

## Création d'un service web avec NetBeans 5.5 et SJAS 9

### Sommaire

---

1.	Présentation .....	2
2.	Création d'un service web avec NetBeans .....	2
2.1.	Création d'une application Web .....	2
2.2.	Création du service web .....	4
2.3.	Ajout d'une méthode au service web .....	6
3.	Génération des javadocs .....	7
4.	Déploiement sur le serveur d'application installé avec NetBeans .....	8
4.1.	Déploiement à partir de NetBeans .....	8
4.2.	Test du service web à partir de NetBeans .....	8
5.	Utilisation du service Web créé .....	10
5.1.	Serveur d'application .....	10
5.2.	Création d'un nouveau projet client .....	11
5.3.	Intégration du fichier WSDL dans NetBeans .....	13
5.4.	Utilisation du service web dans l'application de test .....	15
6.	Déploiement sur un serveur d'application autonome .....	17

## 1. Présentation

---

L'objectif de ce document est d'expliquer la création et l'utilisation d'un service web avec NetBeans 5.5 en respectant la norme JAX-WS 2.0.

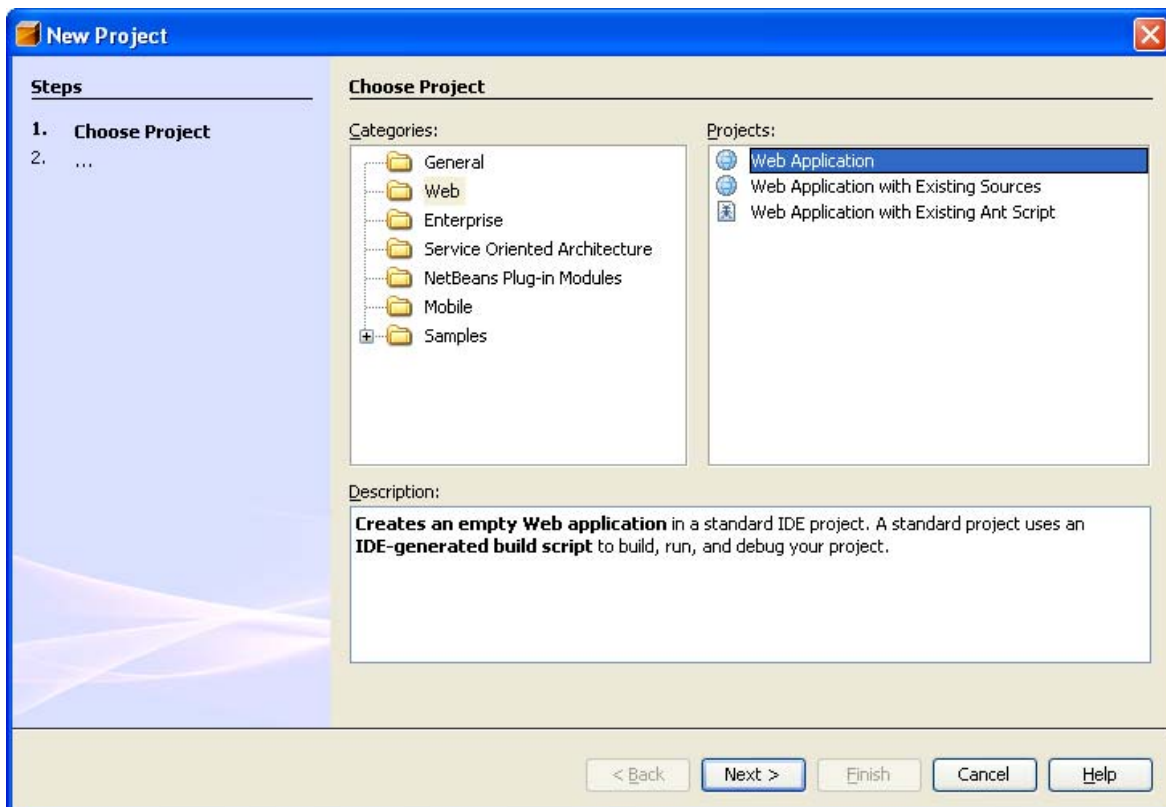
## 2. Création d'un service web avec NetBeans

---

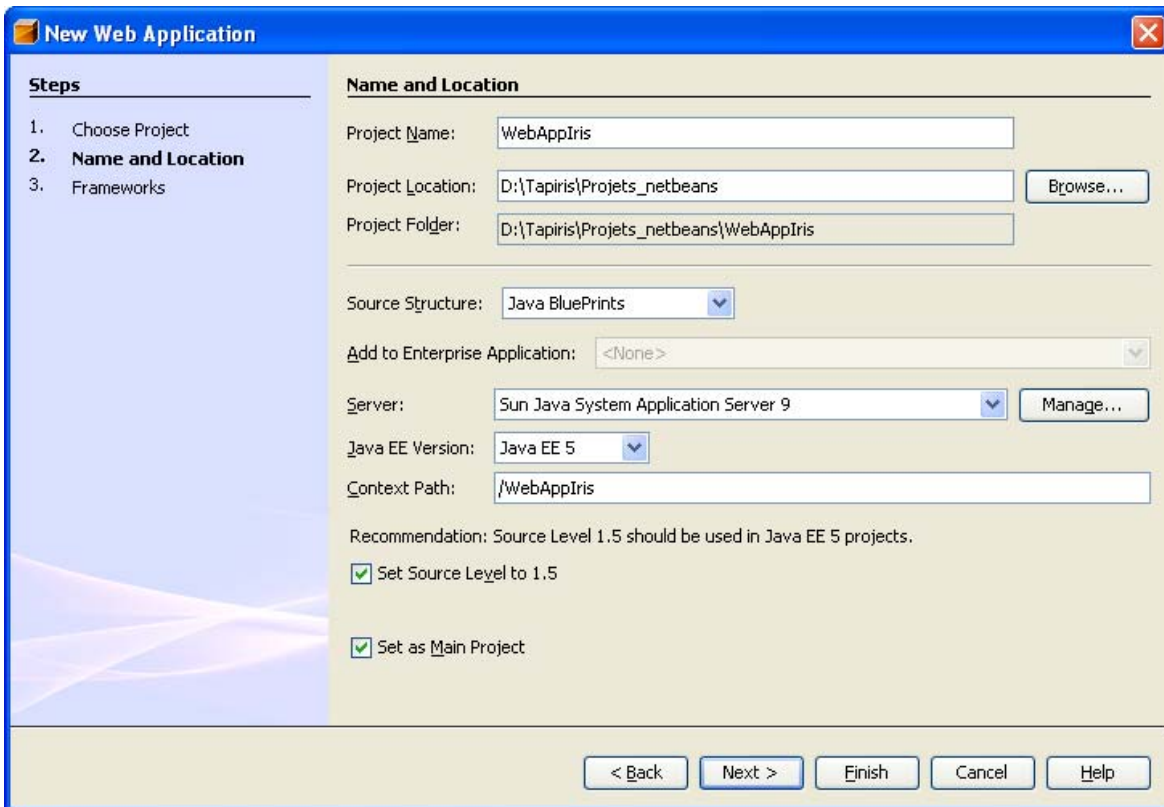
### 2.1. Création d'une application Web

*Objectifs : Mise en place du container web du serveur d'application (SJAS). Il contiendra le site web JSP par défaut (non utilisé dans notre exemple) et nous allons y intégrer un service web.*

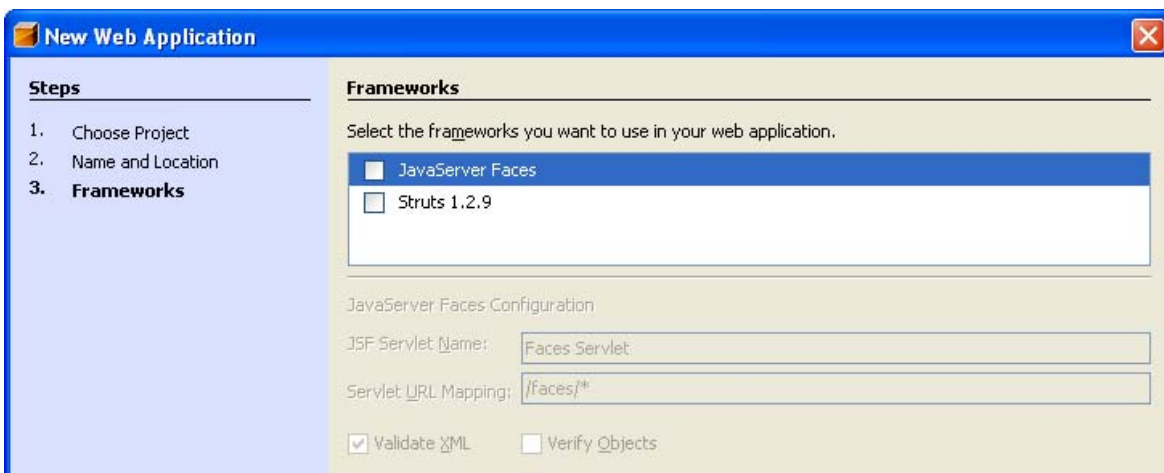
File > New Project > Web > Web Application



Choisissez le serveur d'application sur lequel devra être déployé le service web : Sun Java System Application Server 9 (SJSAS) et Java EE 5.

