1. WebSockets en java (JSR 356)

1.1. Vue d'ensemble sur l'api JSR 356

JSR356 / "Java Api for WebSocket" fait partie des spécifications **JEE 7** .

Une instance d'une classe d'implémentation annotée via **@ServerEndpoint** est automatiquement créée **pour chaque nouvelle connexion établie** .

...

1.2. Configuration des implémentations existantes (de l'appli)

```
package websocket;

import java.util.HashSet;  import java.util.Set;
import javax.websocket.Endpoint;
import javax.websocket.server.ServerApplicationConfig;
import javax.websocket.server.ServerEndpointConfig;
import websocket.xy.XyEndpoint;

public class MyWebSocketsConfig implements ServerApplicationConfig {

   @Override
   public Set<ServerEndpointConfig> getEndpointConfigs(
         Set<Class<? extends Endpoint>> scanned) {

      Set<ServerEndpointConfig> result = new HashSet<>();

      if (scanned.contains(XyEndpoint.class)) {
         result.add(ServerEndpointConfig.Builder.create(
               XyEndpoint.class,
               "/websocket/xy").build());
      }

      return result;
   }
```

```
    @Override
    public Set<Class<?>> getAnnotatedEndpointClasses(Set<Class<?>> scanned) {
        Set<Class<?>> results = new HashSet<>();
        for (Class<?> clazz : scanned) {
            if (clazz.getPackage().getName().startsWith("websocket.")) {
                results.add(clazz);
            }
        }
        return results;
    }
}
```

1.3. Exemple de "Chat" : coté serveur (dans tomcat)

```
package websocket.chat;

import java.io.IOException;
import java.util.Set;
import java.util.concurrent.CopyOnWriteArraySet;
import java.util.concurrent.atomic.AtomicInteger;

import javax.websocket.OnClose;
import javax.websocket.OnError;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.ServerEndpoint;

import org.apache.juli.logging.Log;
import org.apache.juli.logging.LogFactory;

import util.HTMLFilter;

@ServerEndpoint(value = "/websocket/chat")
public class ChatAnnotation {
```

```
private static final Log log = LogFactory.getLog(ChatAnnotation.class);

private static final String GUEST_PREFIX = "Guest";
private static final AtomicInteger connectionIds = new AtomicInteger(0);
private static final Set<ChatAnnotation> connections =
        new CopyOnWriteArraySet<>();

private final String nickname;
private Session session;   //javax.websocket.Session

public ChatAnnotation() {
   nickname = GUEST_PREFIX + connectionIds.getAndIncrement();
}


@OnOpen
public void start(Session session) {
   this.session = session;
   connections.add(this);  // une instance de cette classe construite pour chaque connexion
   String message = String.format("* %s %s", nickname, "has joined.");
   broadcast(message);
}


@OnClose
public void end() {
   connections.remove(this);
   String message = String.format("* %s %s", nickname, "has disconnected.");
   broadcast(message);
}


@OnMessage
public void incoming(String message) {
   // Never trust the client
   String filteredMessage = String.format("%s: %s",
         nickname, HTMLFilter.filter(message.toString()));
```

```
        broadcast(filteredMessage);
    }


    @OnError
    public void onError(Throwable t) throws Throwable {
        log.error("Chat Error: " + t.toString(), t);
    }


    private static void broadcast(String msg) {
        for (ChatAnnotation client : connections) {
            try {
                synchronized (client) {
                    client.session.getBasicRemote().sendText(msg);
                }
            } catch (IOException e) {
                log.debug("Chat Error: Failed to send message to client", e);
                connections.remove(client);
                try {
                    client.session.close();
                } catch (IOException e1) {
                    // Ignore
                }
                String message = String.format("* %s %s",
                        client.nickname, "has been disconnected.");
                broadcast(message);
            }
        }
    }
}
```