|  |  |
| --- | --- |
|  |  |
|  | C:\Users\hamadi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\logo_inti_vector-04.jpg  Institut des Nouvelles Technologies  de l’Information |

QCM

Rest JavaScript

Session 2016

## Guide d'utilisation

* Notez qu'en fonction de la question, il est possible de cocher une ou plusieurs réponses. Ce sera à vous de déterminer le nombre de bonnes réponses possibles.

## Quelques conseils de mise en œuvre

* Ne restez pas bloqué sur une question pour laquelle vous n'avez pas de réponse. Passez à la question suivante. Vous reviendrez par la suite sur les questions sans réponse.
* Identifiez les questions auxquelles vous avez répondu sans certitude. Que ces réponses soient justes ou non, cela vous permettra de mémoriser les questions sur lesquelles vous devrez revenir dans la correction du TEST.
* Ne cherchez pas à aller le plus vite possible. Le temps imparti doit être utilisé au maximum pour vous permettre d'effectuer une synthèse des points abordés dans le module.

**QCM**

**1. Quel(s) protocole(s) utilise(nt) les services *Rest* ?**

* HTTP
* HTTP et JMS
* JMS
* HTTP et WSDL

**2. Comment ajouter des arguments à une méthode *Get* ?**

* Ajouter @PathParam( value = “nomArg”) devant l’argument de la méthode
* Ajouter {nomArg} dans le chemin d’accès de la méthode
* Les deux
* Rien de particulier, on y accède via les request parameters

**3. Quels types de contenu peuvent produire une méthode *Rest* ?**

* Du XML et du JSON
* Du texte et du html
* Des images ou des fichiers
* Tous ces formats, et d’autres

**4. Quelle annotation permet de définir le type de retour d’une méthode *Rest* ?**

* @Produces
* @Return
* @MediaType
* @Consumes

**5. Quelle annotation permet de définir le type du contenu de la request attendu ?**

* @Produces
* @MediaType
* @Consumes
* @ContentType

**6. Que doivent indiquer obligatoirement les annotations d’une méthode *Rest* ?**

* Le type de requête (GET, POST, etc.)
* Le type de retour
* Le type d’entrée
* Les trois

**7. En considérant l’URL “http://exemple.com/rest/demande/go”, quelles annotations ont dû être utilisées dans l’application exemple ?**

* @Path(“/rest”) sur la classe et @Path(“/demande/{argument}”) sur la méthode
* @ApplicationPath(“/rest”) sur la classe de configuration, @Path(“/demande”) sur la classe et @Path(“/go”) sur la méthode
* @ApplicationPath(“rest”) sur la classe de configuration, @ClassPath(“demande”) sur la classe et @MethodPath(“go”) sur la méthode
* @Path(“rest”) sur la classe de configuration et @Path(“demande”) sur la classe et @Path(“go”)

**8. De quoi a besoin un client pour utiliser les méthodes d’un service *Rest* ?**

* Des URL du service
* Des classes proxy générées
* Des librairies JAX­RS
* Des trois

**9. Si une méthode d’un service *Rest* attend une entité comme argument, comment JAX­RS le gèrera-t-il ?**

* On ne peut pas passer d’entité en argument d’une méthode Rest
* En parsant l’argument sérialisé et en faisant correspondre aux setters de l’entité les attributs de l’objet de la requête et en mettant null si un argument est manquant
* En parsant l’argument sérialisé et en faisant correspondre aux setters de l’entité les attributs de l’objet de la requête et en envoyant une exception si un argument est manquant
* En parsant l’argument sérialisé avec JAXB, en utilisant un objectfactory et un unmarshaller liés à un xsd

**10. Comment représenter en JSON un objet ayant un attribut1 en chaîne de caractères, un attribut2 en valeur numérique et un attribut3 en array ?**

* {attribut1: ‘valeur’, attribut2: 2, attribut3: [] }
* [“attribut1”: “valeur”, “attribut2”:”2”, “attribut3”: {}]
* {“attribut1”: “valeur”, “attribut2”: 2, “attribut3”: Array() }
* {“attribut1”:”valeur”, “attribut2”:2, “attribut3”:[]}

**11. Quel langage définit les standards du *Javascript* ?**

* ActionScript
* LiveScript
* ECMAScript
* TypeScript

**12. En *Javascript*, comment transformer un objet en JSON ?**

* objet.json()
* JSON.stringify(objet)
* JSON(objet)
* objet.toJSON()

**13. En *Javascript*, comment récupérer un élément DOM par son id ?**

* document.getById(‘id’)
* document.getElementsById(‘id’)[0]
* document.getElementById(‘#id’)
* document.getElementById(‘id’)

**14. En *Javascript*, comment créer un élément DOM de type paragraphe ?**

* document.createParagraphe()
* document.create(‘p’)
* document.createElement(‘p’)
* document.newElement(‘p’)

**15. En *Javascript*, comment itérer sur les noms de propriétés d’un objet ?**

* forEach(var clef in objet)
* for(var clef : objet)
* objet.forEach()
* for(var clef in objet)

**16. Comment s’appelle l’objet *Javascript* permettant de faire de l’Ajax ?**

* Ajax
* XMLHttpRequest
* $http
* doAjax

**17. Comment rajouter un événement au click sur un élément p en *Javascript* ?**

* elementP.onclick = function(){}
* elementP.addEventListener(‘click’, function() {})
* <p onclick=”fonction()” ></p>
* Les trois

**18. Quel est le substitut aux classes en *Javascript* ?**

* Le prototypage
* Les JavaSclass
* Le JSON
* Il n’y en a pas

**19. Comment rendre possible l’instanciation d’un objet Truc via new Truc() ?**

* class Truc();
* var Truc = function() { }
* var Truc = {};
* Truc = Truc();

**20. Comment accéder aux attributs publics de Truc dans celui-ci ?**

* this.attribut
* that.attribut
* attribut
* Truc.attribut

FIN