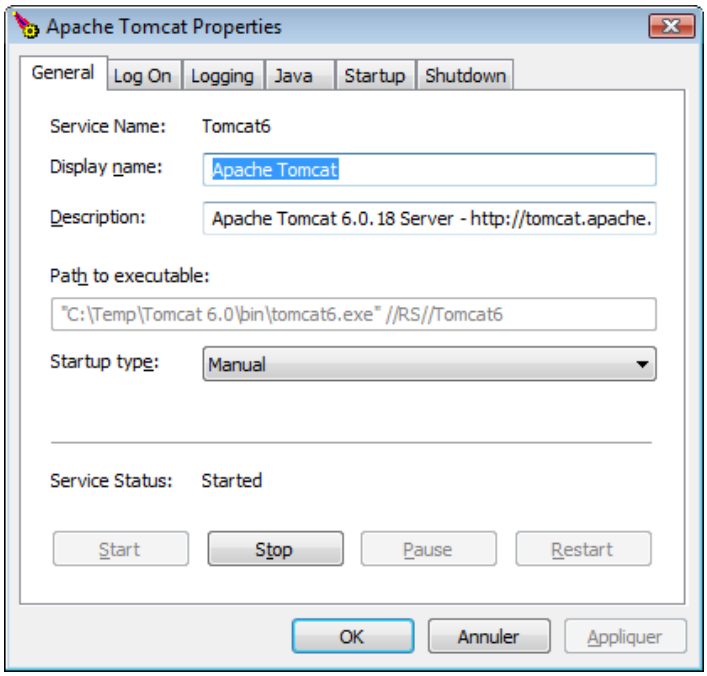
Applications Réparties TD 4
Web Services en Java avec Axis

1 Lancement et Configuration d'Axis

Le serveur de déploiement « Axis » est installé comme une application Web au sein du moteur de servlets et de JSPs de Apache Tomcat.

1.1 Démarrage de Tomcat

Vous devez, pour accéder aux possibilités d'Axis, activer préalablement le serveur *Apache Tomcat*. On installera, si ça n'est déjà fait, la dernière version Tomcat 6.0 depuis le site http://tomcat.apache.org/download-60.cgi.
Une fois Tomcat installé, vous trouverez l'utilitaire Apache Service Manager dans la barre de tâche.



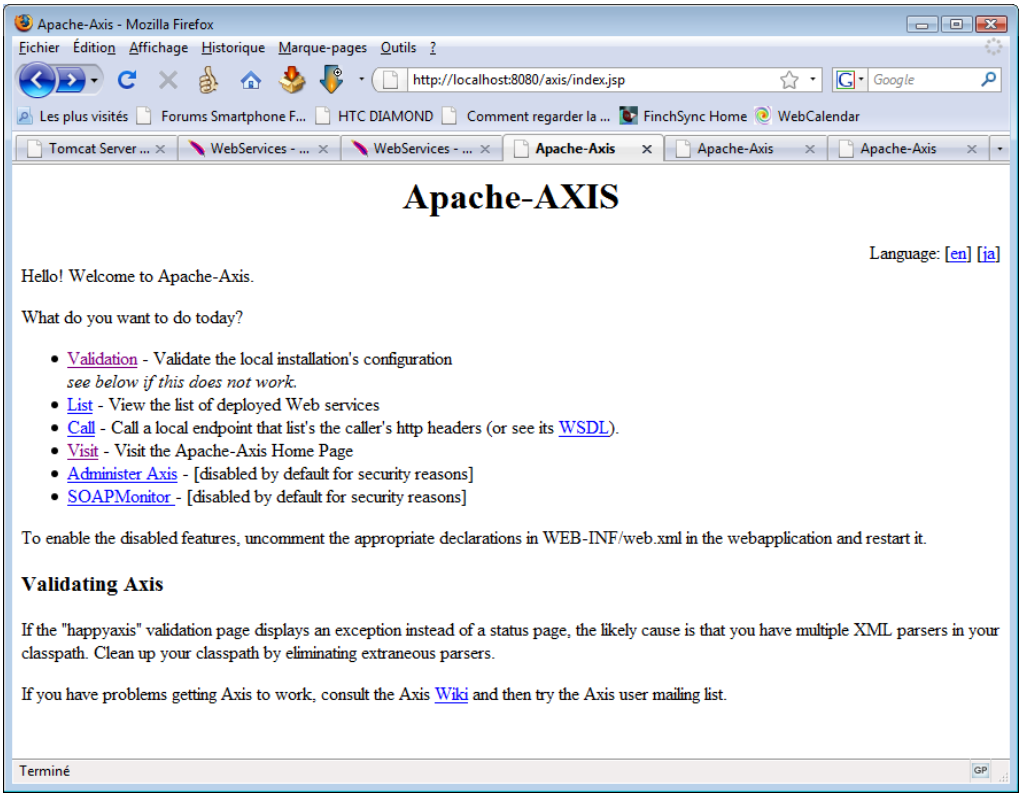
C'est à partir de l'onglet General que vous activerez/désactiverez le serveur Tomcat.
Testez l'activation du serveur avec sur http://localhost:8080/.

