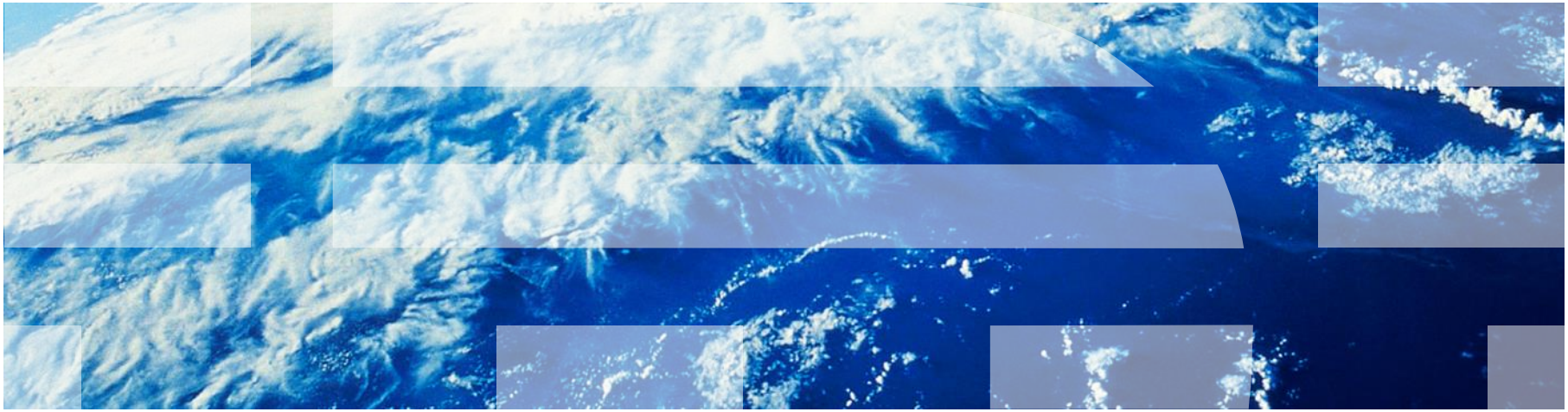
IBM Worklight V6.1.0
入門

MQ Telemetry Transport を使用して
動的なコラボレーティブ・モバイル・アプリケーションを開発する



2014 年 3 月 17 日

商標

- IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。
- Linux は、Linus Torvalds の米国およびその他の国における登録商標です。
- Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- この資料は、事前に IBM の書面による許可を得ずにその一部または全部を複製することは禁じられています。

IBM® について

- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

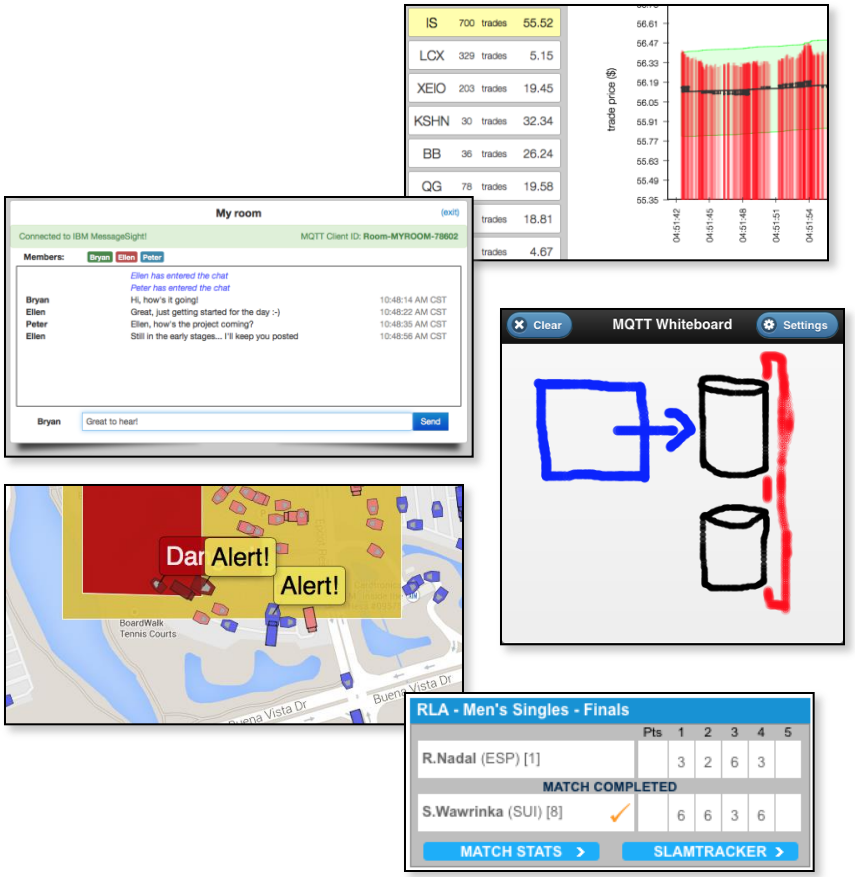
- MQ Telemetry Transport 入門
- コラボレーティブ・アプリケーション (ホワイトボード) を作成する
- MQTT ブローカーに接続する
  - Mosquitto
  - IBM MessageSight

