WebSocket in Java EE

Štěpán Kopřiva
stepan.kopriva@oracle.com

Agenda

- WebSocket Protocol
- Protocol Analysis and Usage
- WebSocket API for Java
- Project Tyrus
- Demo

WebSocket Protocol

- IETF RFC 6455 Standard
- Bi – directional, full duplex communication
- One TCP connection
- Stream of messages
- Communication over standard TCP port 80
- ws:// and wss://

Establish a Connection

Handshake Request

Handshake Request

Http Request

```
GET /endpoint HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: megachat, chat
Sec-WebSocket-Extensions : compress, mux
Sec-WebSocket-Version: 13
Origin: http://example.com
```

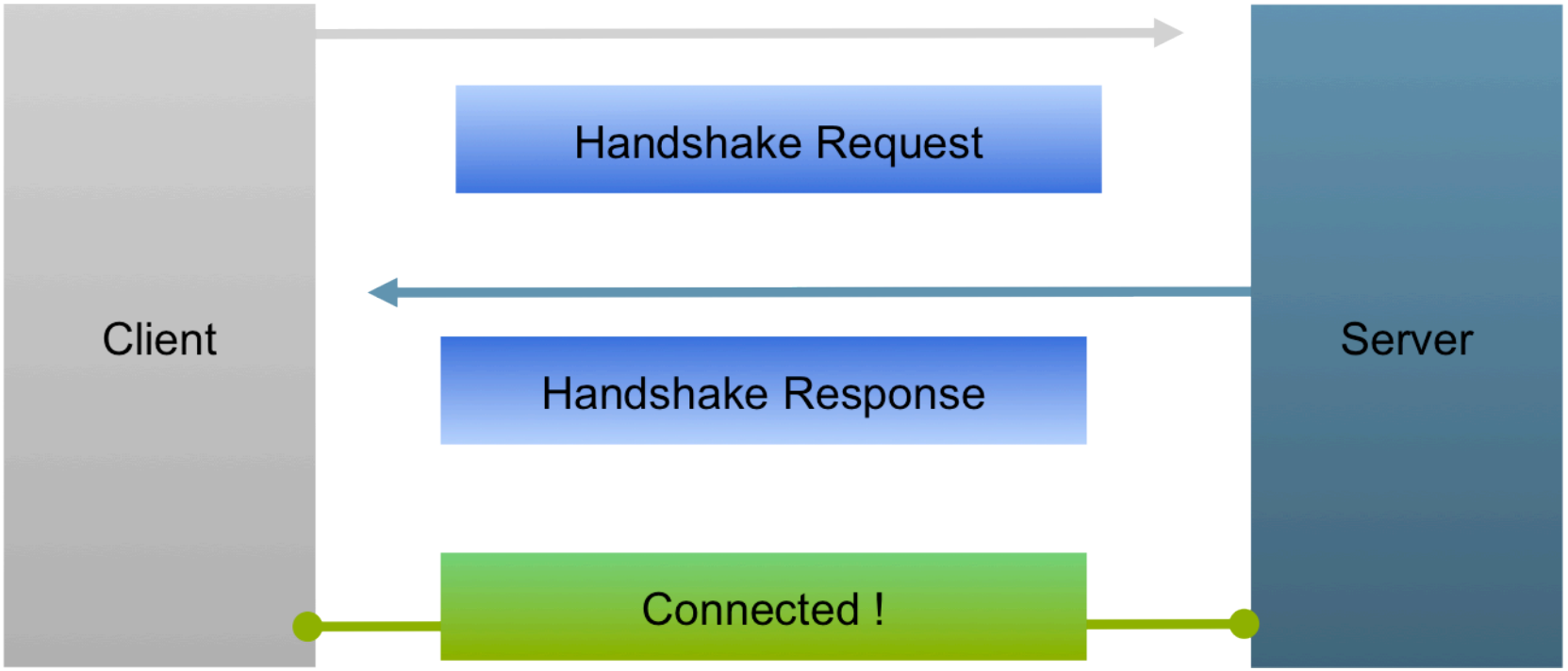
Handshake Response

Handshake Response

Http Response

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Extensions: compress, mux
```

Establish a connection



WebSocket Events & Lifecycle

- Open
- Message
- Error
- Close

WebSocket Pros & Cons (vs. HTTP)

- Small message size (small overhead)
- Decreased latency
