SE 550: Lecture 7

Overview

RMI

RMI Example

RMI and Class Loading

RMI and Stubs (and sometimes Skeletons)

Midterm review

Plus... a question from a student who can't make the lecture: what is a snapshot?

RMI

What is RPC? What is RMI?

What is a remote object?

What is a naming service?

What is a stub? What is a skeleton?

How does RMI make use of serialization?

RMI Example

A simple remote database:

- Put: put a (key,value) pair into the remote database.
- Get: given a key, get the appropriate (key,value) pair from the remote database.
- Quit

How can we implement this using RMI?

RMI Example

The client and server have to agree a *remote interface*:

```
public interface Hash extends Remote {

  public void put (Serializable key, Serializable value) throws RemoteException;
  public Serializable get (Serializable key) throws RemoteException;

}
```

The source code is available on-line.

RMI Example

The client uses Naming.lookup to access the remote database:

```
String url = ...;
final Hash hashTable = (Hash)(Naming.lookup (url));
```

Compile the client with:

```
javac ajeffrey/teaching/hash/client/Main.java
```

RMI Example

The server provides an implementation of the Hash interface:

```
class HashImpl extends UnicastRemoteObject implements Hash {

    protected final Hashtable contents = new Hashtable ();
    protected HashImpl () throws RemoteException {}

    public void put (Serializable key, Serializable value) throws RemoteException  {
        contents.put (key, value);
    }
    public Serializable get (Serializable key) throws RemoteException {
        return (Serializable)(contents.get (key));
    }

}
```

and registers the database with:

```
        System.setSecurityManager (new RMISecurityManager ());
        Naming.rebind ("hashserver", new HashImpl ());
```

RMI Example

Compile the server with:

```
javac ajeffrey/teaching/hash/server/Main.java
rmic ajeffrey.teaching.hash.server.HashImpl
```

Prepare a policy.txt security policy file, such as:

```
grant {
        // Not a realistic policy file!
        permission java.security.AllPermission;
};
```

RMI Example

Putting it all together...

Run the *RMI Naming registry*:

```
rmiregistry
```

Run the server:

```
java -Djava.security.policy=policy.txt ajeffrey.teaching.hash.server.Main
```

Run the client:

```
java ajeffrey.teaching.hash.client.Main
```

Hooray, a remote database access mechanism!

RMI Recipe

To write a client/server pair with RMI:

Write a remote interface:

- extends Remote
- all methods raise RemoteException
- all method arguments and results are serializable

Write a client which accesses the remote object using Naming.lookup (String).

Write a server which:

- implements the remote interface
- extends UnicodeRemoteObject
- registers the object using Naming.rebind (String,Object)

Compile with javac and rmic.

RMI Recipe

To run a client/server pair with RMI:

Make sure you're running the RMI registry.

Run the server, with an appropriate `policy.txt` file.

Run the client (on another machine).

RMI Class Loading

What is a class loader?

Where does the default Java class loader load classes from?

Where does the applet class loader load classes from?

Where does the RMI class loader load classes from?

Where are classes loaded from if we run:

```
java -Djava.rmi.server.codebase=http://foo.bar.com/somewhere/ blah.client.Main
```

Tip: when testing, just run the client and server in the same directory!

RMI Stubs (and sometimes Skeletons)

Under the hood, each remote object has a *stub* object at the client end, which speaks to the RMI infrastructure at the server end.

When a remote method is called at the client, the stub:

- marshals the arguments (ie serializes them to the server)
- asks the server's RMI infrastructure to call the method
- unmarshals the result (ie unserializes it from the server)
- returns the result.

Because RMI uses serialization *you should always use RMI with immutable arguments and results*!

Summary

RMI allows pointers to remote objects across the network, and to call methods of those remote objects.

Under the hood, RMI uses object serialization to marshall and unmarshall arguments and results.

Stub (and sometimes skeleton) classes are used to implement remote objects.

Next week: more RMI.