Exercice : JAX-RS

On se propose de mettre en place un service web RESTful pour la gestion d'articles publiés sur le web. Le service web fournit plusieurs fonctionnalités.

Ajouter d'un article

Requête	**POST**	ws/publication
	Header	Content-type : application/xml
	Body	<article Ref="AB123"> <Titre>Article</Titre> <Date>15/02/2021</Date> <Auteur>Patrick Senior</Auteur> <Section> <Identifiant>102</Identifiant> <Description> Pensez à votre audience. Un artiste…</ Description > <Status>Publié</Status> </Section> </article>
Réponse	**Header**	Content-type : text/plain
	Status	201 Created
	Body	Article ajouté

Afficher la liste de tous les articles

Requête	**GET**	ws/publication
Réponse	**Header**	Content-type : application/json
	Status	200 Ok

	Body	[
		{
		"Ref": "AB123",
		"Titre": "Article",
		"Date": "15/02/2021",
		"Auteur": "Patrick Senior",
		"Section": {
		"Identifiant": "102",
		"Description ": " Pensez à votre audience. Un artiste…",
		"Status": " Publié"
		}

```
    },
    {
      "Ref": "VF584",
      "Titre": "Audience",
      "Date": "05/03/2020",
      "Auteur": "Biatrice Allaume",
      "Section": {
        "Identifiant": "255",
        "Description ": "Quand on parle d'audience on doit forcément faire référence…",
        "Status": Publié "
      }
    }
  ]
```


Travail demandé :

1. **Développez le service web RESTful décrit ci-dessus en complétant les parties manquantes.**
NB : Veuillez ignorer la logique métier.

a. Compléter la classe RestActivator

```
@.................................[1]
public class RestActivator extends Application{
      public RestActivator() {
            // TODO Auto-generated constructor stub}
}
```

[1] @ApplicaitonPath("ws")

b. Compléter les classes des entités Article et section

```
@.................................................. [1]
public class Article {

      private String Ref;
      private String Titre;
      private Date Date;
      private String Auteur;
      private Section section;

     @.................................................. [2]
      public String getRef() {         return Ref;  }
      public void setRef(String ref) {       Ref = ref;   }
      public String getTitre() {nreturn Titre;      }
      public void setTitre(String titre) {   Titre = titre;      }
      public Date getDate() {          return Date; }
      public void setDate(Date date) {       Date = date; }
      public String getAuteur() {            return Auteur;      }
      public void setAuteur(String auteur) {       Auteur = auteur;    }
      public Section getSection() {          return section;     }
      public void setSection(Section section) {          this.section = section;   }

}
```

```
@.......................................... [3]
public class Section {
	public String Ident;
	public String Contenu;
	public boolean Status ;

	@.......................................... [4]
	public String getIdent()   {     return Ident;}
	@.......................................... [5]
	public String getContenu() {     return  Contenu;    }
	@.......................................... [6]
	public boolean getStatus() {     return Status;      }
```

[1]	
[2]	
[3]	
[4]	
[5]	
[6]	

2. Soit la classe Publication ci-dessous, Supposant que nous allons ajouter la methode « `chercherDetailsArticle` » qui permet d'afficher les détails d'un article publié sur le web tel que décrit ci-dessous

```
@Path("publication")
public class Publication {
	Public static List<Article> listeArticles = new ArrayList<Article>();

	Public Publication(){}
//.................
}
```

```
@GET
@Produces("application/xml")
public Response chercherDetailsArticle(
                @QueryParam(value="ref")String id)
{
…
return Repsonse.status(Status.OK).entity(article).build();
}
```

Donnez la requête HTTP permattant de consommer le service ainsi que la réponse HTTP retournée en complétant les tableaux ci-dessous avec des exemples concrets.

Requête	**Type de requête**	**GET**
	URL	ws/publication?ref=AB123
	Header	-
	Body	-
Réponse	**Header**	application/xml
	Status	200
	Body	<article Ref="AB123"> <Titre>Article</Titre> <Date>15/02/2021</Date> <Auteur>Patrick Senior</Auteur> <Section> <Identifiant>102</Identifiant> <Description> Pensez à votre audience. Un artiste…</ Description > <Status>Publié</Status> </Section> </article>

3. Supposant que nous allons ajouter une autre methode qui permet de modifier une section dans un article tel que décrit ci-dessous:

```
@PUT
@Path("{ident}")
@Consumes("application/xml")

public Response ModifierSection(@PathParam("ident") String id)
{
…
return Repsonse.status(Status.OK).entity("Article modifié").build();
}
```

Requête	**Type de requête**	**PUT**
	URL	ws/publication/102
	Header	application/xml
	Body	<Section> <Identifiant>102</Identifiant> <Description> Pensez à votre audience. Un artiste…</ Description >

		<Status>test</Status> </Section>
Réponse	**Header**	text
	Status	200
	Body	`Article modifié`