- Useful for following applications:
  - Small messages (updates)
  - A lot of messages
  - Ticker update, games, chat

WebSocket vs. AJAX experiment

- Localhost
  - AJAX: 1000 iterations in 2.934 s
  - WS: 1000 iterations in 0.466 s
- England - California
  - AJAX: 1000 iterations in 263.7 s
  - WS: 1000 iterations in 227.7 s
- http://www.peterbe.com/plog/are-websockets-faster-than-ajax

WebSocket Scalability

- Extension which
  - Separates logical connection from physical one
  - Allows multiple logical connections to share physical connection

JSR 356 Java API for Web Socket

- Specification
  - http://jcp.org/en/jsr/detail?id=356
  - http://java.net/projects/websocket-spec
  - Early Draft Review
  - Will be in Java EE 7
- Reference Implementation
  - http://tyrus.java.net
  - Bundled in latest Glassfish 4 builds

JSR 356 Java API for Web Socket

- Package javax.websocket
- Annotated API
- Programmatic API
- Client API

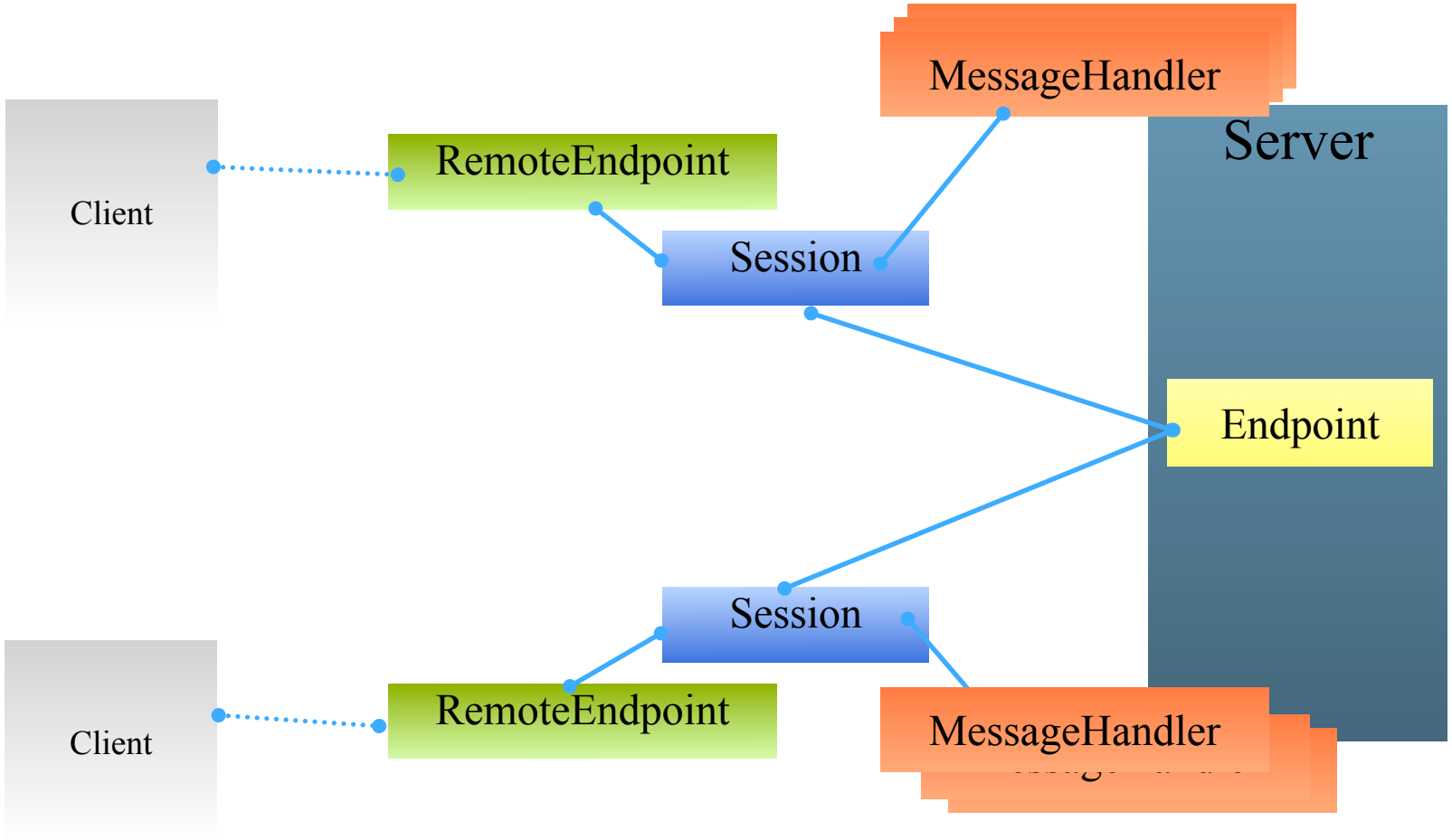
Hello World Server - Annotated

