Software Engineering Lab

Flavia Monti

June 25, 2019

1 SOAP Web Service

1.1 Server

1. Server = Maven project
2. Create an INTERFACE containing all the functionalities that I want to give to the user.
3. Create a class that IMPLEMENTS that interface.
4. Create a class with the MAIN (in the main I need to specify the url of the server, for instance http://localhost:8080/server).

AwsServer is the name of the server of the first lab.

```
package com.mycompany.aaawsserver;

import javax.xml.ws.Endpoint;

public class Server {
    public static void main(String args[]) throws
    InterruptedException {
        AaawsImpl implementor = new AaawsImpl();
        String address = "http://localhost:8080/aaaws";
        Endpoint.publish(address, implementor);
    }
}
```

Listing 1: Server.java

```
package com.mycompany.aaawsserver;

import java.util.List;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebResult;
import javax.jws.WebService;
import javax.xml.bind.annotation.XmlSeeAlso;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import javax.xml.ws.RequestWrapper;
import javax.xml.ws.ResponseWrapper;

@WebService
public interface AaawsIFace {
    public java.lang.String[] getClients();
}
```

Listing 2: AwsIFace.java

```
package com.mycompany.aaawsserver;

import javax.jws.WebService;

@WebService(endpointInterface = "com.mycompany.aaawsserver.
    AaawsIFace")
```

```
public class AaawsImpl implements AaawsIFace {

    private String[] clients = new String[2];

    public AaawsImpl(){
        clients[0] = "1,Massimo Mecella";
        clients[1] = "2,Miguel Ceriani";
    }

    @Override
    public String[] getClients() {
        return clients;
    }
}
```

Listing 3: AwsImpl.java

```
<?xml version="1.0" encoding="UTF-8"?>
<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.mycompany</groupId>
    <artifactId>aaawsserver</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.
    sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-frontend-jaxws</artifactId>
            <version>3.1.6</version>
        </dependency>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-transports-http-jetty</artifactId>
            <version>3.1.6</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <configuration>
                    <mainClass>com.mycompany.aaawsserver.Server</
    mainClass>
                </configuration>
            </plugin>
        </plugins>
```

```
    </build>

</project>
```

Listing 4: pom.xml

1.2 Client

1. Client = normal java project
2. Create a client web server specifying the url where it has to take the functionalities (http://localhost:8080/server?wsdl)

```
package com.mycompany.aaawsserver;

import javax.xml.ws.Endpoint;

public class Server {
    public static void main(String args[]) throws
    InterruptedException {
        AaawsImpl implementor = new AaawsImpl();
        String address = "http://localhost:8080/aaaws";
        Endpoint.publish(address, implementor);
    }
}
```

Listing 5: AwsClient.java

2 RESTful services

2.1 Server

Maven project

```java
package com.mycompany.serverlab2;

import org.apache.cxf.endpoint.Server;
import org.apache.cxf.jaxrs.*;
import org.apache.cxf.jaxrs.lifecycle.SingletonResourceProvider;

public class MyServer {
    public static void main(String[] args) throws Exception{
        JAXRSServerFactoryBean factoryBean = new
    JAXRSServerFactoryBean();
        factoryBean.setResourceClasses(CourseRepository.class);
        factoryBean.setResourceProvider(new
    SingletonResourceProvider(new CourseRepository()));
        factoryBean.setAddress("http://localhost:8080/");
        Server server = factoryBean.create();
    }
}
```

Listing 6: Server.java

```java
package com.mycompany.serverlab2;

import java.util.Objects;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "Student")
public class Student {
    private int id;
    private String name;

    //Generate Setter e Getter of Id, Name

    //Generate hashCode() and equals() (override)
}
```

