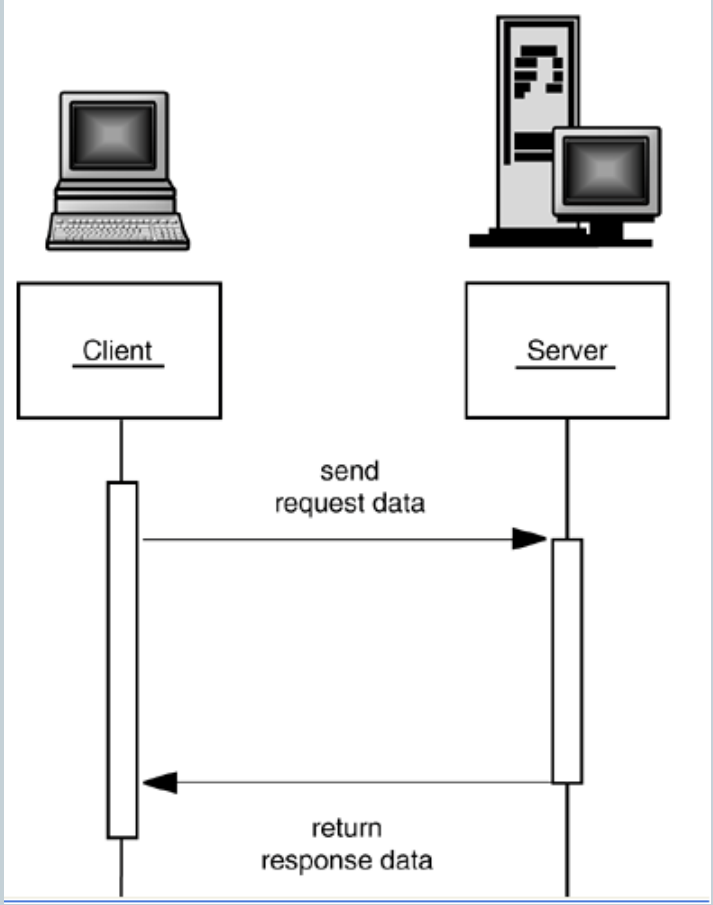
Middleware

د. فادي تركاوي

أنواع الإتصال

- الأول يهتم بكيفيه تبادل الملفات والبيانات بين ال Clientوالـ Serverويتم التعامل مع هذا النوع باستخدام بروتكولات FTP,SMTP,HTTPوغيرها من البروتكولات.
- النوع الثاني من التطبيقات يهتم بكيفيه تشغيل برنامج أو داله في الجهاز الأخر مثل Telnetو Remote Procedure Callأختصارا . RPC
- لغه جافا من اللغات الموجهه للكائنات OO، وبالتالي سوف تطبق مفهوم RPC ولكن عن طريق الكائنات ، ومن هنا جائت تقنيات الـ Distributed Object والتي تسمح لي باستدعاء داله موجوده في كائن بعيد بدون الدخول في تفاصيل ارسال واستقبال البيانات ...
- والطريقه الأولى و التقليديه في كتابه برامج الشكبات Client-Serverمعروفه لدى الأغلب ، حيث يقوم العميل (مثلا طالب) بتعبئه Formويرسل البيانات الى الخادم ، والذي يقوم بمعالجتها ويرجع النتيجه أو تخزينها للأستفاده منها لاحقا ...

