자바 네트워크(Java Network)

JAVA RMI(Remote Method Invocation)

"원격 메소드 호출" 이란!

강의자료/소스코드는 다음 URL 에서 확인 하세요

http://ojc.asia

Java Remote Method Invocation

분산 컴퓨팅 이란?

- 네트워크에서 서로 다른 시스템 간에 응용 프로그램을 분산해서 처리하는 환경을 말한다.
- 즉 하나의 컴퓨터에 존재하는 Application이나 프로세스에서 스스로 처리하거나 수행하기 어려운 작업을 다중 프로세서나 컴퓨터에 분산시키는 것이다.
- 분산 컴퓨팅을 적용한 Application을 Distributed Application 이라고 한다.

분산 객체(Distributed Object)

- 분산 컴퓨팅 기술이 객체 지향과 접목되어 하나의 프로세서나 컴퓨터에서 실행되는 객체가 다른 프로세서나 컴퓨터에서 객체와 통신이 가능 하도록 하는 기술이 분산 객체 기술이다.
- RMI는 java-to-java
- 분산 객체란 자신이 존재하는 런타임 환경과는 다른 런타임에 있는 객체와 통신이 가능한 객체이다.

JAVA RMI

RMI(Remote Method Invocation)

- 직렬화 된 Java 클래스의 전송, 원격 프로시저 호출 (RPC)과 같은 객체 지향적 인 원격 메서드 호출을 수행하는 Java API
- 기존 자바언어의 장점과 풍부한 API를 분산 객체 기술에 이용이 가능하게 됨
- 다른 실행 환경에 있는 객체의 메소드를 로컬에서 생성한 객체의 메소드와 다름 없이 호출할 수 있도록 하는 “자바의 분산 객체 기술”이다.

RMI(Remote Method Invocation)

- RMI를 사용하면 객체가 다른 JVM에서 실행중인 객체의 메서드를 호출 할 수 있다.
- RMI 스펙에 대한 구현은 JRMP (Java Remote Method Protocol) 이다.
- Distributed computing model을 구현하기 위한 자바 Object 간의 통신을 구현한 도구이다.
- RMI는 두 개의 객체 스텁 및 스켈레톤을 사용하여 애플리케이션 간의 원격 통신을 제공한다.

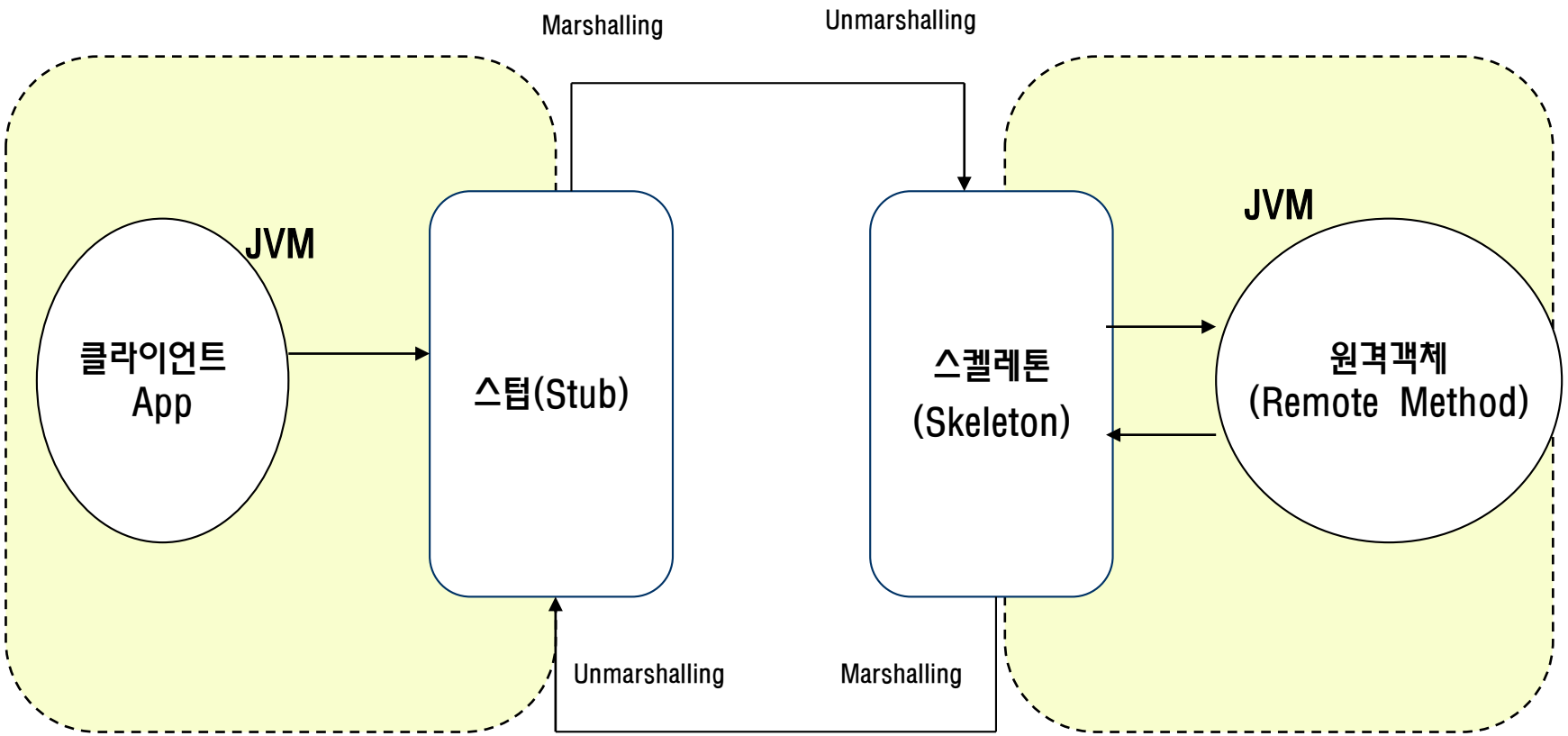
스텁(Stub)