Listing 7: Student.java

```java
package com.mycompany.serverlab2;

import java.util.*;
import javax.ws.rs.*;
import javax.ws.rs.core.*;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "Course")
public class Course {
    private int id;
    private String name;
    private List<Student> students = new ArrayList<>();

    @GET
```

```
    @Path("{studentId}")
    public Student getStudent(@PathParam("studentId")int studentId) {
        return findById(studentId);
    }

    @POST
    @Path("")
    public Response createStudent(Student student) {
        for (Student element : students) {
            if (element.getId() == student.getId()) {
                return Response.status(Response.Status.CONFLICT).build();
            }
        }
        students.add(student);
        return Response.ok(student).build();
    }

    @DELETE
    @Path("{studentId}")
    public Response deleteStudent(@PathParam("studentId") int studentId) {
        Student student = findById(studentId);
        if (student == null) {
            return Response.status(Response.Status.NOT_FOUND).build();
        }
        students.remove(student);
        return Response.ok().build();
    }

    private Student findById(int id) {
        for (Student student : students) {
            if (student.getId() == id) {
                return student;
            }
        }
        return null;
    }

    //Generate Setter e Getter of Id, Name, Students

    //Generate hashCode() and equals() (override)
}
```

Listing 8: Course.java

```
package com.mycompany.serverlab2;

import java.util.*;
import javax.ws.rs.*;
import javax.ws.rs.core.*;

@Path("course")
@Produces("text/xml")
public class CourseRepository {
    private Map<Integer, Course> courses = new HashMap<>();
```

```
11
12      public CourseRepository(){
13          Student student1 = new Student();
14          Student student2 = new Student();
15          student1.setId(1);
16          student1.setName("Student A");
17          student2.setId(2);
18          student2.setName("Student B");
19
20          List<Student> course1Students = new ArrayList<>();
21          course1Students.add(student1);
22          course1Students.add(student2);
23
24          Course course1 = new Course();
25          Course course2 = new Course();
26          course1.setId(1);
27          course1.setName("REST with Spring");
28          course1.setStudents(course1Students);
29          course2.setId(2);
30          course2.setName("Learn Spring Security");
31
32          courses.put(1, course1);
33          courses.put(2, course2);
34      }
35
36      @GET
37      @Path("courses/{courseId}")
38      public Course getCourse(@PathParam("courseId") int courseId) {
39          return findById(courseId);
40      }
41
42      @PUT
43      @Path("courses/{courseId}")
44      public Response updateCourse(@PathParam("courseId") int
        courseId, Course course) {
45          Course existingCourse = findById(courseId);
46          if (existingCourse == null) {
47              return Response.status(Response.Status.NOT_FOUND).build
        ();
48          }
49          if (existingCourse.equals(course)) {
50              return Response.notModified().build();
51          }
52          courses.put(courseId, course);
53          return Response.ok().build();
54      }
55
56      @Path("courses/{courseId}/students")
57      public Course pathToStudent(@PathParam("courseId") int courseId
        ) {
58          return findById(courseId);
59      }
60
61      private Course findById(int id) {
62          for (Map.Entry<Integer, Course> course : courses.entrySet()
        ) {
63              if (course.getKey() == id) {
```

```
                return course.getValue();
            }
        }
        return null;
    }
}
```

Listing 9: CourseRepository.java

```
<?xml version="1.0" encoding="UTF-8"?>
<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.mycompany</groupId>
    <artifactId>ServerLab2</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-frontend-jaxrs</artifactId>
            <version>3.1.7</version>
        </dependency>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-transports-http-jetty</artifactId>
            <version>3.1.7</version>
        </dependency>
    </dependencies>
</project>
```

Listing 10: pom.xml

2.2 Client

Maven project

```
package com.mycompany.clientlab2;

import javax.ws.rs.core.Response;
import org.apache.cxf.jaxrs.client.WebClient;

public class MyClient {

    public static void main(String[] args) throws Exception{
        WebClient client = WebClient.create("http://localhost:8080/course");

        //GET course
```

```
        Course course = client.path("courses/1").accept("text/xml")
    .get().readEntity(Course.class);
        System.out.println(course.getName());

        //POST student
        Student student = new Student();
        student.setId(100);
        student.setName("MASSIMO DECIMO MECELLO");
        Response r = client.path("students").post(student);
        System.out.println(r.getStatus());

        //GET student
        Student mecello = client.path("100").get().readEntity(
    Student.class);
        System.out.println(mecello.getName());
    }
}
```

