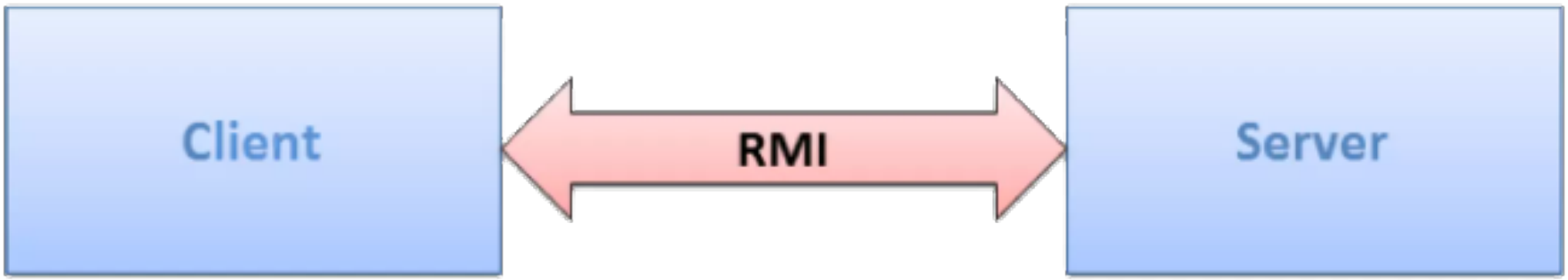
RMI

autoři: **Bc. Radek Novotný, Bc. André Vágner**
fakulta: Fakulta jaderná a fyzikálně inženýrská
České vysoké učení technické v Praze

Co je RMI?

- API, které umožňuje volání metod objektu na vzdáleném Java virtuálním stroji.
- Umožňuje vytvořit distribuované aplikace a pracovat s objekty jako by byly lokální.
- Rozhraní a třídy definovány v balíčku java.rmi.
- Poprvé se objevilo v JDK 1.1 (Java Development Kit).



Definice

- Vzdálený objekt – objekt, jehož metody lze zavolat z jiné JVM.
  - Popsán pomocí rozhraní napsaných v Javě.
- RMI – volání metody vzdáleného rozhraní na vzdáleném objektu.
  - Stejná syntaxe jako volání metody na lokálním objektu.

Cíle

- hladká podpora vzdáleného volání metod objektů v různých virtuálních strojích,
- podpora zpětného volání ze serveru do aplikace,
- přirozená integrace distribuovaného modelu objektů do Javy,
- zjednodušení psaní spolehlivých distribuovaných aplikací,
- zachování kontroly typů poskytovaných běhovým prostředím Javy.

Nevýhody

- Závislost na JVM (řešení: CORBA – RMI-IIOP)
- Blokování komunikace RMI (řešení: REST, SOAP a RMI přes HTTP)

Distribuované objektové aplikace

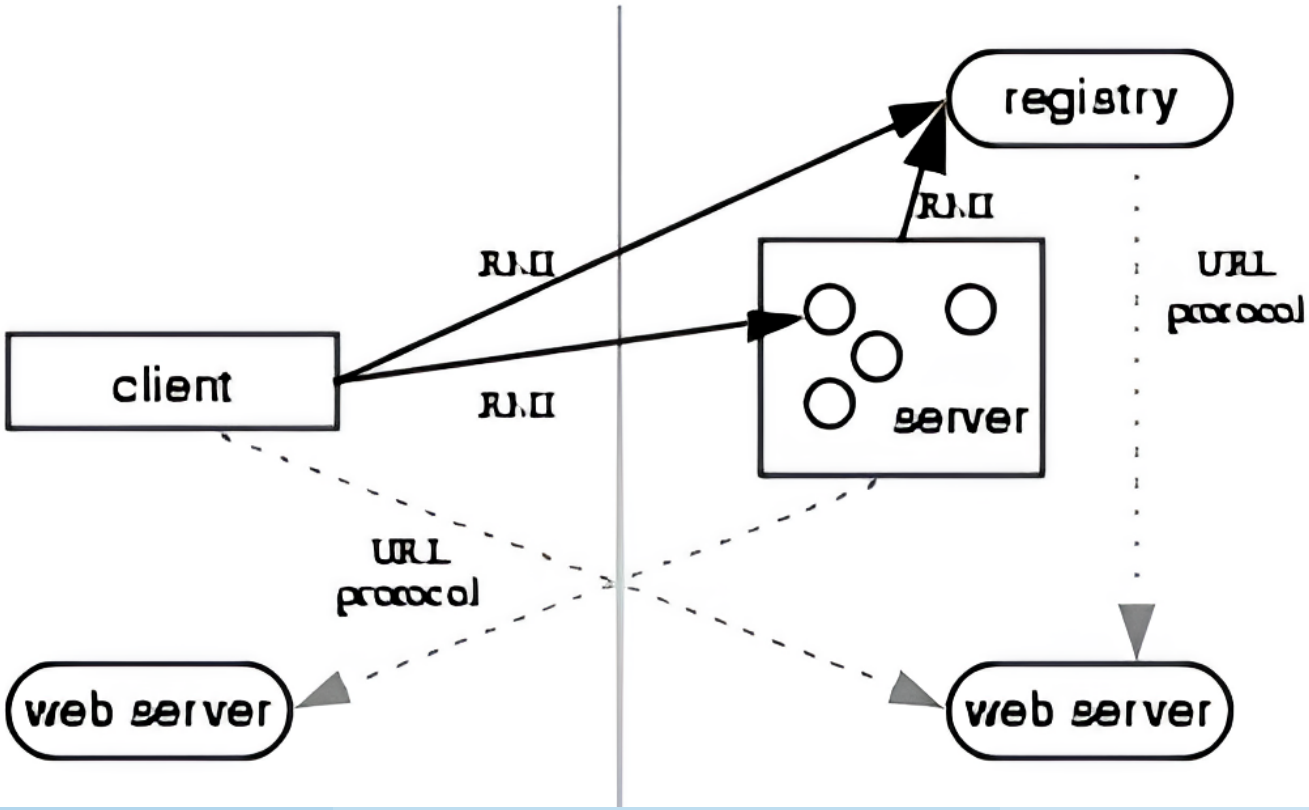
- Klient - Server
- RMI poskytuje mechanismu, pomocí kterého server a klient komunikují a posílají si informace $\Rightarrow$ distribuovaná objektová aplikace.

