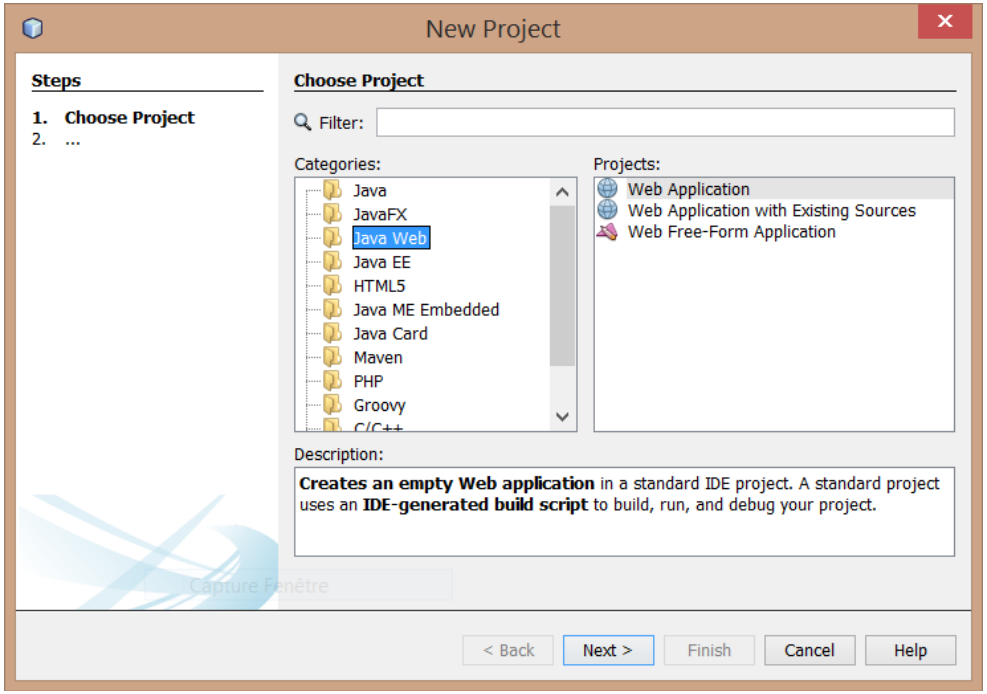
Création des web services

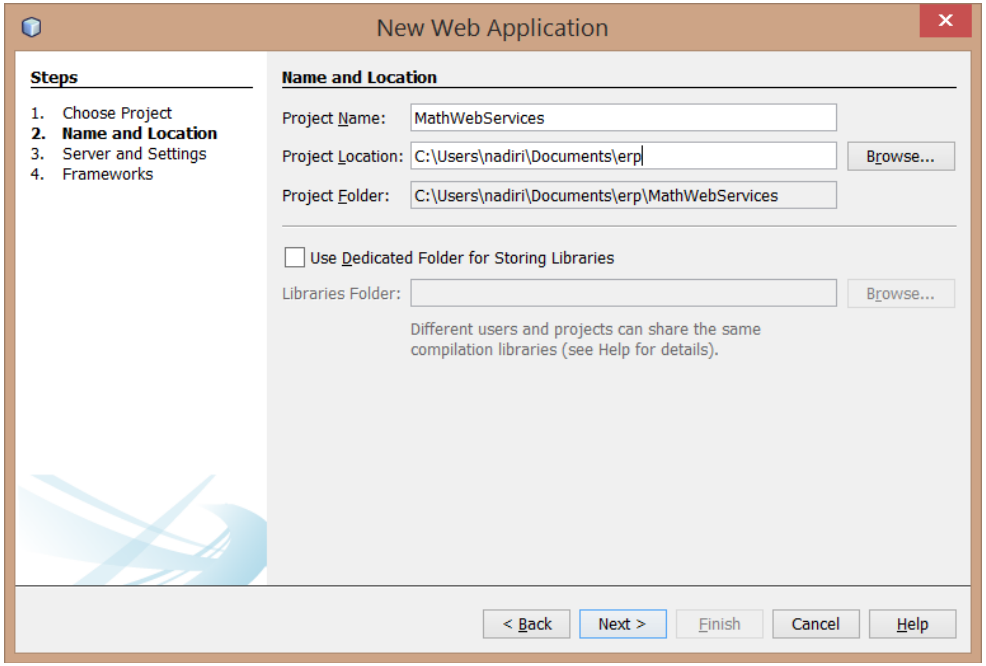
Dans ce TP, nous allons créer un web service à partir d'une classe Java proposant quelques simples opérations mathématiques :