Listing 11: Client.java

In the Client I copy the code of the two classes Course and Student that I found in the Server.

```
package com.mycompany.clientlab2;

import java.util.*;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "Course")
public class Course {
    private int id;
    private String name;
    private List<Student> students = new ArrayList<>();

    private Student findById(int id) {
        for (Student student : students) {
            if (student.getId() == id) {
                return student;
            }
        }
        return null;
    }

    //Generate Setter e Getter of Id, Name, Students

    //Generate hashCode() and equals() (override)
}
```

Listing 12: Course.java

```
package com.mycompany.clientlab2;

import java.util.Objects;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "Student")
public class Student {
    private int id;
```

```
    private String name;

    //Generate Setter e Getter of Id, Name

    //Generate hashCode() and equals() (override)
}
```

Listing 13: Student.java

```
<?xml version="1.0" encoding="UTF-8"?>
<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.mycompany</groupId>
    <artifactId>ClientLab2</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-rs-client</artifactId>
            <version>3.0.15</version>
        </dependency>
    </dependencies>
</project>
```

Listing 14: pom.java

3 JMS

3.1 Client

Maven project

```java
package com.mycompany.lab3_giusto;

import java.util.Properties;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.JMSException;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import javax.jms.Destination;
import javax.jms.Message;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class ClientSynchrono {
    public static void main(String[] args) throws NamingException,
    JMSException {
        Properties props = new Properties();
        props.setProperty(Context.INITIAL_CONTEXT_FACTORY, "org.
    apache.activemq.jndi.ActiveMQInitialContextFactory");
        props.setProperty(Context.PROVIDER_URL, "tcp
    ://192.168.49.81:61616");
        InitialContext jndiContext = new InitialContext(props);
        ConnectionFactory cf = (ConnectionFactory) jndiContext.
    lookup("ConnectionFactory");
        Destination destination = (Destination) jndiContext.lookup(
    "dynamicTopics/Quotazioni");
        Connection connection = cf.createConnection();
        Session session = connection.createSession(false, Session.
    AUTO_ACKNOWLEDGE);
        MessageConsumer consumer = session.createConsumer(
    destination);
        connection.start();
        while (true) {
            Message m = consumer.receive();
            System.out.print(m.getStringProperty("Nome")+" ");
            System.out.println(m.getFloatProperty("Valore"));
        }
    }
}
```

Listing 15: ClientSync.java

```java
package com.mycompany.lab3_giusto;

import java.util.Properties;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
```

```java
import javax.jms.MessageListener;
import javax.jms.Session;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class ClientAsynchrono {
    public static void main(String[] args) throws NamingException, JMSException {
        Properties props = new Properties();
        props.setProperty(Context.INITIAL_CONTEXT_FACTORY, "org.apache.activemq.jndi.ActiveMQInitialContextFactory");
        props.setProperty(Context.PROVIDER_URL, "tcp://192.168.49.81:61616");
        InitialContext jndiContext = new InitialContext(props);
        ConnectionFactory cf = (ConnectionFactory) jndiContext.lookup("ConnectionFactory");
        Destination destination = (Destination) jndiContext.lookup("dynamicTopics/Quotazioni");
        Connection connection = cf.createConnection();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageConsumer consumer = session.createConsumer(destination);
        MessageListener listener = new myListener();
        consumer.setMessageListener(listener);
        connection.start();
    }
}
```

Listing 16: ClientAsyn.java

```java
package com.mycompany.lab3_giusto;

import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;

public class myListener implements MessageListener{
    @Override
    public void onMessage(Message msg){
        try {
            System.out.println("Received message!");
            System.out.println(msg.getStringProperty("Nome")+" "+msg.getFloatProperty("Valore"));
            System.out.println("MECELLONEEEE!!!\n\n");
        } catch (JMSException e) {
            System.out.println("ERRRORREEE");
        }
    }
}
```