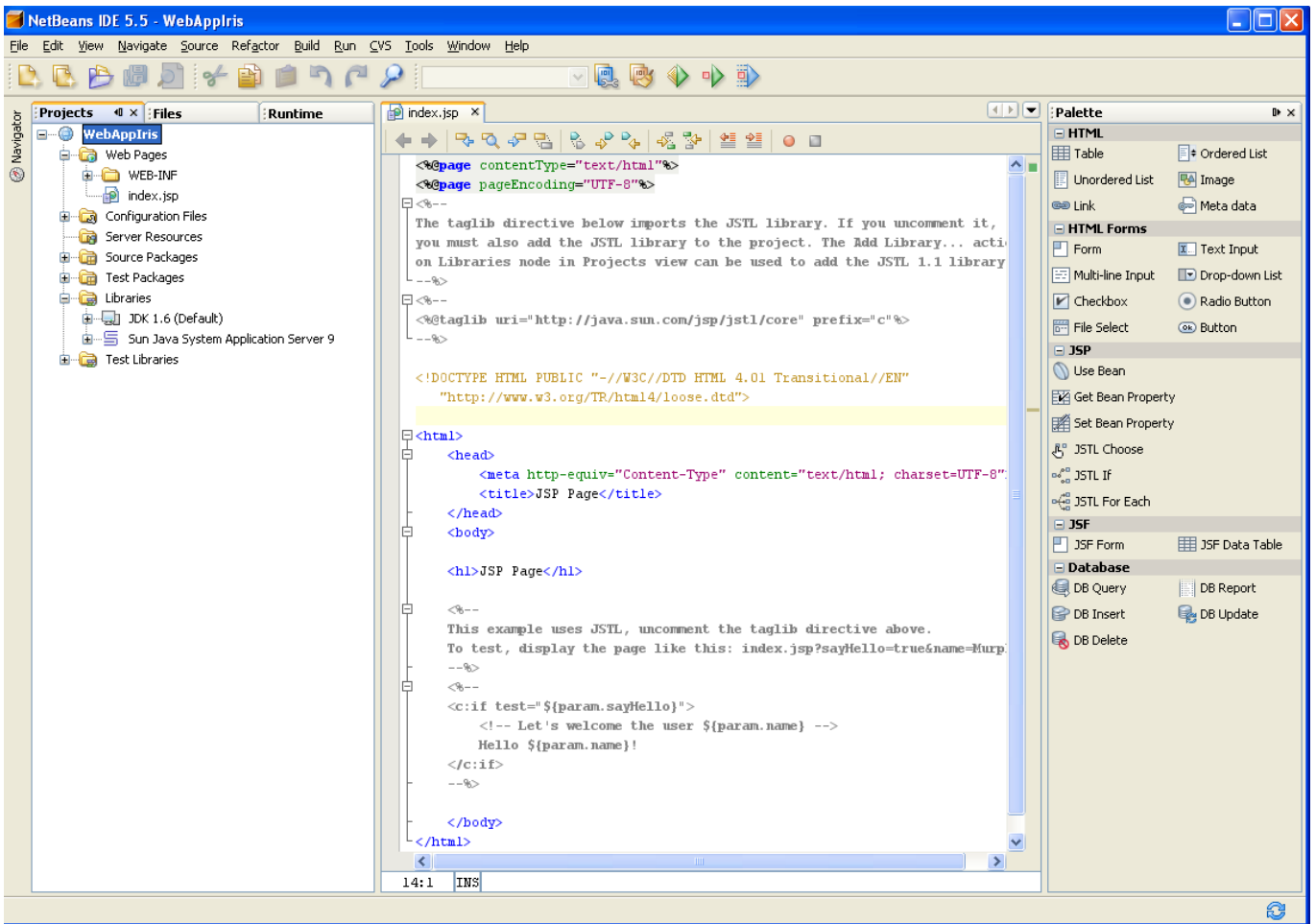


Aucun Frameworks de sélectionné pour le moment.



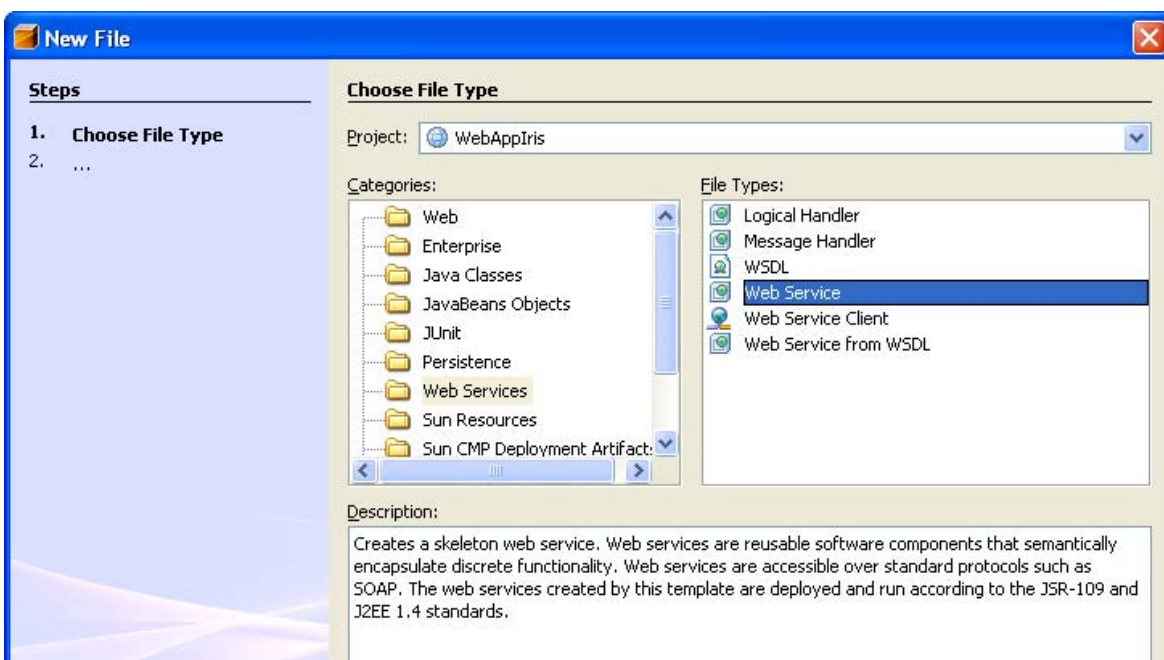
Cliquez sur Finish.

NetBeans créé une application web contenant déjà une page index.jsp :



## 2.2. Création du service web

Clic droit sur le projet > New > File/Folder > Web Services > Web Service



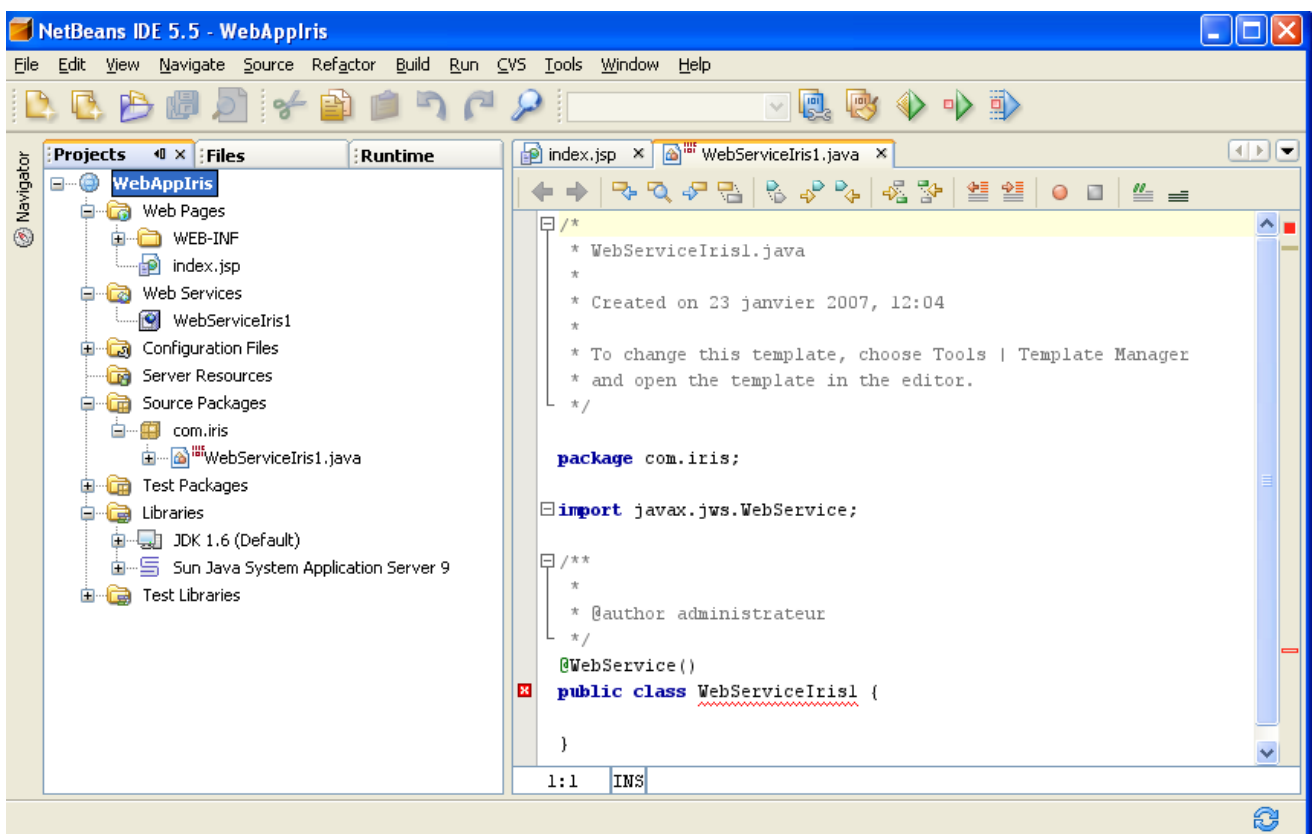
Entrez le nom du service web : « WebServiceIris1 »

