Overview

- RMI mechanism
- Web services
RMI

- Overview of RMI
- Infrastructure
- How to create a RMI server/client
RMI overview

- Allow distributed computing among objects
  - Limited functionalities (RPC among objects)

- Part of core JDK platform since 1.1
  - java.rmi package

- Similar technologies
  - Microsoft .NET Remoting
  - RPC
  - CORBA
  - SOA
RMI overview

- Allow communications between two JVMs over a network

- The distribution is made at the object level
  - “seamless” connection from a client to a remote server

- Relies on serialization
  - Custom classes have to implement the java.io.Serializable interface
  - Primitive types are sent by value
RMI Infrastructure

1: create Remote object
2: register "registrationName"

Client JVM

Server JVM

Registry JVM
RMI Infrastructure

Client JVM

Client → Stub

Server

Skeleton → Remote object

Client JVM

3: ask for registrationName

"registrationName"

4: returns

Register

Registry JVM
5: method invocation and return

Client JVM

Client → Stub → Skeleton → Remote object

Server

Client JVM

“registrationName”

Register

Registry JVM
RMI registry

- Used for looking up Objects
- Servers register their Objects
- Clients use to find and obtain remote references
- Runs on port 1099 by default
How to create a RMI server/client?

- Declare a remote interface
  - extends java.rmi.Remote

- Define a class that implements the remote Interface
  - extends java.rmi.RemoteObject or java.rmi.UnicastRemoteObject
Remote interface declaration

```java
public interface BankAccount extends java.rmi.Remote {
    public void deposit(float amount) throws java.rmi.RemoteException;
    public void withdraw(float amount) throws OverdrawnException,
        java.rmi.RemoteException;
    public float getBalance() throws java.rmi.RemoteException;
}
```
Distributed Garbage Collector

- Local references counted normally
- Clean/dirty state for the remote references
  - java.rmi.server.Unreferenced for notifications when no more clients
- "lease" for a remote reference
  - Duration (java.rmi.dgc.leaseValue to configure, default to 10 minutes)
- Consequence
  - Remote reference must not be trusted
Summary

- RMI provides a “basic” mechanism
  - Distributed garbage collection
  - marshalling/unmarshalling

- Limitations
  - Synchronous calls
    - Asynchronous notifications
  - Lack high-level services
    - persistence, messaging, thread management, logging
  - No way to set a threading policy
  - Servers have to be thread safe
Web services - overview

- Independent of the implementation used
  - e.g. a .net client can access to a java server

- Main technologies
  - XML (Extensible Markup Language)
  - SOAP (Simple Object Access Protocol) allow client to call remote service. The msg format is XML.
  - WSDL (Web Services Description Language)
  - UDDI (Universal Description, Discovery, and Integration) a standard used for publishing/query web services
  - JAX-WS : an API to develop web services in JAVA
Web services architecture

- UDDI
- WSDL document
- Web service provider
- Client
- WSIL
WSDL

- Describe the methods provided by the web service
- Can be requested through a http request
  - e.g.: http://services.aonaware.com/DictService/DictService.asmx?wsdl
- Allow the generation of the client code
Java declaration for class named “Simple”

```java
public String hello(String callerName) {
    return "Hello " + callerName;
}
```

WSDL declaration

- Declaration of exchanged messages
  ```xml
  <message name="hello">
    <part name="parameters" element="tns:hello"/>
  </message>
  ```

- Declaration of available “ports”
  ```xml
  <binding name="SimplePortBinding" type="tns:Simple">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
      <soap:operation soapAction=""/>
      <input> <soap:body use="literal"/> </input>
      <output> <soap:body use="literal"/> </output>
    </operation>
  </binding>
  ```
Allow to marshall/unmarshall the data to be transmitted

Example of a request

POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
Implementation of a web service with eclipse

- Bottom-up approach
  - Extract the wsdl from a .java
- Top-Down
  - Build a skeleton from a wsdl declaration
A bottom-up approach

- A Java class implementing a service:
- The server and client-side code
Bottom-up approach

```java
package fr.inria.dream.test_ws;

import java.util.Date;

public class SimpleServiceImpl {
    public String hello(String callerName) {
        return "Hello " + callerName;
    }

    public int addition(int a, int b) {
        System.out.println("Ajout de " + a + " et de " + b);
        return a + b;
    }

    public Date getServerTime() {
        System.out.println("getServerTime invoked !!!");
        return new Date();
    }

    public String helperMethod() {
        return "Just for the sample";
    }
}
```
Bottom-up approach
Bottom-up approach
Bottom-up approach
Bottom-up: generation of a client

```java
package fr.inria.dream.test_ws;

import java.rmi.RemoteException;

public class ClientExample {

    /**
     * @param args
     */
    public static void main(String[] args) {
        SimpleServiceImpl proxy = new SimpleServiceImplProxy();
        try {
            System.out.println("call to hello = " + proxy.hello("mr test"));
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

}```
Summary

- Data transmission: the classes used for the data must be
  - Public
  - With a constructor without parameter
  - With get/set methods for each property
  - Consequence of the heterogeneity (no serialization possible in general)

- Synchronous calls
Top-Down: build a client from an existing service 1/3
Top-Down: build a client from an existing service 2/3
Top-Down: build a client from an existing service 3/3
View of the list with a browse
Questions?