Listing 17: Listener.java

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">
3     <modelVersion>4.0.0</modelVersion>
4     <groupId>com.mycompany</groupId>
5     <artifactId>Lab3_giusto</artifactId>
6     <version>1.0-SNAPSHOT</version>
7     <packaging>jar</packaging>
8     <dependencies>
9
10        <dependency>
11            <groupId>javax.jms</groupId>
12            <artifactId>javax.jms-api</artifactId>
13            <version>2.0</version>
14        </dependency>
15
16        <dependency>
17            <groupId>org.apache.activemq</groupId>
18            <artifactId>activemq-all</artifactId>
19            <version>5.14.5</version>
20        </dependency>
21
22    </dependencies>
23    <properties>
24        <project.build.sourceEncoding>UTF-8</project.build.
    sourceEncoding>
25        <maven.compiler.source>1.8</maven.compiler.source>
26        <maven.compiler.target>1.8</maven.compiler.target>
27    </properties>
28 </project>
```

Listing 18: pom.java

3.2 Complex

Client and Server Maven project

3.2.1 Client

```java
package com.mycompany.lab3_giusto;

import java.util.Properties;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.JMSException;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import javax.jms.Destination;
import javax.jms.Message;
import javax.jms.TextMessage;
import javax.jms.Topic;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class ClientSynchrono {
    public static void main(String[] args) throws NamingException, JMSException {
        Properties props = new Properties();
        props.setProperty(Context.INITIAL_CONTEXT_FACTORY, "org.apache.activemq.jndi.ActiveMQInitialContextFactory");
        //props.setProperty(Context.PROVIDER_URL, "tcp://192.168.49.81:61616");   //with mecellone
        props.setProperty(Context.PROVIDER_URL, "tcp://localhost:61616");
        InitialContext jndiContext = new InitialContext(props);
        ConnectionFactory cf = (ConnectionFactory) jndiContext.lookup("ConnectionFactory");
        //Destination destination = (Destination) jndiContext.lookup("Ordini");  //mecellone
        Connection connection = cf.createConnection();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        Topic topic = session.createTopic("Ordini");
        MessageConsumer consumer = session.createConsumer(topic);
        connection.start();
        while (true) {
            TextMessage m = (TextMessage) consumer.receive();
            System.out.println(m.getText());
            //System.out.print(m.getStringProperty("Nome")+" "); //mecellone
            //System.out.println(m.getFloatProperty("Valore"));  //mecellone
        }
    }
}
```

Listing 19: ClientSync.java

```java
package com.mycompany.lab3_giusto;

import java.net.URI;
import java.util.Properties;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.jms.MessageListener;
import javax.jms.MessageProducer;
import javax.jms.Session;
import javax.jms.Topic;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import org.apache.activemq.ActiveMQConnectionFactory;
import org.apache.activemq.broker.BrokerFactory;
import org.apache.activemq.broker.BrokerService;

public class ClientAsynchrono {
    public static void main(String[] args) throws NamingException, JMSException, Exception {
        BrokerService broker = BrokerFactory.createBroker(new URI("broker:(tcp://localhost:61616)"));
        broker.start();
        ConnectionFactory cfprod = new ActiveMQConnectionFactory("tcp://localhost:61616");
        Connection connectionprod = cfprod.createConnection();
        Session sessionprod = connectionprod.createSession(false, Session.AUTO_ACKNOWLEDGE);
        Topic topic = sessionprod.createTopic("Ordini");
        MessageProducer producer = sessionprod.createProducer(topic);
        Properties props = new Properties();
        props.setProperty(Context.INITIAL_CONTEXT_FACTORY, "org.apache.activemq.jndi.ActiveMQInitialContextFactory");
        props.setProperty(Context.PROVIDER_URL, "tcp://192.168.49.81:61616");
        InitialContext jndiContext = new InitialContext(props);
        ConnectionFactory cf = (ConnectionFactory) jndiContext.lookup("ConnectionFactory");
        Destination destination = (Destination) jndiContext.lookup("dynamicTopics/Quotazioni");
        Connection connection = cf.createConnection();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageConsumer consumer = session.createConsumer(destination);
        MessageListener listener = new myListener(session, producer);
        consumer.setMessageListener(listener);
        connection.start();
    }
```

```
}
```

Listing 20: ClientAsyn.java

```
package com.mycompany.lab3_giusto;

import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.MessageProducer;
import javax.jms.Session;
import javax.jms.TextMessage;

public class myListener implements MessageListener{
    private Session session;
    private MessageProducer producer;
    private int counter;

    public myListener(Session s, MessageProducer mp){
        session = s;
        producer = mp;
        counter = 0;
    }

