MQTT SmartBoard

Short Description:

Collaboratively design, share and discuss design projects in near real time on a user-specified design room. Automatically retain room designs remotely in IBM MessageSight server. You don't have to design quietly. Brainstorm ideas, draft plans, send pictures to co-designers in a closed client-to-client chat environment.

How it works:

This demo integrates MQTT and IBM MessageSight to provide a highly scalable, low latency collaborative design environment. The Android application subscribes to a user-specified room(topic) to receive all drawing and chat information. SmartBoard allows you to add, scale, modify, drag, delete objects on the room canvas, send chat messages -text and images - immediately publishing drawing and chat information to a dynamic topic structure in MessageSight. The result is an incredibly scalable low-latency collaborative design experience.

Try it:

Free draw, add objects of your own size and color, drag, format objects, remove what you don't like or clear everything and start all over again, discuss the design project with co-designers and when you are done print and save a copy of the design on your Android device. If you don't have space in memory, don't worry. The design is already saved in IBM MessageSight. Just login to same room every time and the design is intact.

Subscription:
smartboard/<room>/#

Topic And Sample Payloads:

Topic	Sample Topic	Sample Payload
smartboard/<room>/objects/<id> [retained]	[smartboard/room5/objects/2167f29d-3b38-432a-8091-6f20bfbff7b5]	{"id":"2167f29d-3b38-432a-8091-6f20bfbff7b5","dimens":"382 330 620 517","type":"Rectangle","color":-16777216,"clientId":"paho-917066198329565","size":5}
smartboard/<room>/chat	[smartboard/room5/chat]	{"selfie":"tjaXL...\n","message":"Allan: Hey dude!","time":"2014-09-02 16:07:55","username":"Allan%adb7177b-d39c-46cb-a6f1-235392b99873","type":"Chat","clientId":"paho-917066198329565","direction":false}
smartboard/<room>/points	[smartboard/room5/points]	{"color":-16777216,"brushSize":5,"mX":784.2857055664063,"drawActionFlag":0,"type":"Point","clientId":"paho-917066198329565","mode":"pencil","mY":448.71429443359375}
smartboard/<room>/users/<name> [retained]	[smartboard/room5/users/Allan%71afd4f2-5950-4bfd-a89a-baf73271f80c]	{"selfie":"tja...\n","time":"2014-09-02 15:59:39","type":"User","status":"online","clientId":"paho-916727881160387","userId":"Allan%3e743630-8cfa-4f74-bce0-3705bb83fe2b"}
smartboard/<room>/clear	[smartboard/room5/clear]	{"type":"ClearScreen"}

Installation:

In order to compile and run this code you will need to add two dependencies to the project structure:

- Eclipse Paho Mqtt Client library

http://www.eclipse.org/paho/clients/java/

- Eclipse Paho Android Service

http://www.eclipse.org/paho/clients/android/

This application is compiled to run on Android 4.1 (Jelly Bean) or later versions.
This code was originally written with IntelliJ IDEA and compiled on Android sdk level 19.
To use another IDE such as eclipse you must export the project.

Design Notes:

In development, I found it structurally coherent and logically meaningful to abstract MQTT client operations from activity classes. This reduces the amount of work an activity has to do on top of UI lifecycle operations.

In this Smartboard application, for instance, you will notice that the client operations, i.e publish,subscribe are done in an MqttHandler singleton. This abstraction makes this class – MqttHelper - dynamically useful for any other activity that may want to use the MQTT client services in future development. From a design perspective, this model is centralized hence very scalable, reusable and easy to debug.

This design practice (where no MQTT clients are instantiated/called in activity classes) also prevents loss of the MQTT client handles and/or MqttConnection in garbage collection when the activity is recreated such when mobile device or tablet is rotated to either landscape or potrait mode etc.

I have included in the appendix a relational diagram of the classes in the project. This schematic representation is a useful resource for consolidating the classes, and understanding design methodology used.

Future Development:

Enable concurrent multi-room experience.
Send short Audio/Video recordings in the chat room
Undo
Dynamic scaling of canvas for multiple screen sizes
Up navigation. (Check Android developer website)
Last will option, delete user prescence info not behaving as expected. Potential bug in Android Service paho libraries.
Secure sign-on. Etc.

External Resources

http://www.eclipse.org/paho/files/android-javadoc/index.html
http://developer.android.com/index.html
http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html
http://www.slideshare.net/BryanBoyd/dynamic-apps-with-websockets-and-mqtt-ibm-impact-2014
http://www.slideshare.net/AllanMarube/smart-boarddiagram-38627291

Appendix

Relational Diagram