MQ Telemetry Transport 入門

- MQ Telemetry Transport (MQTT) は、Internet of Things (IoT) およびモバイル接続のために設計された軽量のメッセージング・プロトコルです。
- MQ Telemetry Transport は以下を提供します。
  - 信頼性の低い接続を介した信頼性の高いメッセージ・デリバリー
  - エンタープライズに対するセキュア・メッセージ・デリバリー
  - 1 対多メッセージ・デリバリー (パブリッシュ/サブスクライブ)
  - サーバーからのデータの準リアルタイム・プッシュ (ポーリングなし)
  - 最小フットプリント・オンザワイヤー (2 バイト・ヘッダーのみ)
  - バッテリー使用量の削減

MQ Telemetry Transport 入門 (1/3)

- MQ Telemetry Transport を使用すれば、サーバーからモバイル・デバイスへのリアルタイム・データ・プッシュが可能になります。これは動的モバイル・アプリケーションには理想的です。
- 例:
  - ストック・ティッカー
  - チャット・アプリケーション
  - コラボレーション・アプリケーション
    - (ホワイトボード共有)
  - リアルタイムの緊急警報
  - ライブの試合スコア更新



MQ Telemetry Transport 入門 (2/3)

- MQTT クライアント・ライブラリーは、Java™、C、JavaScript™、C++、Python、Objective-C、およびその他のプログラミング言語で使用できます。
- Eclipse Paho プロジェクト (http://www.eclipse.org/paho/) では、数多くのオープン・ソース MQTT クライアント (http://git.eclipse.org/c/paho) が提供されています。
- IBM では、Mobile Messaging & M2M Client Pack で MQTT クライアントを提供しています。developerWorks の IBM Messaging コミュニティーで入手可能です。
- このモジュールでは、IBM Worklight® アプリケーションでのパブリッシュ/サブスクライブ・メッセージングに Eclipse Paho JavaScript MQTT クライアントを使用しています。

MQ Telemetry Transport 入門 (3/3)

- MQTT クライアントには、パブリッシュ/サブスクライブ・メッセージング用にシンプルな API が用意されています。
- 以下の例では、MQTT ブローカーへの**接続**、MQTT トピックの**サブスクライブ**、ブローカーからのメッセージの受信と**処理**、および MQTT トピックでのメッセージの**パブリッシュ**に、Eclipse Paho JavaScript MQTTクライアントを使用しています。

接続

```
function connect() {
  client = new Messaging.Client(hostname, port, clientId);
  client.onMessageArrived = onMsgCallback;
  client.onConnectionLost = onConnLostCallback;
  client.connect({ onSuccess: onSuccessCallback });
}
```

サブスクライブ

```
function subscribe(topic) {
  client.subscribe(topic);
}
```

パブリッシュ

```
function publish(topic, data) {
  var msg = new Messaging.Message(data);
  msg.destinationName = topic;
  client.send(msg);
}
```

受信

```
function onMsgCallback(msg) {
  var topic = msg.destinationName;
  var data = msg.payloadString;
  console.log(topic, data);
}
```

アジェンダ

- MQ Telemetry Transport 入門
- コラボレーティブ・アプリケーション (ホワイトボード) を作成する
- MQTT ブローカーに接続する
  - Mosquitto
  - IBM MessageSight

コラボレーティブ・アプリケーションを作成する

- このモジュールでは、Worklight および MQ Telemetry Transport を使用して動的アプリケーション (ホワイトボード) を作成します。これらを使用すると、ユーザーはリアルタイムで共有キャンバスに描画することができます。2 つのステップでこれを行います。
  - **Worklight アプリケーションのビルド**
    - 最初に、「単一ユーザー」モードを実装します。
    - アプリケーションは、タッチ/クリック・イベントをキャプチャーし、キャンバスでのペイントを可能にします。
  - **MQTT クライアントおよびサーバーを使用してコラボレーションを実装**
    - JavaScript MQTT クライアントを、MQTT サーバーを通じて他のクライアントと通信するホワイトボード・アプリケーションに追加します。
    - MQTT サーバー・オプションは以降のスライドで説明します。
    - **注**: MQTT データには、IBM Worklight のセキュリティーおよび認証のメカニズムは使用されません。

