Websockets

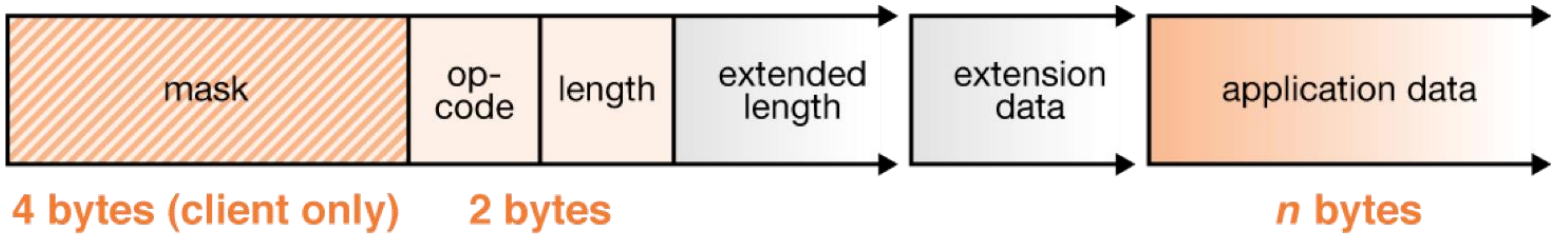

```
Part of WebSocket Interface:
attribute Function onopen;
attribute Function onmessage;
attribute Function onerror;
attribute Function onclose;
```

websockets

- an IETF standard recognized by RFC 6455. To be specific,
- it's a protocol which works **on top of TCP.**

websockets

- Same underlying TCP/IP connection.
  - HTTP replaced by the WebSocket connection
  - same ports as HTTP (80) and HTTPS (443), by default.
- WebSocket data frames
  - can be sent back and forth between the client and the server in full-duplex mode.
  - Both



It is a standard

The WebSocket API

Editor's Draft 4 June 2014

Latest Published Version:
http://www.w3.org/TR/websockets/
Latest Editor's Draft:
http://dev.w3.org/html5/websockets/
Previous Versions:
http://www.w3.org/TR/2009/WD-websockets-20090423/
http://www.w3.org/TR/2009/WD-websockets-20091029/
Editor:
Ian Hickson, Google, Inc.

Abstract

http://dev.w3.org/html5/websockets/

Type to search

Java WebSocket API Handbook

Published with GitBook

About

Table of Contents

> **Head over to Leanpub to grab a PDF version of this book**

The book is what it says it is - a handbook, a quick reference, a fast track guide. It covers the nitty gritty of the **Java API for WebSocket**: a Java based standard (specification) for building WebSocket based applications. As with most standard (Java) APIs, the WebSocket API has multiple (competing) implementations which comply with the specification. It is also a part of the **Java EE Platform**.

> What it implies is that you can use this standard API in a standalone format as well as part of a larger platform in concert with other Java EE based APIs such as JAX-RS, CDI, JPA, EJB, JMS etc.

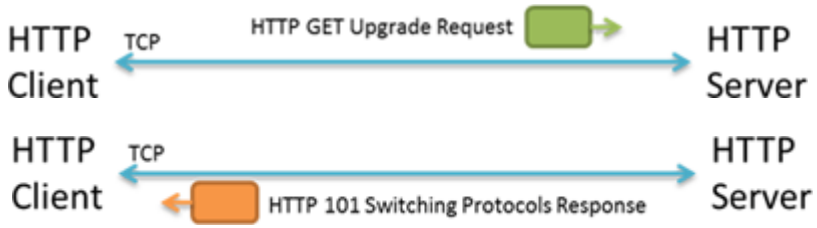
java websocket api handbook
https://abhirockzz.gitbooks.io/java-websocket-api-handbook/content/

