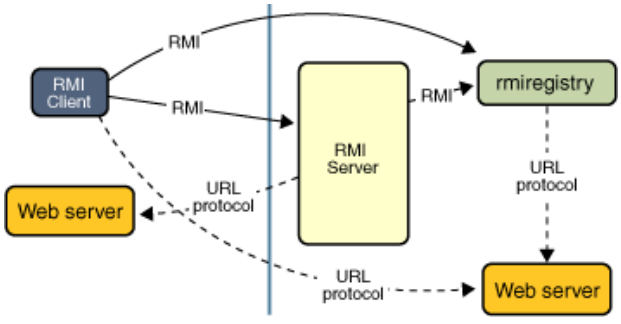
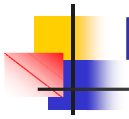
Programowanie w języku Java

Wykład 5: Programowanie rozproszone: RMI.

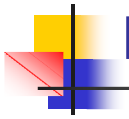
RMI (Remote Method Invocation)





Deklaracja zdalnych obiektów

- Pakiet: java.rmi
- Zdalny obiekt jest widziany poprzez odległy interfejs:
  - **publiczny**
  - **rozszerzający java.rmi.Remote**
  - **wszystkie metody generują wyjątek java.rmi.RemoteException**
  - **każdy obiekt przekazywany jako argument też jest widziany jako odległy interfejs**



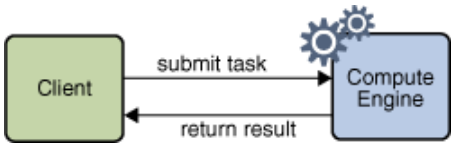
Dostęp do zdalnych obiektów

- Na lokalny komputer przesyłana jest namiastka (stub) zdalnego obiektu
- Namiastka przekierowuje wszystkie wywołania metod do zdalnego obiektu
- Tylko metody definiowane przez zdalny interfejs są dostępne w zdalnym obiekcie
- Transmisja lokalnych obiektów będących parametrami poprzez serializację

Projektowanie aplikacji RMI

- Projekt interfejsu
- Implementacja zdalnych obiektów
- Implementacja aplikacji klienta

Projekt zdalnego interfejsu




```
package compute;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Compute extends Remote {
    <T> T executeTask(Task<T> t) throws
     RemoteException;
}
```

```
package compute;

public interface Task<T> {
    T execute();
}
```

Implementacja zdalnych obiektów

- Deklaracja implementowanych interfejsów:

```
public class ComputeEngine implements Compute
```

- Implementacja zdalnych metod:

```
public <T> T executeTask(Task<T> t) {
return t.execute();
}
```

- Inicjalizacja zdalnych obiektów

Inicjalizacja zdalnego obiektu

- Instalacje menedżera bezpieczeństwa:

```
if (System.getSecurityManager() == null) {
        System.setSecurityManager(new SecurityManager()); }
```

- Utworzenie obiektu i udostępnienie go klientom:

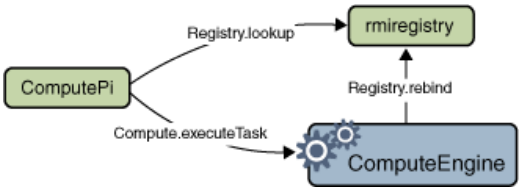
```
Compute engine = new ComputeEngine();
Compute stub = (Compute)
        UnicastRemoteObject.exportObject(engine, 0);
```

Port

- Udostępnienie namiastki obiektu poprzez rejestr RMI:

```
Registry registry = LocateRegistry.getRegistry();
registry.rebind("Compute", stub);
```

Implementacja aplikacji klienta



- Instalacja menedżera bezpieczeństwa:
- Uzyskanie dostępu do zdalnego obiektu:

```
Registry registry = LocateRegistry.getRegistry(”serwer.pl”);
Compute comp = (Compute) registry.lookup(”Compute”);
```


Wywołanie zdalnej metody

```
Pi task = new Pi(250);
BigDecimal pi = comp.executeTask(task);
```

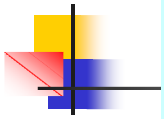
Wynik: Liczba PI obliczona z dokładnością 250 miejsc po przecinku.

$$\frac{\pi}{4} = 4 \cdot \arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right)$$