コラボレーティブ・アプリケーションを作成する

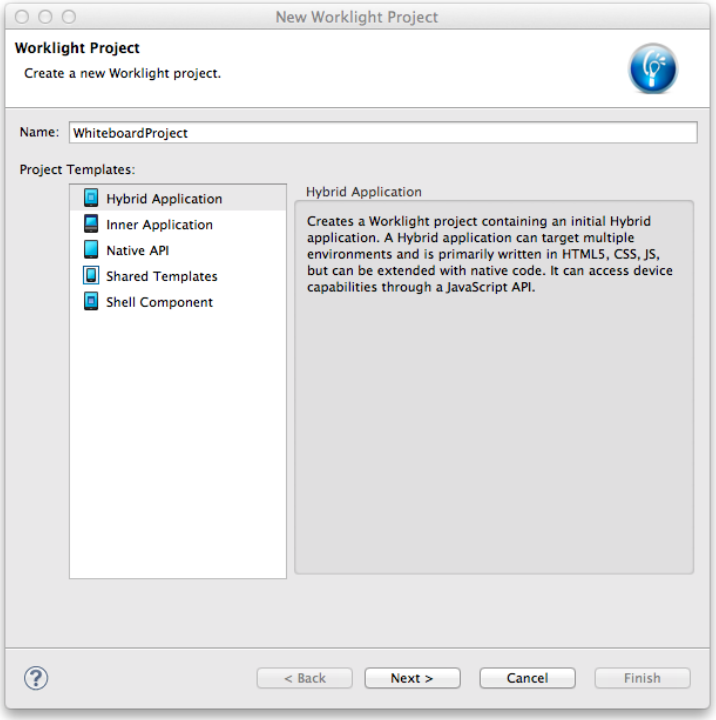
- ホワイトボード・アプリケーションは、MQTT メッセージングを使用して、すべてのユーザーのための共有キャンバスを提供します。
- ホワイトボードはそれぞれ、すべての描画アクションを MQTT メッセージとして、アプリケーション・セッションに固有のトピックにパブリッシュします。
- 各ホワイトボードはまた、すべての描画トピックのセットに対してサブスクライブを行い、このデータに基づいて他のユーザーのアクションを描画します。
- MQTT パブリッシュ/サブスクライブ・メッセージング機能により、リアルタイムのコラボレーティブ・エクスペリエンスを、接続された数多くのアプリケーションへ拡大することができます。

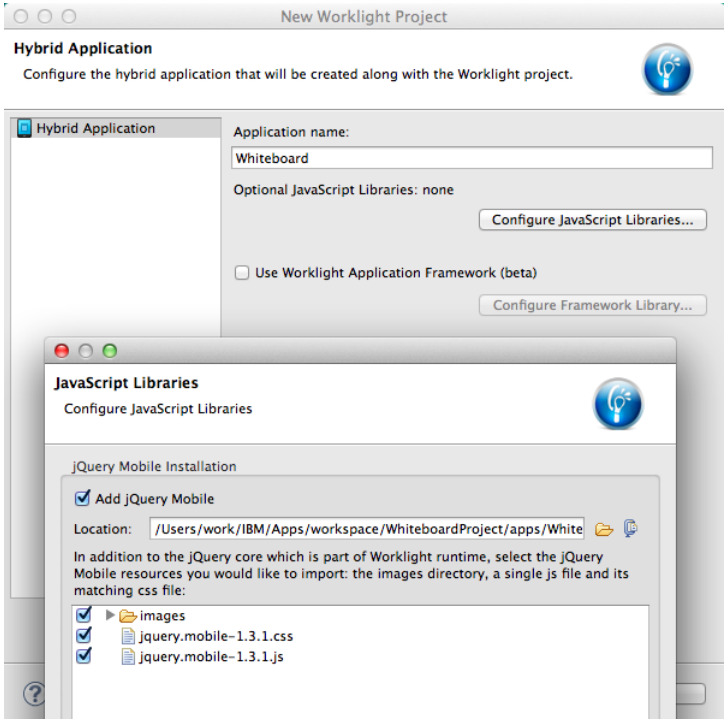
コラボレーティブ・アプリケーションを作成する

- このシナリオは、従来のポーリング (HTTP) またはモバイル・プッシュ通知を使用しても効率的に行うことはできません。
  - リアルタイム・エクスペリエンスを提供するには、描画アクションをミリ秒以内で他のキャンバスに反映する必要があります。
  - HTTP ポーリングは高帯域幅 (各要求に新しいクライアント接続が必要) であり、待ち時間が長くなります。
  - プッシュ通知は帯域幅を最小化しますが、小さなアプリケーション内更新には適していません。
  - MQTT パブリッシュ/サブスクライブ・メッセージングは、帯域幅および待ち時間を最小化します。MQTT 接続がクライアントからサーバーへ一回確立され、メッセージが少ない待ち時間でアプリケーションに直接プッシュされます。

ホワイトボード– プロジェクトを作成する

- **Whiteboard** というハイブリッド・アプリケーションを持つ **WhiteboardProject** プロジェクトを作成します。別途入手する jQuery Mobile 1.3.1 ライブラリーを追加します。
  - 詳しくは、**『UI フレームワークで作業する』**モジュールを参照してください。