- Additionner deux nombres
- Tester si un nombre est premier
- Décomposer un nombre en facteurs premiers

Pour cela, créer sous NetBeans un nouveau projet de type Web Application :



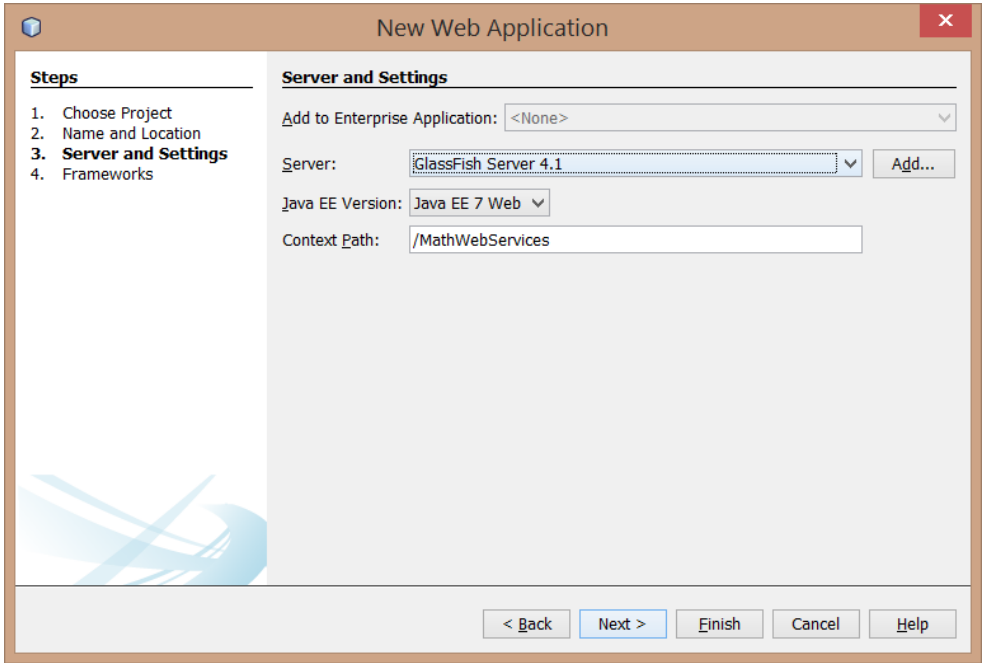
Nouveau projet Web



Nommage et emplacement

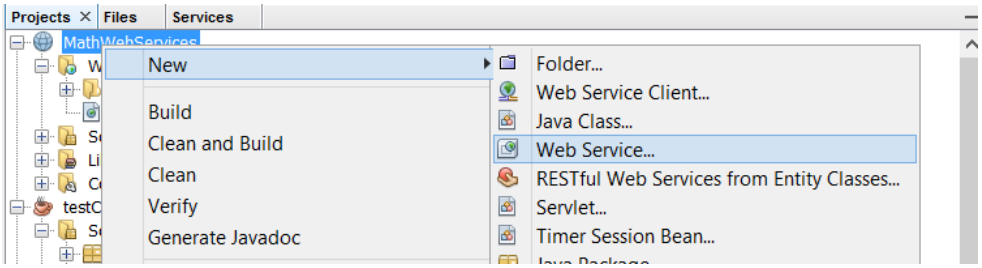
Nous utilisons ici Glassfish pour héberger notre code métier et exposer le service. D'autres conteneurs sont disponibles, tels que Tomcat, Geronimo, Websphere, Oracle Application Server, Wildfly. Chaque serveur peut utiliser des frameworks différents pour prendre en charge la génération du code, la sérialisation/désérialisation des messages, etc.

Glassfish repose sur L'implémentation Metro qui prend en charge JAX-WS (Java API for XML Web Services) pour prendre en charge la traduction WSDL/Java et SOAP/Java.