Classical Client-Server Model

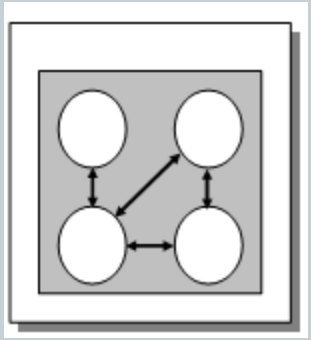


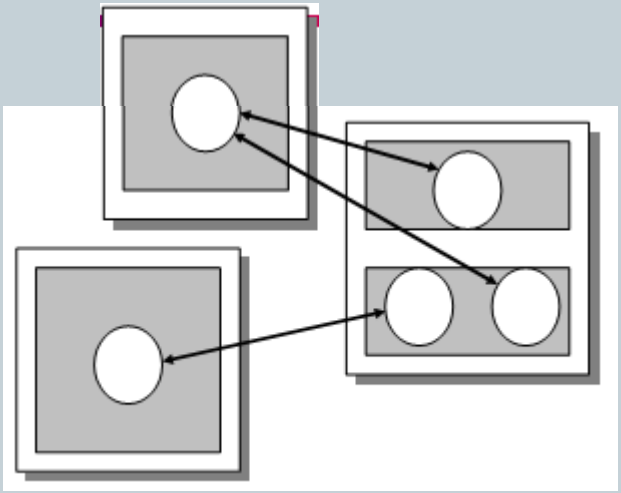
- لكي يقوم المبرمج ببرمجه برنامج مثل هذا عليه أن يقوم بتحديد عده امور مهمه :
- طريقه برمجه الـ Client-Server، هل هم بالأعتماد على TCP-Socketأو UDP-Socket.
- كيفيه ارسال البيانات الى الجهه الأخرى بصوره مفهومه للجهه الأخرى Formatted Understandable Data
- كيفيه عمل parseللرساله القادمه واستخلاص المعلومات منها ..
- كل هذه الأمور على المبرمج أن يضعها في الأعتبار قبل و أثناء كتابه البرنامج ، والا فلن يعمل بالشكل المطلوب ..فمثلا لو قام الطالب Clientبالضفط على زر Delete .. يجب أن تذهب هذه المعلومه الى الطرف الأخر بطريقه ما .. ويقوم الطرف الأخر بمعرفه هذه المعلومه و القيام بالوظيفه المطلوبه ...ويرجع النتيجه بطريقه مفهومه يفهمها الكلاينت .

و المطلوب من الـــ Middleware

- الذي نريده هنا هو ميكانيكه تسمح لي عمل نفس الـ Client-Serverبدون الحاجه الى تلك الخطوات أعلاه ، فقط أقوم بالضغط على الزر ولا أهتم بكيفيه الأرسال ، ولا كيفيه فهم المعلومه ولا توجد حاجه ل parse ولا غيره ، أيضا قد يكون الخادم مكتوب باللغه أخرى مثلا سي++. المشكله هنا أن الكائن الذي يقوم بهذا العمل غير موجود لدينا في الجهاز المحلي ..
- الحل سوف يكون عن طريق عمل "واجهه تستطيع التعامل مع الخادم" Proxy في جهه العميل ، وعندما نضغط على ذالك الزر يقوم العميل باستدعاء داله الموجوده في الـ Proxy .. وبعدها يقوم هذا الـ Proxyبالأتصال مع الخادم بطريقه يعرفها هو ... بنفس الأمر السيرفر قد لا يعلم عن الكلاينت شيء ، لذلك يكون فيه Proxyهو الأخر يتخاطب مع الكلاينت .. أي أن الـ Server-Proxy يقوم بالتخاطب مع الـ client-Proxy .

