用户或者开发人员打开index.html页面，然后点击"Send To Client"按钮：

POST http://xxxx:9999/push

返回内容：MPUSH返回推送响应内容

JDK HTTP server在接收到请求时，会根据context找到相应HttpHanlder，调用PushHandler#handle()

```java
final class PushHandler implements HttpHandler {
    private final Logger logger = LoggerFactory.getLogger(this.getClass());
    private final PushSender pushSender = PushSender.create();
    private final AtomicInteger idSeq = new AtomicInteger();

    public void start() {
        pushSender.start();
    }

    public void stop() {
        pushSender.stop();
    }

    @SuppressWarnings("unchecked")
    @Override
    public void handle(HttpExchange httpExchange) throws IOException {
        String body = new String(readBody(httpExchange), Constants.UTF_8);
        Map<String, Object> params = Jsons.fromJson(body, Map.class);

        sendPush(params);

        byte[] data = "服务已经开始推送,请注意查收消息".getBytes(Constants.UTF_8);
        httpExchange.getResponseHeaders().set("Content-Type", "text/plain; charset=utf-8");
        httpExchange.sendResponseHeaders(200, data.length);//200, content-length
        OutputStream out = httpExchange.getResponseBody();
        out.write(data);
        out.close();
        httpExchange.close();
    }

    private void sendPush(Map<String, Object> params) {
        String userId = (String) params.get("userId");
        String hello = (String) params.get("hello");
        Boolean broadcast = (Boolean) params.get("broadcast");
        String condition = (String) params.get("condition");


        NotificationDO notificationDO = new NotificationDO();
        notificationDO.content = "MPush开源推送，" + hello;
        notificationDO.title = "MPUSH推送";
        notificationDO.nid = idSeq.get() % 2 + 1;
        notificationDO.ticker = "你有一条新的消息,请注意查收";
        PushMsg pushMsg = PushMsg.build(MsgType.NOTIFICATION_AND_MESSAGE, Jsons.toJson(notificationDO));
        pushMsg.setMsgId("msg_" + idSeq.incrementAndGet());

        pushSender.send(PushContext
                .build(pushMsg)
                .setUserId(Strings.isBlank(userId) ? null : userId)
                .setBroadcast(broadcast != null && broadcast)
                .setCondition(Strings.isBlank(condition) ? null : condition)
                .setCallback(new PushCallback() {
                    @Override
                    public void onResult(PushResult result) {
                        logger.info(result.toString());
                    }
                })
        );
```

```
        );
    }

    private byte[] readBody(HttpExchange httpExchange) throws IOException {
        InputStream in = httpExchange.getRequestBody();
        String length = httpExchange.getRequestHeaders().getFirst("content-length");
        if (length != null && !length.equals("0")) {
            byte[] buffer = new byte[Integer.parseInt(length)];
            in.read(buffer);
            in.close();
            return buffer;
        } else {
            ByteArrayOutputStream out = new ByteArrayOutputStream(1024);
            byte[] buffer = new byte[1024];
            int len = 0;
            while ((len = in.read(buffer)) != -1) {
                out.write(buffer, 0, len);
            }
            in.close();
            return out.toByteArray();
        }
    }

    public static final class NotificationDO {
        public String msgId;
        public String title;
        public String content;
        public Integer nid; //主要用于聚合通知，非必填
        public Byte flags; //特性字段。 0x01:声音  0x02:震动  0x03:闪灯
        public String largeIcon; // 大图标
        public String ticker; //和title一样
        public Integer number;
        public Map<String, String> extras;
    }
}
```