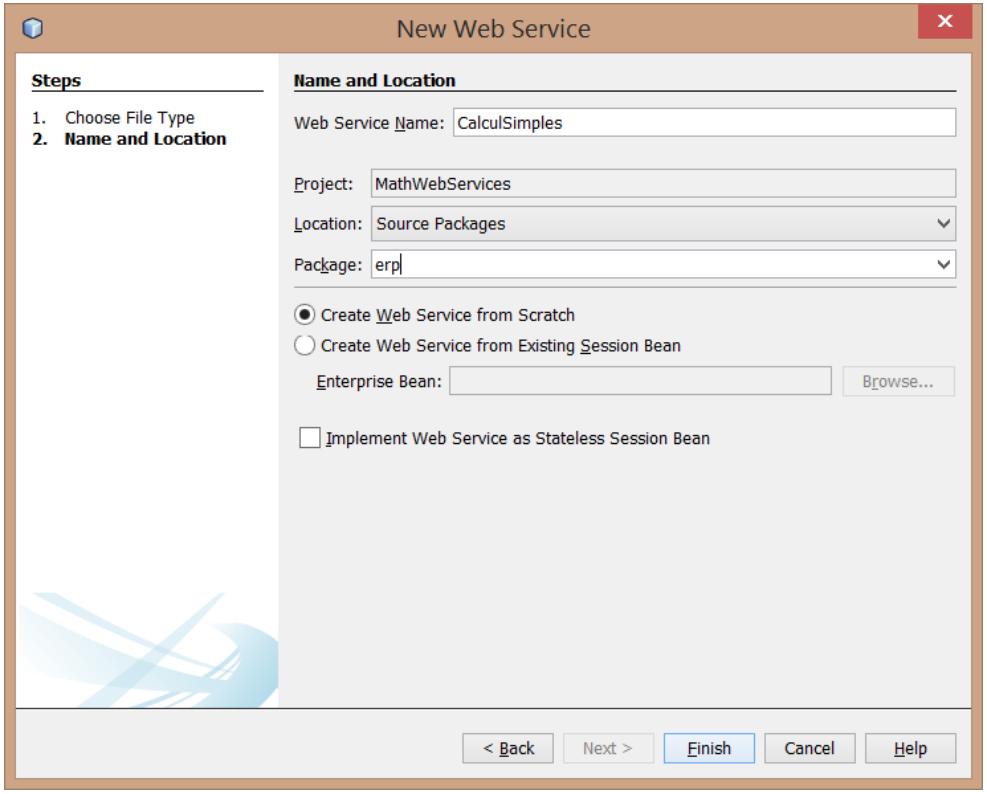


Environnement d'exécution

Nous allons ensuite créer un nouveau web service dans le projet :



Création d'un nouveau service



Nommage et emplacement

Une fois les paramètres saisis, Netbeans présente le code métier généré, avec une opération par défaut : hello. L'utilisation de l'API JAX-WS, intégrée à la plateforme depuis la version JAVA EE 5, permet d'utiliser des métadonnées directement dans le code source pour transformer une classe et certaines de ses méthodes en web service. Ces métadonnées sont appelés Annotations, et sont exploitées par l'environnement pour extraire des informations qu'il fallait auparavant définir dans des descripteurs XML annexes. Le développement d'un web service est ainsi grandement simplifié. Trois annotations sont utilisées ici.

@WebService

Définit la classe comme étant un web service, identifié par un nom

@WebMethod

Expose une méthode de la classe en tant qu'opération proposée par le web service, sous un nom donné. Par défaut, toutes les méthodes publiques d'une classe sont exposées, le paramètre (exclude=true) permet d'annuler ce comportement

