esprit Se former autrement	**EXAMEN** Semestre : 1 ■ 2 ☐ Session : Principale ■ Rattrapage ☐

Module: SOA
Enseignant(s) :UP_WEB
Classe(s) :4SE
Documents autorisés : OUI ☐ NON ■ Nombre de pages : 8
Calculatrice autorisée : OUI ☐ NON ■ Internet autorisée : OUI ☐ NON ■
Date : 19-03-2021 Heure : 09h Durée : 1h30

Partie I: QCU, Architecture Orientée Service « SOA » [15 pts]

1. **Quel est le principe d'un Service Web ayant la caractéristique « Stateless » :**
   A. Sauvegarde les sessions des autres services pour une utilisation ultérieure.
   B. Sauvegarde les différentes réponses dans une session côté serveur.
   C. Ne garde jamais en mémoire des informations spécifiques à un client, comme des sessions, en vu de les utiliser pour les prochaine requêtes.
   D. Aucune de ces réponses.
2. **Les Services Web doivent être :**
   A. Fortement couplés.
   B. Développés avec les mêmes technologies.
   C. Faiblement couplés.
   D. Non interopérables.
3. **Soit l'extrait suivant d'un schema XML :**

```
<xs:element name="email" type="emailType" >
  <xs:simpleType name ="emailType" >
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]*@[a-zA-Z]+.[a-zA-Z]{3}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Quel élément « email » respecte la restriction définit par la grammaire ci-dessus ?

   A. <email value= "aZ9@exemple.com" />
   B. <email> aBzYeF </email>
   C. <email> aZ9@exemple.com </email>
   D. <email> aZ9@exp.tn </email>

4. **Lequel des éléments XML ci-dessous est bien formé ?**

A.
```
<product id="123">
    <name>user1<name>
    <price>218</price>
</product>
```

B.
```
<product id=123>
    <name>user1<name>
    <price>218</price>
</Product>
```

C.
```
<product id="123">
    <name>"user1"</name>
    <price>"218"</price>
</product>
```

D.
```
<product id="123">
    <name>user1</name>
    <price>218</price>
</product>
```

5. **Soit l'objet Java suivant :**

```
@XmlRootElement
public class Product{
      @XmlAttribute
      public int id;
      public  String name;
      public float  price;
 }
```

Quel est le schéma XML généré en utilisant la commande schemagen ?

A.

```
<xs:schema xmlns:xs="……..">
      <xs:element name="product" type="prodType" />
      <xs:complexType  name="prodType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="price" type="xs:float" />
            <xs:attribute name = "id" type = "xs:integer" />
        </xs:sequence>
      </xs:complexType>
</xs:schema>
```

B.

```
<xs:schema xmlns:xs="……..">
      <xs:element name="product" type="prodType" />
      <xs:complexType  name="prodType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="price" type="xs:float" />
        </xs:sequence>
       <xs:attribute name = "id" type = "xs:integer" />
      </xs:complexType>
</xs:schema>
```

C.

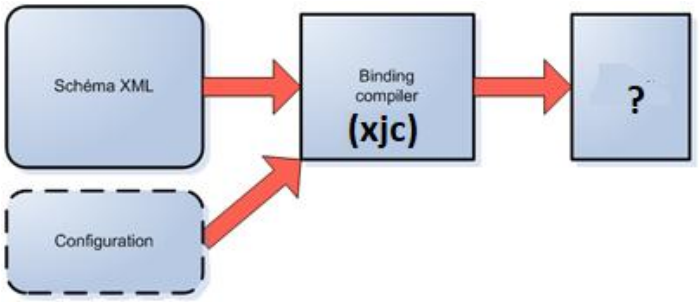
```
<xs:schema xmlns:xs="……...">
      <xs:element name="product" type="prodType" />
      <xs:simpleType  name="prodType">
      <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="name" type="xs:float" />
      </xs:sequence>
    <xs:attribute name = "id" type = "xs:integer" />
    </xs:complexType>
</xs:schema>
```

D.