Distributed System





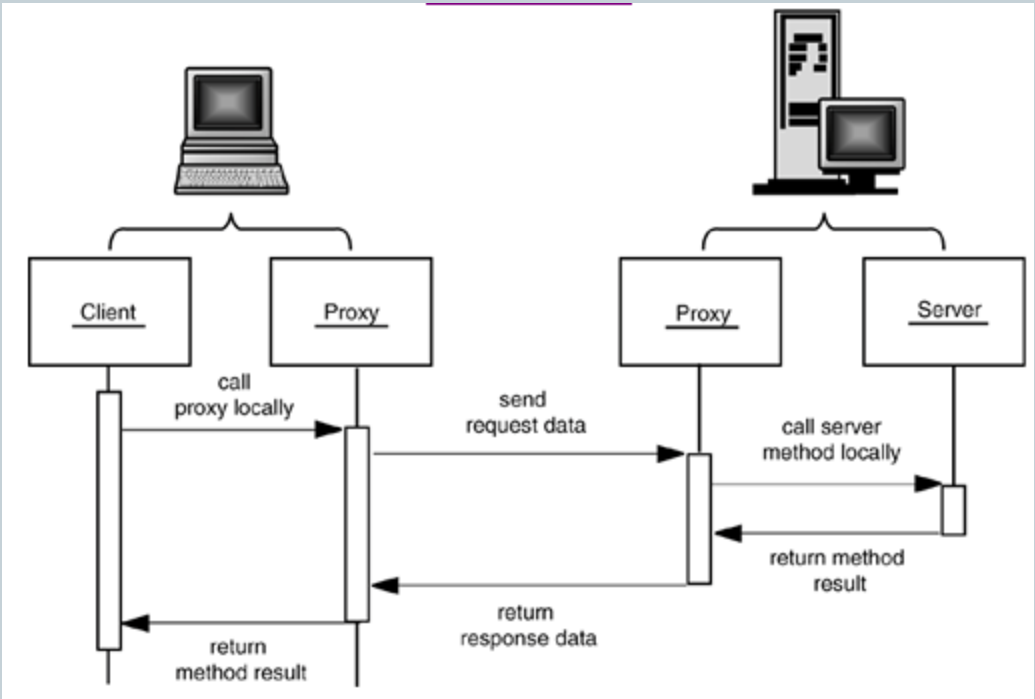
Local System

Distributed System

computer

Process

Object



- أنظر الصوره التاليه لتبين لك :
- هنا طريقه التخاطب بين تلك الـ Proxiesقد يعتمد على التنقيه المستخدمه ، وأشهر ثلاث تقنيات هما :
- RMI
- CORBA
- SOAP

RMI

- أختصار لـ Remote Method Invocationوتعتمد على فكره أستدعاء الكلاينت (يعمل في JVMلداله في كائن بعيد موجود في الخادم (يعمل على JMVمختلفه) ، ويشترط أن يكون الخادم والعميل مكتوبين بجافا ، وتستخدم هذه التقنيه RMI Protocolلتطبيق التخاطب الذي يتعامل مع . TCP Socketوباستخدام RMIسوف نعطي العميل الشعور بأنه لا وجود لكائنات بعيده .. بالضبط كأنه يتعامل مع كائن محلى ، وهذه أحد ميزات الـ . RMI

CORBA

- أختصار لـ Comman Object Request Broker Architecture ..
وظيفتها نفس وظفيه الـ RMIوهي أستدعاء داله في كائن بعيد ، لكن يمكن أن يكون الكائن البعيد مكتوب بأي لغه لا يهم .. وتستخدم CORBA بروتكول يسمى Internet Inter-ORB Protocolواختصارا IIOP .

SOAP

- أختصار لـ Simple Object Access Protocol
وهي تقريبا نفس وظيفه CORBA، ولكن تعتمد في عملها على ملفات XML .
- CORBAو SOAPتعمل على أي لغه لا يشترط جافا ، وهنا سوف نستخدم لغه خاصه لوصف الدوال الموجوده في السيرفر والتي يستطيع الكلاينت التعامل معها
- في CORBAنستخدم لغه تسمى Interface Defenition Langauge واختصارا IDL يمكن أن نستخدم RMI مع بروتكول IIOPولن نحتاج الى IDL. أما في SOAPنستخدم Web Services Description Languageأختصارا WSDL . طبعا هذه ليست لغات كجافا أو سي ، لكنها فقط لغه لوصف الـ Interfaceالذي يستطيع أستخدامه الكلاينت ...
- طبعا أسهل هذه التقنيات هي RMIوهي موضوع حديثنا اليوم ، أما CORBA فهي تحتاج موضوع اخر لها هي والـ SOAPوالتي تستخدم عند العمل مع Web-Service .

