.18.33.34:5701", "172.18.33.34:5702");    HazelcastInstance client = HazelcastClient.newHazelcastClient(clientConfig);  IMap map = client.getMap( "jwttoken" );        System.out.println( "Token" + map.values() );  System.out.println("key"+map.keySet().toString());    String authvalue= map.keySet().toString();  String result = authvalue.replaceAll("[\\[\\]]","");  System.out.println("authvalue-- :"+result);      LOGGER.info("Inside createnewShipping() new Shipping Details::");  RestTemplate restTemplate = new RestTemplate();  Order resultObj = null ;      String usernameVal = null;    LOGGER.info("Headers::::::" + headers);  List<String> token = headers.get("jwtToken");  System.out.println("JWT token:::::" + token);  LOGGER.info("JWT token:::::" + token.get(0));  String listObj = token.get(0);  String jwtUsername = JwtUtil.getSubject(httpServletResponse,listObj, signingKey);  String jwtPassword = JwtUtil.getPassword(httpServletResponse,listObj, signingKey);  LOGGER.info("::::::Jwt User credentials after Parsing:::::");  LOGGER.info("username::" + jwtUsername);  LOGGER.info("password::" + jwtPassword);    LOGGER.info("Customer Object from Customer Service ::::");    String valueShipping = new StringBuilder(ipAddressShipping).append("-").append(jwtUsername).toString();    System.out.println("valueShipping----"+valueShipping);    try {    boolean validation=false;      if (jwtUsername != null) {      if ((token.equals(map.values())) && (result.equals(valueShipping)) ) {  validation = true;    LOGGER.info("JWT Token Validataion Sucessful in Shipping Service.");  return "if!"+token;  }  else if(result.equals(null)||result.equals("")){    System.out.println("inside result null");  return result;  }  else  {      LOGGER.info("JWT Token Validataion failed in Shipping Service.");  validation = false;  return "else!"+token;  }  } else {  validation = false;  return result;  }      } catch (Exception e) {    }    return result;      }    } |

|  |
| --- |
| **JwtUtil.java** |
| **Path 🡪 src/main/java/com/tech/cadt/shipping/rest/ JwtUtil.java** |
| package com.tech.cadt.catalogue.rest;  import io.jsonwebtoken.Claims;  import io.jsonwebtoken.Jws;  import io.jsonwebtoken.Jwts;  import javax.servlet.http.HttpServletResponse;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  public class JwtUtil {    private static final Logger LOGGER = LoggerFactory.getLogger(JwtUtil.class);    public static String getSubject(HttpServletResponse httpServletResponse,String token, String signingKey) {    LOGGER.info("JWTUtil Service Started::::::");  LOGGER.info("Inside getSubject() method::::");  Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);  String username = claims.getBody().getSubject();  LOGGER.info("Subject :::"+username);  return username;  }  public static String getPassword(HttpServletResponse httpServletResponse,String token, String signingKey) {  LOGGER.info("JWTUtil Service Started::::::");  LOGGER.info("Inside getPassword() method::::");  Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);  Object password = claims.getBody().get("password");  LOGGER.info("password :::"+password);  return password.toString();  }  } |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **ApplicationConfig.java** | | **Path 🡪 src/main/java/com/tech/cadt/start/ApplicationConfig.java** |   package com.tech.cadt.start;  import javax.inject.Named;  import org.glassfish.jersey.server.ResourceConfig;  import org.springframework.context.annotation.Configuration;  import org.springframework.stereotype.Component;  @Configuration  @Component  public class ApplicationConfig {  @Named  static class JerseyConfig extends ResourceConfig {  public JerseyConfig() {  this.packages("com.tech.cadt.shipping");  }  }  } |

|  |
| --- |
| **ShippingService.java** |
| **Path 🡪 src/main/java/com/tech/cadt/start/ ShippingService.java** |
| package com.tech.cadt.start;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.annotation.ComponentScan;  import org.springframework.stereotype.Component;  @SpringBootApplication  @Component  @ComponentScan("com.tech.cadt.shipping")  public class ShippingService {  public static void main(String[] args) {  SpringApplication.run(ShippingService.class, args);}} |

|  |
| --- |
| **application.properties** |
| **Path 🡪 src/main/resources/application.properties** |
| server.port = 8084  server.host =172.18.33.34 |