Entrez le nom du package dans lequel doit être créé les classes du service web : « com.iris »

Remarque : Pas de doublon de nom du service web sur le même serveur d'applications



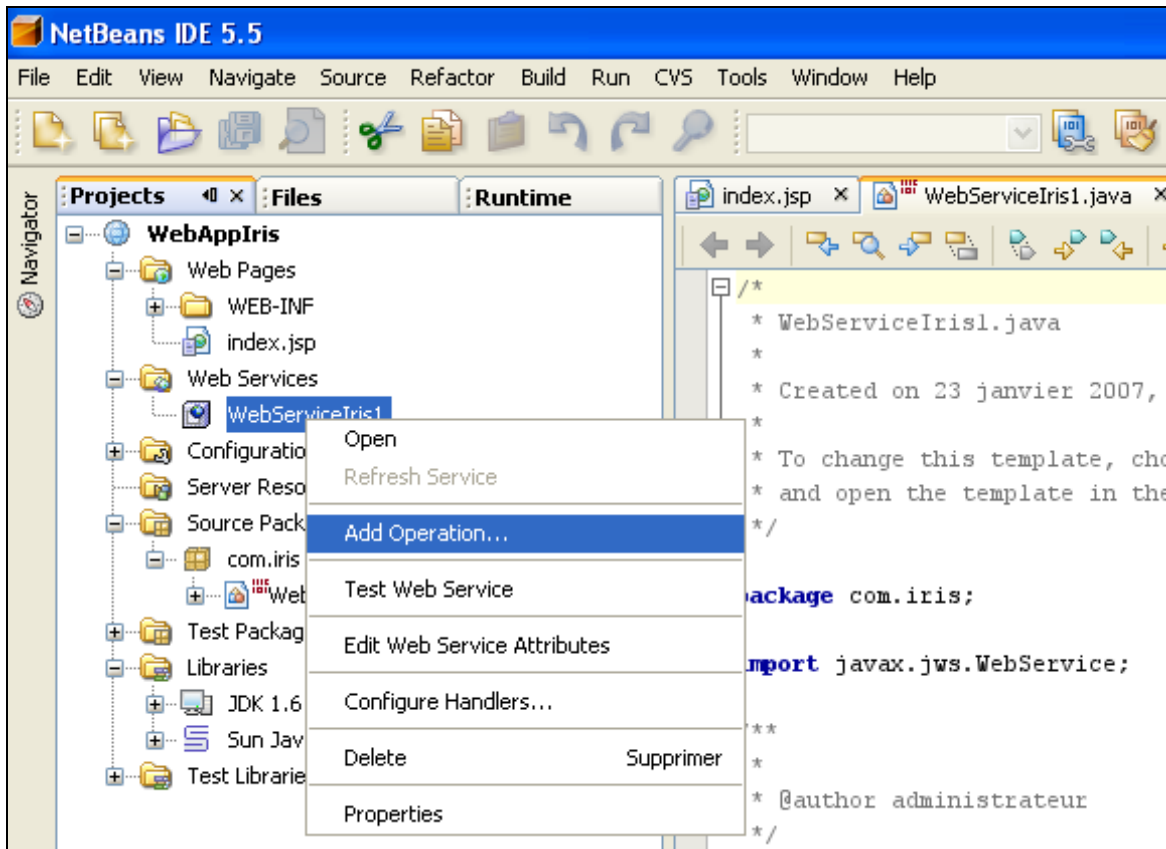
Cliquez sur Finish



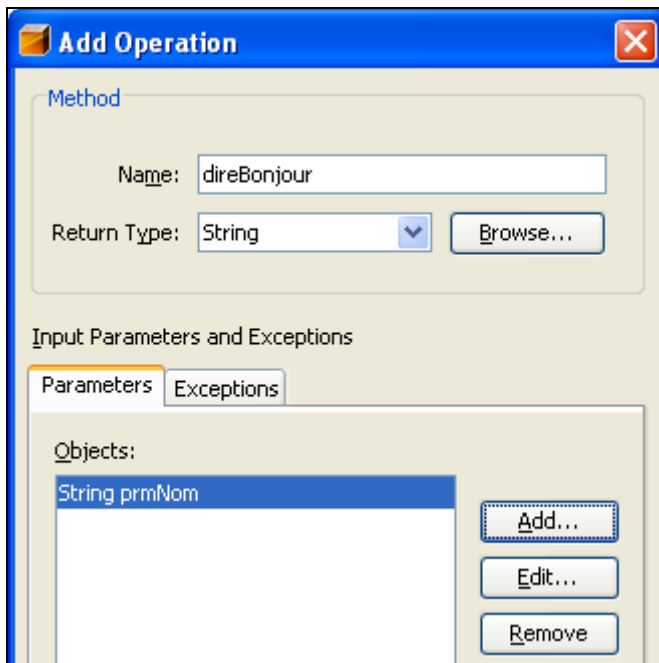
Après la création du service web, NetBeans signale une erreur sur la classe WebServiceIris1 tant qu'aucune méthode n'a été créée.

### 2.3. Ajout d'une méthode au service web

Dans l'onglet projects > Dossier Web Services > Clic droit > Add Operation



Ajoutez la méthode direBonjour() en respectant l'écran suivant :



Voici le code généré avec les annotations « web services » commençant par le caractère @ (spécifications du package javax.jws de Java EE 5)

```
@WebMethod
public String direBonjour(@WebParam(name = "prmNom") String prmNom) {
    // TODO implement operation
    return null;
}
```

Ajoutez le code suivant à la méthode :

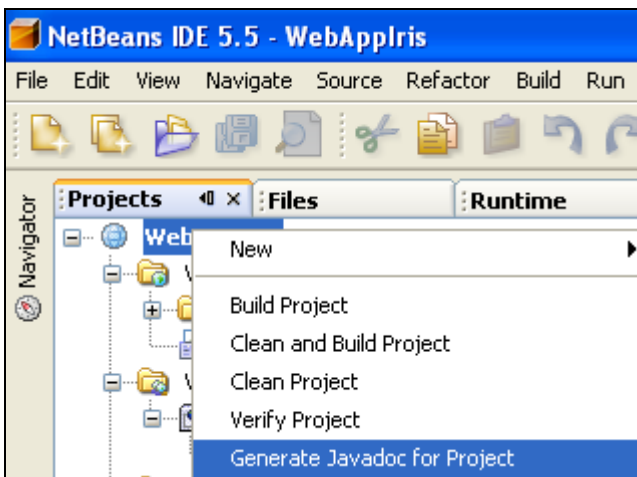
```
@WebMethod
public String direBonjour(@WebParam(name = "prmNom") String prmNom) {
    String s = "" ;
    s = "Bonjour " + prmNom ;
    return s ;
}
```