    @Override
    public void onMessage(Message msg){
        try {
            System.out.println("Received message!");
            System.out.println(msg.getStringProperty("Nome")+" "+
    msg.getFloatProperty("Valore"));
            System.out.println("MECELLONEEEE!!!\n\n");
            counter += 1;
            if (counter%4 == 0) {
                String s = msg.getStringProperty("Nome");
                producer.send(session.createTextMessage("O' compro
    "+s));
            }
        } catch (JMSException e) {
            System.out.println("ERRRORREEE");

        }
    }
}
```

Listing 21: Listener.java

3.2.2 Server

```
package com.mycompany.lab3_giusto;

import java.net.URI;
import java.util.Properties;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
```

```
11 import javax.jms.MessageListener;
12 import javax.jms.MessageProducer;
13 import javax.jms.Session;
14 import javax.jms.Topic;
15 import javax.naming.Context;
16 import javax.naming.InitialContext;
17 import javax.naming.NamingException;
18 import org.apache.activemq.ActiveMQConnectionFactory;
19 import org.apache.activemq.broker.BrokerFactory;
20 import org.apache.activemq.broker.BrokerService;
21
22 public class myProducer {
23     public static void main(String[] args) throws NamingException,
      JMSException, InterruptedException, Exception {
24         BrokerService broker = BrokerFactory.createBroker(new URI("
      broker:(tcp://localhost:61616)"));
25         broker.start();
26         ConnectionFactory cf = new ActiveMQConnectionFactory("tcp
      ://localhost:61616");
27         Connection connection = cf.createConnection();
28         Session session = connection.createSession(false, Session.
      AUTO_ACKNOWLEDGE);
29         Topic topic = session.createTopic("Ordini");
30         MessageProducer producer = session.createProducer(topic);
31         connection.start();
32         while (true) {
33             producer.send(session.createTextMessage("Ciao"));
34             Thread.sleep(1000);
35         }
36     }
37 }
```

Listing 22: Producer.java

4 Complex system

4.1 Catalog Microservice

Maven project

```java
package com.mycompany.catalog;

import java.util.List;

public class Product {
    final private int id;
    private String Name;
    public Product(int id,String Name){
        this.id = id;
        this.Name = Name;
    }
    public int getId(){return this.id;}
    public String getName(){return this.Name;}

}
```

Listing 23: Product.java

```java
package com.mycompany.catalog;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.stream.Stream;
import com.mycompany.catalog.Product;

public class Products{
    private ArrayList<Product> products;
    public Products(){
        products = new ArrayList<Product>();
    }
    public void addProduct(String name){
        Stream<Product> stream = this.products.stream();
        Stream<Integer> ids = stream.map(p -> new Integer(p.getId()
    ));
        Object[] idsArray;
        idsArray = ids.toArray();
        Random rand = new Random();
        int idx;
        while(true){
            idx = rand.nextInt();
            boolean used = false;
            for (int i = 0;i<idsArray.length;i++){
                if(idsArray[i].equals(new Integer(idx))){
                    used = true;
                    break;
                }
            }
            if(!used)break;
            else continue;
        }
        Product p = new Product(idx,name);
```

```
34          this.products.add(p);
35      }
36      public List<Product> getProducts(){
37          return (List<Product>)this.products.clone();
38      }
39  }
```

Listing 24: Products.java