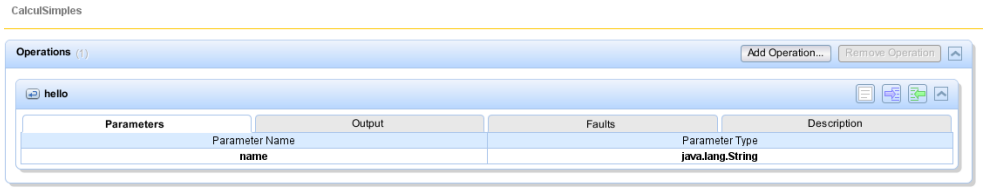
@WebParam

Définit un paramètre de l'opération

```java
package erp;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author nadiri
 */
@WebService(serviceName = "CalculSimples")
public class CalculSimples {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

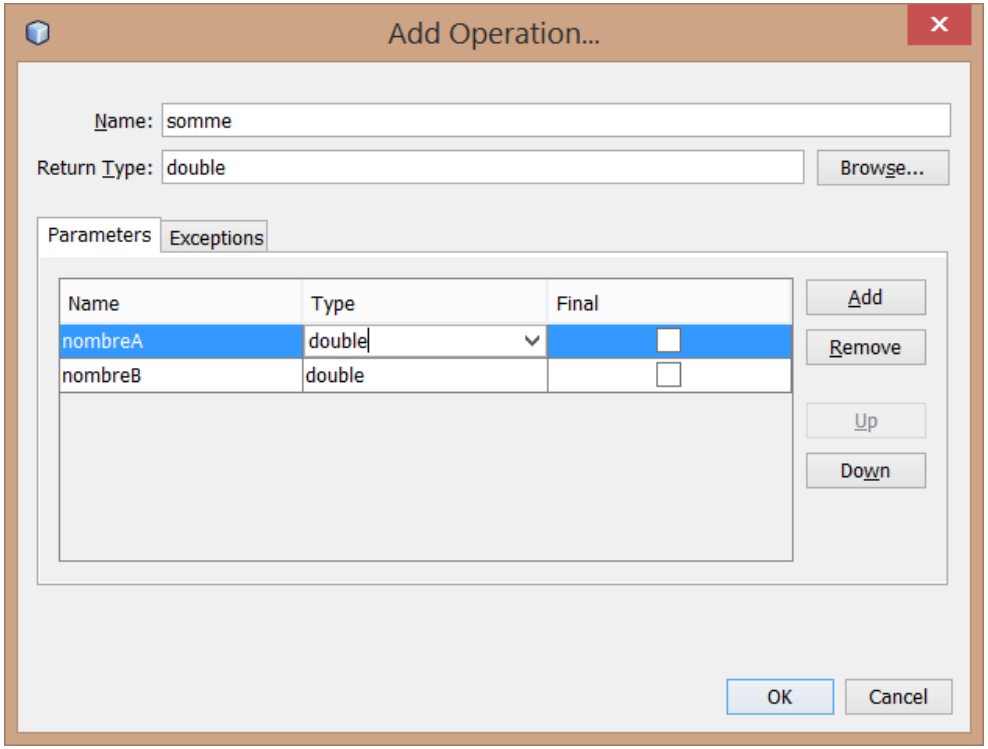
Code généré et annotations

Netbeans propose un assistant pour créer des opérations. Pour cela, passer de la vue "Source" à la vue "Design." :



Vue graphique du service

Créer ensuite une nouvelle opération additionner avec les paramètres suivants :



Création d'une opération

En basculant à nouveau vers la vue Source, nous pouvons voir le code généré avec les informations :

```java
/**
 * Web service operation
 */
@WebMethod(operationName = "somme")
public double somme(@WebParam(name = "nombreA") double nombreA, @WebParam(name = "nombreB") double nombreB) {
    //TODO write your implementation code here:
    return 0.0;
}
```

Code généré

Créer de la même manière deux autres opérations "decomposer" et "premier". Vous pouvez également éditer le code sans passer par l'assistant.

```
    @WebMethod(operationName = "decomposer")
    public List decomposer(@WebParam(name = "entier") int entier) {
        List l = new ArrayList();
        int i=2;
        while(i<=entier){
            if(entier%i==0){
                l.add(i);
                entier=entier/i;
            }else
                i++;
        }
        return l;
    }

    @WebMethod(operationName = "premier")
    public boolean premier(@WebParam(name="entier") int entier){
        return decomposer(entier).size()==1;
    }


}
```

Autres opérations

La dernière étape consiste à tester si le code fonctionne correctement avant de passer à la phase suivante. Pour cela nous créons une méthode main pour exécuter chacune des fonctions et vérifier leur comportement. Pour exécuter la classe en tant que simple application, nous utilisons le menu *Run » Run File*

```
    public static void main(String[] args){
        CalculsSimples m=new CalculsSimples();
        System.out.println("premier 8 ? "+m.premier(8));
        System.out.println("premier 5 ? "+m.premier(5));
        System.out.println("decomposer 8 : "+m.decomposer(8));
        System.out.println("decomposer 362880 :"+m.decomposer(362880));
    }
```