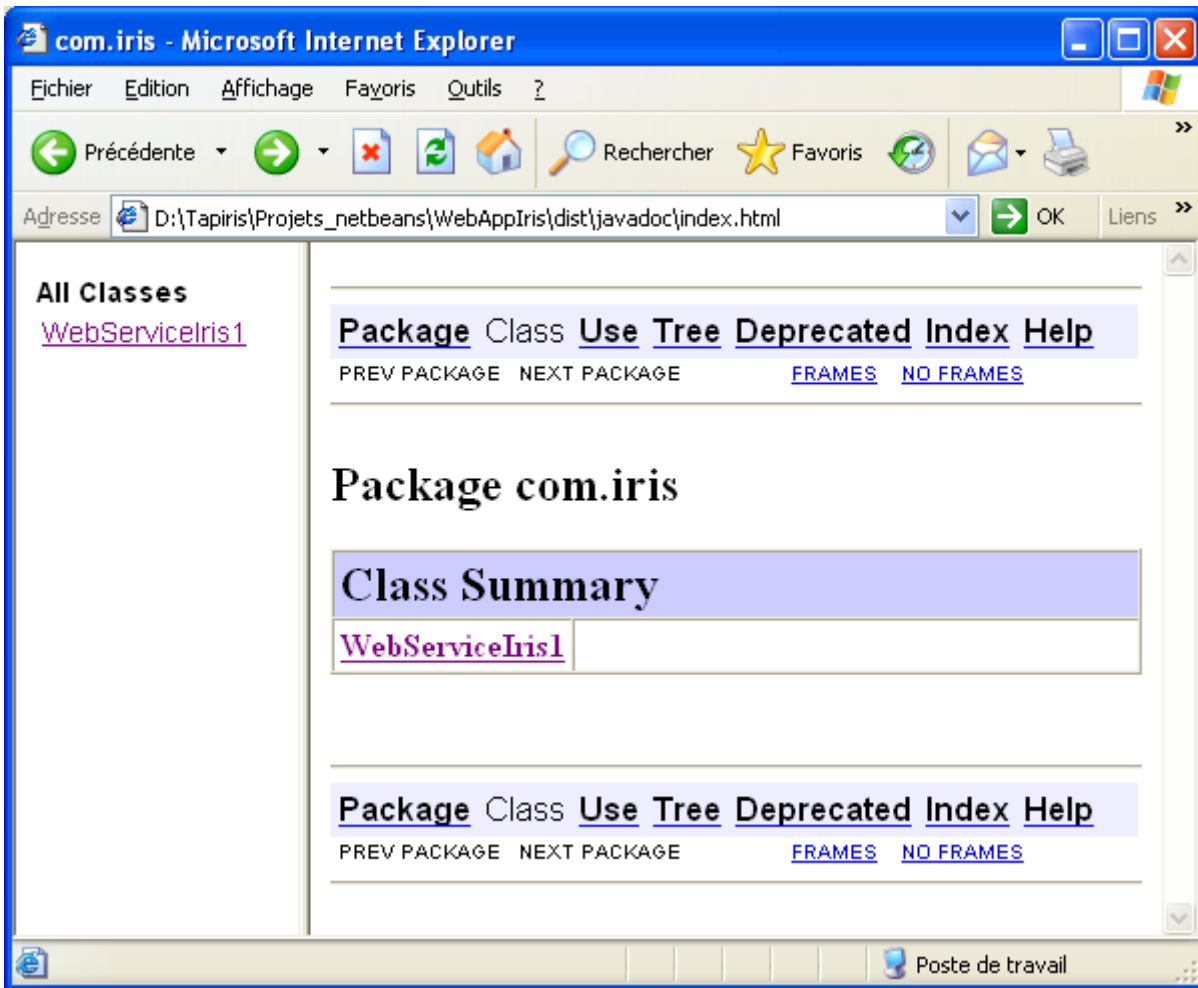
### 3. Génération des javadocs

---

Pour respecter la norme de création des services web, il faut générer les javadocs du projet.

Clic droit sur le projet > Generate Javadocs for project





#### 4. Déploiement sur le serveur d'application installé avec NetBeans

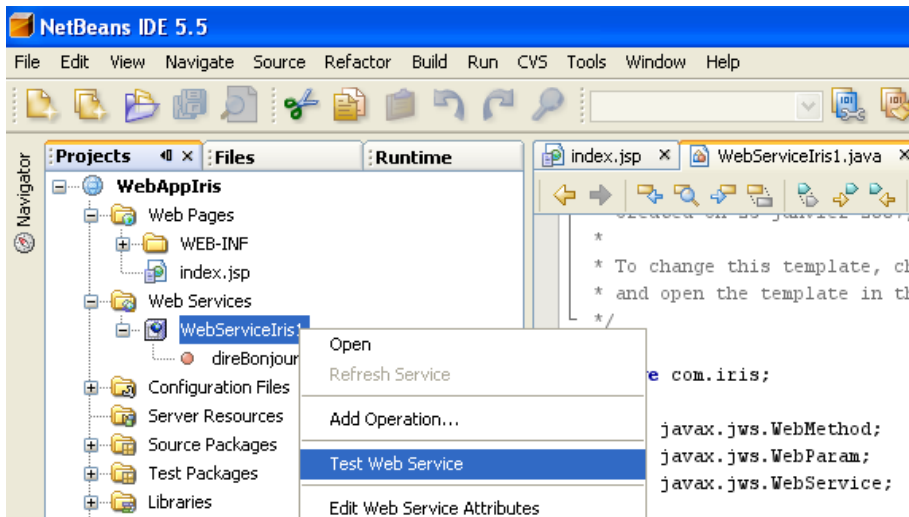
---

##### 4.1. Déploiement à partir de NetBeans

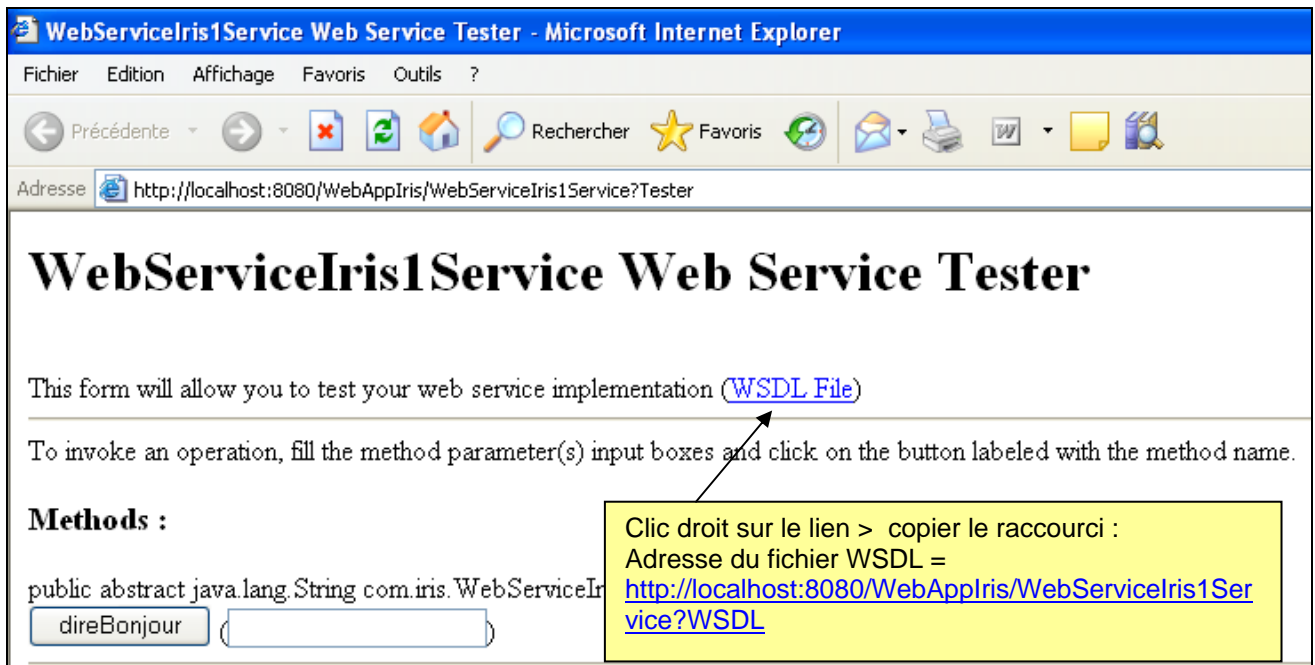
Clic droit sur le projet > Deploy Project

##### 4.2. Test du service web à partir de NetBeans

Dans l'onglet « Projects » > Clic droit sur le service web « WebServiceIris1 » > Test Web Service



Ce test ouvre une page web qui permet d'appeler la méthode du service web et de récupérer l'adresse du fichier WSDL (utilisé par les clients du web service)



Entrez un texte dans la zone de texte et cliquez sur le bouton « direBonjour ». Vous devez obtenir le résultat suivant :

Method invocation trace - Microsoft Internet Explorer

Adresse <http://localhost:8080/WebAppIris/WebServiceIris1Service?Tester>

### direBonjour Method invocation

Method parameter(s)

Type	Value
java.lang.String	Armentières

Method returned

java.lang.String : "Bonjour Armentières"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns1="http://iris.com/">
  <soapenv:Body>
    <ns1:direBonjour>
      <prnNom>Armentières</prnNom>
    </ns1:direBonjour>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns1="http://iris.com/">
  <soapenv:Body>
    <ns1:direBonjourResponse>
      <return>Bonjour Armentières</return>
    </ns1:direBonjourResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Terminé Intranet local

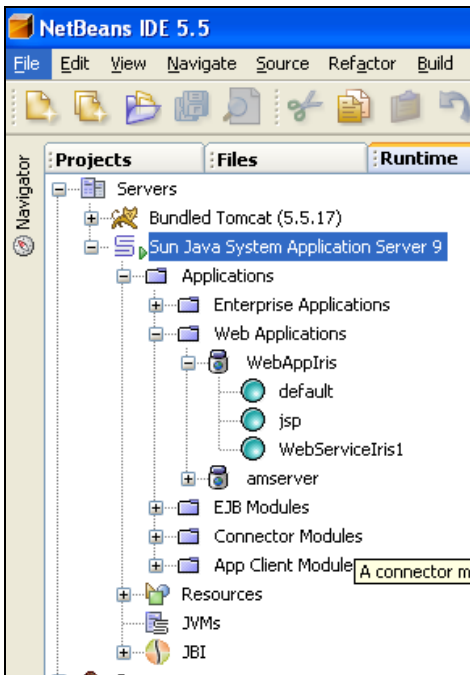
Le service web est maintenant opérationnel.

