***Web Service:*** des composant web permettant à une application d’appeler des fonctionnalités distantes indépendamment de leur plateforme d’exécution et de leur langage d’implémentation, basé sur le protocoles http et respect un format précis tel que XML ou JSON.

***CORBA :*** premier api permettant la communication entre des applications hétérogènes (multiplateformes multi-langages).

Pour communiquer les applications distribuées il y’a 2 modèle synchrone ou asynchrone :

***Modèle synchrone :*** est une communication bloquante ou le client reste bloqué tant qu’il n’a pas reçu de réponse du serveur. (RMI, CORBA, http, SOAP, REST).

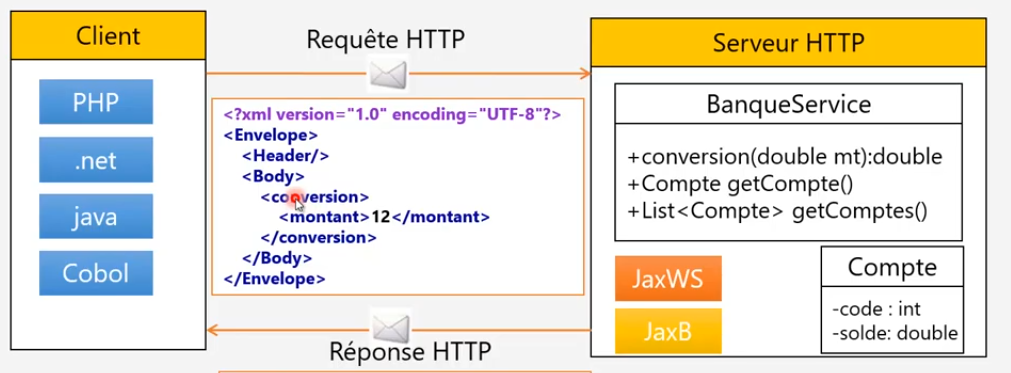
**Modèle asynchrone** : est une communication non bloquante, se fait à l’aide d’un outil intermédiaire appelé broker qui conserve le message jusqu’à ce que le récepteur soit disponible.

***Brokers :*** permettant la communication entre les applications distribuées de manière asynchrone (Rabbit Mq, JMS 🡪 Java Message Serice).

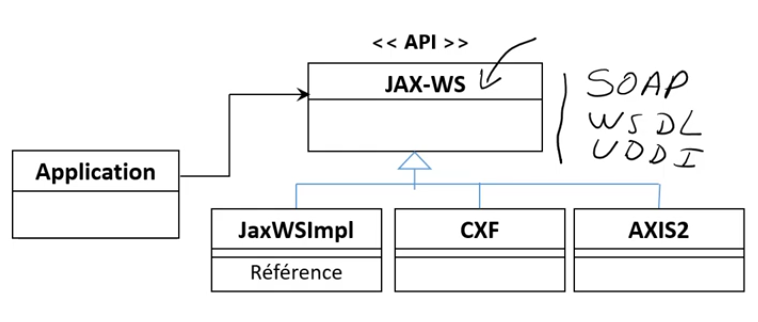
***Les Web Services basé sur le protocole SOAP (Simple Object Access Protocole):***

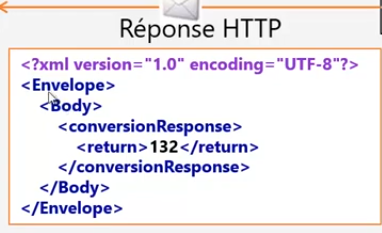
Est un protocole qui pour l’envoie des requêtes se base sur le protocole http et le format XML.

Client veut exécuter un service distant situant sur le serveur, donc il doit envoyer une requête http dont le corps contient un fichier XML qui indique la méthode voulue et ses paramètres



Le serveur pour répondre à cette requête se base sur une Api (JaxWs le cas du langage java) qui comprend facilement le format XML, ensuite elle exécute la méthode demandée dont le résultat sera après parser en format XML, puis transporter vers le client.