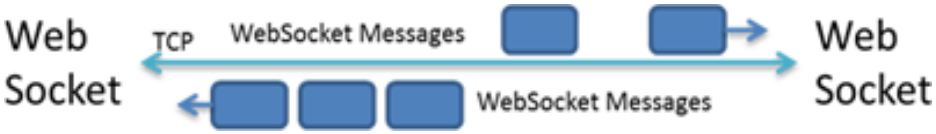
websockets

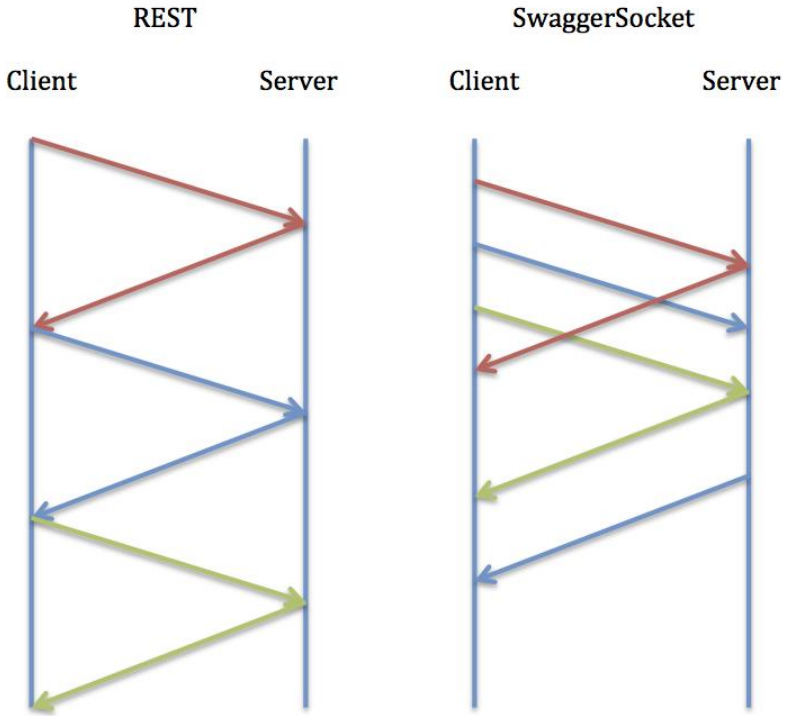
- **Bi-directional**:
  - both server and client can initiate a communication
- **Full duplex**:
  - once the WebSocket session is established, both server and client can communicate independent of each other

- connection piggybacks initial handshake (HTTP Upgrade mechanism).
- Once established, the underlying TCP **connection remains open**