## 5. Utilisation du service Web créé

### 5.1. Serveur d'application

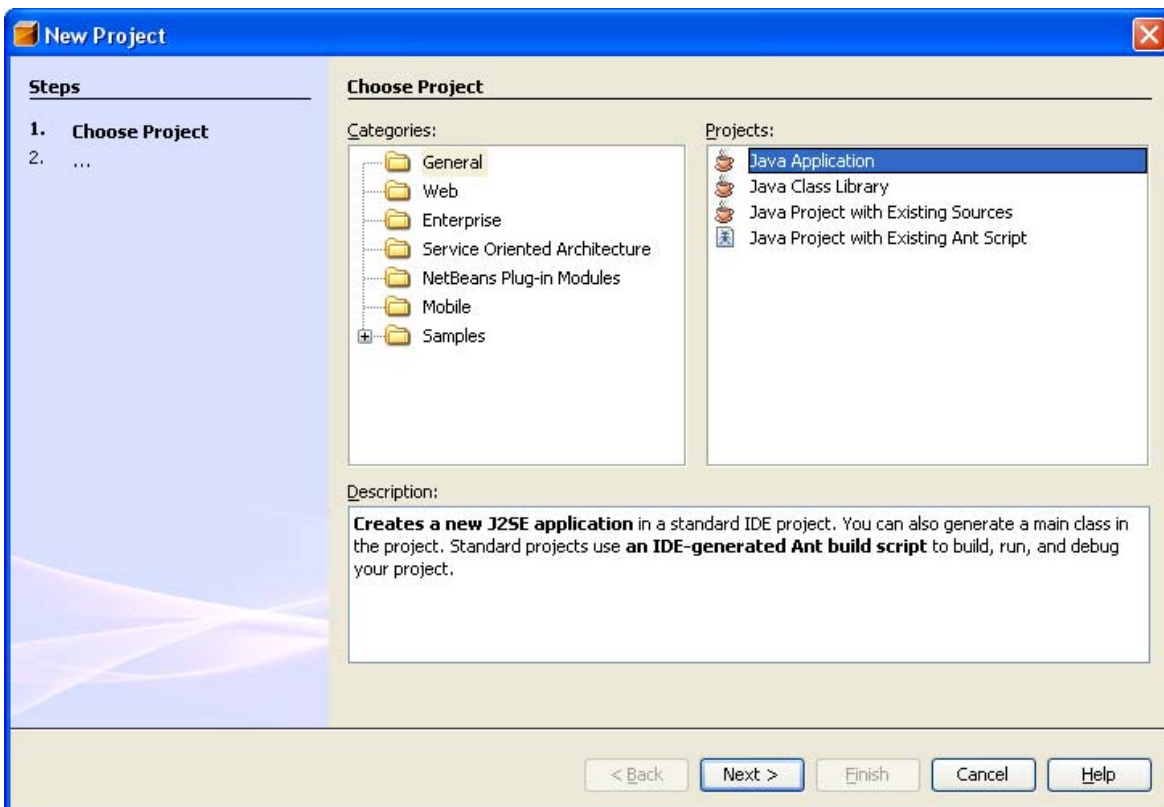
Vérifiez que le serveur d'application utilisé par le service web est bien démarré.

Dans NetBeans > Onglet Runtime > Clic droit sur « Sun Java System Application Server 9 » > Start

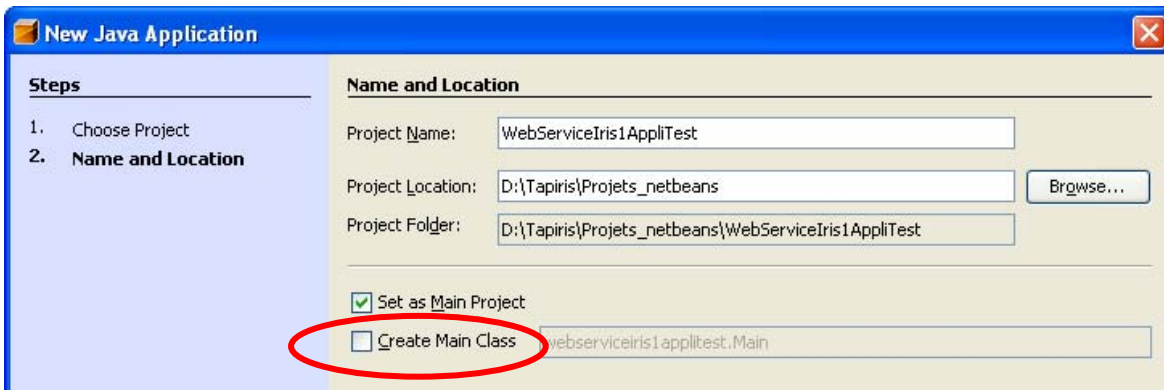


## 5.2. Création d'un nouveau projet client

File > New Project > General > Java Application

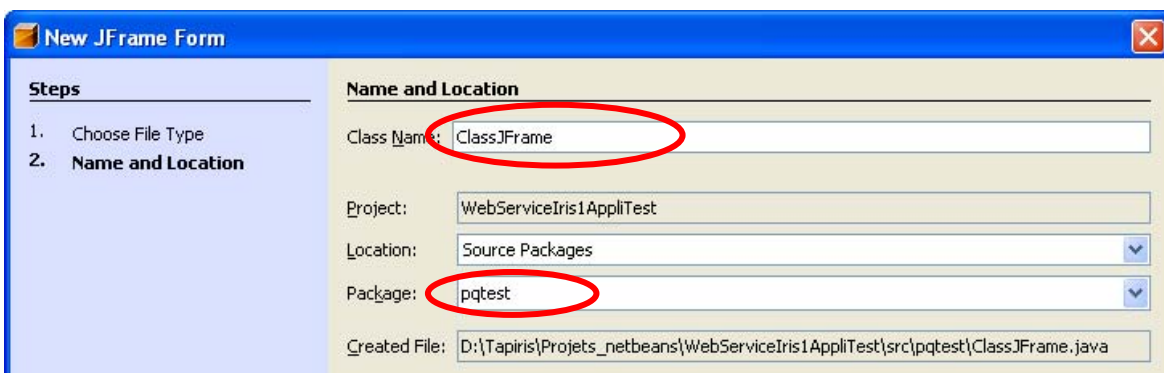
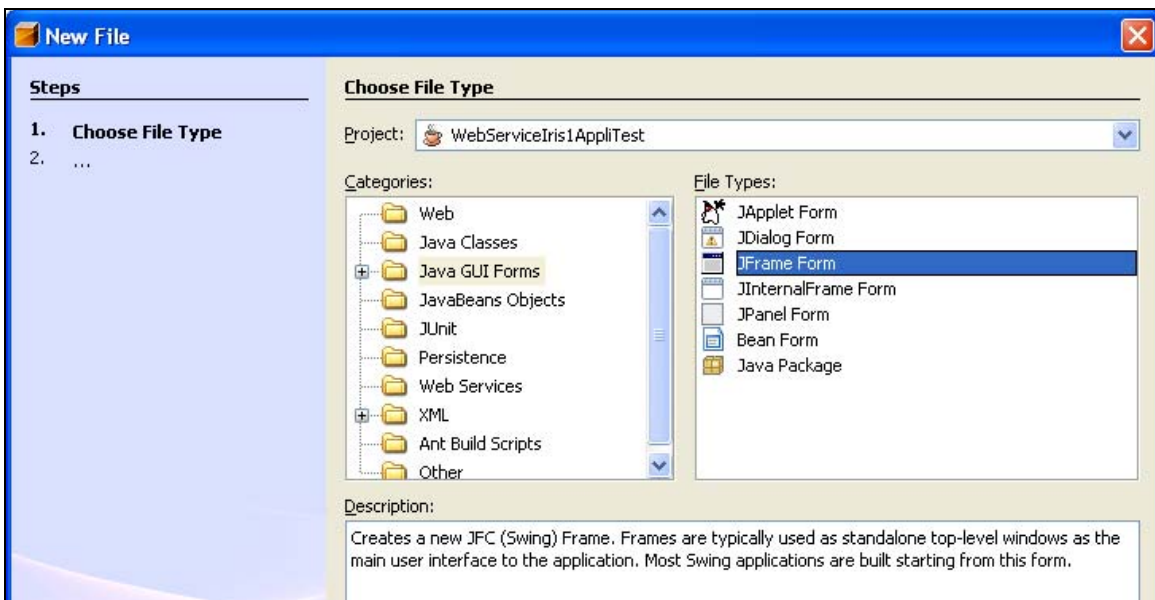


Décochez « Create Main Class »



Création d'une fiche graphique JFrame :

Clic droit sur le projet > New > File/Folder > Java GUI Forms > JFrame Form

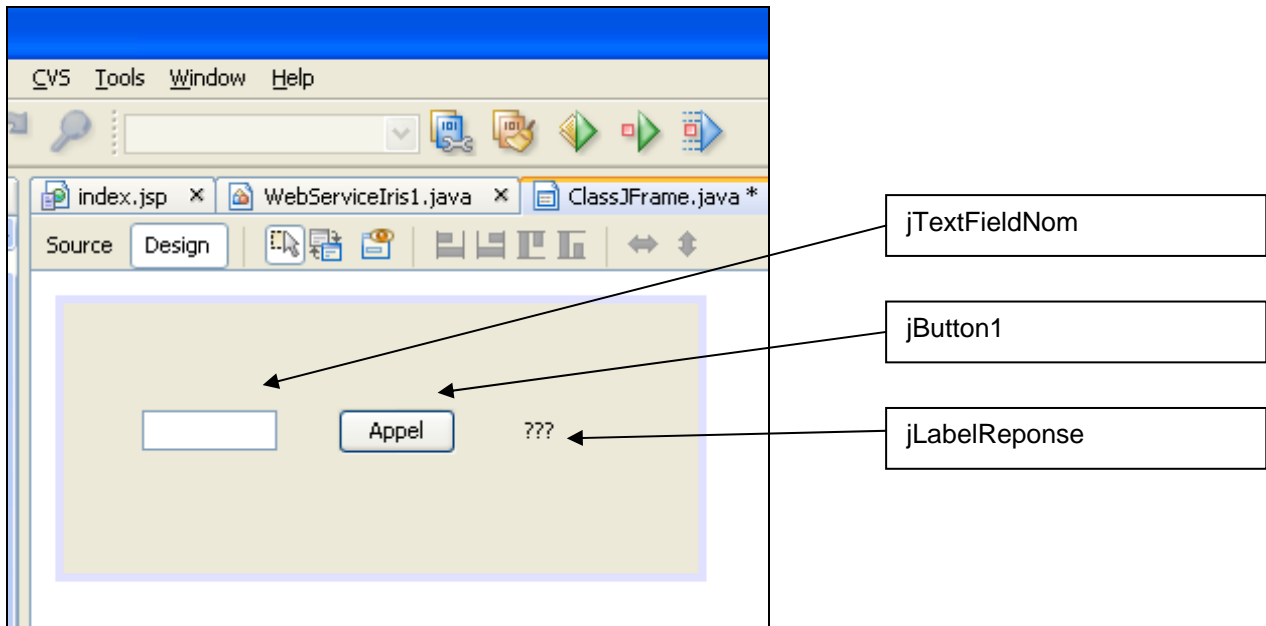


Donnez un nom à la classe : ClassJFrame  
Créez en même temps le package « pqtest »  
Cliquez sur Finish