Distribuovaná objektová aplikace vyžaduje:

- nalézt vzdálené objekty,
- komunikovat se vzdálenými objekty,
- načíst bytecode třídy objektu, který je předán jako parametr nebo vracející hodnota.

Distribuované objektové aplikace - průběh

1. Server zavolá registry, aby přiřadil jméno k danému vzdálenému objektu,
2. klient vyhledá vzdálený objekt podle jména v registrech serveru, a poté na něj zavolá metodu.



Distribuovaný vs objektový model - podobnosti

- Odkaz lze vzdálenému objektu předat jako argument nebo vrátit jako výsledek v jakémkoliv volání metody (lokálně i vzdáleně).
- Vzdálený objekt lze přetypovat na jakékoliv vzdálené rozhraní podporované implementací.
- *instanceof* operátor lze využít na testování vzdáleného prostředí podporovaného vzdáleným objektem.

Distribuovaný vs objektový model - rozdíly

- Klienti vždy interagují se vzdáleným rozhraním, nikdy se třídami, které dané rozhraní implementují.
- Lokální argumenty z a do RMI jsou vždy předávány kopií.
- Vzdálené objekty jsou předávány odkazem, nikoliv kopírováním vzdálené implementace.
- Je potřeba ošetřit dodatečné výjimky, které mohou nastat při volání vzdálených metod.

Rozhraní java.rmi.Remote

- Definice sad metod, které lze zavolat ze vzdálené JVM
- Musí splňovat:
  - rozhraní rozšiřuje rozhraní *java.rmi.Remote* nebo jeho následníka,
  - metody očekávají výjimku *java.rmi.RemoteException*,

```
public interface MojeRozhrani extends java.rmi.Remote {
        public void MojeMetoda()
                throws java.rmi.RemoteException;
}
```

Výjimka java.rmi.RemoteException

Případy, kdy je vyhozena:

- chyba komunikace,
- chyba během převodu nebo zpětného převodu parametrů či návratové hodnoty,
- chyba protokolu.

Třída RemoteObject

Serverové funkce RMI

- java.rmi.server.RemoteObject
  - implementace metod java.lang.Object (hashCode, equals a toString)
  - java.rmi.server.RemoteServer
  - java.rmi.server.UnicastRemoteObject
    - singleton (jedináček)
  - java.rmi.activation.Activatable