1.2 Démarrage d'Axis

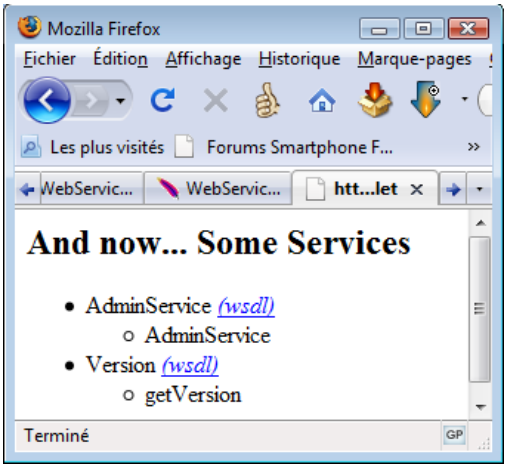
Axis (Apache eXtensible Interaction System) est un projet open-source du groupe Apache. Son but est de proposer un ensemble d'outils pour faciliter le développement, le déploiement et l'utilisation des services web écrits en java. Axis propose de simplifier au maximum les taches pour la création et l'utilisation des services web. Il permet notamment de générer automatiquement le fichier WSDL à partir d'une classe java et le code nécessaire à l'appel du service web.
Pour son utilisation, Axis nécessite un J.D.K. 1.3 minimum et un conteneur de servlet (nous utiliserons ici Tomcat 6.0 déjà installé). L'installation d'Axis est facile. Il faut télécharger la dernière version sur le site du groupe Apache : http://ws.apache.org/axis/
La version utilisée dans cette section est la 1.4. Il faut dézipper le fichier dans un répertoire (par exemple dans c:\java). Il faut ensuite copier le répertoire axis de l'arborescence dézippée dans le répertoire webapps de Tomcat et lancer ou relancer Tomcat.

Applications Réparties TD 4
Web Services en Java avec Axis

Pour tester si l'installation s'est déroulée correctement, il suffit de saisir l'url http://localhost:8080/axis/index.jsp dans un browser



Un clic sur le lien « Validation » permet d'exécuter une JSP qui fait un état des lieux de la configuration du conteneur et des API nécessaires et optionnelles accessibles.
Un clic sur le lien « List » permet de voir quels sont les services web qui sont installés.



Pour plus d'informations sur l'installation ou en cas de problème, il suffit de consulter la page correspondante sur le site d'Axis : http://ws.apache.org/axis/
Axis propose deux méthodes pour développer et déployer un service web :

Applications Réparties TD 4
Web Services en Java avec Axis

- le déploiement automatique d'une classe java
- l'utilisation d'un fichier WSDD

2 Création et test d'un Service Web sous Axis

2.1 Création simple d'un Service Web sous Axis

La première étape consiste à définir la classe du service web qui retournera au client la somme de 2 entiers.
La définition de cette classe est la suivante :

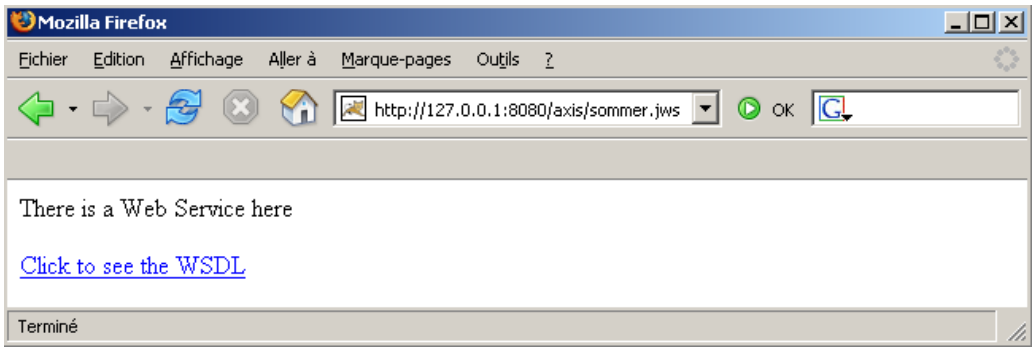
```
public class sommer {
    public int getsomme(int a, int b) {
        return a+b;
    }
}
```

Utilisez un éditeur de texte simple, type Bloc-notes sous Windows. Attention, vous devez sauvegardez votre classe sous le fichier portant le même nom de la classe et suffixé par « jws ».