**5.OrderService**

**Project Structure**

├──src

│   └──main

│   ├──java

│   │   └──com

│   │   └──techm

│   │   └──cadt

│ │   ├──order

│ │ │ └─rest

│ │   │   ├──OrderServiceController.java

│ │   │   └──JwtUtil.java

│ │ └── start

│ │ ├──ApplicationConfig.java

│ │ └──OrderService.java

│ └──resources

│   └── application.properties

└── pom.xml

**Source**

**Pom.xml**

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>Customer</groupId>  <artifactId>OrderService</artifactId>  <version>1.0</version>    <packaging>jar</packaging>    <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.1.RELEASE</version>  </parent>    <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>    <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jersey</artifactId>  </dependency>    <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>    <dependency>  <groupId>com.h2database</groupId>  <artifactId>h2</artifactId>  <scope>runtime</scope>  </dependency>      <dependency>  <groupId>io.jsonwebtoken</groupId>  <artifactId>jjwt</artifactId>  <version>0.6.0</version>  </dependency>    <dependency>  <groupId>org.json</groupId>  <artifactId>json</artifactId>  <version>20090211</version>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast</artifactId>  <version>3.8</version>  </dependency>  <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast-client</artifactId>  <version>3.8</version>  </dependency>    </dependencies>    </project> |

**Steps involved in this approach**

1. In the target micro service once you reach the request headers data decode the jwt token.

2. Compare the Logged in user details against decoded JWT token values.

3. On successful validation of these credentials provide access to resource.

|  |
| --- |
| **OrderRestServiceController.java** |
| **Path 🡪 src/main/java/com/tech/cadt/order/rest/ OrderRestServiceController.java** |
| package com.tech.cadt.order.rest;  import java.util.List;  import javax.persistence.criteria.Order;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.ws.rs.Consumes;  import javax.ws.rs.GET;  import javax.ws.rs.Produces;  import javax.ws.rs.core.MediaType;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.HttpHeaders;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestHeader;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.ResponseBody;  import org.springframework.web.bind.annotation.RestController;  import org.springframework.web.client.RestTemplate;  import com.hazelcast.client.HazelcastClient;  import com.hazelcast.client.config.ClientConfig;  import com.hazelcast.core.HazelcastInstance;  import com.hazelcast.core.IMap;  import com.tech.cadt.order.rest.JwtUtil;  @RestController  @RequestMapping("/orders")  @CrossOrigin  public class OrderRestServiceController {  private static final Logger LOGGER = LoggerFactory.getLogger(OrderRestServiceController.class);  private static final String signingKey = "signingKey";      @Consumes(MediaType.APPLICATION\_JSON)  @RequestMapping(value="/createOrder", method = RequestMethod.POST)  @CrossOrigin  @ResponseBody  public String createnewOrder(HttpServletResponse httpServletResponse,@RequestHeader HttpHeaders headers,HttpServletRequest httpServletRequest) {    String ipAddressOrder = httpServletRequest.getRemoteAddr();  System.out.println("IP Address: "+ipAddressOrder);    ClientConfig clientConfig = new ClientConfig();    clientConfig.getGroupConfig().setName("dev").setPassword("dev-pass");  clientConfig.getNetworkConfig().addAddress("172.18.33.34:5701", "172.18.33.34:5702");    HazelcastInstance client = HazelcastClient.newHazelcastClient(clientConfig);  IMap map = client.getMap( "jwttoken" );        System.out.println( "Token" + map.values() );  System.out.println("key"+map.keySet().toString());    String authvalue= map.keySet().toString();  String result = authvalue.replaceAll("[\\[\\]]","");  System.out.println("authvalue--"+result);              LOGGER.info("Inside createnewOrder() new Order Details::");  RestTemplate restTemplate = new RestTemplate();  Order resultObj = null ;      String usernameVal = null;    LOGGER.info("Headers::::::" + headers);  List<String> token = headers.get("jwtToken");  LOGGER.info("JWT token:::::" + token.get(0));  String listObj = token.get(0);  String jwtUsername = JwtUtil.getSubject(httpServletResponse,listObj, signingKey);  String jwtPassword = JwtUtil.getPassword(httpServletResponse,listObj, signingKey);  LOGGER.info("::::::Jwt User credentials after Parsing:::::");  LOGGER.info("username::" + jwtUsername);  LOGGER.info("password::" + jwtPassword);    LOGGER.info("Customer Object from Customer Service ::::");  String valueOrder = new StringBuilder(ipAddressOrder).append("-").append(jwtUsername).toString();    System.out.println("valueOrder----"+valueOrder);    try {    boolean validation=false;  if (jwtUsername != null) {  if ((token.equals(map.values())) && (result.equals(valueOrder)) ) {  validation = true;    LOGGER.info("JWT Token Validataion Sucessful in Order Service.");  return "if!"+token;  }  else if(result.equals(null)||result.equals("")){    System.out.println("inside result null");  return result;  }  else  {      LOGGER.info("JWT Token Validataion failed in Order Service.");  validation = false;  return "else!"+token;  }  } else {  validation = false;  return result;  }      } catch (Exception e) {    }    return result;      }    } |