initial handshake over http

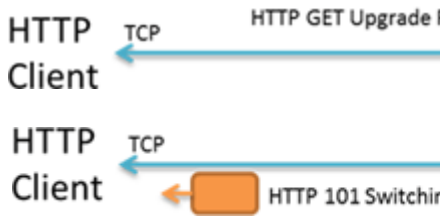


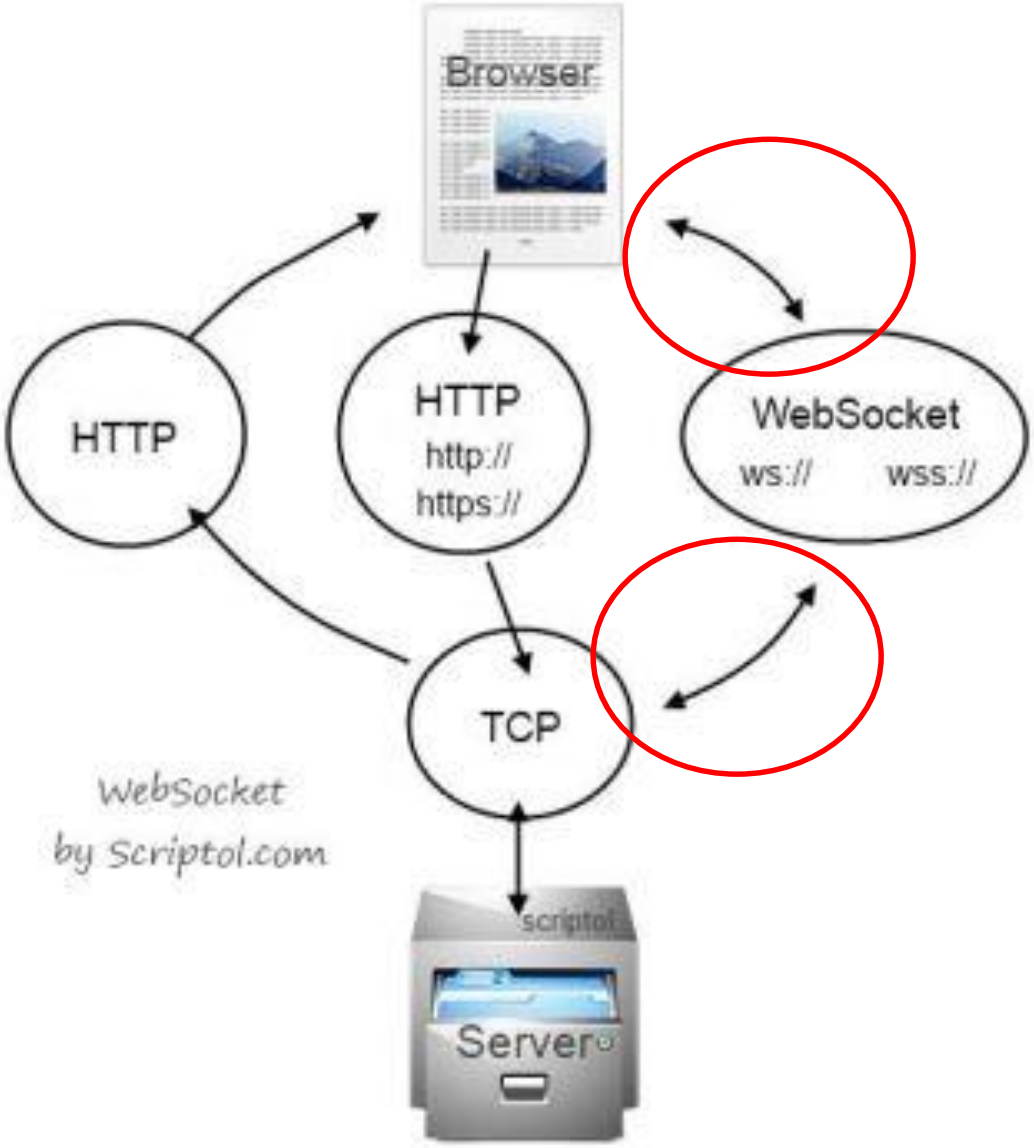




http://blog.wordnik.com/introducing-swaggersocket-a-rest-over-websocket-protocol
http://blogs.msdn.com/b/ie/archive/2012/03/19/websockets-in-windows-consumer-preview.aspx

Some c nous





http://blog.wordnik.com/introducing-swaggersocket-a-rest-over-websocket-protocol
http://blogs.msdn.com/b/ie/archive/2012/03/19/websockets-in-windows-consumer-preview.aspx
http://www.scriptol.com/ajax/xhr.php

Browsers…

Web Sockets - CR

Bidirectional communication technology for web apps

Global	84.54% + 1.42% =	85.96%
unprefixed:	84.54% + 1.27% =	85.81%

Current aligned | Usage relative | Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		35						
		36					4.1	
8		37					4.3	
9	34	38					4.4	
10	35	39	7.1	26	7.1		4.4.4	
11	36	40	8	27	8.1	8	37	40
TP	37	41		28				
	38	42		29				
	39	43						

= Supported = Not supported = Partial support = Support unknown

http://caniuse.com/#feat=websockets

Browsers…

Web Sockets - CR

Bidirectional communication technology for web apps

Global 84.54% + 1.42% = 85.96%

unprefixed: 84.54% + 1.27% = 85.81%

Current aligned | Usage relative | Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		35						
		36						
8		37						
9	34	38					4.4	
10	35	39					4.4.4	
11	36	40				8	37	40
TP	37	41						
	38	42		29				
	39	43						

Browser side client (e.g. java script)
OUT OF SCOPE

= Supported = Not supported = Partial support = Support unknown

http://caniuse.com/#feat=websockets

Servers…

NetBeans | NetBeans IDE | NetBeans Platform | Enterprise | Plugins | Docs & Support | Community