ホワイトボード – UI を作成する

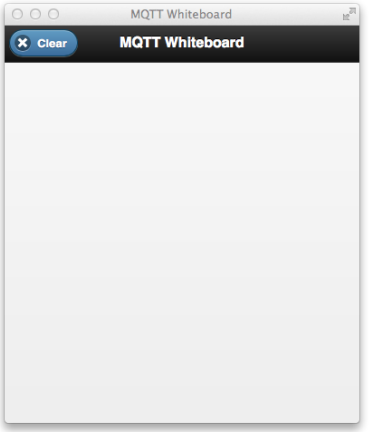
- ヘッダー、キャンバス、およびキャンバスをクリアするボタンを追加します。

index.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Whiteboard</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=0">
    <link rel="shortcut icon" href="images/favicon.png">
    <link rel="apple-touch-icon" href="images/apple-touch-icon.png">
    <link href="jqueryMobile/jquery.mobile-1.3.1.css"
rel="stylesheet">
    <link rel="stylesheet" href="css/main.css">
    <script>window.$ = window.jQuery = WLJQ;</script>
    <script src="jqueryMobile/jquery.mobile-1.3.1.js"></script>
  </head>
  <body style="display: none;">
    <div data-role="page" id="page">
      <div data-role="header" id="header" data-position="fixed">
        <h3>MQTT Whiteboard</h3>
        <a onclick="app.clear()" class="ui-btn-left" data-
role="button" data-icon="delete" data-theme="b">Clear</a>
      </div>
      <canvas id="whiteboard"></canvas>
    </div>
    <script src="js/initOptions.js"></script>
    <script src="js/main.js"></script>
    <script src="js/Whiteboard.js"></script>
    <script src="js/messages.js"></script>
  </body>
</html>
```

main.css

```
/* Reset CSS */
a, abbr, address, article, aside, ……
{
  margin: 0;
  padding: 0;
}

body {
  margin: 0 auto;
}

canvas {
  position: absolute;
}
```

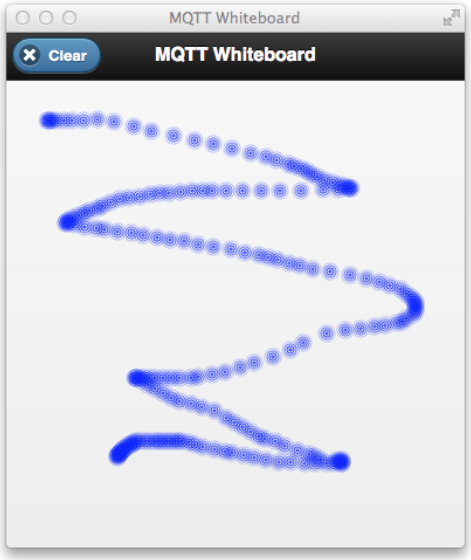


ホワイトボード – イベントをキャプチャーする

- タッチ・イベントをキャプチャーし、キャンバスに描画します。

main.js

```
function wlCommonInit(){
  window.app = new WhiteboardApp();

  $("#whiteboard").on("vmousemove", function(event) {
    if (app.drawOn) {
      var x = event.pageX;
      var y = event.pageY - $("canvas").offset().top;
      app.draw(x, y, app.size, app.color);
    }
    event.preventDefault();
  });
  $("#whiteboard").on("vmousedown", function(event) {
    var x = event.pageX;
    var y = event.pageY - $("canvas").offset().top;
    app.draw(x, y, app.size, app.color);
    app.drawOn = true;
  });
  $("#whiteboard").on("vmouseup", function(event) {
    app.drawOn = false;
  });
  $("#whiteboard").bind("tap", function(event) {
    event.preventDefault();
  });

  $(window).resize(function() { resize(); });
  resize();
}
```

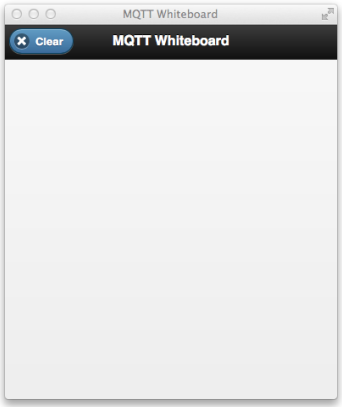


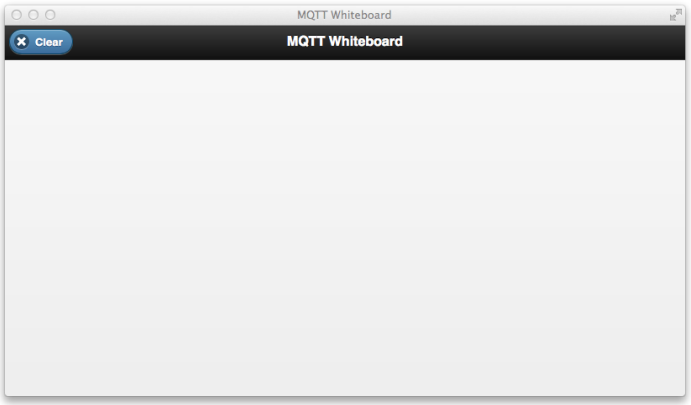
ホワイトボード – イベントをキャプチャーする

- 画面サイズが変わったときに、使用可能スペースを埋めるためにキャンバス・サイズを調整します。

main.js (cont.)

