RMI Continued

IS 313

5.29.2003
Outline

- Review of RMI
- Programming example
RMI

- Remote interface
- Remote object
- Registry
- Server
- Client
- RMI compilation
Remote Interface

- Service
  - What a client wants from the server

- Remote Interface
  - Define a Java interface for the service
  - Must extend java.rmi.Remote
  - All methods must throw java.rmi.RemoteException
Example

```java
public interface IFFService extends java.rmi.Remote {

    public List findByName (String firstName, String lastName)
    throws RemoteException;

    public List findByLevel (int level)
    throws RemoteException;

    ...
    etc ...

}
```
Remote object

- Remote object
  - implements remote interface
  - usually extends `java.rmi.server.UnicastRemoteObject`
  - constructor must throws `RemoteException`
Example

public class FFRMIMode extends UnicastRemoteObject implements IReservationServer
{
    public FFRMIMode () throws RemoteException
    {
        ...
    }
    public List findByName (String firstName, String lastName)
        throws RemoteException
    {
        ...
    }
    ...
    etc ...
}
Registry

- Must be running for objects to be registered and looked up
  - `rmiregistry [port]`
  - default port is 1099
Server

- Creates remote object
- Registers (names) it via the registry
- Example

```java
String name = "FrequentFlyer";
try {
    Naming.rebind (name, new FFRMIMode ());
} catch (RemoteException e) {
    ...
}

... also must catch MalformedURLException and AccessException..
```
Client

- Looks up the relevant remote object
- Executes its methods
Example

```java
try {
    String rmiUrl = "//" + host + "/FrequentFlyer";
    IFFServer server =
        (IFFServer) Naming.lookup (rmiUrl);
    List members = server.findByName ("Marilyn", "Monroe");
} ... exception handling omitted ...
```
Serialization

- Turning Java data into stream data
- Can be written
  - to a file
  - to a data stream
  - as an RMI argument or return value
- All primitive types
  - not all objects
RMI at Run Time

Server program

Remote object

RMI Registry

Serializable object

Client program

Stub class
Example

- So far
  - remote interface
  - remote object

- Need
  - server to register
  - client to use
Execution

- Run registry
- Run server
- Run client
Distribution

- Client programmers need
  - Remote interface and associated classes
- Executing clients need
  - Stub class
  - Serializable classes
Full distribution

- Send .java
  - Remote interface
- Send .class
  - Serializable classes
  - stub class
Minimal distribution

- Send .java
  - Remote interface
- Client app
  - requests needed class files at run time
  - uses HTTP if available
  - can use registry also
Activation

- **Problem**
  - Need to have server program running
  - Server program
    - creates object
    - registers object
    - provides execution environment (JVM)
Remote Activation

1. Register object without creating it
2. When client requests object
   - create a JVM
   - instantiate the object
Differences

- `java.rmi.activation.Activatable` instead of `UnicastRemoteObject`
- `Run rmid` as well as `registry`
- `Register ActivationDescriptor` with `rmid`
Example