HOME / Docs & Support

PRINTABLE VERSION

Using the WebSocket API in a Web Application

This tutorial demonstrates how to create a simple web application that enables collaboration between client browsers that are connected to a single server

application

When the b

binary mes

In this tutor

and the ap

and creatin

353) to pro

The applica

the page is

implementa

Note

blog

Web

Links

- Docs Home
- FAQ
- User Comment

User Guide

- 1) Introduction
- 2) Setup
- 3) First webapp
- 4) Deployer
- 5) Manager
- 6) Realms and
- 7) Security Ma
- 8) JNDI Resou
- 9) JDBC DataS
- 10) Classloadi
- 11) JSP
- 12) SSL/TLS
- 13) SSI
- 14) CGI
- 15) Proxy Sup
- 16) MBean De

Apache Tomcat 7

Version 7.0.59, Jan 28 2015

The Apac

Software Founda

http://www.apache.

jetty://

Version: 9.2.6.v20141205

Google™ Custom Search

Jetty WebSocket Server API

Chapter 30. Jetty Websocket API

Previous

Home

Contact the core Jetty developers at www.webtide.com

private support for your internal/customer projects … custom extensions and distributions … versioned s

scalability guidance for your apps and Ajax/Comet projects … development services from 1 day to full pro

Jetty WebSocket Server API

The Jetty WebSocketServlet

Using the WebSocketCreator

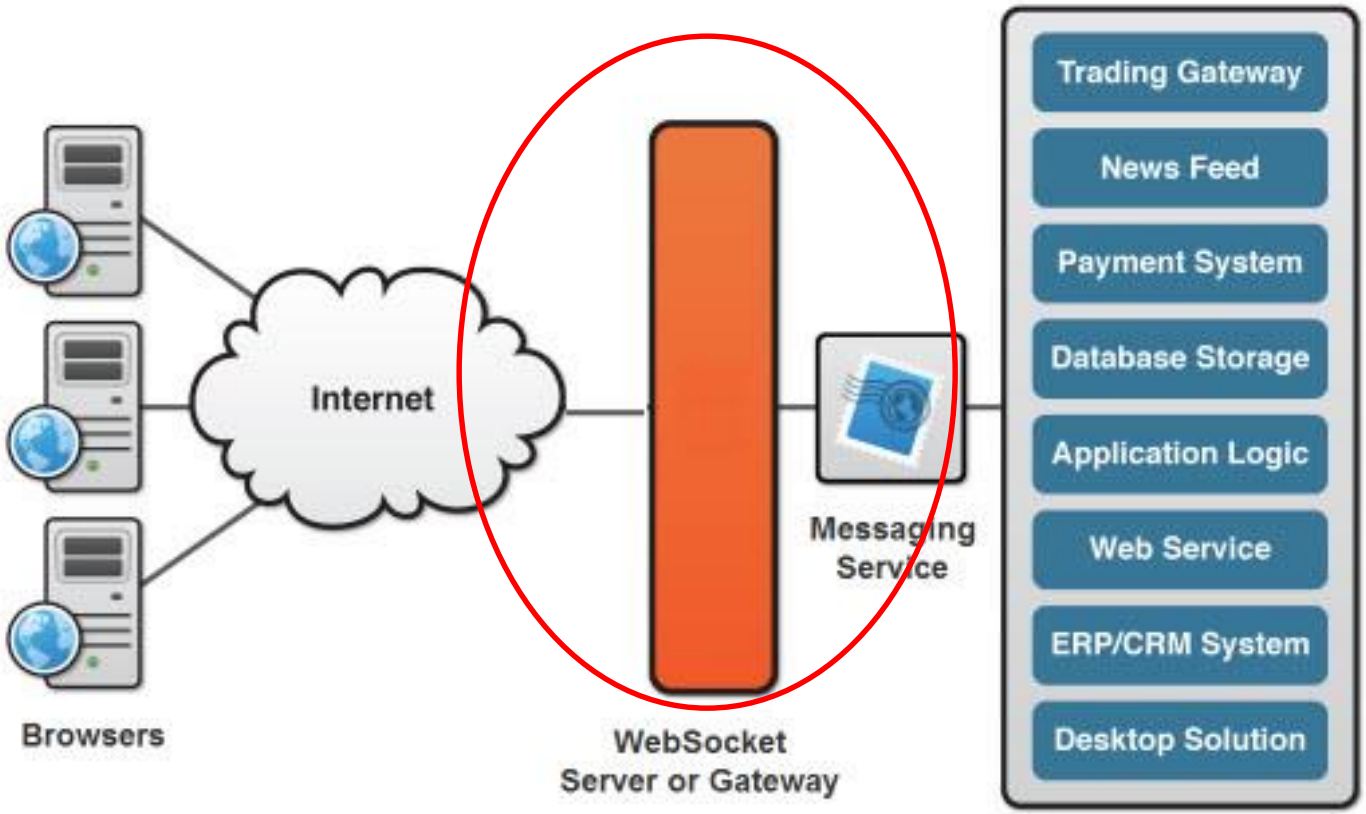
Jetty provides the ability to wire up WebSocket endpoints to Servlet Path Specs via the use of a WebSocke