```
var resize = function() {
  var winSize = {
    width: window.innerWidth ||  document.body.clientWidth,
    height: window.innerHeight || document.body.clientHeight
  };
  // make canvas fill space under header
  $("canvas").css("top", $("#header").innerHeight() + "px");
  $("canvas")[0].width = winSize.width;
  $("canvas")[0].height = winSize.height - $("#header").innerHeight();
}
```



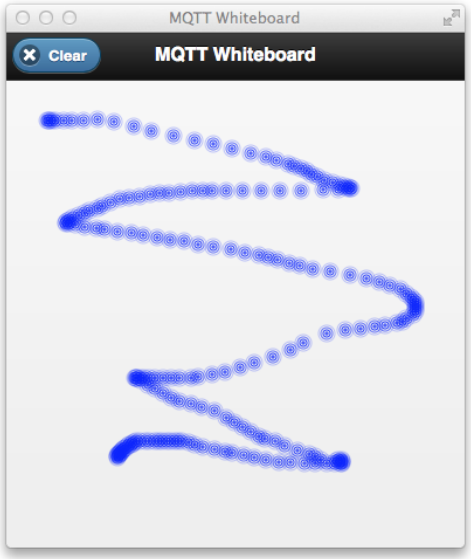


ホワイトボード– 描画する

- キャンバス描画アクションを実行する新しいオブジェクトを定義します。

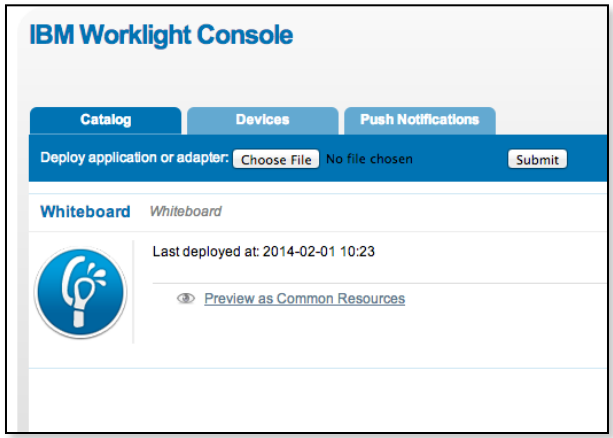
Whiteboard.js

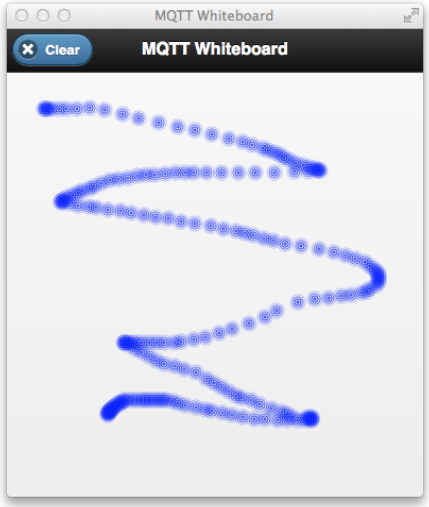
```
var Colors = [
  "rgb(255,0,0)", "rgb(0,170,0)", "rgb(0,0,255) »,
  "rgb(0,0,0), "rgb(0,255,255)", "rgb(127,255,212)",
  "rgb(139,69,19)"];

function WhiteboardApp() {
  this.size = 8;
  this.color = Colors[Math.floor(Math.random() * Colors.length)];
  this.drawOn = false;
  this.canvas = $("canvas")[0];
}

WhiteboardApp.prototype.draw = function(x, y, size, color) {
  var context = this.canvas.getContext("2d");
  for (var i = 1; i <= size; i+=2) {
    context.save();
    context.beginPath();
    var alpha = 1.0 - Math.pow(i/size, 2);
    context.globalAlpha = alpha;
    context.strokeStyle = color;
    context.arc(x, y, i, 0, 2*Math.PI);
    context.stroke();
    context.restore();
  }
}

WhiteboardApp.prototype.clear = function() {
  var context = this.canvas.getContext("2d");
  context.clearRect(0, 0, this.canvas.width, this.canvas.height);
}
```



Whiteboard – Run the Application

- MQTT メッセージングを通じてコラボレーションを追加する前に、アプリケーションを、ローカル Worklight Server にデプロイすることによってテストします。
  - **「実行 (Run As)」 > 「Development Server 上で実行 (Run on Development Server)」**を選択します。
  - http://localhost:10080/WhiteboardProject/console を開きます。
  - **「共通リソースとしてプレビュー (Preview as Common Resources)」**を選択します。
  - クリックまたはタップして、キャンバス上に描画します。





ホワイトボード – コラボレーションを追加する

- この時点で、コラボレーティブ機能を、パブリッシュ/サブスクライブ MQTT メッセージング機能を持つアプリケーションに追加できます。MQTT クライアントをホワイトボード・アプリケーションに追加し、アプリケーションのロード時に MQTT ブローカーに接続します。
- 共有キャンバスを作成するために、各ホワイトボード・クライアントは、draw アクションおよび clear アクションを、クライアントに固有のトピック上にパブリッシュします。アクションのタイプ、およびオプションについては、JSON データとしての MQTT メッセージ・ペイロードで説明されています。
- 例: **draw**
  - トピック: `whiteboard/<clientId>`
  - ペイロード: `{“type” : “draw”, “x” : “127”, “y” : “53”, “color”: “rgb(255,0,0)”}`
- 例: **clear**
  - トピック: `whiteboard/<clientId>`
  - ペイロード: `{“type” : “clear” }`
- 別のホワイトボード・クライアントがこのメッセージを受け取ると、When another Whiteboard client receives this message, `WhiteboardApp.draw()` または `WhiteboardApp.clear()` が、JSON メッセージからのパラメーターを使用して呼び出されます。

ホワイトボード – MQTT クライアントを追加する

- Eclipse Paho JavaScript MQTT クライアントをプロジェクトに追加します。
  - `WhiteboardProject/apps/Whiteboard/common/js/mqttws31.js`
- `index.html` で .js ファイルをロードします。

index.html

