Client-Server Architecture

Client-Server Example

Socket Programming – Client

Client-Server Interaction as Function Calls

Remote Procedure Call
### RMI

1. **lookup**
2. **Stub (proxy)**
3. **Method invocation**
4. **Return value**

**Type of the local object??**

### Interface
- Must extends `java.rmi.Remote`
- Shared by both client and server code
- E.g. `AuthInterface`

```java
public interface AuthInterface extends java.rmi.Remote {
  User auth( String username, String password )
  throws java.rmi.RemoteException;
}
```

### Remote Object

```java
public class AuthImpl implements AuthInterface {
  public User auth( String username, String password )
    throws java.rmi.RemoteException {
    // user authentication
  }
}
```

### Stub
- Created automatically

```java
public class AuthStub implements AuthInterface {
  public User auth( String username, String password )
    throws java.rmi.RemoteException {
    // connect to the server
    // send username and password to the server
    // return the result
  }
}
```

### More About RMI
  - Compilation and Execution

### Spring RMI Support
- POJO interface and implementation
- RmiServiceExporter handles remote object registration, lookup, stub generation etc.
  - service, serviceInterface, and serviceName
  - registryPort
- Evelyn example
Alternatives to RMI

<table>
<thead>
<tr>
<th>Name</th>
<th>Language</th>
<th>Message Type</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMI</td>
<td>Java-to-Java</td>
<td>Binary</td>
<td>Default 1099</td>
</tr>
<tr>
<td>Hessian</td>
<td>Mostly Java-to-Java</td>
<td>Binary</td>
<td>HTTP</td>
</tr>
<tr>
<td>Burlap</td>
<td>Any</td>
<td>XML</td>
<td>HTTP</td>
</tr>
<tr>
<td>Spring HTTP Invoker</td>
<td>Java-to-Java</td>
<td>Binary</td>
<td>HTTP</td>
</tr>
<tr>
<td>Web services</td>
<td>Any</td>
<td>XML</td>
<td>HTTP</td>
</tr>
</tbody>
</table>

Web Services

- Roughly speaking, anything that encodes RPC calls in XML messages and transport them over HTTP
- Simple Object Access Protocol (SOAP)
- Web Service Description Language (WSDL)
- Universal Description, Discovery, and Integration (UDDI)

SOAP

- [http://www.w3.org/TR/soap/](http://www.w3.org/TR/soap/)
- Format conventions for message content and routing directions in the form of an envelope
- Rules for encoding custom data types
- Application of the envelope and the data encoding rules for representing RPC calls and responses
- Transport protocol binding (usually HTTP)

A Sample SOAP Message

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doSpellingSuggestion
xmlns:ns1="urn:GoogleSearch">
      <key xsi:type="xsd:string">00000000000000000000000000000000</key>
      <phrase xsi:type="xsd:string">britney speers</phrase>
    </ns1:doSpellingSuggestion>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Things to Note

- Namespaces
- `<Envelop>`
  - Optional `<Header>` - for information related to processing of the message
  - `<Body>`
- EncodingStyle
- `<Fault>`
  - Only sub-element of `<Body>` defined by SOAP

SOAP Encoding

- [http://schemas.xmlsoap.org/encoding](http://schemas.xmlsoap.org/encoding)
- Include all built-in data types of XML Schema Part 2: Datatypes
  - xsi and xsd name spaces
SOAP Encoding Examples

```plaintext
int a = 10;  <a xsi:type="xsd:int">10</a>
float x = 3.14159;  <x xsi:type="xsd:float">3.14159</x>
String s = "SOAP";  <s xsi:type="xsd:string">SOAP</s>
```

Compound Values and Other Rules

```plaintext
<Array xsi:type=SOAP-ENC:Array SOAP-ENC:arrayType= "xsd:int[3]">
<val>10</val>
<val>20</val>
<val>30</val>
</Array>

<Sample>
<iVal xsi:type="xsd:int">10</iVal>
<sVal xsi:type="xsd:string">Ten</sVal>
</Sample>
```

References, default values, custom types, root attribute, complex types, custom serialization ...

---

SOAP RPC Elements

- Target object URI in HTTP header
- Namespace qualified method name and method parameters
- Optional SOAP header for additional data that’s not part of the parameter list

---

A Sample SOAP RPC Response

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:doSpellingSuggestionResponse xmlns:ns1="urn:GoogleSearch" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:string">britney spears</return>
</ns1:doSpellingSuggestionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

---

A Sample Fault Response

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcode>SOAP-ENV:Client</faultcode>
<faultstring>Client Error</faultstring>
<detail>
<m:dowJonesfaultdetails xmlns:m="DowJones">
<message>Invalid Currency</message>
<errorcode>1234</errorcode>
</m:dowJonesfaultdetails>
</detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

---

WSDL

- A language for describing web services
  - Where the service is
  - What the service does
  - How to invoke the operations of the service
- **Why do we need WSDL when we have API documentation??**
Sample WSDL Documents

- Amazon ECS -
  http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl
- Google Web APIs -
  http://api.google.com/GoogleSearch.wsdl

How Do We Describe an API?

```java
interface Foo {
    int bar(String, BigDecimal);
}
```

Return value  Method name  Parameters

How Do We Describe a Web Service API?