```
1   package com.mycompany.catalog;
2   import com.google.gson.JsonArray;
3   import com.google.gson.JsonObject;
4   import com.mycompany.catalog.Products;
5   import com.rabbitmq.client.Channel;
6   import com.rabbitmq.client.Connection;
7   import com.rabbitmq.client.ConnectionFactory;
8   import com.rabbitmq.client.DeliverCallback;
9   import java.util.List;
10
11  public class CatalogMS {
12      private final static String QUEUE_NAME = "CATZ_CATALOG";
13      private final static String QUEUE_NAME_RET = "CATZ_CATALOG_RET"
        ;
14      private final static String HOST = "192.168.49.81";
15
16      public static void main(String[] argv) throws Exception {
17
18          // CREATE PRODUCTS
19          Products prod = new Products();
20          for(int i=0;i<10;i++)prod.addProduct("Control");
21          for(int i=0;i<10;i++)prod.addProduct("Durex");
22          for(int i=0;i<10;i++)prod.addProduct("Akuel");
23          for(int i=0;i<10;i++)prod.addProduct("Pesante");
24          for(int i=0;i<10;i++)prod.addProduct("Esp");
25          for(int i=0;i<10;i++)prod.addProduct("Masculan");
26          for(int i=0;i<10;i++)prod.addProduct("Serena");
27
28          // JSON CATALOG
29          JsonArray jcat = new JsonArray();
30          List<Product> lprods = prod.getProducts();
31          for(int i=0;i<lprods.size();i++){
32              Product p = lprods.get(i);
33              JsonObject jp = new JsonObject();
34              jp.addProperty("name", p.getName());
35              jp.addProperty("id", p.getId());
36              jcat.add(jp);
37          }
38          //-----------------------------------------------------
39          ConnectionFactory factory = new ConnectionFactory();
40          factory.setHost(HOST);
41          Connection connection = factory.newConnection();
42          Channel channel = connection.createChannel();
43
44          // RECEIVE QUEUE
45          channel.queueDeclare(QUEUE_NAME, false, false, false, null)
        ;
46          System.out.println(" [*] Waiting for messages. To exit
        press CTRL+C");
```

```
47          DeliverCallback deliverCallback = (consumerTag, delivery)
    -> {
48              String message = new String(delivery.getBody(), "UTF-8"
    );
49              System.out.println(" [x] Received '" + message + "'");
50              if(message.equals("DammeErCatalogo")){
51                  System.out.println("\t->Sending Catalog");
52                  channel.basicPublish("", QUEUE_NAME_RET, null,jcat.
    toString().getBytes("UTF-8"));
53              System.out.println(" [x] Sent '" + message + "'");
54              }
55          };
56          // SEND QUEUE
57              channel.queueDeclare(QUEUE_NAME_RET, false, false,
    false, null);
58
59          // START CONSUME
60          channel.basicConsume(QUEUE_NAME, true, deliverCallback,
    consumerTag -> { });
61      }
62  }
```

Listing 25: CatalogService.java

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <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">
3      <modelVersion>4.0.0</modelVersion>
4      <groupId>com.mycompany</groupId>
5      <artifactId>Catalog</artifactId>
6      <version>1.0-SNAPSHOT</version>
7      <packaging>jar</packaging>
8      <properties>
9          <project.build.sourceEncoding>UTF-8</project.build.
       sourceEncoding>
10         <maven.compiler.source>1.8</maven.compiler.source>
11         <maven.compiler.target>1.8</maven.compiler.target>
12     </properties>
13     <dependencies>
14         <dependency>
15             <groupId>com.rabbitmq</groupId>
16             <artifactId>amqp-client</artifactId>
17             <version>5.7.0</version>
18         </dependency>
19         <dependency>
20             <groupId>com.google.code.gson</groupId>
21             <artifactId>gson</artifactId>
22             <version>2.8.0</version>
23         </dependency>
24 </dependencies>
25 </project>
```