```
@WebSocketEndpoint(value="/hello")
public class HelloServer {
    @WebSocketMessage
    public void onMessage(String message, Session session) {
        session.getRemote().sendString("Hello "+text);
    }
}
```

Main API classes

Endpoint	Intercepts websocket lifecycle events
MessageHandler	Handles all incoming messages for an Endpoint
RemoteEndpoint	Represents the ‘other end’ of this conversation
Session	Represents the active conversation

Object Model: Server Side



WebSocketEndpoint Annotation

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface WebSocketEndpoint {

    public String value();

    public String[] subprotocols() default {};

    public Class<? extends Decoder>[] decoders() default {};

    public Class<? extends Encoder>[] encoders() default {};
}
```

WebSocketMessage Annotation

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface WebSocketMessage {
    public long maxMessageSize() default -1;
}
```

Other Method Annotations

- WebSocketOpen
- WebSocketClose
- WebSocketError

Hello World Server - Programmatic

```
public class HelloServer extends Endpoint {
    @Override
    public void onOpen(Session session) {
      session.addMessageHandler(new MessageHandler.Basic<String> {
           public void onMessage(String text) {
              session.getRemote().sendString("Hello "+text);
           }
      });
    }
    ...
}
```

Endpoint

```
public abstract class Endpoint {
    public abstract EndpointConfiguration
    getEndpointConfiguration();

    public abstract void onOpen(Session session);

    public void onClose(CloseReason closeReason) { }

    public void onError(Throwable thr) { }
}
```

Endpoint Configuration

```
public interface EndpointConfiguration {

    List<Encoder> getEncoders();

    List<Decoder> getDecoders();
}
```

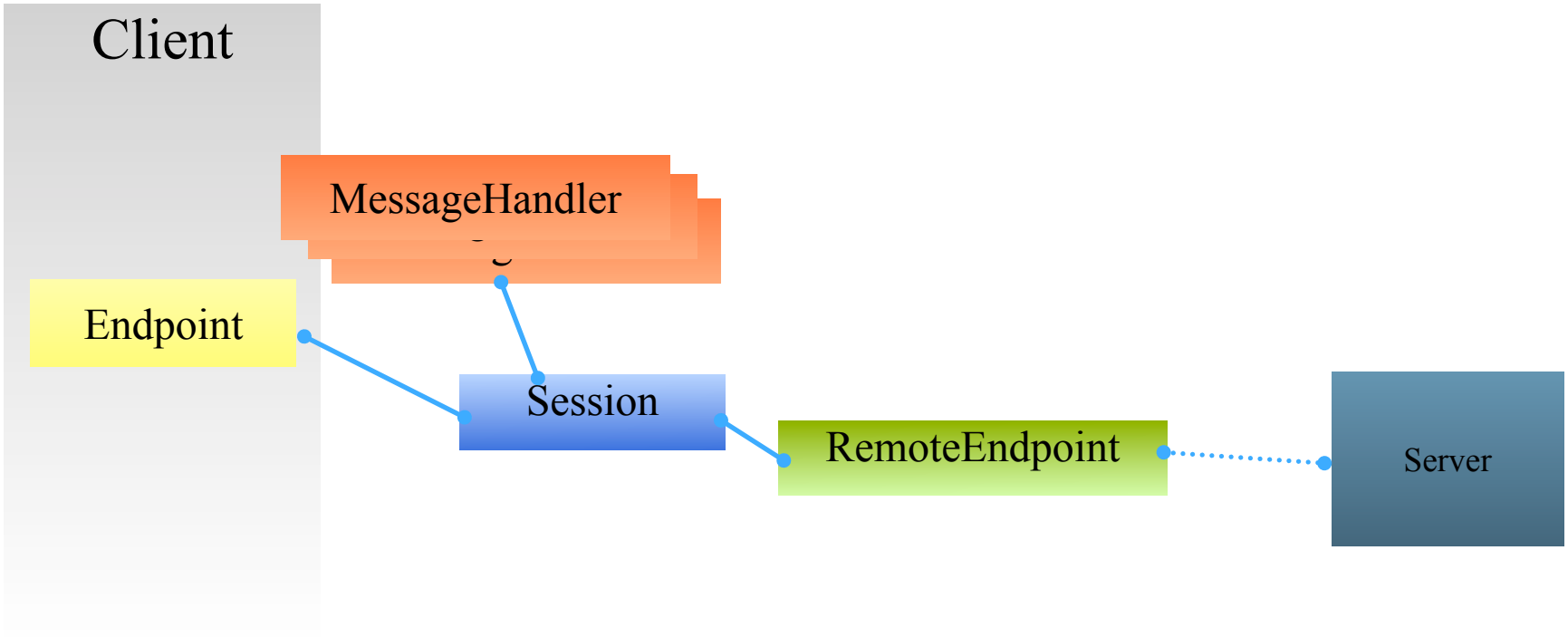
Session