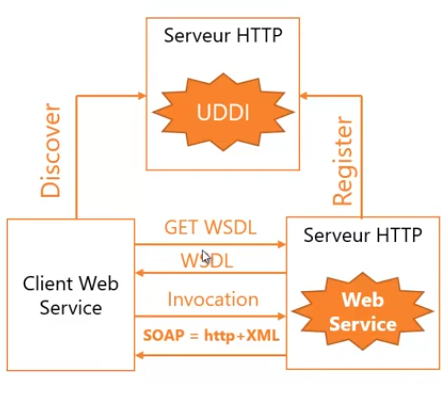




***WSDL (Web Service Description Language):*** contrat d’échange de donné qui fournit la description de Web Service au format XMl en précisant les méthodes pouvant être invoquer, ainsi que leur signature et le point d’accès au web services (URL, port, etc).

***UDDI (Universal Description Discovery and Integretion)*** : pour qu’un client puisse consomme un web service, il a besoin de savoir son adresse, et pour y avoir il peut la demander auprès d’un annuaire s’appelle UDDI (lui aussi est un Web Service), qui est un outil de publication des Web Services.

1. Server fait son publication sur un annuaire UDDI afin d’être repérer par les clients.
2. Le client récupère l’adresse de Web Service à partir de l’annuaire.
3. Le client demande le WSDL à partir de serveur pour savoir les méthodes à invoquer.
4. Le client dispose de WSDL alors il lui reste que d’appeler la méthode voulue en envoyant une requête SOAP au serveur.



***SOAP application sous Java :( Simple java project jdk8) :***

Pour crée un web service SOAP on a besoin 2 classes, une classe simple annoter pour être consommée par distant, et une classe Serveur pour déployer la 1ère classe.

Classe de web Service :



JaxB : une api java permettant la sérialisation et desérialisation d’un objet Java vers le format XML.

Pour développer les web service sous java il faut utiliser l’API JAX-WS qui fournit un ensemble d’annotation (permettant aussi la génération de WSDL) qui s’applique sur les classes choisies pour être consommer par le web service.

@WebService( serviceName = ‘’BanqueWs’’) : pour dire que la classe est un web service, et qu’il va être consommé avec le nom BanqueWs.

@WebMethod(operationName = ‘’Conversion’’) : utiliser sur une méthode pour qu’il fasse partir de web service.

@WebParam(name = ‘’mantant’’): pour indiquer les noms des paramètres d’une méthode de web service, si les paramètres ne sont pas annoter, par défaut jaxWs leur donnera un nom arg1, arg2, … ce qui est un comportement indésirable.

Le JaxWs se base sur ses annotations pour générer le fichier WSDL.

Pour déployer le Web service on crée une classe Serveur JaxWs:



Le 0.0.0.0 pour que le Web Service soit accessible par toute les interface réseau de la machine.

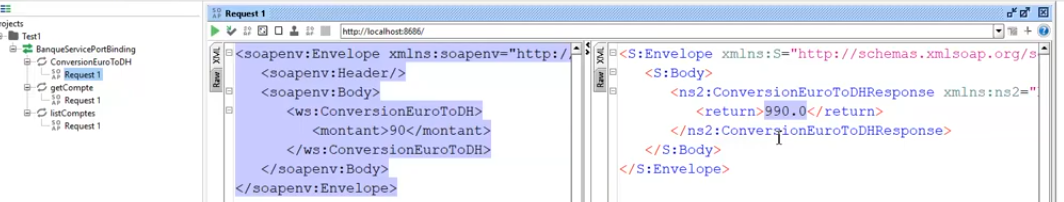
EndPoint.publish utilisé pour publier le web service, demande 2 paramètres, la classe de web service, et l’adresse sur lequel elle est déployée.

**Consommation de Web Service :**

<http://localhost:8686/BanqueWS?wsdl> 🡪 Lien de récupération de WSDL á partir de serveur.