Listing 26: pom.java

4.2 Identiry Microservice

Maven project

```
package com.mycompany.identity.microservice;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.DeliverCallback;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.Timestamp;
import javax.xml.bind.DatatypeConverter;

class TokenMapEntry {
    long token;
    long timestamp;
    TokenMapEntry(long token, long timestamp) {
        this.token = token;
        this.timestamp = timestamp;
    }
}

public class IdentityService {
    private final static String QUEUE_AUTH = "CATZ_AUTH";
    private final static String QUEUE_AUTH_RET = "CATZ_AUTH_RET";
    private final static String QUEUE_VERIFY = "CATZ_VERIFY";
    private static final Map<String, TokenMapEntry> users_tokens =
    new HashMap<>();
    private static final Map<String, String> users_pw = new HashMap
    <>();
    private static final Random RND = new Random(42);

    public static void main(String[] args) throws Exception{
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("192.168.49.81");
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();
        channel.queueDeclare(QUEUE_AUTH, false, false, false, null)
    ;
        channel.queueDeclare(QUEUE_AUTH_RET, false, false, false,
    null);
        channel.queueDeclare(QUEUE_VERIFY, false, false, false,
    null);

        DeliverCallback authCallback = (consumerTag, delivery) -> {
            String message = new String(delivery.getBody(), "UTF-8"
    );
            System.out.println(" [x] Received in AUTH'" + message +
     "'");
            // retrieve user and pw
```

```
48              String[] msg_split = message.split(":");
49              if (msg_split.length != 2) {
50                  System.out.println("skipping message "+message);
51                  return;
52              }
53              String user = msg_split[0];
54              String pw = msg_split[1];
55              try {
56                  MessageDigest digest = MessageDigest.getInstance("
    SHA-256");
57                  byte[] hash = digest.digest(pw.getBytes(
    StandardCharsets.UTF_8));
58                  String hash_str = bytesToHex(hash);
59                  System.out.println("hashed "+hash_str);
60                  // hash from db
61                  Boolean user_contained = users_pw.containsKey(user)
    ;
62                  if (!user_contained) {
63                      System.out.println("not in db");
64                      return;
65                  }
66                  String true_hash = users_pw.get(user);
67                  if (true_hash.equals(hash_str)) {
68                      System.out.println("authenticated");
69                      long new_token = RND.nextLong();
70                      long timestamp = System.currentTimeMillis();
71                      TokenMapEntry entry = new TokenMapEntry(
    new_token, timestamp);
72                      users_tokens.put(user, entry);
73                      channel.basicPublish("", QUEUE_AUTH_RET, null,
    String.valueOf(new_token).getBytes("UTF-8"));
74                      System.out.println("new token "+String.valueOf(
    new_token));
75                  } else {
76                      System.out.println("not authenticated");
77                      channel.basicPublish("", QUEUE_AUTH_RET, null,
    "NO".getBytes("UTF-8"));
78                  }
79              } catch (NoSuchAlgorithmException ex) {
80                  System.out.println("mucho errore");
81              }
82          };
83
84          DeliverCallback verifyCallback = (consumerTag, delivery) ->
     {
85              String message = new String(delivery.getBody(), "UTF-8"
    );
86              System.out.println(" [x] Received in VERIFY'" + message
     + "'");
87          };
88
89          channel.basicConsume(QUEUE_AUTH, true, authCallback,
    consumerTag -> { });
90          channel.basicConsume(QUEUE_VERIFY, true, authCallback,
    consumerTag -> { });
91
92          // TEST
```

```
93          String dummy_hash = "30
    C952FAB122C3F9759F02A6D95C3758B246B4FEE239957B2D4FEE46E26170C4"
    ;
94          users_pw.put("user", dummy_hash);
95          String message = "user:pw";
96          channel.basicPublish("", QUEUE_AUTH, null, message.getBytes
    ("UTF-8"));
97          System.out.println("message sent");
98      }
99
100     private static String bytesToHex(byte[] hash) {
101         return DatatypeConverter.printHexBinary(hash);
102     }
103 }
```

Listing 27: IdentityService.java

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <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">
3      <modelVersion>4.0.0</modelVersion>
4      <groupId>com.mycompany</groupId>
5      <artifactId>identity-microservice</artifactId>
6      <version>1.0-SNAPSHOT</version>
7      <packaging>jar</packaging>
8      <properties>
9          <project.build.sourceEncoding>UTF-8</project.build.
       sourceEncoding>
10         <maven.compiler.source>1.8</maven.compiler.source>
11         <maven.compiler.target>1.8</maven.compiler.target>
12     </properties>
13
14     <dependencies>
15         <dependency>
16             <groupId>com.rabbitmq</groupId>
17             <artifactId>amqp-client</artifactId>
18             <version>5.7.0</version>
19         </dependency>
20     </dependencies>
21
22 </project>
```

Listing 28: pom.java

4.3 Server

Maven project

```
package com.mycompany.serverrest;

import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.DeliverCallback;
import java.util.*;
import javax.ws.rs.*;
import javax.xml.bind.annotation.XmlRootElement;
import org.json.JSONObject;