Sélection du nouveau JFrame comme classe principale du projet :  
Clic droit sur le projet > Properties > Run > Main Class = « pqtest.ClassJFrame »

Création des composants de l'IHM (affichage en mode « Design ») :

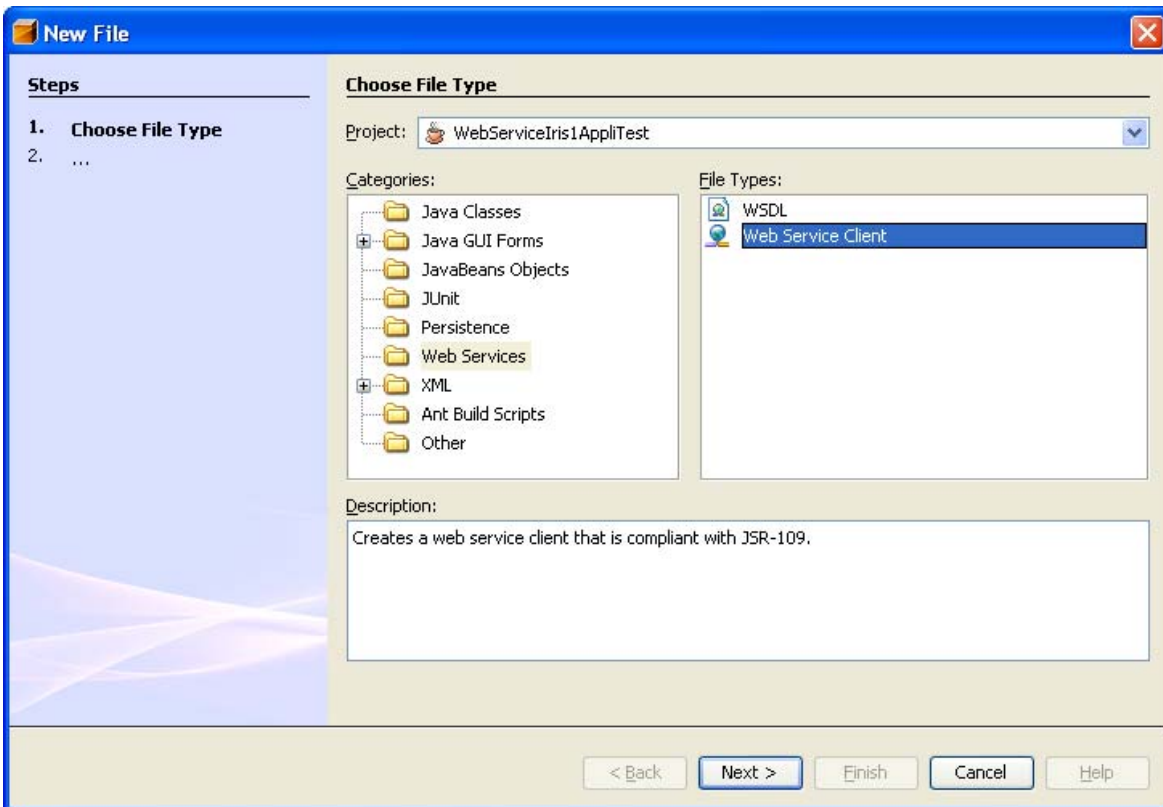
En cliquant – glissant de la palette vers la JFrame, créez les composants suivants :



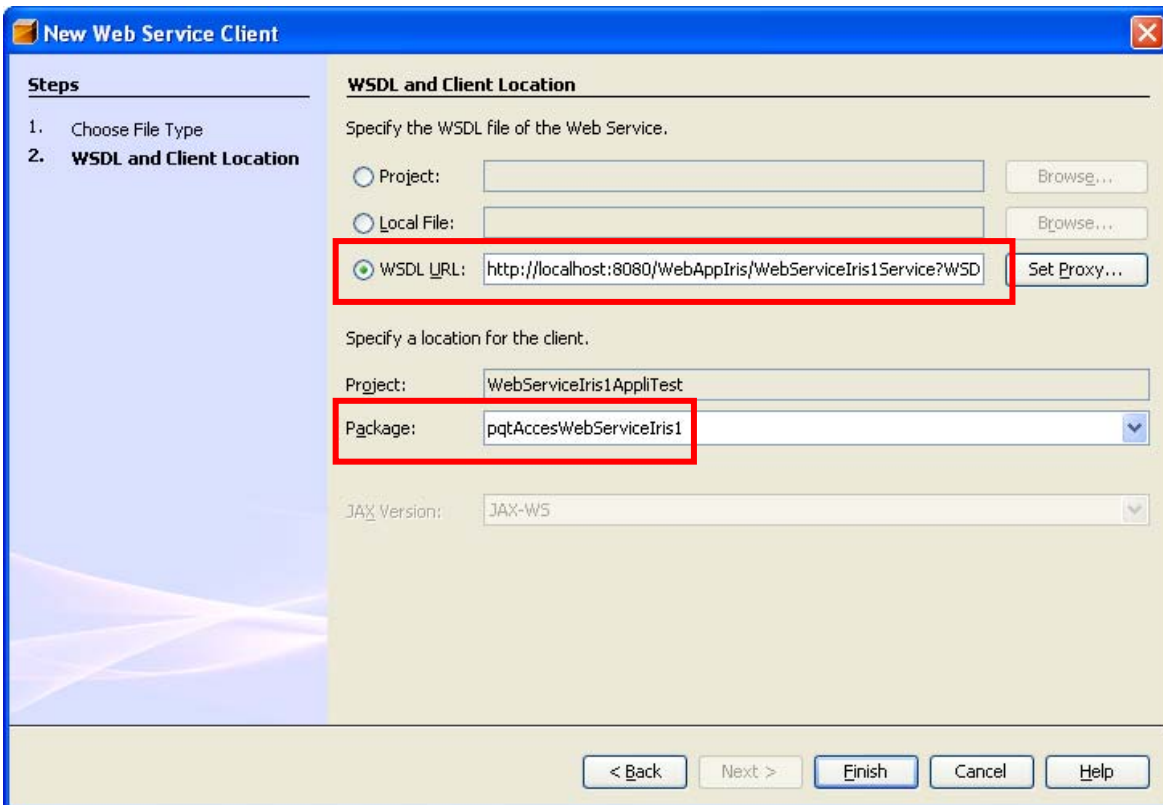
*Remarque : les composants graphiques de la JFrame sont tous créés dans la méthode initComponents(). Cette méthode n'est pas modifiable à partir du code source, NetBeans ne présente pas son code directement, il est replié dans l'éditeur de code source.*

### 5.3. Intégration du fichier WSDL dans NetBeans

Clic droit sur le projet > New > File / Folder > Web Services > Web Service Client



Remplir le champ « WSDL URL » avec l'adresse du fichier WSDL récupéré précédemment  
Remplir le champ « Package » avec pqtAccesWebServiceIris1  
Cliquez sur Finish

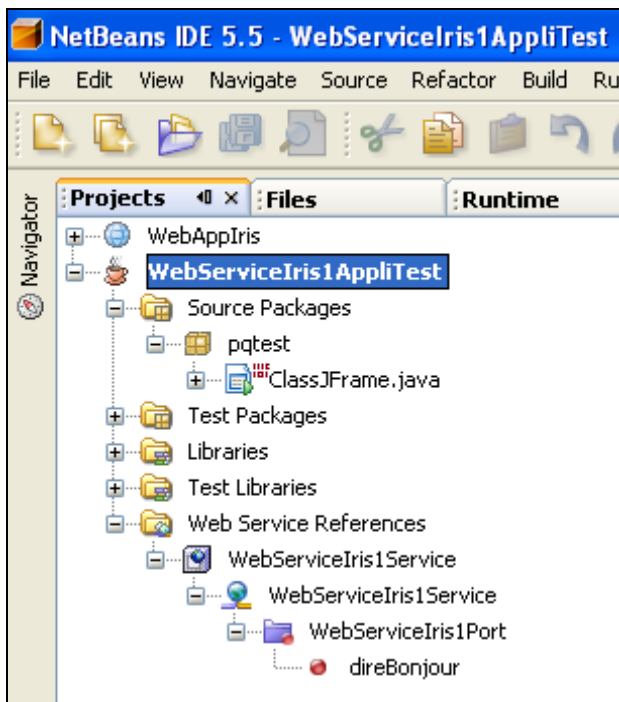


NetBeans génère le paquetage `pqtAccesWebServiceIris1`. Les classes générées se situent dans le dossier `\WebServiceIris1AppliTest\build\classes\pqtAccesWebServiceIris1`.