https://netbeans.org/kb/docs/javaee/maven-websocketapi.html

http://tomcat.apache.org/tomcat-7.0-doc/web-socket-howto.html

http://www.eclipse.org/jetty/documentation/9.2.6.v20141205/jetty-websocket-server-api.html

Architecture: remind you something?



https://www.websocket.org/aboutwebsocket.html

Events

The Events

We'll be using three events:

- **onopen:** When a socket has opened
- **onmessage:** When a message has been received
- **onclose:** When a socket has been closed


```
Part of WebSocket Interface:
attribute Function onopen;
attribute Function onmessage;
attribute Function onerror;
attribute Function onclose;
```

http://code.tutsplus.com/tutorials/start-using-html5-websockets-today--net-13270

First the server

```java
package com.shekhar.wordgame.server;

import java.io.BufferedReader;
import java.io.InputStreamReader;

import org.glassfish.tyrus.server.Server;

public class WebSocketServer {

    public static void main(String[] args) {
        runServer();
    }

    public static void runServer() {
        Server server = new Server("localhost", 8025, "/websockets", WordgameServerEndpoint.class);

        try {
            server.start();
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please press a key to stop the server.");
            reader.readLine();
        } catch (Exception e) {
            throw new RuntimeException(e);
        } finally {
            server.stop();
        }
    }
}
```

How to build Java WebSocket Applications Using the JSR 356 API
https://blog.openshift.com/how-to-build-java-websocket-applications-using-the-jsr-356-api/

Server endpoint

```java
@ServerEndpoint(value = "/game")
public class WordgameServerEndpoint {

    private Logger logger = Logger.getLogger(this.getClass().getName());

    @OnOpen
    public void onOpen(Session session) {
        logger.info("Connected ... " + session.getId());
    }

    @OnMessage
    public String onMessage(String message, Session session) {
        switch (message) {
        case "quit":
            try {
                session.close(new CloseReason(CloseCodes.NORMAL_CLOSURE, "Game ended"));
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            break;
        }
        return message;
    }

    @OnClose
    public void onClose(Session session, CloseReason closeReason) {
        logger.info(String.format("Session %s closed because of %s", session.getId(), closeReason));
    }
}
```

How to build Java WebSocket Applications Using the JSR 356 API

https://blog.openshift.com/how-to-build-java-websocket-applications-using-the-jsr-356-api/

Client endpoint

```java
@ServerEndpoint(value = "/game")
public class WordgameServerEndpoint {

    private Logger logger = Logger.get

    @OnOpen
    public void onOpen(Session sessio
        logger.info("Connected ... " +
    }

    @OnMessage
    public String onMessage(String me
        switch (message) {
        case "quit":
            try {
                session.close(new Clo
            } catch (IOException e) {
                throw new RuntimeExce
            }
            break;
        }
        return message;
    }

    @OnClose
    public void onClose(Session sessi
        logger.info(String.format("Se
    }
}
```

```java
@ClientEndpoint
public class WordgameClientEndpoint {

    private Logger logger = Logger.getLogger(this.getClass().getName());

    @OnOpen
    public void onOpen(Session session) {
        logger.info("Connected ... " + session.getId());
        try {
            session.getBasicRemote().sendText("start");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    @OnMessage
    public String onMessage(String message, Session session) {
        BufferedReader bufferRead = new BufferedReader(new InputStreamReader(System.in));
        try {
            logger.info("Received ...." + message);
            String userInput = bufferRead.readLine();
            return userInput;
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    @OnClose
    public void onClose(Session session, CloseReason closeReason) {
        logger.info(String.format("Session %s close because of %s", session.getId(), closeReason));
    }

}
```