```
<html>
  <head>
    ...
  </head>
  <body style="display: none;">
    ...
    <script src="js/initOptions.js"></script>
    <script src="js/main.js"></script>
    <script src="js/mqttws31.js"></script>
    ...
```

- アプリケーションのロード時に、クライアントの**接続**を呼び出します (下で定義)。

main.js

```
function wlCommonInit(){
  window.app = new WhiteboardApp();
  app.connect();
  ...
```

ホワイトボード – MQTT クライアントを追加する

- MQTT クライアントを初期化し、そのクライアントに接続する際には、以下の構成パラメーターと接続オプションを指定します。

MQTT クライアント構成	
host	MQTT ブローカー・ホストの DNS 名または IP
port	ホストのポート番号
clientId	このクライアントの固有 MQTT ID (1 文字から 23 文字)
onMessageArrived	メッセージのクライアントへの送信時に呼び出されるコールバック
onConnectionLost	接続の逸失時に呼び出されるコールバック

接続オプション	
keepAliveInterval	クライアント切断後の非アクティブ期間 (秒)
onSuccess	ブローカーへの正常接続後に呼び出されるコールバック
onFailure	ブローカーへの接続失敗後に呼び出されるコールバック

ホワイトボード – MQTT クライアントを追加する

- 構成パラメーター、コールバック、および接続オプションを追加し、MQTT クライアントを接続します (下に赤で示されているコールバックを定義します)。
- 使用可能な MQTT ブローカーがない場合は、自分専用のブローカーを見つける方法に関する詳細について、このモジュールで後述されている**『MQTT ブローカーに接続する』**を参照してください。

Whiteboard.js

```
function WhiteboardApp() {
  ...
  this.host = "< My MQTT broker IP >";
  this.port = "< My MQTT broker port >";
  // generate a 6-character alpha-num unique
ID
  this.uuid = Math.random().toString(36).
    slice(2).substring(0, 6);
  this.clientId = "whiteboard-"+this.uuid;
  this.client = new Messaging.Client(
    this.host,
    this.port,
    this.clientId);
}
```

```
WhiteboardApp.prototype.connect = function() {
  this.client.onMessageArrived = (function(self) {
    return function(msg) { self.onMsg(msg); }
  })(this);
  this.client.onConnectionLost = function() {
    alert("Connection lost!");
  };
  var connectOptions = new Object();
  // extend keep-alive to one hour
  connectOptions.keepAliveInterval = 3600;
  connectOptions.onSuccess = (function(self) {
    return function() { self.onConn; }
  })(this);
  connectOptions.onFailure = function() {
    alert("Failed to connect!");
  };
  this.client.connect(connectOptions);
}
```

ホワイトボード – コールバックを追加する

- 次に、**onMsg** コールバックおよび **onConn** コールバックを実装します。
- MQTT 接続が正常に行われたら、すべての Whiteboard アプリケーションのワイルドカード・トピックにサブスクライブします: `whiteboard/+`

Whiteboard.js

```
WhiteboardApp.prototype.onConn = function() {
  this.client.subscribe("whiteboard/+");
}

WhiteboardApp.prototype.onMsg = function(msg) {
  var topic = msg.destinationName;
  var payload = msg.payloadString;

  // make sure the topic matches whiteboard/*
  if (topic.indexOf("whiteboard/") == 0) {
    var sourceUUID = topic.split("/")[1];

    // do not process own actions
    if (sourceUUID == this.uuid) { return; }

    var data = JSON.parse(payload);
    if (data.type == "draw") {
      this.draw(
       data.x, data.y,
       this.size, data.color, true);
    } else if (data.type == "clear") {
      this.clear(true);
    }
  }
}
```

他のホワイトボードからの **draw** アクションおよび **clear** アクションと、自分のホワイトボードが開始するそれらのアクションとを区別する必要があります。

自分が開始するアクションの場合、MQTT メッセージをパブリッシュすることが必要になります。MQTT メッセージからアクションを受け取った場合、このアクションを新規メッセージとして再パブリッシュする必要はありません。

ホワイトボード – アクションを MQTT メッセージとしてパブリッシュする

- 次に、**draw** アクションおよび **clear** アクション用に MQTT メッセージを作成およびパブリッシュします。

Whiteboard.js

```
WhiteboardApp.prototype.draw =
  function(x, y, size, color, fromOutside) {
  ...
  if (!fromOutside) {
    this.publishDraw(x, y, color);
  }
}