- 스텁은 객체이며 클라이언트 측의 게이트웨이 역할을 한다.
- 모든 나가는 요청은 Stub 통해 라우팅 된다.
- 클라이언트 측에 위치하며 원격 개체를 나타내는데 실제 원격객체가 아니라 원격 객체의 껍데기 이다.
- 클라이언트가 Stub 객체에서 메서드를 호출하면 다음 작업을 수행한다.
  - 원격 가상 머신 (JVM)과의 연결을 시작한다.
  - 원격 가상 머신 (JVM)에 매개 변수를 쓰고 전송 (마샬링, 직렬화)합니다.
  - 결과를 기다린다
  - 반환 값 또는 예외를 읽는다.(언마샬링, 역직렬화)
  - 마지막으로 클라이언트에게 값을 반환합니다.

스켈레톤(Skeleton)

- Skeleton 은 객체이며 서버 측의 게이트웨이 역할을 한다.
- 서버로 들어오는 모든 요청은 Skeleton을 통해 라우팅 된다.
- Skeleton 이 수신 요청을 받으면 다음 작업을 수행한다.
  - 원격 메소드의 매개 변수를 읽는다.
  - 실제 원격 개체에서 메서드를 호출한다.
  - 결과를 작성하고 호출자에게 전송 (마샬링, 직렬화)한다.
  - Java 2 SDK에서는 스켈레톤이 필요없는 스텁 프로토콜이 도입되었다.

RMI Concept


Marshalling
Unmarshalling
JVM
클라이언트
App
스텁(Stub)
스켈레톤
(Skeleton)
JVM
원격객체
(Remote Method)
Unmarshalling
Marshalling

RMI 작성 과정 – 원격 인터페이스 정의

- 원격 객체 사이의 메시지 전송을 위한 것
- 자바 RMI에서는 RMI 클라이언트와 RMI 서버 간의 메시지 전송을 위한 방법으로 자바의 인터페이스를 이용한다. 즉 인터페이스에서 정의한 메소드로 클라이언트와 서버간의 통신이 이루어 진다.

```
public interface Hello extends java.rmi.Remote {
      public String sayHello(String name) throws
            java.rmi.RemoteException;
}
```

RMI 작성 과정 – 원격 인터페이스 구현 클래스 정의(servant)

- 원격 인터페이스에서 정의한 메소드를 구현 한 클래스(이것이 원격개체 이다. 원격에서 호출 가능한 객체.)

```
public class HelloImpl extends UnicastRemoteObject  implements Hello {
	public HelloImpl() throws RemoteException {
		super();
	}

	//원격 메소드 구현
	public String sayHello(String name) {
		return "Hello World ... " + name + "!";
	}
}
```

RMI 작성 과정 – RMI 서버 App 작성

- 원격 인터페이스를 구현한 클래스(원격객체)의 인스턴스를 생성하고 이를 원격의 RMI 클라이언트가 접근하여 구현된 비즈니스 메소드를 원격 호출 할 수 있도록 하는 프로그램 이다.
- 즉 원격 인터페이스를 구현한 클래스의 인스턴스를 생성하고 이를 원격의 클라이언트가 접근할 수 있도록 이름으로 등록하는 일을 담당하는 프로그램

```
HelloImpl remoteObj = new HelloImpl();
java.rmi.Naming.rebind("rmi://localhost:1099/HelloRemote", remoteObj);
```

RMI 작성 과정 – RMI 클라이언트 App 작성

- RMI 서버 Application이 생성한 원격 객체를 접근함으로써 원격 인터페이스에 정의된 비즈니스 메소드를 필요에 따라 호출하는 프로그램

```
Object obj = Naming.lookup("rmi://localhost:1099/HelloRemote");
Hello remoteObj = (Hello)obj;
String msg = remoteObj.sayHello(args[0]);
System.out.println(msg);
```

“Hello World” RMI Application

실습은 다음 강좌에서…