Les classes générées lors de l'importation sont les suivantes :

DireBonjour.class	1 Ko	Java Class
DireBonjourResponse.class	1 Ko	Java Class
ObjectFactory.class	2 Ko	Java Class
package-info.class	1 Ko	Java Class
WebServiceIris1.class	1 Ko	Java Class
WebServiceIris1Service.class	2 Ko	Java Class

Les méthodes du service web sont visibles dans l'inspecteur de projet dans le dossier « Web Service References ».



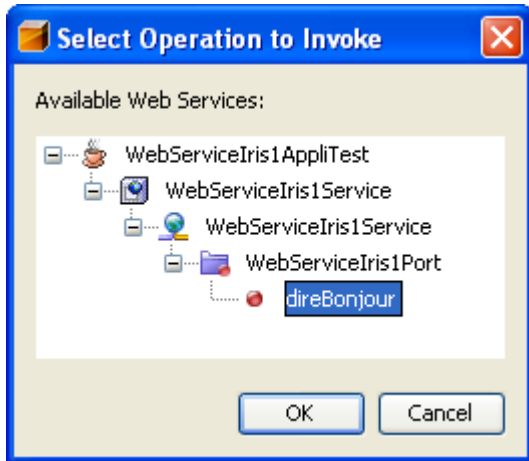
#### 5.4. Utilisation du service web dans l'application de test

Création de l'évènement « onClick » sur le bouton JButton1 :

Dans l'affichage Design, clic droit sur le bouton JButton1 > Events > Mouse > mouseClicked

L'appel de la méthode `direBonjour()` du service web peut se faire de deux façons :

- Cliquer – glisser la méthode à partir de l'inspecteur de projets vers le code source
- Clic droit dans le code > Web Service Client Resources > Call web Service Operation > Sélectionnez la méthode `direBonjour()` > OK



Voilà le code généré :

```
private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {  
  
    try { // Call Web Service Operation  
        pqtAccesWebServiceIris1.WebServiceIris1Service service = new  
            pqtAccesWebServiceIris1.WebServiceIris1Service();  
        pqtAccesWebServiceIris1.WebServiceIris1 port =  
            service.getWebServiceIris1Port();  
        // TODO initialize WS operation arguments here  
        java.lang.String prmNom = "";  
        // TODO process result here  
        java.lang.String result = port.direBonjour(prmNom);  
        System.out.println("Result = "+result);  
    } catch (Exception ex) {  
        // TODO handle custom exceptions here  
    }  
  
}
```

*Remarque : par souci de lisibilité, nous avons supprimé le nom du paquetage `pqtAccesWebServiceIris1` devant chaque objet et nous avons importé le paquetage au début du code source :*  
`import pqtAccesWebServiceIris1.* ;`

Modifiez le code pour prendre en compte les éléments de l'interface graphique :

```
private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {  
  
    try { // Call Web Service Operation  
        WebServiceIris1Service service = new WebServiceIris1Service();  
        WebServiceIris1 port = service.getWebServiceIris1Port();  
        // TODO initialize WS operation arguments here  
        String nom = jTextFieldNom.getText();  
        // TODO process result here  
        jLabelReponse.setText(port.direBonjour(nom)) ;  
    } catch (Exception ex) {  
        // TODO handle custom exceptions here  
    }  
  
}
```

*Remarque : Pour insérer automatiquement la gestion des exceptions : clic droit sur la ligne > Surround with Try-catch (génère aussi les importations de packages nécessaires)*

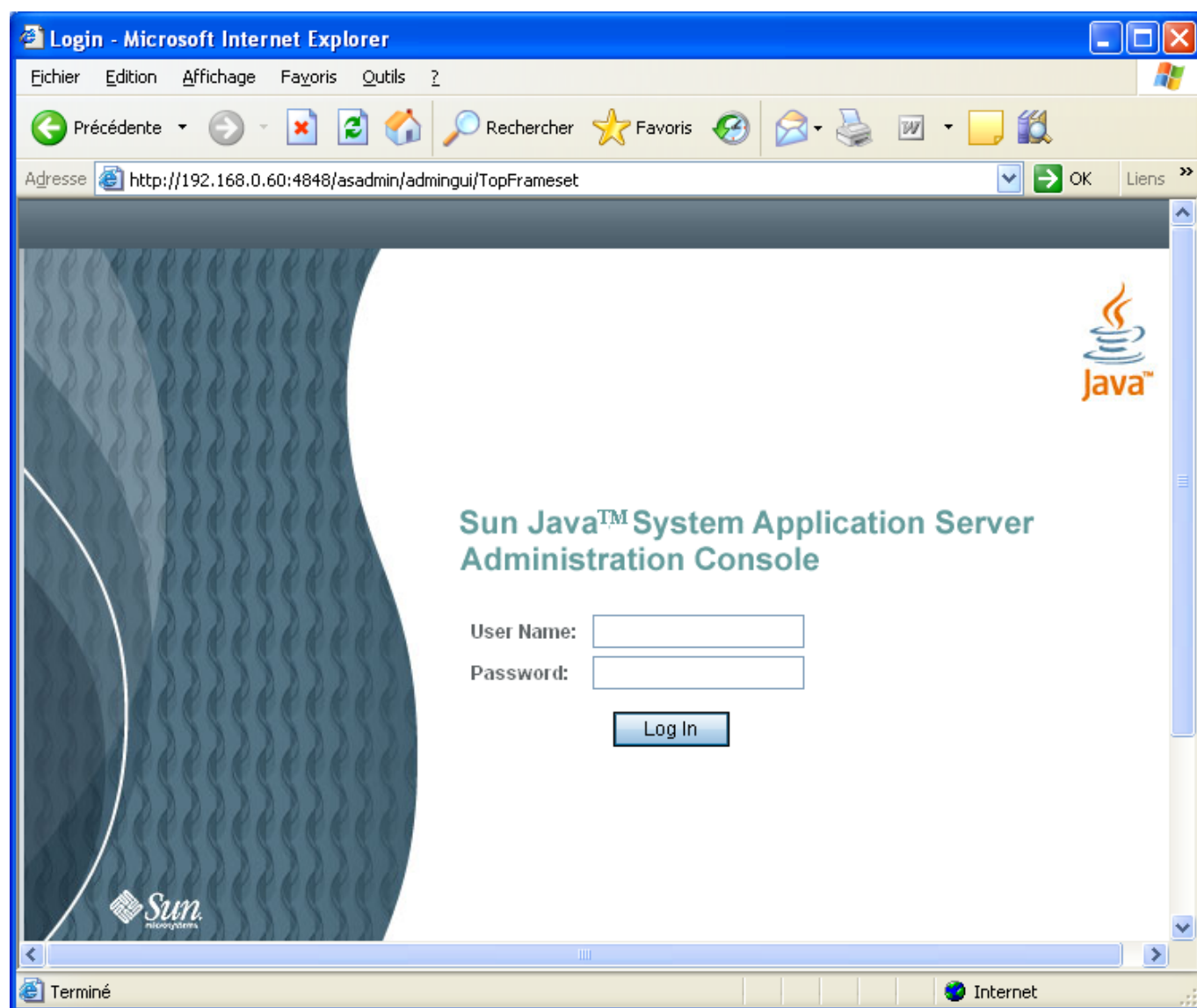
Capture d'écran en fonctionnement :



## 6. Déploiement sur un serveur d'application autonome

---

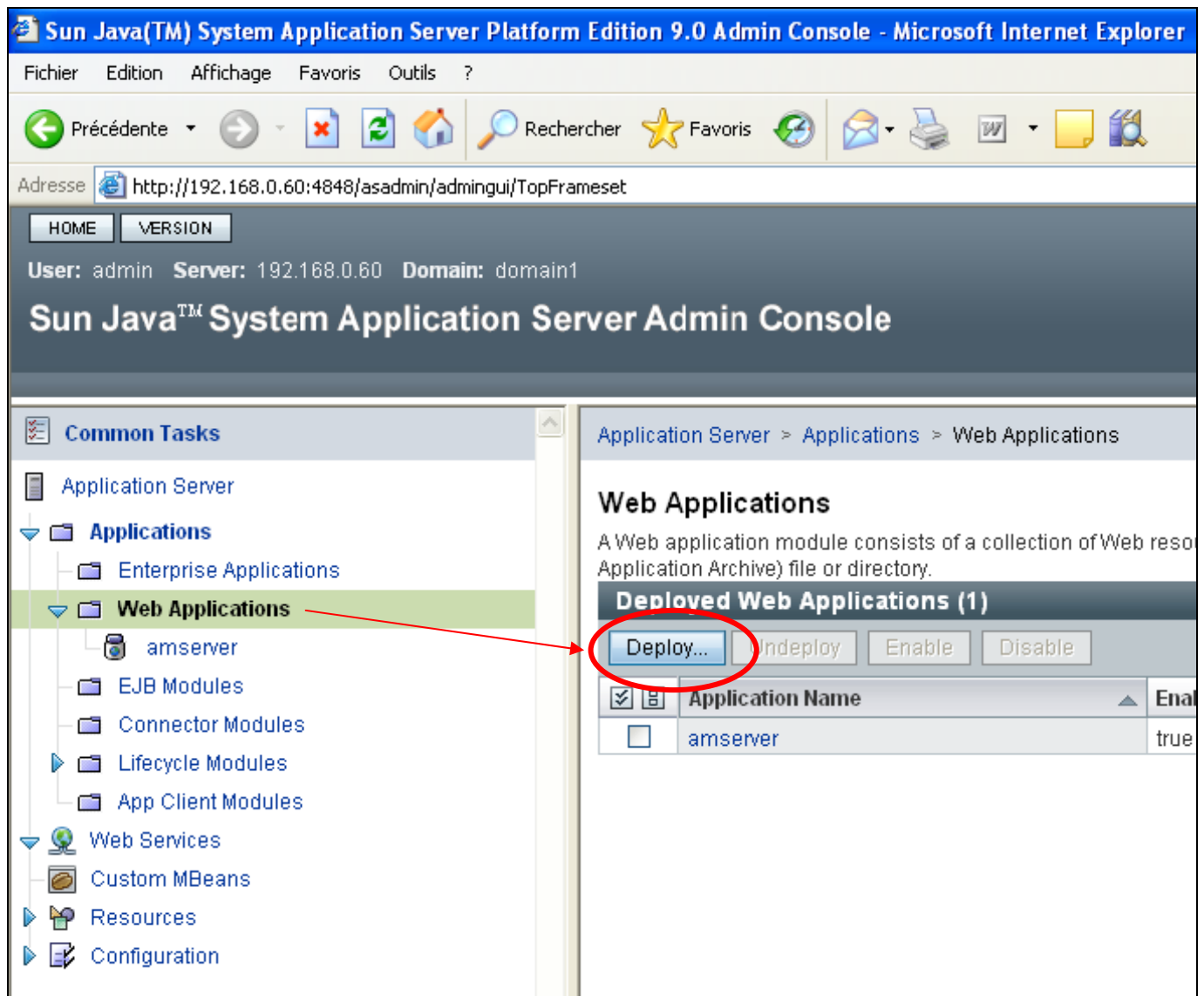
Notre serveur d'application de déploiement se trouve sur la machine distante 192.168.0.60. Dans un navigateur web, il faut se connecter sur le port 4848 pour accéder à l'interface d'administration du serveur d'application SJAS 9 :