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \ldots + (-1)^{m-1}\frac{x^{2m-1}}{2m-1} + o(x^{2m})$$

```
package engine;

import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import compute.Compute;
import compute.Task;

public class ComputeEngine implements Compute {

   public ComputeEngine() {
      super();
   }

   public <T> T executeTask(Task<T> t) {
      return t.execute();
   }

   public static void main(String[] args) {
      if (System.getSecurityManager() == null) {
         System.setSecurityManager(new SecurityManager());
      }
      try {
         String name = "Compute";
         Compute engine = new ComputeEngine();
         Compute stub =
            (Compute) UnicastRemoteObject.exportObject(engine, 0);
         Registry registry = LocateRegistry.getRegistry();
         registry.rebind(name, stub);
         System.out.println("ComputeEngine bound");
      } catch (Exception e) {
         System.err.println("ComputeEngine exception:");
         e.printStackTrace();
      }
   }
}
```

11



```
package client;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.math.BigDecimal;
import compute.Compute;

public class ComputePi {
   public static void main(String args[]) {
      if (System.getSecurityManager() == null) {
         System.setSecurityManager(new SecurityManager());
      }
      try {
         String name = "Compute";
         Registry registry = LocateRegistry.getRegistry(args[0]);
         Compute comp = (Compute) registry.lookup(name);
         Pi task = new Pi(Integer.parseInt(args[1]));
         BigDecimal pi = comp.executeTask(task);
         System.out.println(pi);
      } catch (Exception e) {
         System.err.println("ComputePi exception:");
         e.printStackTrace();
      }
   }
}
```

```
package client;

import compute.Task;
import java.io.Serializable;
import java.math.BigDecimal;

public class Pi implements Task<BigDecimal>,
Serializable {
   private static final long serialVersionUID = 227L;
   private static final BigDecimal FOUR =
      BigDecimal.valueOf(4);
   private static final int roundingMode =
      BigDecimal.ROUND_HALF_EVEN;
   private final int digits;

public Pi(int digits) {
      this.digits = digits;
   }

public BigDecimal execute() {
      return computePi(digits);
   }

public static BigDecimal computePi(int digits) {
      int scale = digits + 5;
      BigDecimal arctan1_5 = arctan(5, scale);
      BigDecimal arctan1_239 = arctan(239, scale);
      BigDecimal pi =
arctan1_5.multiply(FOUR).subtract(
                  arctan1_239).multiply(FOUR);
      return pi.setScale(digits,
                  BigDecimal.ROUND_HALF_UP);
   }
```

```
public static BigDecimal arctan(int inverseX,
                  int scale)
   {
      BigDecimal result, numer, term;
      BigDecimal invX =
         BigDecimal.valueOf(inverseX);
      BigDecimal invX2 =
         BigDecimal.valueOf(inverseX * inverseX);

      numer = BigDecimal.ONE.divide(invX,
                  scale, roundingMode);

      result = numer;
      int i = 1;
      do {
         numer =
            numer.divide(invX2, scale, roundingMode);
         int denom = 2 * i + 1;
         term =
            numer.divide(BigDecimal.valueOf(denom),
                  scale, roundingMode);
         if ((i % 2) != 0) {
            result = result.subtract(term);
         } else {
            result = result.add(term);
         }
         i++;
      } while (term.compareTo(BigDecimal.ZERO)
                  != 0);
      return result;
   }
}
```

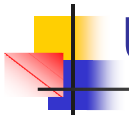

Uruchomienie serwera

- Plik: server.policy:

```
grant codeBase "file:/home/sd/serwer" {
   permission java.security.AllPermission;
};
```

- Uruchomienie rejestru RMI:
  - start rmiregistry 1099
- Uruchomienie programu:

```
java -cp c:\home\sd\src;c:\home\sd\public_html\classes\compute.jar
-Djava.rmi.server.codebase=file:/c:/home/sd/public_html/classes/compute.jar
-Djava.rmi.server.hostname=serwer.pl
-Djava.security.policy=server.policy engine.ComputeEngine
```



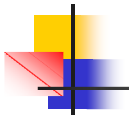
Uruchomienie klienta

- Plik: client.policy:

```
grant codeBase "file:/home/sd/client" {
    permission java.security.AllPermission;
 };
```

- Uruchomienie klienta:

```
java -cp
c:\home\sd\src;c:\home\sd\public_html\classes\compute.jar -
Djava.rmi.server.codebase=file:/c:/home/sd/public_html/classes
/ -Djava.security.policy=client.policy client.ComputePi serwer.pl
45
```

Koniec