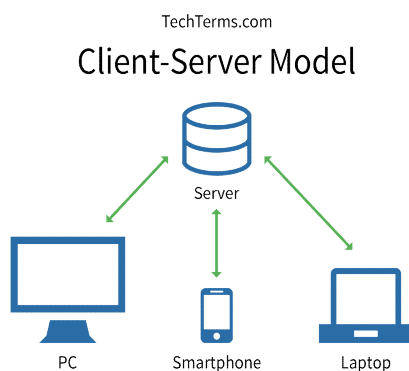


Objects in Client-Server applications

CSCI 2300

Client-Server applications



- Server accepts connections from client (socket)
- Client creates a connection to server (socket)
- Client sends data to the server via the socket
- Server receives data from client via the socket
- Can send “objects” as data

Sockets in Java

- `java.net.Socket` – socket class in Java
- Create a socket connection to server:
 - `Socket socket = new Socket(HostIP, HostPort);`
 - Server must be running on the “HostIP”
 - Server must be listening for connections on HostPort
- Write to a socket:
 - Use `OutputStream` object of the socket
`socket.getOutputStream()`

So what do we “write” to the socket

- Custom message format
- Serializable object
- Object stored in commonly accepted format:
 - XML
 - JSON

Example: Pass a “TextMessage” object from Client to Server: client_server/proprietary/

- Client and Server have an identical serializable “TextMessage” class
- Client
 - instantiates a TextMessage object and fills it with details
 - Client connects to the server via a Socket
 - `Socket socketToServer = new Socket(hostIP, hostPort);`
 - Client sends the TextMessage object to the server
 - `ObjectOutputStream streamToServer;`
`// initialize streamToServer`
`streamToServer.writeObject(textMessage);`

What allows us to pass textMessage to the writeObject() method:

`streamToServer.writeObject(textMessage);`

- A. toString() method of TextMessage
- B. Serializable interface that TextMessage implements
- C. The constructor of TextMessage
- D. The getMessage() method of TextMessage
- E. textMessage cannot be passed to writeObject()

Server can be running on the same machine as Client

- If Server is on the same machine as client, connect to 'localhost' network interface: 127.0.0.1
- Useful for testing
- Create a server socket:
 - `ServerSocket serverSocket = new ServerSocket(port);`
 - Creates a server socket that can listen on all network interfaces of the server, using the given port number
- Server has to listen on a given interface and port
 - `Socket incoming = serverSocket.accept();`
 - Waits for client connections on `serverSocket`

Example: client_server/proprietary

- `TextMessage` implements `Serializable`
- Client
 - instantiates a `TextMessage` object
 - Sets the message
 - Sends the message to the server
- Server
 - Creates a server socket
 - Listens for client connections on the server socket
 - Reads the `TextMessage` object from the client

What if Client and Server ran on different machines. What would happen if the `TextMessage` class changed on the client machine and not on the server machine?

- A. This example would continue to work fine
- B. Client would not be able to send `TextMessage`
- C. Server would not be able to interpret received `TextMessage`
- D. Code on the client machine would not compile

Let's try it

- Added `'String timeStamp'` to client
- Kept old version of `TextMessage` on server

Run the example

- But what if we have to make a change and we have several servers and clients?
- In practice, you can't upgrade all clients and servers simultaneously
- We don't want to break compatibility
- Proprietary format is not the best option in this case

JSON – Commonly used format

- JavaScript Object Notation
- Used to format data
- Commonly used in Web as a vehicle to describe data being sent between systems

JSON example

- “JSON” stands for “JavaScript Object Notation”
 - Despite the name, JSON is a (mostly) language-independent way of specifying objects as name-value pairs
- Example
 - ```
{ "TextMessage": {
 "name": "Kate",
 "message": "Hello"
}}
```

## JSON syntax

- An *object* is an unordered set of name/value pairs
  - The pairs are enclosed within braces, { }
  - There is a colon between the name and the value
  - Pairs are separated by commas
  - Example: { "name": "Kate", "message": "Hello" }
- An *array* is an ordered collection of values
  - The values are enclosed within brackets, [ ]
  - Values are separated by commas
  - Example: [ "html", "xml", "css" ]

## JSON syntax

- A *value* can be: A string, a number, **true**, **false**, **null**, an object, or an array
  - Values can be nested
- *Strings* are enclosed in double quotes, and can contain the usual assortment of escaped characters
- *Numbers* have the usual C/C++/Java syntax, including exponential (E) notation
  - All numbers are decimal--no octal or hexadecimal
- *Whitespace* can be used between any pair of tokens

## Mapping between JSON and Java entities

| JSON       | Java              |
|------------|-------------------|
| string     | java.lang.String  |
| number     | java.lang.Number  |
| true false | java.lang.Boolean |
| null       | null              |
| array      | java.util.List    |
| object     | java.util.Map     |

## JSON reading in Java example

- [http://www.tutorialspoint.com/json/json\\_java\\_example.htm](http://www.tutorialspoint.com/json/json_java_example.htm)



## Client Server example using JSON format: client\_server/json/

- Additional classes from javax.json package
  - JsonObject
  - JsonObjectBuilder
  - JsonWriter
  - JsonReader
- Convert TextMessage to JsonObject:
  - TextMessage: `toJson()`
- Send JsonObject from client to server:
  - Client: `sendTextMessage()`
- Re-build TextMessage from JsonObject:
  - Server: `readTextMessage()`
  - TextMessage: constructor with `JsonObject` parameter

## Uses javax.json-1.1.4.jar

- Modify CLASSPATH from client\_server directoryy  
source `configure.sh`
- `cd json`
- `javac -cp $CLASSPATH *.java`
- `java -cp $CLASSPATH Server`
- `java -cp $CLASSPATH Client`

Imaginee Client and Server ran on different machines. What would happen if we added a class variable to the TextMessage class on the client machine and not on the server machine?

- A. This example would still work, the Server would read TextMessage in the old format.
- B. Client would not be able to send TextMessage
- C. Server would not be able to interpret received TextMessage
- D. Code on the client machine would not compile

Let's try it

- Added 'String timeStamp' to client
- Kept old version of TextMessage on server

Run the example

- As long as we didn't remove any fields from TextMessage, we can upgrade all client machines and then upgrade the server
- Can we upgrade the server first?

A. Yes                      B. No