|  |
| --- |
| **JwtUtil.java** |
| **Path 🡪 src/main/java/com/tech/cadt/order/rest/ JwtUtil.java** |
| package com.tech.cadt.order.rest;  import io.jsonwebtoken.Claims;  import io.jsonwebtoken.Jws;  import io.jsonwebtoken.Jwts;  import javax.servlet.http.HttpServletResponse;  public class JwtUtil {    public static String getSubject(HttpServletResponse httpServletResponse,String token, String signingKey) {  Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);  String username = claims.getBody().getSubject();  return username;  }    public static String getPassword(HttpServletResponse httpServletResponse,String token, String signingKey) {  Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);  Object password = claims.getBody().get("password");  return password.toString();  }  } |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **ApplicationConfig.java** | | **Path 🡪 src/main/java/com/tech/cadt/start/ApplicationConfig.java** |   package com.tech.cadt.start;  import javax.inject.Named;  import org.glassfish.jersey.server.ResourceConfig;  import org.springframework.context.annotation.Configuration;  import org.springframework.stereotype.Component;  @Configuration  @Component  public class ApplicationConfig {  @Named  static class JerseyConfig extends ResourceConfig {  public JerseyConfig() {  this.packages("com.tech.cadt.order");  }  }  } |

|  |
| --- |
| **OrderService.java** |
| **Path 🡪 src/main/java/com/tech/cadt/start/ OrderService.java** |
| package com.tech.cadt.start;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.annotation.ComponentScan;  import org.springframework.stereotype.Component;  @SpringBootApplication  @Component  @ComponentScan("com.tech.cadt.order")  public class OrderService {  public static void main(String[] args) {  SpringApplication.run(OrderService.class, args);  }  } |

|  |
| --- |
| **application.properties** |
| **Path 🡪 src/main/resources/application.properties** |
| server.port = 8088  server.host =172.18.33.34 |

**Build and run the application**

Navigate to the project where pom file is located and use the below command

mvn clean spring-boot:run

D:\headers\Microservices\_JWT-in Headers\OnlineShoppingCart>mvn clean spring-boot:run

D:\headers\Microservices\_JWT-in Headers\AuthService>mvn clean spring-boot:run

D:\headers\Microservices\_JWT-in Headers\CatalogueService>mvn clean spring-boot:run

D:\headers\Microservices\_JWT-in Headers\ShippingService>mvn clean spring-boot:run

D:\headers\Microservices\_JWT-in Headers\OrderService>mvn clean spring-boot:run

**Test and run the application**

|  |  |  |
| --- | --- | --- |
| Service Name | Port Number | Main Class |
| OnlineShoppingCart | 8000 | OnlineShoppingCart.java |
| AuthService | 8001 | AuthService.java |
| CatalogueService | 8022 | CatalogueService.java |
| ShippingService | 8084 | ShippingService.java |
| OrderService | 8088 | OrderService.java |