```
<xs:schema xmlns:xs="……...">
      <xs:element name="product" type="prodType" />
      <xs:complexType  name="prodType">
      <xs:all>
            <xs:element name="name" type="xs:string" />
            <xs:element name="name" type="xs:float" />
      </xs:all>
    <xs:attribute name = "id" type = "xs:integer" />
    </xs:complexType>
</xs:schema>
```

6. **JAXB permet de**
   A. Mapper une ou plusieurs classes annotées vers un schéma XML.
   B. Mapper une ou plusieurs classes annotées vers un document XML.
   C. Développer un client SOAP directement à partir de classes annotées
   D. Générer un document XML directement à partir d'un schéma XML.

7. **La commande xjc présentée par la figure suivante permet de générer :**



   A. Un document XML
   B. Deux classes java : une classe annotée ainsi que la fabrique ObjectFactory.java
   C. Une seule classe java annotée
   D. Une instance d'un document XSD

Nom…… Prenom …… Classe : ……

8. **Quel élément du contrat WSDL indique le statut d'une réponse :**
   A. <types>
   B. <message>
   C. <service>
   D. Aucune réponse

9. **Quelle(s) critère(s) correspond (ent) aux services Web étendus :**
   A. Couplage fort
   B. Expose son descripteur WADL
   C. Sans état
   D. Aucune de ces réponses

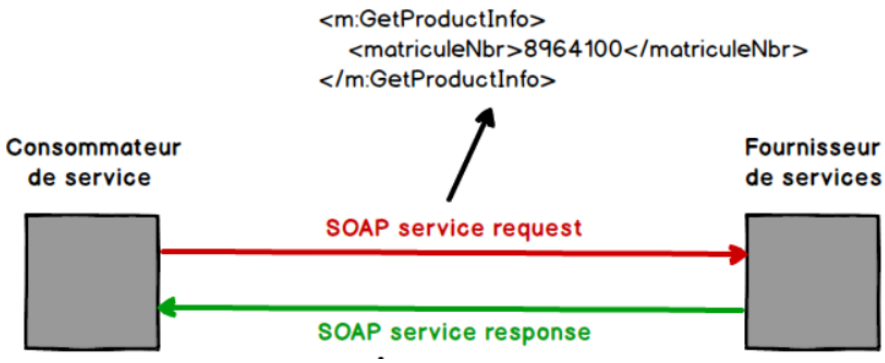
10. **Quel(s) format(s) de données utilisé(s) pour les requêtes et les réponses SOAP :**
    A. XML
    B. Text
    C. JSON
    D. Les réponses A et C

11. **Quel est l'entête à renvoyer pour préciser que le résultat retourné est en JSON :**
    A. Content-Format: application/json
    B. Content-Type: application/json
    C. Body-Format: application/json
    D. Body-Type: application/json

12. **Pour consommer une ressource REST, il faut préciser :**
    A. Le nom de l'opération qui l'implémente
    B. Le nom de l'opération qui l'implémente ainsi que les inputs
    C. Le path de la ressource ainsi que les entrées
    D. Touts les réponses sont correctes

13. **Le JWT permet de :**
    A. Sécuriser une session d'échange de messages entre le consommateur et fournisseur d'un Service Web
    B. Sécuriser le contrat WSDL d'un Service Web
    C. Crypter les données échangées entre le consommateur et le fournisseur d'un Service Web
    D. A et C

14. **Soit la figure suivante :**



Quelle réponse SOAP correspond au résultat d'exécution de à la requête ci-dessus :

A.

```
<productResponse>
        <name>pr1</name>
        <price>8.90</price>
</productResponse>
```

B.

```
 {
    "name": "pr1",
    "price": "8.90"
 }
```

C.

```
<GetProductInfoResult>
    <name>pr1</name>
    <price>8.90</price>
</GetProductInfoResult>
```

D.

```
<GetProductInfoResponse>
    <name>pr1</name>
    <price>8.90</price>
</GetProductInfoResponse>
```

15. En considérant les URLs suivants :

URL1: http://localhost:8081/Products/get/matriculeNbr=8964100

URL2: http://localhost:8081/Products/matriculeNbr=8964100

URL3: http://localhost:8081/Products/get/8964100

URL4: http://localhost:8081/Products/8964100

Quelle(s) URL(s) utiliser pour consommer une ressource REST en passant le matricule du produit via l'annotation **@PathParam** tout **en appliquant les bonnes pratiques du REST**:

A. URL1
B. URL3
C. URL2 et URL4
D. URL4

Partie II: Web Services étendus (SOAP) [5 pts]:

Ci joint une structure d'un fichier XML partagé entre deux applications A et B hétérogènes :

```
<emp id = "12345678 " >
        <adresseMail> foulen.benfoulen@esprit.tn </adresseMail>
        <isAvailable> true </isAvailable>
        <nom> Ben foulen </nom>
        </prenom> Foulen </prenom>
        <age> 45 </age>
</emp>
```

a- Annotez la classe Employe.java pour obtenir exactement la même structure XML ci-dessus : (2pts)

```
[1].....
[2].....
[3].....
public class Employe {

        [4].....
        private String cin;
        [5].....
        private String name;
        [6].....
        private String lastName;
        [7].....
        private int age ;
        [8].....
        private String mail;
        [9].....
        private boolean isAvailable;
        [10].....
        private RoleEmploye role;
}
```

Ci joint un contrat WSDL décrivant un service web de gestion d'employés :

```
<definitions xmlns:wsp="http://www.w3.org/ns/ws-policy"  xmlns:tns="http://greeting.tn"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://gestionEmploye.tn" name="GestionemployeService">
   <types>
      <xsd:schema>
         <xsd:import namespace="http://gestionEmploye.tn"
         schemaLocation="http://localhost:8080/GestionEmployeSW/employe?xsd=1"/>
      </xsd:schema>
   </types>
   <message name="afficherEmploye">
   </message>
   <message name="affichage">
   </message>
   <message name="ajouterEmployee">
   </message>
   <message name="ajout">
   </message>
   <portType name="gestionEmployePortType">
  </portType>
   <binding name="gestionEmployePortBinding" type="tns:gestionEmployePortType">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
      <operation name="afficherEmploye">
         <soap:operation soapAction=""/>
         <input>
            <soap:body use="literal"/>
         </input>
         <output>
            <soap:body use="literal"/>
         </output>
      </operation>
      <operation name="ajouterEmployee">
         <soap:operation soapAction=""/>
         <input>
            <soap:body use="literal"/>
         </input>
         <output>
            <soap:body use="literal"/>
         </output>
      </operation>
   </binding>
   <service name="GestionemployeService">
      <port name="gestionEmployePort" binding="tns:gestionEmployePortBinding">
<soap:address location="http://localhost:8080/GestionEmployeSW/employe"/>
      </port>
   </service>
</definitions>
```

b- **Complétez la classe du Web Service etendu pour avoir le contrat WSDL décrit ci-dessus : (3 pts)**

```
[1] ……..
        ……
                ……...

public class GestionEmploye {

        [2] …….

        [3] …….
        public String ajoutEmploye(Employe e) {
                //implémentation
        }

       [4] …….

       [5] ……..

        public String affichageEmploye() {
                // implémentation
        }

}
```