How to build Java WebSocket Applications Using the JSR 356 API
https://blog.openshift.com/how-to-build-java-websocket-applications-using-the-jsr-356-api/

The client

```
@ClientEndpoint
public class WordgameClientEndpoint {

    private static CountDownLatch latch;

    private Logger logger = Logger.getLogger(this.getClass().getName());

    @OnOpen
    public void onOpen(Session session) {
        // same as above
    }

    @OnMessage
    public String onMessage(String message, Session session) {
        // same as above
    }

    @OnClose
    public void onClose(Session session, CloseReason closeReason) {
        logger.info(String.format("Session %s close because of %s", session.getId(), closeReason));
        latch.countDown();
    }

    public static void main(String[] args) {
        latch = new CountDownLatch(1);

        ClientManager client = ClientManager.createClient();
        try {
            client.connectToServer(WordgameClientEndpoint.class, new URI("ws://localhost:8025/websocke
            latch.await();

        } catch (DeploymentException | URISyntaxException | InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}
```

How to build Java WebSocket Applications Using the JSR 356 API
https://blog.openshift.com/how-to-build-java-websocket-applications-using-the-jsr-356-api/

SimpleEchoClient using Jetty

```
public class SimpleEchoClient {
    public static void main(String[] args) {
        String destUri = "ws://echo.websocket.org";
        if (args.length > 0) {
            destUri = args[0];
        }
        WebSocketClient client = new WebSocketClient();
        SimpleEchoSocket socket = new SimpleEchoSocket();
        try {
            client.start();
            URI echoUri = new URI(destUri);
            ClientUpgradeRequest request = new
ClientUpgradeRequest();
            client.connect(socket, echoUri, request);
            System.out.printf("Connecting to : %s%n",
echoUri);
            socket.awaitClose(5, TimeUnit.SECONDS);
        } catch (Throwable t) {
            t.printStackTrace();
        } finally {
            try {
                client.stop();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
 //Basic Echo Client Socket
(…)
public class SimpleEchoSocket {
    private final CountDownLatch closeLatch;
     private Session session;
    public SimpleEchoSocket() {  (…)     }
    public boolean awaitClose(int duration, TimeUnit unit)
throws InterruptedException { (…)  }

    @OnWebSocketClose
    public void onClose(int statusCode, String reason) {  (-
--)     }
     @OnWebSocketConnect
    public void onConnect(Session session) {
        System.out.printf("Got connect: %s%n", session);
        (…)
    }
     @OnWebSocketMessage
    public void onMessage(String msg) {
System.out.printf("Got msg: %s%n", msg);     }
}
```

http://www.eclipse.org/jetty/documentation/9.2.2.v20140723/jetty-websocket-client-api.html

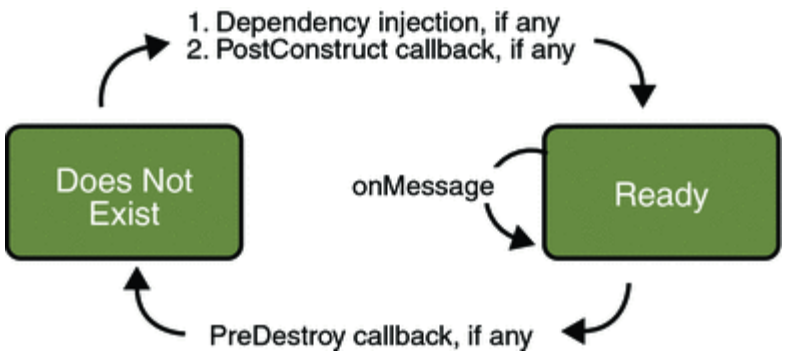
Websocket vs message EJB

```
@ServerEndpoint("/websocket")
public class WebSocketTest {

  @OnMessage
  public void onMessage(String message, Session session) {  (…)}

  @OnOpen
  public void onOpen() {(…)  }

  @OnClose
  public void onClose() {    (…) }
}
```




```
@MessageDriven(
  name = "BookMessageHandler",
  activationConfig = {
    @ActivationConfigProperty( propertyName = "destinationType",
                               propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty( propertyName = "destination",
                               propertyValue ="/queue/BookQueue")
  }
)
public class LibraryMessageBean implements MessageListener {

  @Resource
  private MessageDrivenContext mdctx;

  @EJB
  LibraryPersistentBeanRemote libraryBean;

  public LibraryMessageBean(){
  }

  public void onMessage(Message message) {
  }

  @PostConstruct
   public void initialize() { // Initialization logic }

  @PreDestroy
   public void clean() { // Initialization logic }
}
```

http://docs.oracle.com/javaee/5/tutorial/doc/bnbmt.html
https://docs.oracle.com/html/E13981_01/mdb30cfg006.htm
http://www.informit.com/articles/article.aspx?p=2210834&seqNum=7