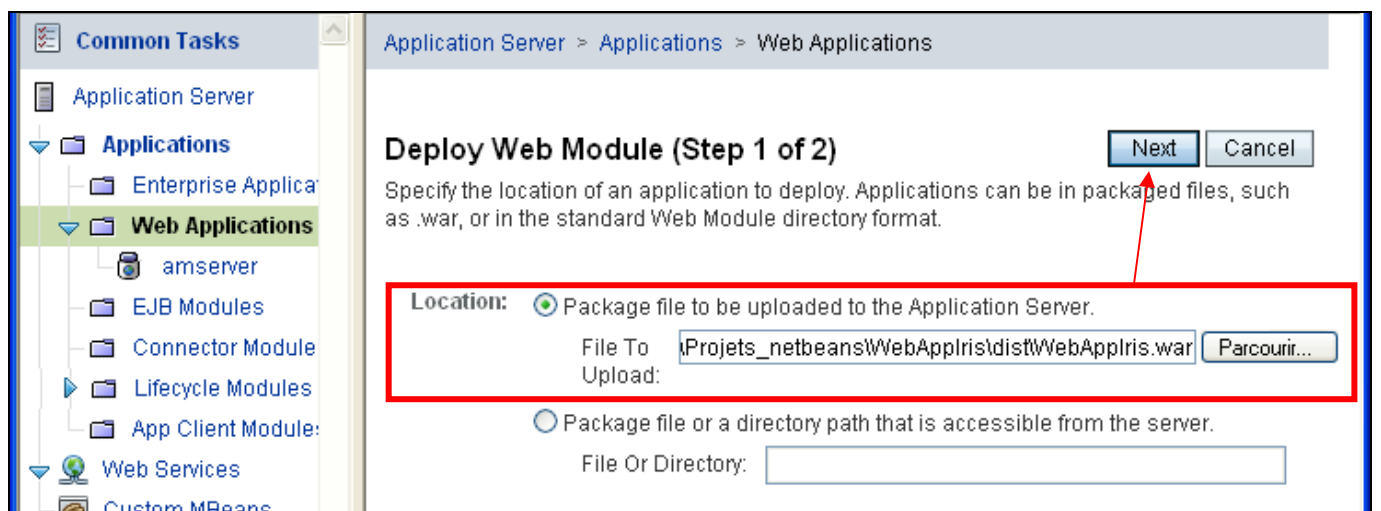
La configuration par défaut du serveur d'application est la suivante :

- User Name = admin
- Password = adminadmin
- Domaine par défaut = domain1

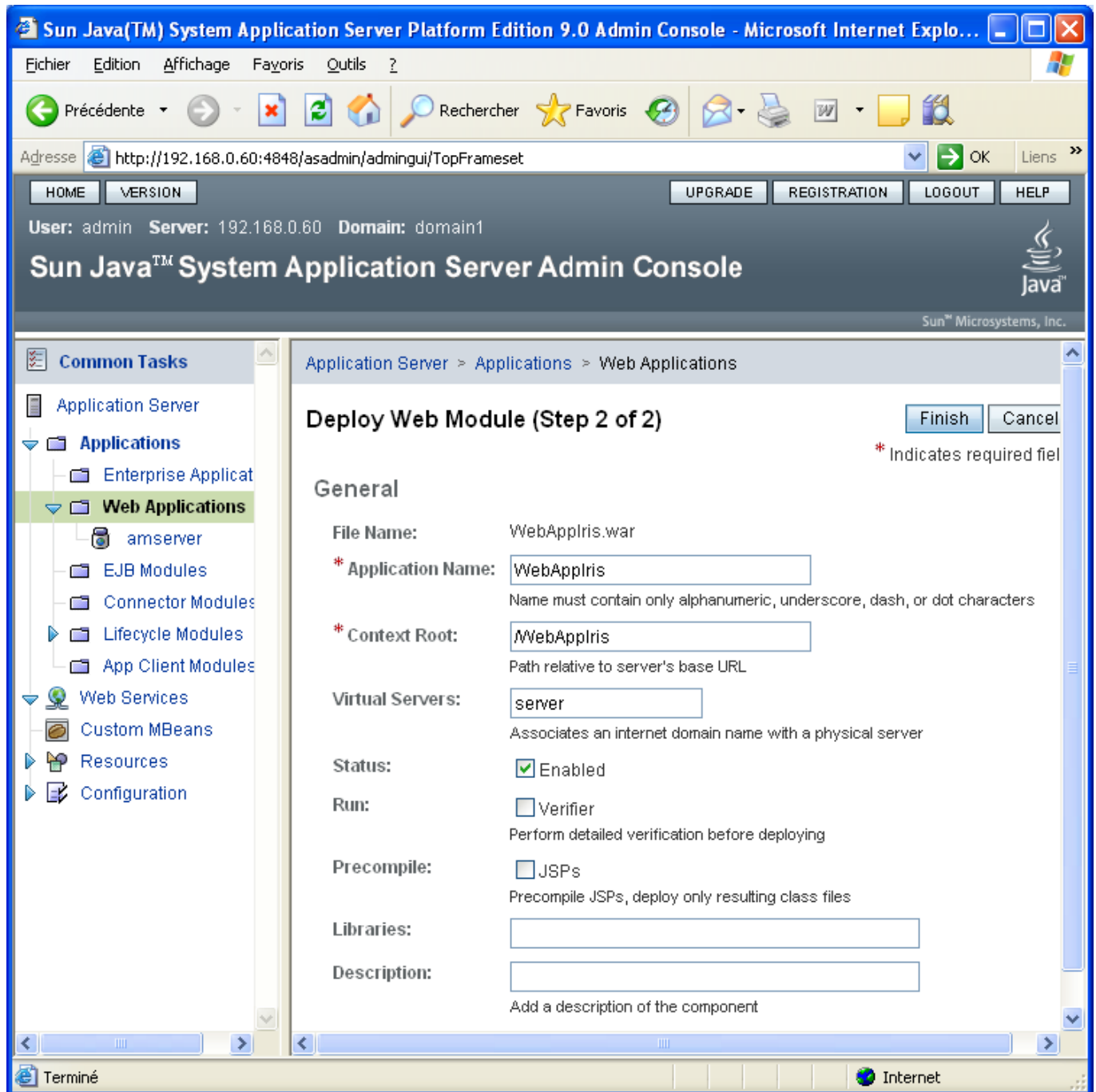
Cliquez sur Applications > Web Applications > Deploy



Il faut transférer sur le serveur d'application le fichier WAR du projet « WebAppIris » :  
 Cliquez sur Parcourir pour sélectionner dans le dossier NetBeans du projet du service web :  
 \WebAppIris\dist\WebAppIris.war



Cliquez sur Next



Cliquez sur Finish et vérifier le résultat :

The screenshot shows the Sun Java System Application Server Admin Console interface. The browser title is "Sun Java(TM) System Application Server Platform Edition 9.0 Admin Console - Microsoft Internet Explorer". The address bar shows "http://192.168.0.60:4848/asadmin/amingui/TopFrameset". The user is logged in as "admin" on server "192.168.0.60" in domain "domain1".

The left sidebar shows a tree view of the server configuration. Under "Applications", "Web Applications" is expanded, showing "amserver" and "WebAppIris". Red arrows point from these items in the tree to the table in the main content area.

The main content area shows "Web Applications" and a table of "Deployed Web Applications (2)". The table has columns for "Application Name", "Enabled", and "Context Root".

<input checked="" type="checkbox"/>	Application Name	Enabled	Context Root
<input type="checkbox"/>	amserver	true	amserver
<input type="checkbox"/>	WebAppIris	true	/WebAppIris

## 7. Utilisation du service web déployé sur le serveur distant