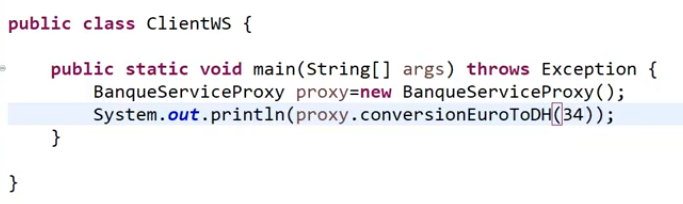
SoapUi : est un outil simple pour tester le Web Service, permet d’envoyer et de recevoir des requêtes et des réponses auprès de serveur.

Il suffit de lui introduire le lien WSDL du serveur, SoapUi ainsi charge automatiquement les méthodes disponibles sur le Web service.



Tester SOAP sous Java :

Il faut ajouter un **Web service client** en lui donnant le lient WSDL, ce qui va générer un package ou il y’a des classes et des interfaces généralement c’est un proxy, ensuite on crée une classe client avec la méthode main dans laquelle on instance l’objet proxy auquel on fait des appels aux méthodes distantes.



Problème des Web Service basé sur SOAB c’est qu’il utilise XML qui est un langage verbeux qui prend beaucoup en bande passante juste pour envoyer une simple information, et également XML prend beaucoup plus de temp pour le parser, et que pour consommer un WS basé sur SOAP, le WSDL est indispensable.

Web Service basé sur le protocole REST(Representation State Transfer) :

REST utilise les spécifications originales du protocole http, plutôt que de réinventer une surcouche (XML)

SOAP peut utiliser d’autre protocole que http mais REST utilise que http .

Les 5 règles de RESTful :

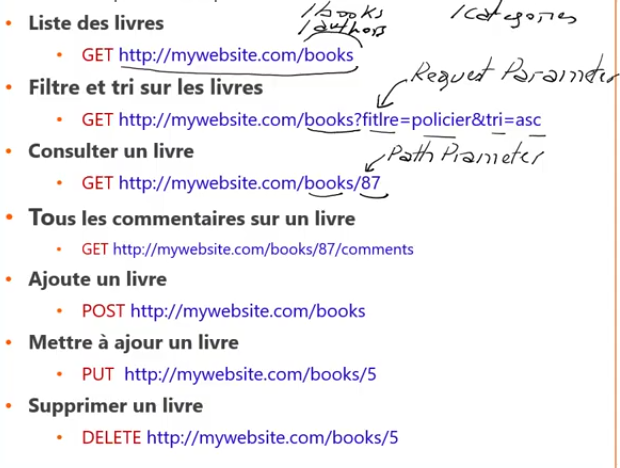
1. l’URI comme identifiant de la source.
2. Les méthodes http comme identifiant des opérations, on peut avoir différentes opérations avec le même URI, <http://monSite.com/books/5>, avec GET c’est consulter, PUT pour mis á jour, DELETE pour supprimer, et POST pour ajouter le book qui a 5 comme id.
3. Réponses http en différent format (XML, JSON, …) en fonction de la demande du client qui peut être spécifier au header accept de la requête http.
4. Les liens comme relations entre les ressources, produit appartient á une catégorie, alors quand vous récupérer un produit, un lien <a> propose la consultation de catégorie de ce produit.
5. Envoyer un jeton d’authentification diffèrent pour chaque requêtte, token contient les informations de session du client (username, rôle, droit d’accès). Raison de sécurité.

RESTful c’est lorsqu’on respect tous les règles.

REST c’est généralement les 3 premier règles.

RESTful n’est pas un protocole, mais c’est de http plus le respect de 5 règles.

MIS EN OUVRE DE RESTFUL :



Serveur d’application JEE, glassfish, jboss(wildfly open source, le plus performant), WebSphere (payant), lorsqu’on démarre un serveur d’application, en effet on démarre plusieurs serveurs (web container tomcat,… ) , un serveur d’application doit fournir une implémentation pour chaque spécification JEE, JPA 🡪 HIBERNAT, JAXWS 🡪 JAXWSImpl … .