WhiteboardApp.prototype.clear =
  function(fromOutside) {
  ...
  if (!fromOutside) {
    this.publishClear();
  }
}
```

MQ Telemetry Transport は、クライアントとサーバー間のメッセージ送信用に 3 つの Quality of Service を提供します。

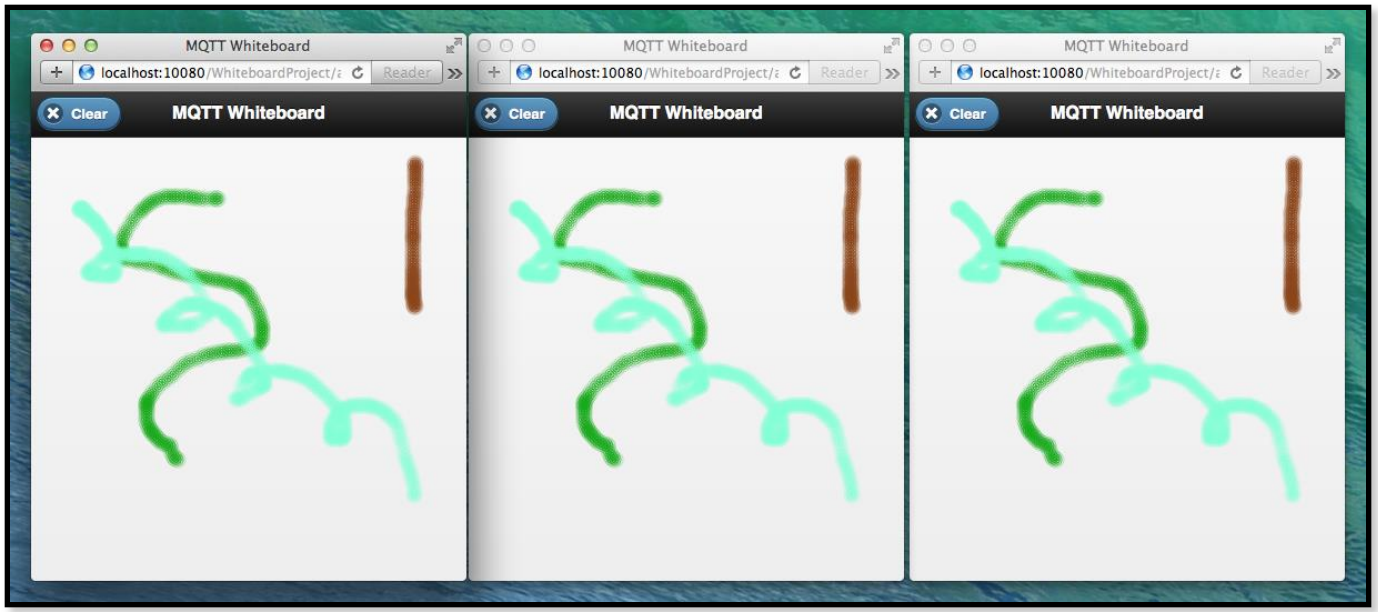
QoS 0 = “at most once”
QoS 1 = “at least once”
QoS 2 = “exactly once”

retained メッセージは、MQTT サーバー上に保持され、メッセージ・トピックに対する新規サブスクライバーに送信されます。

```
WhiteboardApp.prototype.publishDraw = function(x, y,
color) {

  var topic = "whiteboard/” + this.uuid;
  var data = JSON.stringify({
    type: "draw",
    x: x,
    y: y,
    color: color,
  });

  var msg = new Messaging.Message(data);
  msg.destinationName = topic;
  msg.qos = 0;
  msg.retained = false;

  this.client.send(msg);
}

WhiteboardApp.prototype.publishClear = function() {

  var topic = "whiteboard/” + this.uuid;
  var data = JSON.stringify({
    type: "clear"
  });

  var msg = new Messaging.Message(data);
  msg.destinationName = topic;
  msg.qos = 0;
  msg.retained = false;