La deuxième étape consiste à déployer le service au sein du fournisseur de services web Axis. Le premier mode de déploiement sous Axis que nous allons mettre en œuvre est le plus simple qui soit. Il s'agit du déploiement instantané par fichier « jws » . Vous noterez qu'il n'est même pas nécessaire de compiler le fichier source java.

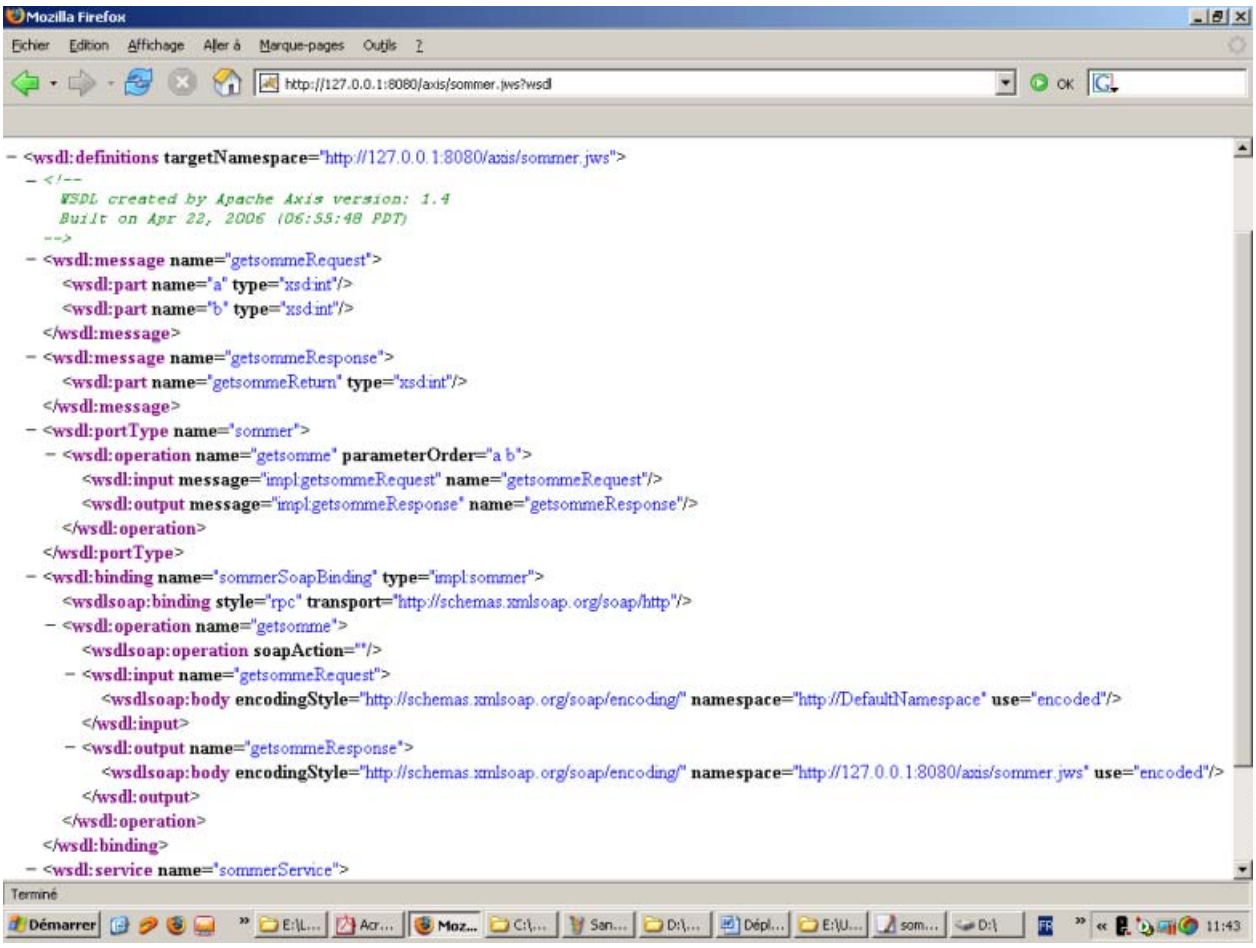
Pour réaliser le déploiement, il suffit de copier le fichier *sommer.jws* dans le domaine applicatif d'Axis. Le domaine d'application d'Axis est : *<dossier de Tomcat>/webapps/axis*, vous devez donc copier le fichier « jws » dans ce dossier.