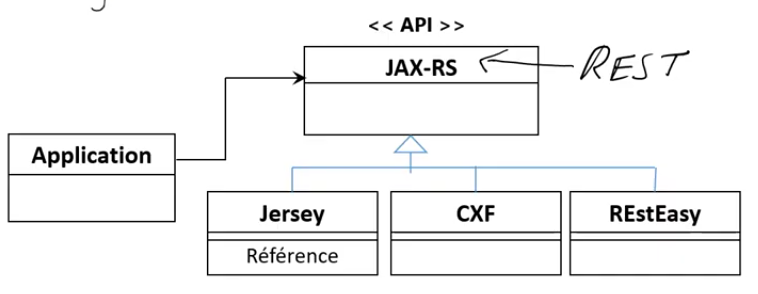
Pour démarrer un serveur d’application comme WebSphere c’est trop lourd, car il démarre tous les implémentations des API existe et il ne donne pas le choix des implémentations, par contre Spring démarre que les API en cas d’utilisation, et en peut utiliser l’implémentation qui nous chante.

Serveur d’application : un ensemble de conteneurs, démarre le Framework qui fait l’invertion de contrôle, ainsi que de démarrer tous les implémentations des spécifications JEE 🡪 conteneur lourd.

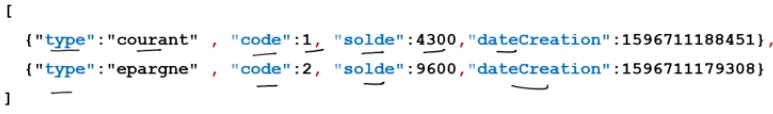
Pour démarrer le serveur web dans spring, il suffit de démarrer que springIoc + tomcat , du coup spring est dite conteneur léger.

SpringIOC : Framework spring qui fait l’injection de dépendance

Pour utiliser RESTful sous Java il faut utiliser une API JAX-RS :



JSON(Java Object Notation) un format léger d’échange de données, indépendant de tous langages, simple á lire pour l’humain, facile á traduire par les langage de programmation, objet JSON ressemble beaucoup á l’objet javaScript, sauf que les attributs sont délimiter par des double quottes.

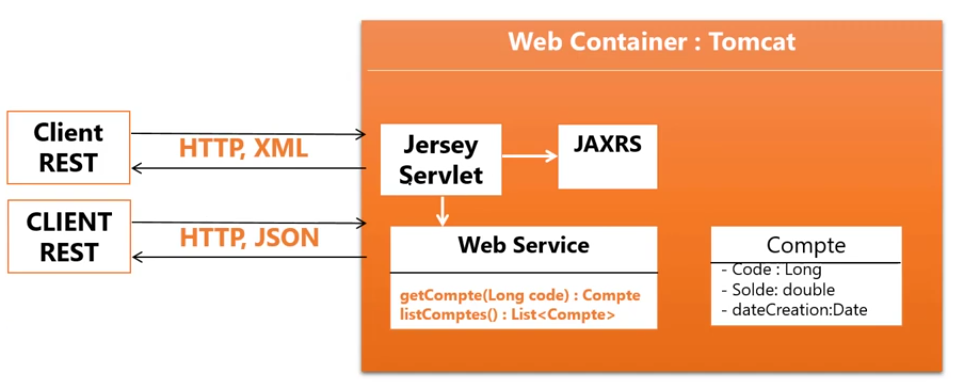


XML fournit un schéma sur la ressource envoyée du coup les données sont bien structurés et prête de les enregistrer directement dans la BD, par contre JSON il se peut que les donnés ne soient pas valide, du coup il faut les validées d’abord.

RESTfull sous java (client-serveur) :

Client envoie une requête http/json ou bien http/xml á un serveur web avec tomcat, alors que tomcat travaille qu’avec des servlets ce qui fait que JAXRS á une implémentations Jersey Servlet qui reçoit c’est requête http, il la traite pour savoir le service demandé dont on envoie son résultat dans une réponse http .

Si le client envoie une requête http/json il reçoit une réponse http/json, si la requête de type http/xml.