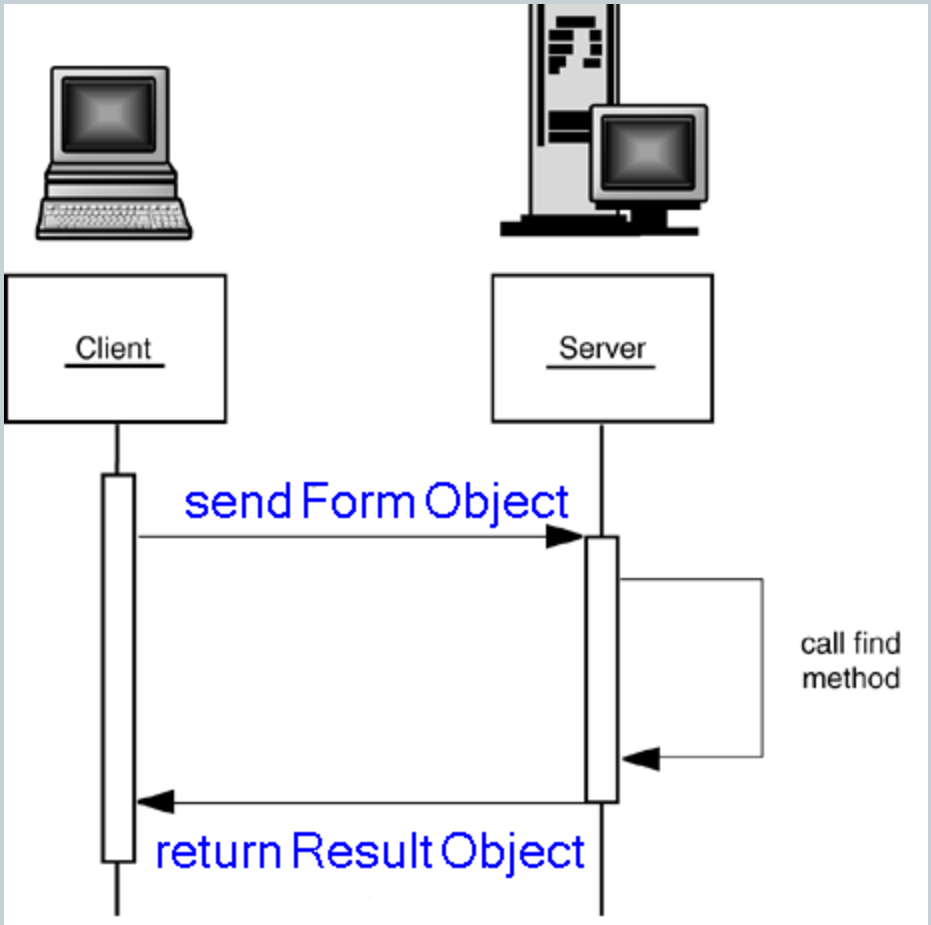
RMI Conecpt

- بنظره سريعه في أي كود RMI، سوف نجد أنه يبعدنا تماما من التفاصيل ، فعندما نريد أن نستدعي داله موجوده في الخادم Server، كل ما على أن أكتب سطر واحد وسأحصل على Reference لهذا الكائن . بعدها أستدعى الداله بشكل عادي كما لو أنها كانت Local . اذا كانت هناك قيمه مرسله Parametersو قيم راجعه return Value، ستذهب بطريقه ما الى المكان الصحيح وبدون تدخل منك .. فعلا RMIتجعل الحياه أكثر سهوله .

- Fish one;

- One.swim();

- أنظر للصوره التاليه والتي توضح هذه العمليه بشكل Abstract :



