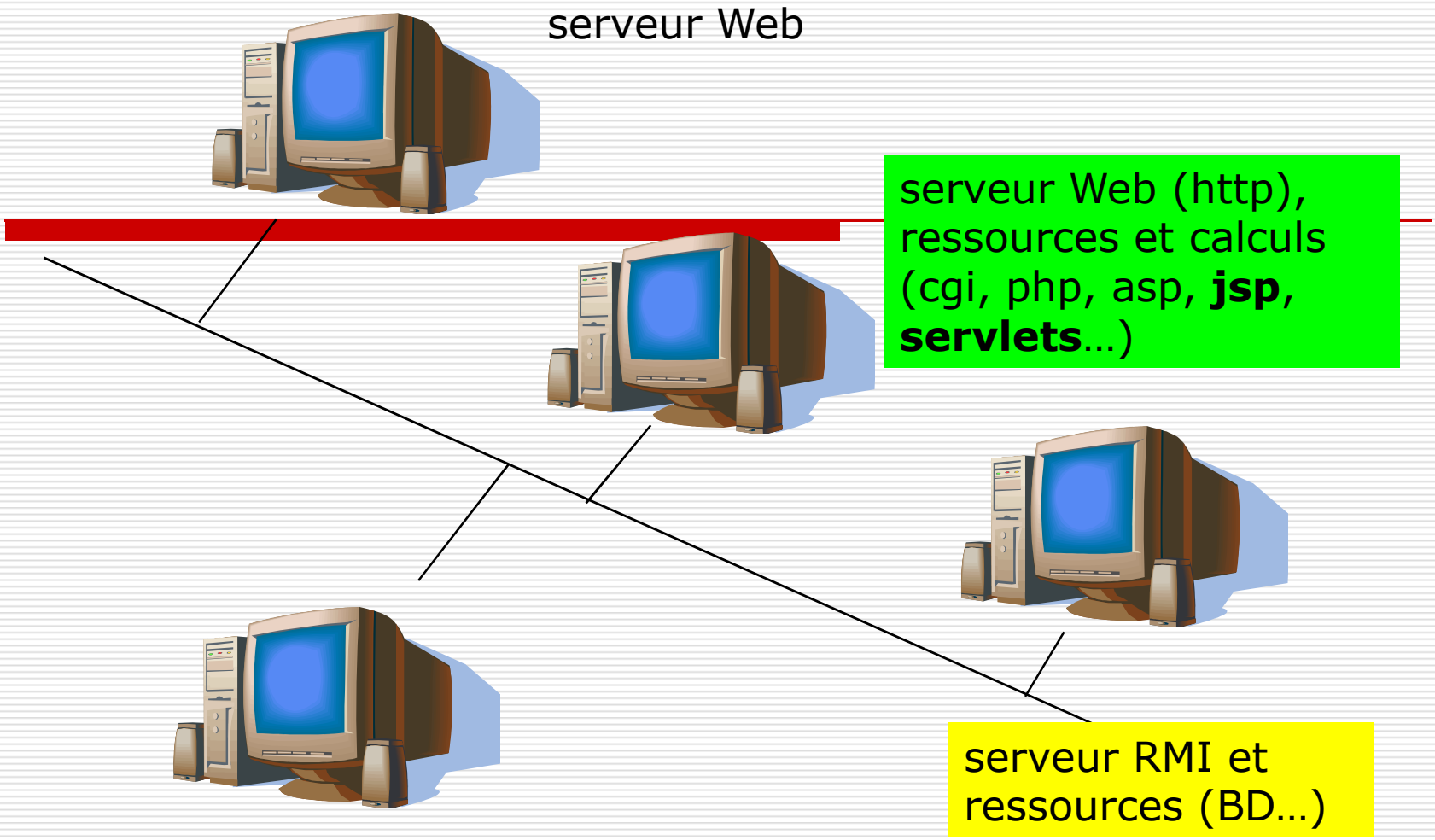
iUT ORSAY

Java et le Web

Jean-Pierre Fournier, 2008

http://www.iut-orsay.fr/~fournier


serveur Web
serveur Web (http), ressources et calculs (cgi, php, asp, **jsp**, **servlets**…)
serveur RMI et ressources (BD…)
client Web et/ou RMI

Java et RMI

- RMI = remote method invocation
  - La possibilité de faire exécuter un sous-programme par un autre ordinateur
    - Pour utiliser sa vitesse de calcul,
    - Pour se consacrer à d'autres tâches côté client,
    - Pour exploiter à distance des ressources plutôt que de les déplacer ou copier
    - Pour répartir des travaux simultanés entre plusieurs serveurs
    - …

Java et RMI…Étape 1

- Définir dans une interface les méthodes que le serveur implémentera (ou obtenir une copie de cette interface si le serveur est déjà réalisé)

```
import java.rmi.*;

/* le serveur RMI qui implémente cette méthode
affiche chez lui le texte donné et retourne sa
longueur */
interface MesMethodesCoteServeur extends Remote{
    int affiche(String donnee) throws RemoteException;
    }
```

Java et RMI…Etape 2

- Réaliser du côté du serveur une implémentation des méthodes annoncées par l'interface
- Dans cet exemple, le serveur et la classe implémentant les méthodes sont confondus, mais ce n'est pas nécessaire (ni conseillé)

```
import java.rmi.*;
import java.rmi.server.*;

class ServeurRMI implements MesMethodesCoteServeur{

    public int affiche(String donnee)throws RemoteException{
        System.out.println("Ici le service. J'ai recu le texte : "+donnee+"
    envoye par un client.");
        return donnee.length();
        }

    public static void main(String[] args) throws Exception{
        System.out.println("Le serveur est actif.");
        ServeurRMI uneImplementation = new ServeurRMI();
        UnicastRemoteObject.exportObject(uneImplementation);
        Naming.rebind("/ImplementationDistante", uneImplementation);
        System.out.println("ImplementationDistante est exportée et publié
        }
}
```

Java et RMI…Etape 3

- Réaliser un client, qui utilise les méthodes de l'interface :

```
import java.rmi.*;

public class Client{
    public static void main(String [] args)
        throws Exception{
        MesMethodesCoteServeur unLien=
                (MesMethodesCoteServeur) Naming.
                        lookup("rmi://"+args[0]+
                                "/ImplementationDistante");
    System.out.println("J'ai fait afficher un texte de
    longueur "+unLien.affiche(args[1])+" par le serveur.");
        }
    }
```

Java et RMI…Etape 4 (et fin)

- Faire compiler les 3 éléments (ou les 4 si la classe contenant l'implémentation est distincte de la classe principale du serveur)
  - **`javac MesMethodes.java`**
  - **`javac Client.java`**
  - **`javac Serveur.java`**

Inutile sous Eclipse (déjà fait automatiquement)

- Adapter la classe d'implémentation aux liaisons rmi :
  - **`rmic ServeurRMI`**

Sous Eclipse, un plug-in automatise cela aussi

- Lancer en arrière-plan le système d'enregistrement :
  - **`Rmiregistry`**
  - Il est également possible, si un seul service est envisagé, de glisser, dans le main du serveur :

    `LocateRegistry.createRegistry(1099);` *// port par défaut*

- Lancer le serveur, puis le client…