Vous êtes désormais en mesure d'accéder à votre service à l'URL suivante : http://localhost:8080/axis/sommer.jws. Vous devez alors constater que votre service a bien été déployé sur Axis en ayant en retour la page « html » suivante :



Si vous cliquez sur ce dernier lien, vous verrez la définition « WSDL » (générée automatiquement par Axis) de votre service web.

Applications Réparties TD 4
Web Services en Java avec Axis



2.2 L'utilisation d'un fichier WSDD

Cette solution est un peu moins facile à mettre en œuvre mais elle permet d'avoir un meilleur contrôle sur le déploiement du service web.

Il faut écrire la classe java qui va contenir les traitements proposés par le service web.

```
public class soustraire {
    public int getsoustraction(int a, int b) {
        return a-b;
    }
}
```

Il faut compiler cette classe et mettre le fichier .class dans le répertoire WEB-INF/classes de *<dossier de Tomcat>*/ /webapps/axis.

Il faut créer le fichier WSDD qui va contenir la description du service web (deploysoustraire.wsdd) :

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
            xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="soustraire" provider="java:RPC">
    <parameter name="className" value="soustraire"/>
    <parameter name="allowedMethods" value="*"/>
  </service>
</deployment>
```

Applications Réparties TD 4
Web Services en Java avec Axis

Expliquons les différents éléments :

Balise de début avec comme attributs les espaces de noms des balises mises en œuvre dans un descripteur de déploiement.	*<deployment xmlns="http://xml.apache.org/axis/wsdd/" xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">*
Le nom d'invocation du service, avec le mode, ici "RPC". Ce sera le principal mode que nous mettrons en œuvre.	*<service name="soustraire" provider="java:RPC">*
Ensuite, la classe compilée associée au service	*<parameter name="className" value="soustraire"/>*
Ensuite, les autorisations d'accès au service (ici toutes les méthodes)	*<parameter name="allowedMethods" value="*"/>*
Fin de la balise "service"	*</service>*
Fin de la balise "deployment"	*</deployment>*

Il faut ensuite déployer le service web en utilisant l'application AdminClient fournie par Axis.

```
java org.apache.axis.client.AdminClient deploysoustraire.wsdd
```

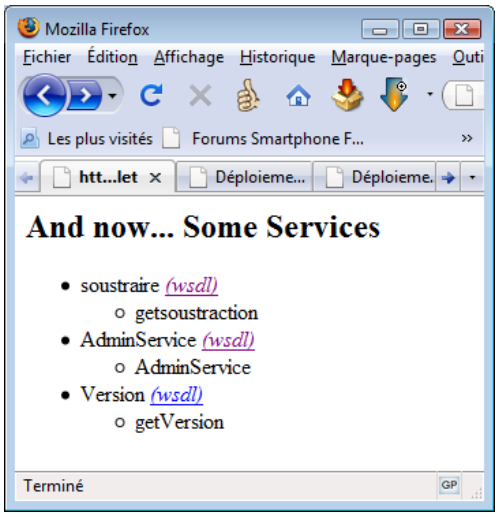
Si les librairies Axis ne sont pas spécifiées dans le classpath, vous devez le faire comme suit dans un fichier "classpath.bat" comme suit :

```
set CATALINA_HOME=C:\temp\tomcat50-jwsdp
set AXIS_HOME=C:\temp\tomcat50-jwsdp\webapps\axis\WEB-INF
set CLASSPATH=%CLASSPATH%;%CATALINA_HOME%\common\lib\activation.jar
set CLASSPATH=%CLASSPATH%;%CATALINA_HOME%\common\lib\mail.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\axis.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\axis-ant.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\jaxrpc.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\wsdl4j-1.5.1.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\commons-discovery-0.2.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\commons-logging-1.0.4.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\saaj.jar
set CLASSPATH=%CLASSPATH%;%AXIS_HOME%\lib\log4j-1.2.8.jar
set CLASSPATH=%CLASSPATH%;%CATALINA_HOME%\common\lib\xerces.jar
set CLASSPATH=%CLASSPATH%;%CATALINA_HOME%\common\lib\servlet-api.jar
set CLASSPATH=%CLASSPATH%;%CATALINA_HOME%\common\lib\naming-factory.jar
```

Attention :

- Vérifier pour chacun des .jar de classpath.bat leur nom et leur version
- Il vous faut positionner les variables d'environnements CATALINA_HOME (répertoire de base de Tomcat) et AXIS_HOME (répertoire de base d'axis) avant le lancement de classpath.bat.

Applications Réparties TD 4
Web Services en Java avec Axis



2.3 L'utilisation d'un Service Web par un Client

Pour faciliter l'utilisation d'un service web, Axis propose l'outil WSDL2Java qui génère automatiquement à partir d'un document WSDL des classes qui encapsulent l'appel à un service web. Grace à ces classes, l'appel d'un service web par un client ne nécessite que quelques lignes de code.

L'utilisation de l'outil WSDL2Java nécessite une url vers le document WSDL qui décrit le service web. Il génère à partir de ce fichier plusieurs classes dans le package localhost. Ces classes sont utilisées dans le client pour appeler le service web.