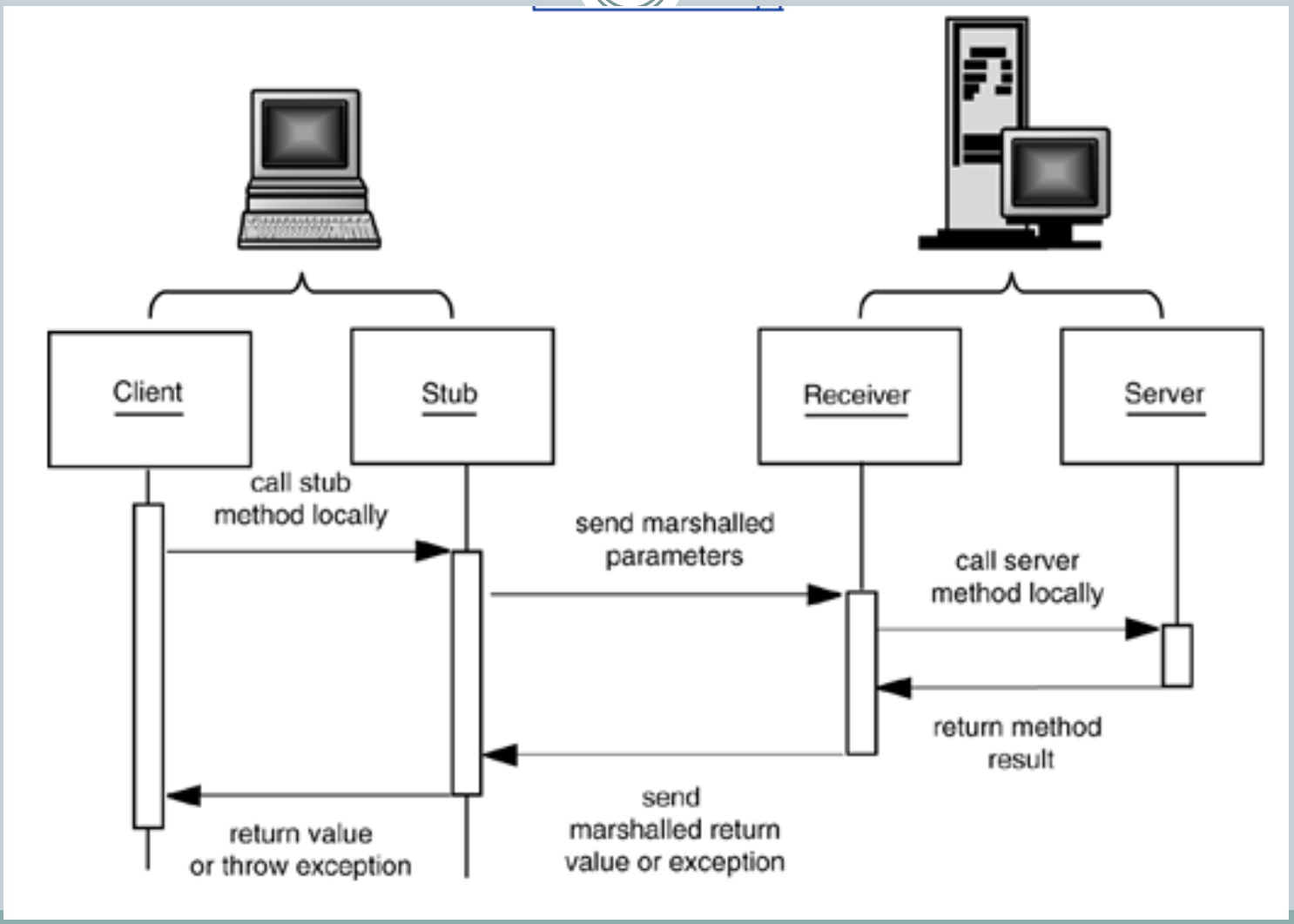
تـــــــــــوضيـــــــح RMI

- لندخل الى العمق أكثر ولنرى كيف يقوم RMIبارسال القيم المرسله والراجعه .. أولا عندما يقوم الكلاينت باستدعاء داله موجوده في كائن بعيد سوف يقوم client باستدعاء الداله stub الموجوده في Proxyلديه .. ويمرر لها المتغيرات المطلوب ارسالها .. بعد ذلك يبدأ الـ Stubبعمل معالجه لتلك المتغيرات حتى يرسلها .. فلن يرسلها هكذا .. هذه العمليه تسمى Parameter Marshaling
- في عمليه الـ Marshalling سوف ينظر أولا الى نوع المتغير أو الكائن ، فاذا كان متغير عادي Primitive Data Type فيقوم بارساله كما هو بالطبع بعد تحويله الى بايت Byte بترميز يعرف بـ .. Big-Endian

تـــــــــــوضيـــــــح RMI

- أما اذا كان المتغير هو كائن reference فلا يمكن أن يرسل بهذه الطريقه ، لأنه يمكن أن يحتوي على كائنات أخرى فيه وتلك الكائنات تحتوي أيضا على كائنات وهكذا . لذلك في عمليه ارسال الكائن سوف نقوم بترميزه بـ serialization وبالتالي تذهب البايتات بطريقه معينه متسلسله الى الجهه الأخرى .وعندما تصل الى الجهه الثانيه سوف يقوم بتجميع هذه البايتات بعمليه تدعى De-Serialization . لذلك في حاله التعامل مع الكائنات (قيم مرسله أو راجعه) يجب أن يكون الكائن لديك يطبق impelements الـ serialization .
- الأن بعدما أنتهي الـ stub من عمليه Encoded سوف يرسل هذه البايتات الى الجهه الثانيه ، ويقوم ال Receiver Object باستقبال هذه البايتات وتجميعها unmarshals the parameters وبعدها يقوم بتحديد الكائن المراد ثم استدعاء الداله المراده وفي حال كان هناك قيمه راجعه منها يقوم هذا الـ receiver object بعمل Marshals لهذه القيمه الراجعه ويعيدها الى Stub الذي يقوم بعمليه unmarshals للقيمه الراجعه ويرجعها الى الكلاينت الذي استدعى الـ Stub .

RMI توضيح


Client
Stub
Receiver
Server
call stub
method locally
send marshalled
parameters
call server
method locally
return method
result
return value
or throw exception
send
marshalled return
value or exception

تـــــــــــوضيـــــــح RMI

- ربما تبدوا العمليه معقده وغير مفهومه ، لكن كما ذكرت لا حاجه للمبرمج بمعرفه تلك الخطوات ، لأن من ميزات RMIهي عمل أخفاء لكل هذه التفاصيل Good Transparency.