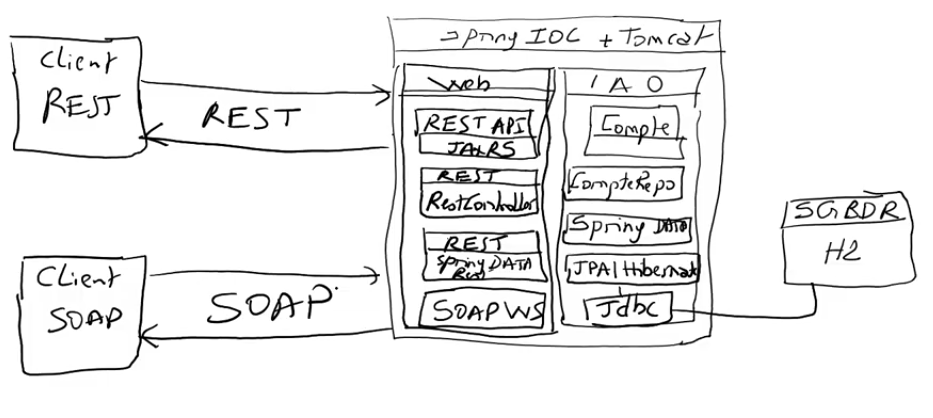
Un web service avec JaxRs est une classe annoter par @Path(‘’nomWebService’’ ) ;

Pour qu’une méthode fasse partie du web service on utilise aussi l’annotation @Path(‘’nomDeMethode’’) @GET (ou @POST, @PUT, @DELETE) pour spécifier la méthode http, @Produce({mediaType.APPLICATION\_JSON, mediaType.APPLICATION\_XML) pour répondre soit avec xml soit json, selon le demande de client.

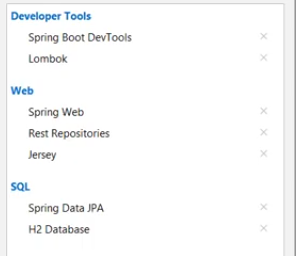
Jackson est une librairie qui fait le mapping objet/json

**Web Services sous Spring :**

Sous spring il y’a 3 façon d’utiliser les web service basé sur REST : JAXRX, RESTCONTROLLER, SPRING DATA REST

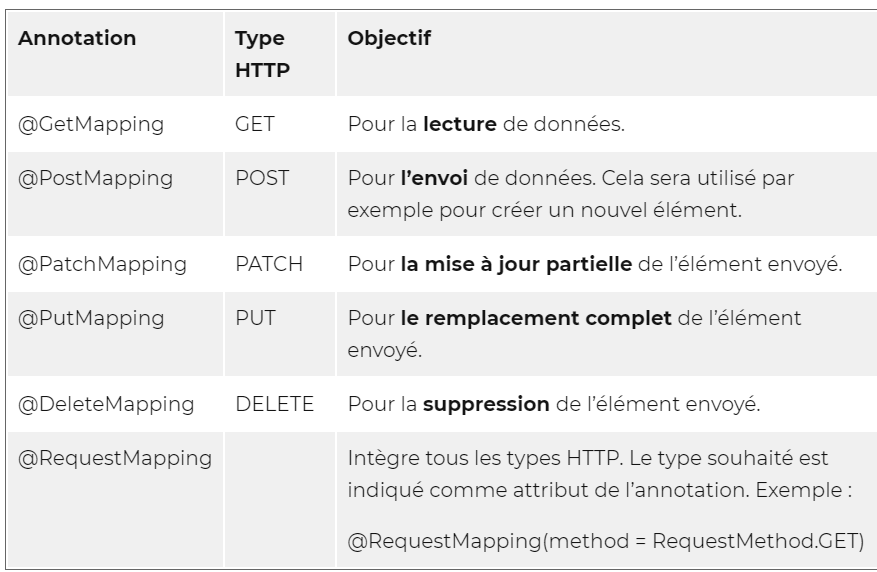


Les dépendances nécessaires pour créer un web service sous Spring :



RestController sous spring :

La classe doit être annoter par @RestController, les méthodes annoter par @GetMapping, @PostMapping, @PatchMapping, @PutMapping, @DeleteMapping, @RequestMapping.



@RequestBody : signifie que ce paramètre existe dans le corps de la requête.

@PathVariable : signifie que ce paramètre est présent dans l’URL de requête