Předávání parametrů v RMI

Argumenty a vracející hodnoty

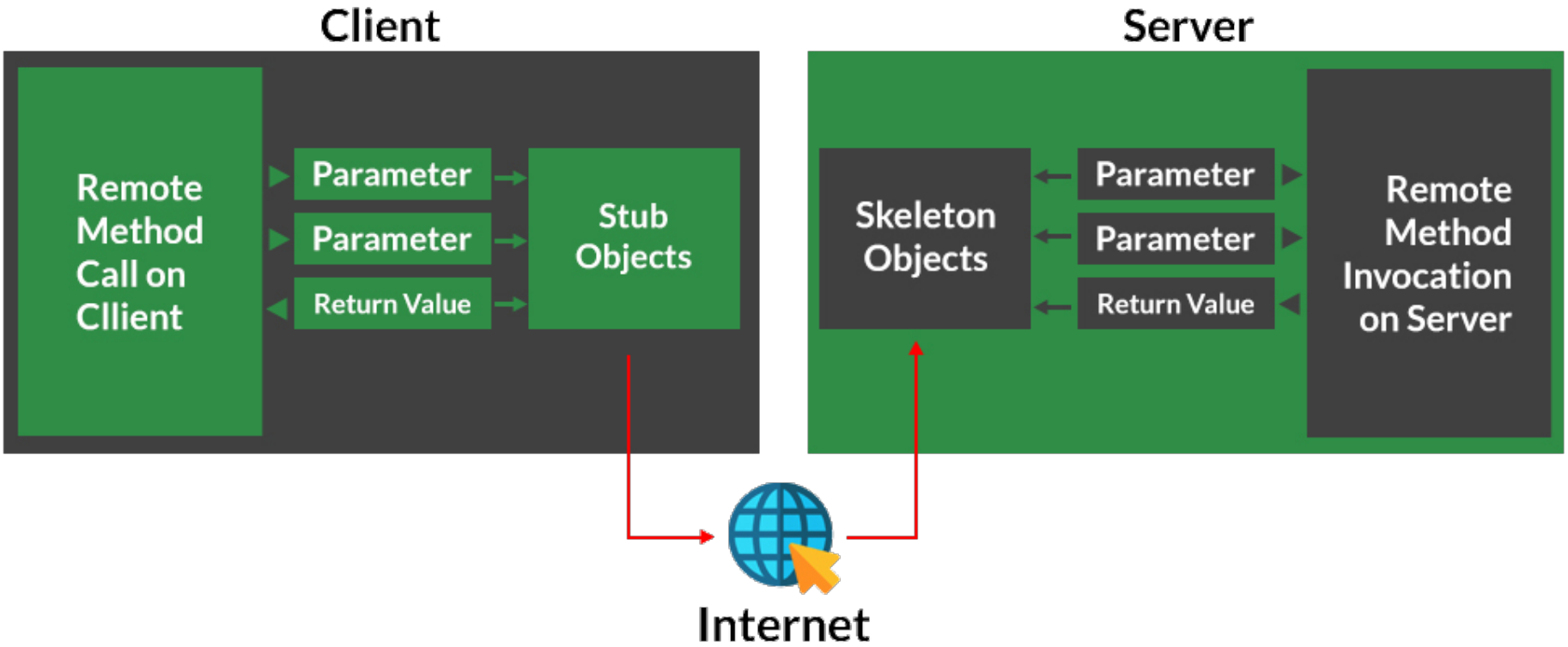
- Serializovatelné – implementují java.io.Serializable.

Předávání vzdálené metodě:

- Nevzdálené objekty
  - Kopírováním
  - Jako argument: kopie před zavoláním
  - Vrácení hodnoty: vytvoření nového objektu v zavolané JVM
- Vzdálené objekty
  - Stub

Architektura

- Klient - Server
- Stuby, Kostry, Remote reference a Transport.
- Stub na klientské straně zajišťuje komunikaci s objektem na serveru.
- Kostra na serverové straně zajišťuje volání metod objektu.



Stuby a kostry

- Stub – lokální reprezentant serveru (proxy vzdáleného objektu).
- Volající zavolá metodu na lokálním stubu.
- Stub implementuje stejná rozhraní jako implementuje vzdálený objekt.

Průběh volání metody stubu:

- naváže spojení s JVM obsahující vzdálený objekt,
- serializuje parametry a přenese je,
- čeká na výsledek vzdálené metody,
- deserializuje návratovou hodnotu nebo výjimku,
- vrátí hodnotu volajícímu.

Stub – skrývá serializaci parametrů a komunikaci na úrovni sítě.

Stuby a kostry

Ve vzdálené JVM může mít každý stub odpovídající kostru.

- Java 2 – kostry nejsou potřeba.

Kostra na straně vzdáleného JVM přijímá volání a předává ho vzdálenému objektu. Průběh:

- deserializuje parametry metody,
- vyvolá metodu na vzdáleném objektu,
- serializuje návratovou hodnotu nebo výjimku a odešle ji zpět.

RMI přes HTTP

- Transportní vrstva RMI
  - Přímá komunikace přes sockety – problém s firewall
  - Komunikace přes HTTP
- Proces volání přes HTTP protokol:
  - HTTP POST požadavek (request).
  - odpověď HTTP (response).
- *java.rmi.server.disableHttp* přepínač.

Problémy RMI přes HTTP

- HTTP požadavky – alespoň o řád pomalejší.
- Jednostranná komunikace.

Příklad

- rmiRozhrani.java – metoda pozdrav
- Klient.java
- Server.java

Příklad – volání

```
start java -classpath . -Djava.rmi.server.codebase=file:./ Server "
    Ahoj %s, zprava od Serveru 1"

Server poslouchá na portu 1098
odpov d m Klient1 # pri zavolani metody
#-------------------------------------------------------------------
start java -classpath . -Djava.rmi.server.codebase=file:./ Server "
    Ahoj %s, zprava od Serveru 2" 1099

Server poslouchá na portu 1099
odpov dam Klient2 # pri zavolani metody
#-------------------------------------------------------------------
java  -classpath . Klient Klient1

Volani metody serveru...
Odpoved serveru: Ahoj Klient1, zprava od Serveru 1
#-------------------------------------------------------------------
java  -classpath . Klient Klient2 localhost 1099

Volani metody serveru...
Odpoved serveru: Ahoj Klient2, zprava od Serveru 2
```

Děkujeme za pozornost