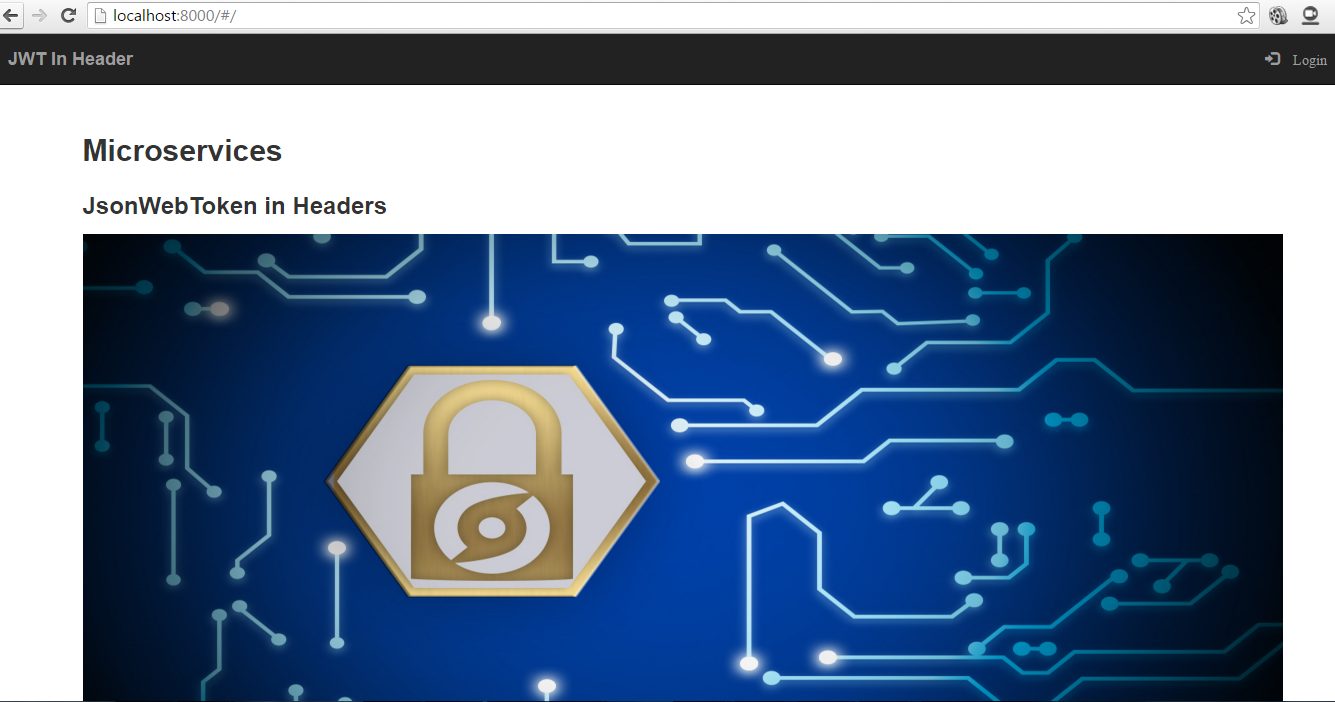
All the services should be executed for executing Online Shopping Cart service.

After all the services are running, application can be executed with following url in the browser:

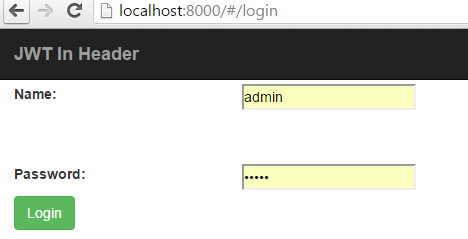
<http://localhost:8000/>

Username : admin

Password   :admin

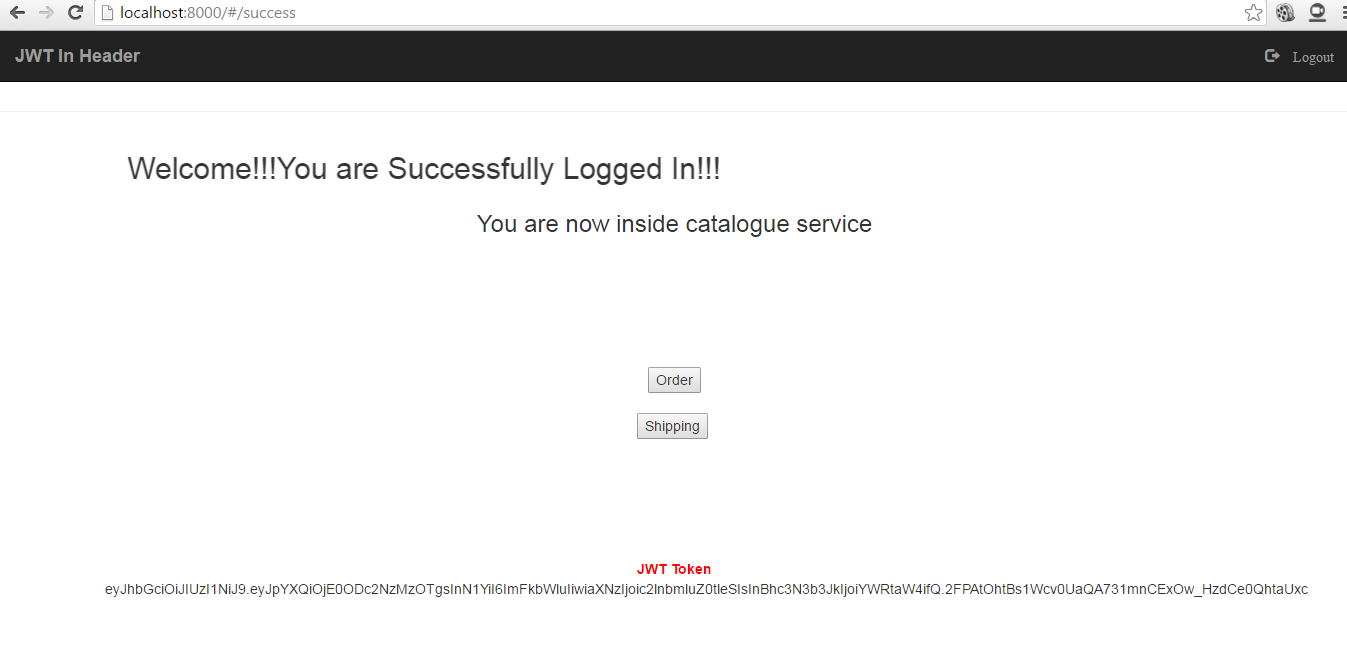


Click on login

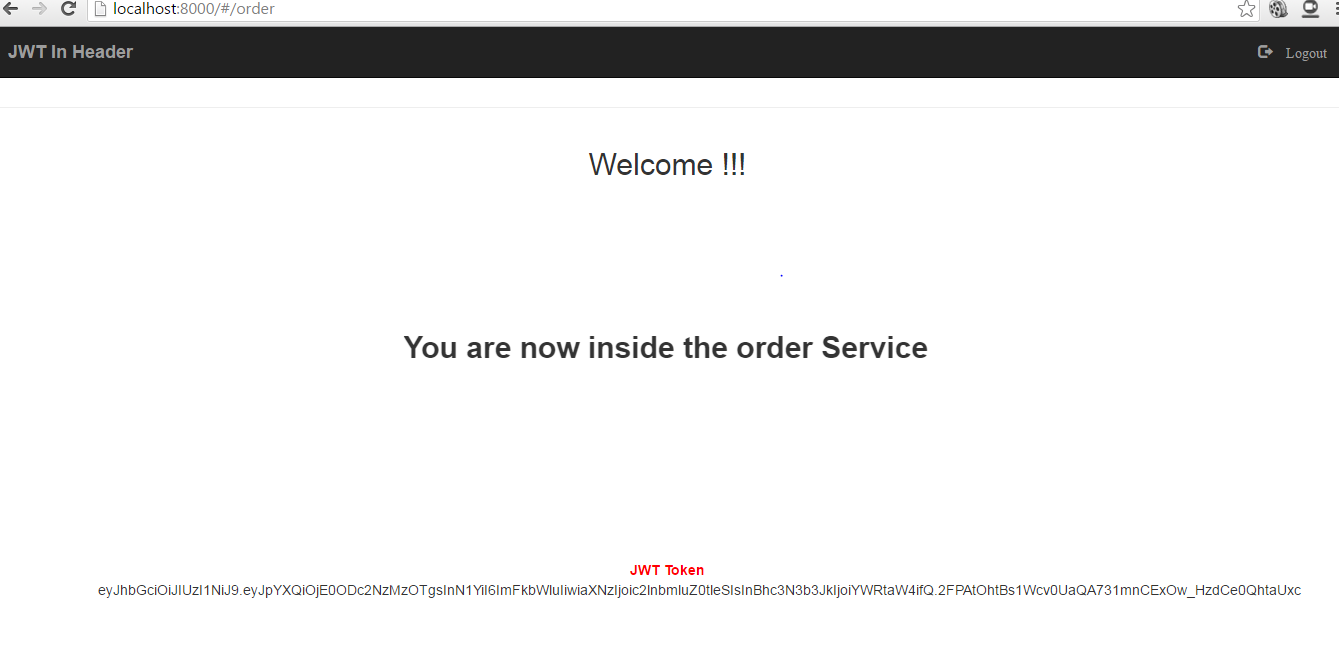


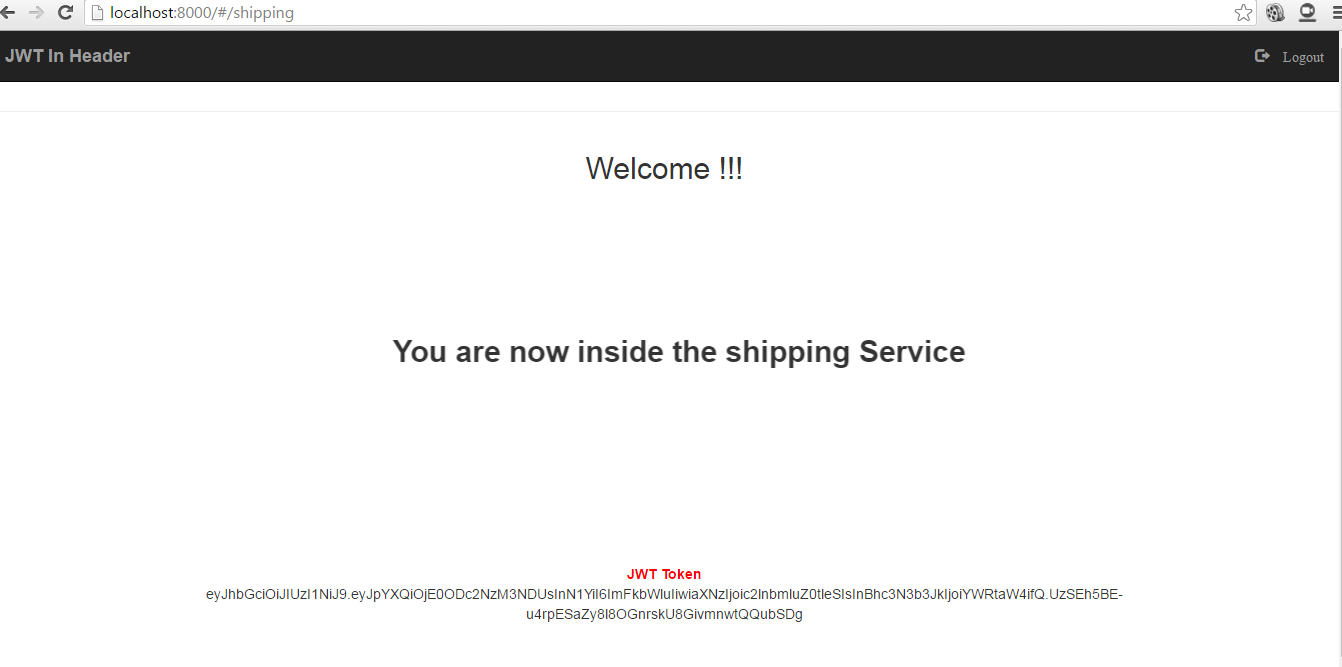
Enter the user id and password and click log in.

Jwt token is generated during login ,and it will be passed to the target service in headers, , It will redirect to the appropriate url



If we click on order and shipping services the same token will be carried to the target service in headers





Now click “logout”