@XmlRootElement(name = "Catalog")
public class Catalog {
    private static List<String> catalogList;
    private final static String queueName = "CATZ_CATALOG";
    private final static String queueNameRet = "CATZ_CATALOG_RET";
    private String message = "DammeErCatalogo";
    private String messageRet;

    public static List<String> getCatalogList() {
        return catalogList;
    }

    public static void setCatalogList(List<String> catalogList) {
        Catalog.catalogList = catalogList;
    }

    @GET
    @Path("catalog")
    public List<String> getCatalog() throws Exception{
        //SEND MESSAGE
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("192.168.49.81");
        try (Connection connection = factory.newConnection();
             Channel channel = connection.createChannel()) {
                channel.queueDeclare(queueName, false, false, false
    , null);
                //queue = channel;
                channel.basicPublish("", queueName, null, message.
    getBytes());
            }

        //RECEIVE RESPONSE
        ConnectionFactory factoryRet = new ConnectionFactory();
        factoryRet.setHost("192.168.49.81");
        Connection connectionRet = factoryRet.newConnection();
        Channel channel = connectionRet.createChannel();
        channel.queueDeclare(queueNameRet, false, false, false,
    null);
        DeliverCallback deliverCallback = (consumerTag, delivery)
    -> {
            String messageRet = new String(delivery.getBody(), "UTF
    -8");
```

```
            System.out.println(" [x] Received '" + messageRet + "'");
            this.messageRet = messageRet;
        };
        channel.basicConsume(queueNameRet, true, deliverCallback, consumerTag -> { });
        return new ArrayList<>();
    }
}
```

Listing 29: Catalog.java

```
package com.mycompany.serverrest;

import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.DeliverCallback;
import java.util.*;
import javax.ws.rs.*;
import javax.xml.bind.annotation.XmlRootElement;
import org.json.JSONObject;

@XmlRootElement(name = "Identity")
public class Identity {
    private final static String queueName = "CATZ_AUTH";
    private final static String queueNameRet = "CATZ_AUTH_RET";
    private String message = "user:pw";
    private String messageRet;

    public void getToken() throws Exception{
        //SEND MESSAGE
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("192.168.49.81");
        try (Connection connection = factory.newConnection();
        Channel channel = connection.createChannel()) {
            channel.queueDeclare(queueName, false, false, false, null);
            channel.basicPublish("", queueName, null, message.getBytes());
        }
        //RECEIVE RESPONSE
        ConnectionFactory factoryRet = new ConnectionFactory();
        factoryRet.setHost("192.168.49.81");
        Connection connectionRet = factoryRet.newConnection();
        Channel channel = connectionRet.createChannel();
        channel.queueDeclare(queueNameRet, false, false, false, null);
        DeliverCallback deliverCallback = (consumerTag, delivery) -> {
            String messageRet = new String(delivery.getBody(), "UTF-8");
            System.out.println(" [x] Received '" + messageRet + "'");
            this.messageRet = messageRet;
        };
        channel.basicConsume(queueNameRet, true, deliverCallback, consumerTag -> { });
```

```
    }

    public String getMessageRet() {
        return messageRet;
    }

    public void setMessageRet(String messageRet) {
        this.messageRet = messageRet;
    }
}
```

Listing 30: Identity.java

```
package com.mycompany.serverrest;

import org.apache.cxf.endpoint.Server;
import org.apache.cxf.jaxrs.*;
import org.apache.cxf.jaxrs.lifecycle.SingletonResourceProvider;

public class ServerMain {
    public static void main(String[] args) throws Exception{
        Identity auth = new Identity();
        Catalog catalogo = new Catalog();
        auth.getToken();
        catalogo.getCatalog();
    }
}
```

Listing 31: Server.java

```
<?xml version="1.0" encoding="UTF-8"?>
<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.mycompany</groupId>
    <artifactId>serverRest</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.
    sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-frontend-jaxrs</artifactId>
            <version>3.1.7</version>
        </dependency>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-transports-http-jetty</artifactId>
            <version>3.1.7</version>
        </dependency>
```

```
        <!-- https://mvnrepository.com/artifact/com.rabbitmq/amqp-client -->
        <dependency>
            <groupId>com.rabbitmq</groupId>
            <artifactId>amqp-client</artifactId>
            <version>5.7.0</version>
        </dependency>
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20180813</version>
        </dependency>
    </dependencies>
</project>
```

Listing 32: pom.java