```
java org.apache.axis.wsdl.WSDL2Java http://localhost:8080/axis/services/soustraire?wsdl
```

Il faut utiliser les classes générées pour appeler le service web.

Voici le squelette de SoustraireClient.java, complétez-le, compilez-le et exécutez-le.

```
import localhost.axis.services.soustraire.Soustraire;
import localhost.axis.services.soustraire.*;

public class SoustraireClient{

  public static void main(String[] args) throws Exception{

// Création du service depuis le endpoint, par défaut localhost port 8080
// soustraireService correspond au nom du service dans le fichier “wsdl”
// c’est la balise : <wsdl:service name="soustraireService">

    SoustraireService service = ... ;

// Utilisation du service pour obtenir un stub qui implemente le SDI
// (Service Definition Type ; i.e. PortType).
// Pour le typage, c’est la balise : <wsdl:portType name="soustraire">
// Pour le getsoustraire(), le sommer correspond à la balise :
// <wsdl:port binding="impl:sommerSoapBinding" name="soustraire">

    Soustraire monService = ... ;
// Mise en oeuvre du service par application directe des méthodes
```

Applications Réparties TD 4
Web Services en Java avec Axis

```
    int ret = ... ;
    System.out.println(ret);
  }
}
```

2.4 Désactivation du Service Web

Enfin pour retirer un service déployé, il suffit d'appliquer l'utilitaire *AdminClient*, créer un fichier *undeploysoustraire.wsdd* correspondant au service.
Le contenu du fichier *undeploysoustraire.wsdd* sera toujours le même, au nom de service près. Pour le service « sommer », nous aurons :

```
<undeployment xmlns="http://xml.apache.org/axis/wsdd/">
<service name="soustraire"/>
</undeployment>
```

Ensuite, la ligne de commande est simplement :

```
java org.apache.axis.client.AdminClient  undeploysoustraire.wsdd
```

Vérifiez ensuite la désactivation du service.

2.5 Gestion des Accès aux Méthodes des Services Web

Empêchez l'accès à la méthode *getsoustraction* par deux approches différentes :

- Par la modification de la classe *soustraire*,
- Par la modification du descripteur dans *deploysoustraire.wsdd* avec le paramètre *allowedMethods*.

2.6 Gestion des Sessions

Complétez la classe *soustraire* en ajoutant :

- un attribut privé *valeurPi* de type double et affecté d'une valeur par défaut 3.14,
- une méthode public *double getPi()* pour obtenir la valeur de Pi,
- une méthode public *void setPi(double nouvelleValeur)* pour redéfinir la valeur de Pi (augmentation ou diminution de la précision).

Compilez et déployez la nouvelle classe en utilisant l'utilitaire AdminClient, vérifiez le déploiement.

Complétez le code de votre client *SoustraireClient* et exécutez la méthode *setPi(double nouvelleValeur)* avec la nouvelle valeur 3,1415. Consultez la valeur de Pi avec la méthode *getPI()*. Que constatez-vous ? Pourquoi ? En fait chaque fois qu'une méthode du service est appelée, Axis génère une nouvelle instance de la classe.

Pour qu'Axis n'utilise qu'une unique instance par session utilisateur, il est nécessaire de spécifier le paramètre de configuration supplémentaire suivant dans le descripteur.

```
<parameter name= "scope" value= "application"/>
```

Désactivez le déploiement, ajoutez le nouveau paramètre à notre descripteur et déployez à nouveau le service.
Vérifiez qu'une unique instance de classe est créée (en appelant successivement les méthodes *setPi* et *getPi* depuis votre client).