```
CalculsSimples > main
Output
Java DB Database Process | GlassFish Server 4.0 | MathsWebService (run)
run:
premier 8 ? false
premier 5 ? true
decomposer 8 : [2, 2, 2]
decomposer 362880 :[2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 5, 7]
BUILD SUCCESSFUL (total time: 1 second)
```

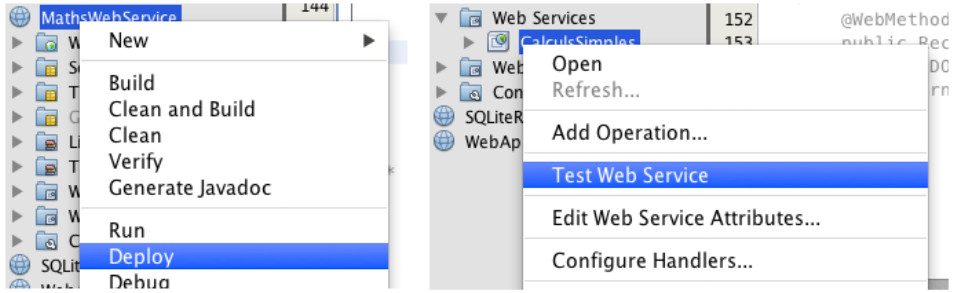
Test du code métier

Tester Invocation du web service par le navigateur

Le web service est maintenant prêt à être déployé sur un serveur. Rappelons que les web services ont vocation à être exécutés dans un environnement qui prend en charge les requêtes provenant des clients, la traduction des messages SOAP, l'appel du code métier et la transmission du résultat après retraduction en SOAP.

Dans Netbeans, sélectionner le projet, et dans le menu du bouton droit choisir l'action "Deploy". Si le serveur Glassfish n'est pas lancé, il faudra patienter quelques secondes...

Une fois le serveur lancé et le projet déployé, nous pouvons tester le web service. Sélectionner le web service dans le projet, puis choisir l'action Test Web Service dans le menu du bouton droit.



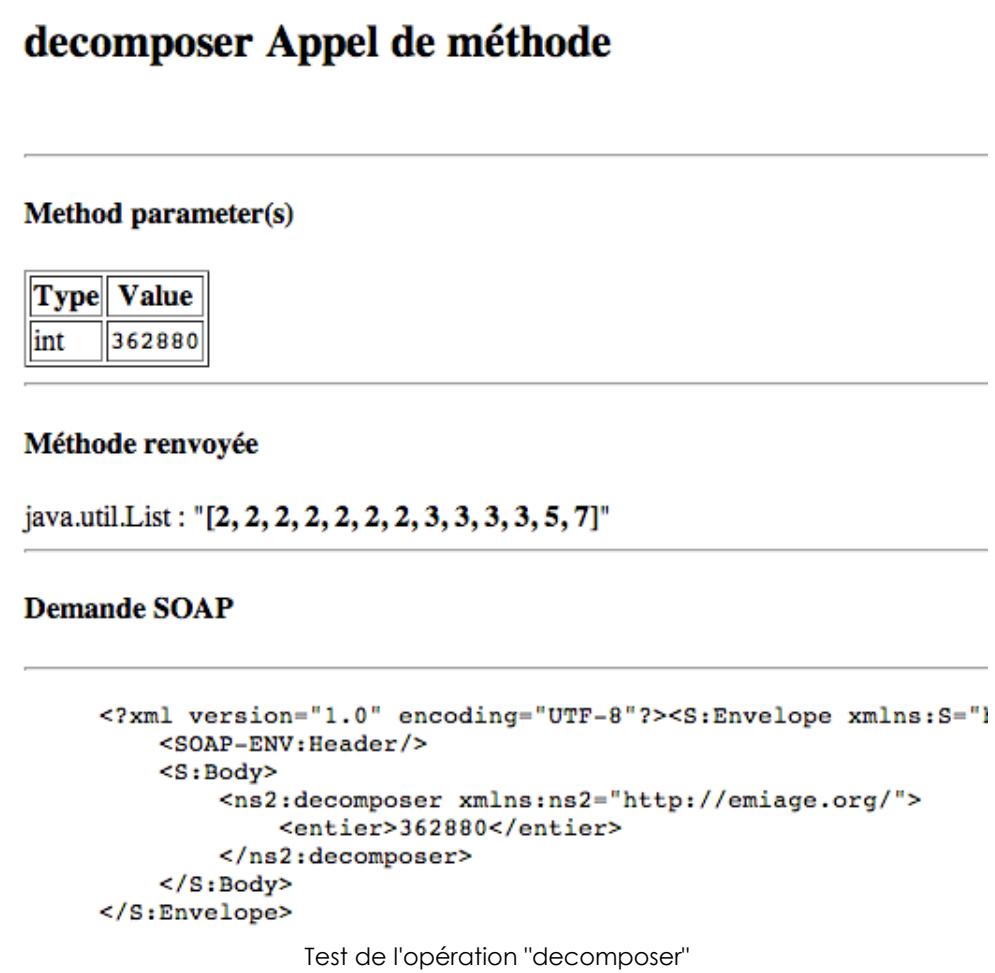
Déploiement et test

Glassfish/JAX-WS propose une page résumant les opérations exposées par le web service, et la possibilité de les tester en passant les paramètres éventuels.

Aperçu du web service

Nous pouvons noter que le testeur propose une vue des messages SOAP échangés durant les requêtes :

Test de l'opération "additionner"



Test de l'opération "decomposer"

Notre web service est à présent fonctionnel, nous pouvons passer à la phase suivante : consommer le web service à partir d'une application.