WSDL

<table>
<thead>
<tr>
<th>Type</th>
<th>&lt;types&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>&lt;message&gt; (request and response)</td>
</tr>
<tr>
<td>Return values</td>
<td>&lt;operation&gt;</td>
</tr>
<tr>
<td>Method name</td>
<td>&lt;portType&gt;</td>
</tr>
<tr>
<td>Interface name</td>
<td></td>
</tr>
</tbody>
</table>

A Little More Details

- The name attribute uniquely identifies a `<message>`, an `<operation>`, or a `<portType>`
- Operation behavior patterns
  - Client initiated
    - Request-response
    - One-way
  - Server initiated
    - Solicit-response
    - Notification
- `<fault>`

Other WSDL Elements

- `<definitions>`
  - targetNamespace
- `<import>`
- `<binding>` - concrete protocol and format specification for a `<portType>`
  - E.g. `<input>` should be in SOAP header or body, what encoding rules should be used etc.
- `<service>`

Service

```
<service>
  <port>
    <binding>
      Defies a set of operations
      Implementation of the operations with a certain protocol
    </binding>
    Where the operations can be access (endpoint)
  </port>
</service>
```
JAX-RPC

- A specification for building XML-based web services and clients using Java

JAX-RPC Architecture

- Service Client
- Stub
- WSDL description
- Service Endpoint
- Container
- Dispatch
- JAX-RPC API
  - Client Side JAX-RPC Runtime System
  - Server Side JAX-RPC Runtime System
  - Transport

Source: [http://www.pankaj-k.net/axis4tag/](http://www.pankaj-k.net/axis4tag/)

Service Invocation Patterns

- Static binding
  - statically generated stub
- Dynamic binding
  - statically generated interface
  - `javax.xml.rpc.Service.getPort()`
- Dynamic Invocation Interface (DII)
  - `javax.xml.rpc.Call`

Apache Axis

- An implementation of SOAP for Java
- Simplifies producing and consuming web services
  - Create WSDL document from Java source code
  - Create Java classes from WSDL document
  - Encode and decode XML requests and responses
  - ...
UDDI
- A registry for web services
  - Information about the service providers
  - Classifications of services
  - Technical information about the service interfaces
- A web API for publishing, retrieving, and managing information in the registry

Registries

Core Data Types

- `<businessEntity>`
- `<businessService>`
- `<bindingTemplate>`
- `<tModel>`

```xml
http://www.uddi.org/schema/uddi_v1.xsd
http://www.uddi.org/schema/uddi_v2.xsd
http://www.uddi.org/schema/uddi_v3.xsd
```

```xml
<businessEntity>
  <businessKey>uuid:xxxxxxxxxxxxxxxxxxxx</businessKey>
  <operator>http://some.com</operator>
  <authorizedName>John Doe</authorizedName>
  <name>Some Company</name>
  <description>We provide web services</description>
  <contacts>...</contacts>
  <businessServices>...</businessServices>
</businessEntity>
```

```xml
<businessService>
  <name>Hello World</name>
  <description>A great web service</description>
  <bindingTemplates>...</bindingTemplates>
</businessService>
```

```xml
<bindingTemplate>
  <description xml:lang="en">SOAP binding for Hello World</description>
  <accessPoint URLType="http">http://localhost:8080/soap</accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo tModelKey="xxxxxxxxxxxxx" />
  </tModelInstanceDetails>
</bindingTemplate>
```
<tModel>
  ◆ Interface specification about services
    <tModel TModelKey="uuid:xxxxxxxxxxxxxxxxxxxxxxxxx"
      operator="http://some.com"
      authorizedName="John Doe">
      <name>Hello World Port Type</name>
      <description>
        Interface for a great web service
      </description>
      <overviewDoc>
        <overviewURL>
          http://localhost:8080/soap/helloworld.wsdl
        </overviewURL>
      </overviewDoc>
    </tModel>
</tModel>

UDDI APIs
  ◆ Node API Sets
    - Interaction among registry nodes
  ◆ Client API Sets
    - Publish services to a registry
    - Search a registry for services

WSDL for UDDI Client API
  ◆ http://www.uddi.org/wsdI/publish_v2.wsdl
  ◆ http://www.uddi.org/wsdI/inquire_v2.wsdl

Tools and Libraries
  ◆ http://uddi.org/solutions.html
  ◆ Ruddi - http://www.ruDDi.biz/