```
public interface Session {
    RemoteEndpoint getRemote();

    void addMessageHandler(MessageHandler listener);

    Set<MessageHandler> getMessageHandlers();

    void removeMessageHandler(MessageHandler listener);

    void close() throws IOException;
    ...
}
```

RemoteEndpoint

```
public interface RemoteEndpoint {
    void sendString(String text) throws IOException;

    void sendBytes(ByteBuffer data) throws IOException;

    void sendPartialString(String fragment, boolean isLast)
       throws IOException;

    void sendPartialBytes(ByteBuffer partialByte, boolean
          isLast) throws IOException;

    void sendObject(Object o) throws IOException,;
}
```

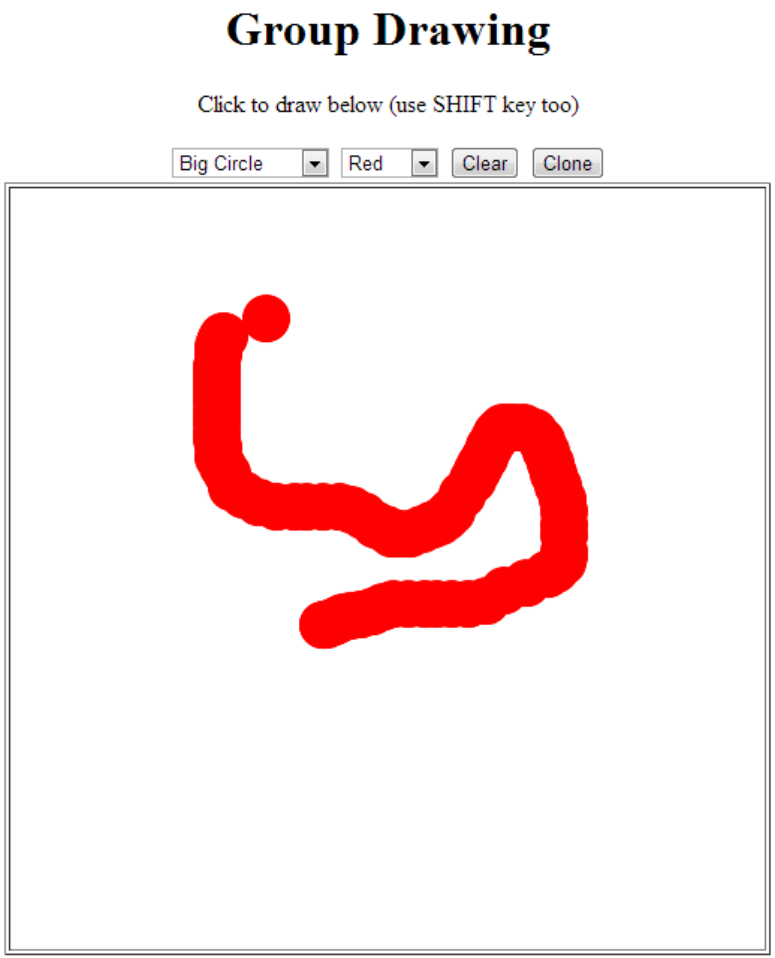
Hello Client - Programmatic



Hello World Client

```
public class HelloClient extends Endpoint {
    @Override
    public void onOpen(Session session) {
      session.addMessageHandler(new MessageHandler.Basic<String> {
           public void onMessage(String text) {
              System.out.println("Server said: "+text);
           }
      });
      session.getRemote().sendString("Peter");
    }
    ...
}
```

Draw Server Demo



Thank You!

- Give it a try
  - http://tyrus.java.net
  - https://github.com/jersey/hol-sse-websocket
- Specification
  - http://jcp.org/en/jsr/detail?id=356
  - http://java.net/projects/websocket-spec
  - Early Draft Review
  - Will be in Java EE 7