  this.client.send(msg);
}
```

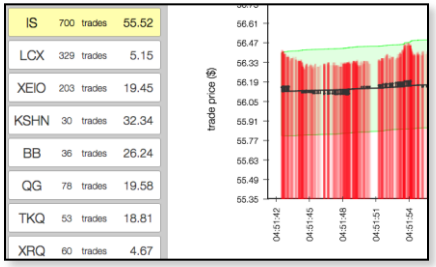
ホワイトボード– すべて同時に書き込む

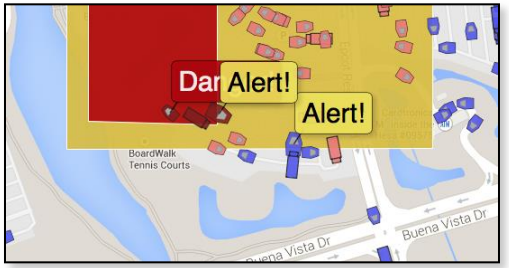
- Worklight 開発サーバーに再度デプロイし、アプリケーションを 2 つ (以上) のウィンドウで開きます。
- 1 つのホワイトボードに描画し、MQTT メッセージが他のすべてのサブスクライブ・ホワイトボード・アプリケーションにパブリッシュされます。



コラボレーション・シナリオを拡張する

- このコラボレーション・シナリオは、他のタイプのリアルタイム・アプリケーションを作成するために変更および拡張することができます。
  - **ストック・ティッカー**: クライアントは為替の取引データ (例えば、MarketData/NYSE/IBM) にサブスクライブし、リアルタイムに更新します。
  - **チャット・アプリケーション**: クライアントは特定のチャット・ルーム (例えば、chat/room123/user1) のトピックに対してパブリッシュおよびサブスクライブします。
  - **リアルタイム緊急警報**: クライアントは特定の地理的位置のトピックにサブスクライブし、 緊急サービスはアラートをこれらのトピックにパブリッシュします (例えば、emergency/<X>/<Y>/alert)。





アジェンダ

- MQ Telemetry Transport 入門
- コラボレーティブ・アプリケーション (ホワイトボード) を作成する
- MQTT ブローカーに接続する
  - Mosquitto
  - IBM MessageSight

MQTT ブローカーに接続する

- MQTT クライアント間でメッセージの受け渡しを行うには、MQTT サーバーが必要です。JavaScript MQTT クライアント (Eclipse Paho など) をサポートするには、MQTT サーバーは WebSocket トランスポートを実装する必要があります。
- オープン・ソースで業務用の、複数の MQTT サーバーが使用可能です。
  - 開発用 MQTT サーバー:
    - Mosquitto (オープン・ソース、マルチプラットフォーム)
    - IBM MessageSight for Developers (仮想アプライアンス・イメージ)
  - エンタープライズ MQTT サーバー:
    - IBM MessageSight (アプライアンス)

Mosquitto

- Mosquitto は、Eclipse Paho メッセージング・プロジェクトの一部である、MQ Telemetry Transport V3.1 のオープン・ソースかつ軽量な実装環境です。
- Mosquitto ブローカーは、数多くのプラットフォーム (Windows、OS X、Linux ディストリビューション) で使用可能です。
- 詳細:
  - http://mosquitto.org
  - http://projects.eclipse.org/projects/technology.mosquitto
- Mosquitto のダウンロード:
  - http://mosquitto.org/download/

IBM MessageSight for Developers

- IBM MessageSight for Developers は、仮想 IBM MessageSight アプライアンス・イメージです。
- MessageSight は、強力なセキュリティーと管理の容易さを備えた、低遅延で信頼性が高く、拡張が容易なメッセージング・サーバーです。
- 開発者イメージを、VMWare、Oracle VirtualBox、KVM などの仮想化ソフトウェアで実行することができます。
- 詳細:
  - http://ibm.com/messagesight
  - https://www.ibmdw.net/messaging/messagesight/
  - http://www.youtube.com/watch?v=kcEDoRqhkhA

特記事項

- これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。
- 本書は米国 IBM が提供する製品およびサービスについて作成したものです。
- 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。
- IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。
  - 〒103-8510
    東京都中央区日本橋箱崎町19番21号
    日本アイ・ビー・エム株式会社
    法務・知的財産
    知的財産権ライセンス渉外
- **以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含む すべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。**
- この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。
- 本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。
- IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、 自ら適切と信ずる方法で、使用もしくは配布することができるものとします。
- 本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- 本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。
- 本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。
- IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

- 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。
- それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。
  - © (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

プライバシー・ポリシーの考慮事項

- サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。
- このソフトウェア・オファリングは、展開される構成に応じて、(アプリケーション・サーバーが生成する) セッション情報を収集するセッションごとの Cookie を使用する場合があります。これらの Cookie は個人情報を含まず、セッション管理のために要求されるものです。加えて、匿名ユーザーの認識および管理のために持続的な Cookie が無作為に生成される場合があります。これらの Cookie も個人情報を含まず、要求されるものです。
- この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

サポートおよびコメント

- IBM Worklight の一連の文書、トレーニング資料、および質問をポストできるオンライン・フォーラムはすべて、次の IBM Web サイトからご覧になれます。
  - http://www.ibm.com/mobile-docs
- **サポート**
  - ソフトウェア・サブスクリプション & サポート (ソフトウェア・メンテナンスと呼ばれる場合もあります) は、パスポート・アドバンテージおよびパスポート・アドバンテージ・エクスプレスから購入されたライセンスに含まれています。International Passport Advantage Agreement および IBM International Passport Advantage Express Agreement の追加情報については、次のパスポート・アドバンテージ Web サイトを参照してください。
    - http://www.ibm.com/software/passportadvantage
  - ソフトウェア・サブスクリプション & サポートが有効になっている場合、IBM は、インストールおよび使用法 (ハウツー) に関する短期間の FAQ に対するサポートや、コード関連の質問に対するサポートを提供します。詳しくは、次の IBM ソフトウェア・サポート・ハンドブックを参照してください。
    - http://www.ibm.com/support/handbook
- **ご意見**
  - 本資料に関するご意見をお寄せください。本資料の具体的な誤りや欠落、正確性、編成、題材、または完成度に関するご意見をお寄せください。 お寄せいただくご意見は、本マニュアルまたは製品の情報、およびその情報の提示方法に関するもののみとしてください。
  - 製品の技術的な質問および情報、および価格については、担当の IBM 営業所、IBM ビジネス・パートナー、または認定リマーケターにお問い合わせください。
  - IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。IBM またはいかなる組織も、お客様から提示された問題についてご連絡を差し上げる場合にのみ、お客様が提供する個人情報を使用するものとします。
  - どうぞよろしくお願いいたします。
  - 次の IBM Worklight Developer Edition サポート・コミュニティーにご意見をお寄せください。
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - IBM からの回答を希望される場合は、以下の情報をご連絡ください。
    - 氏名
    - 住所
    - 企業または組織
    - 電話番号
    - E メール・アドレス

ありがとうございました