@Singleton on server JavaEE side

Developing singleton session beans

Create a bean implementation class for a singleton session bean, introduced by the Enterprise JavaBeans (EJB) 3.1 specifi
The EJB container initializes only one instance of a singleton session bean, and that instance is shared by all clients. Becau
single instance is shared by all clients, singleton session beans have special life cycle and concurrency semantics.

Before you begin

Make sure that you understand the inheritance rules for each annotation you implement. For example, the @ConcurrencyManagement annotation is coded on the singleton session bean class only. You cannot use the @ConcurrencyManagement annotation in the class that it extends, or any class higher in the class inheritance tree.

About this task

Singleton session beans can have business local, business remote, and web service client views; they cannot have EJB 2.1
remote client views. This singleton session bean support replaces the proprietary startup bean functionality, which has bee

https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.0.0/com.ibm.websphere.nd.doc/info/ae/ae/tejb_ssb.html

The following example shows a basic singleton session bean:

```
public interface Configuration {
```

```
public interface Configuration {
        Object get(String name);
        void set(String name, Object value);
}

@Singleton
public class ConfigurationBean implements Configuration {
        private Map<String, Object> settings = new HashMap<String, Object>();

        public Object get(String name) {
                return settings.get(name);
        }

        public void set(String name, Object value) {
                settings.put(name,value);
        }
}
```

remote client views. This singleton session bean support replaces the proprietary startup bean functionality, which has bee

https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.0.0/com.ibm.websphere.nd.doc/info/ae/ae/tejb_ssb.html

The following example shows a basic singleton session bean:

Dev

Create
EJB) 3.1 specifi
The EJB
ll clients. Becau
single i
antics.

Before

Make su
@Concu
he
@Concu
ce tree.

About t

Singleto
ot have EJB 2.1
remote client views. This singleton session bean support replaces the proprietary startup bean functionality, which has bee

```
@Singleton
public class DatabaseBean {
        @PostConstruct
        public void initialize() {
                // Create database tables.
        }
}

@Singleton
@DependsOn({"DatabaseBean"})
public class ConfigurationBean implements Configuration {
        @PostConstruct
        public void initialize() {
                // Initialize settings from a database table.
        }

        // ...
}
```

https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.0.0/com.ibm.websphere.nd.doc/info/ae/ae/tejb_ssb.html

The following example shows a basic singleton session bean:

```
public interface Configuration {
```

```
@Singleton
@Startup
public class ConfigurationBean implements Configuration {
        @PostConstruct
        public void initialize() {
                // 1. Create the database table if it does not exist.
                // 2. Initialize settings from the database table.
                // 3. Load a cache.
                // 4. Initiate asynchronous work (for example, work to a messaging queue or to
                //    calls to asynchronous session bean methods.
        }

        // ...
}
```

specifi
Becau

Singleton session beans can have business local, business remote, and web service client views; they cannot have EJB 2.1 remote client views. This singleton session bean support replaces the proprietary startup bean functionality, which has bee

https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.0.0/com.ibm.websphere.nd.doc/info/ae/ae/tejb_ssb.html

The following example shows a basic singleton session bean:

```
public interface Configuration {
```

```
@Singleton
@ConcurrencyManagement(BEAN)
public class ConfigurationBean implements Configuration {
        private Map<String, Object> settings = new HashMap<String, Object>();

        synchronized public Object get(String name) {
                return settings.get(name);
        }

        synchronized public void set(String name, Object value) {
                settings.put(name, value);
        }
}
```

remote client views. This singleton session bean support replaces the proprietary startup bean functionality, which has bee

https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.0.0/com.ibm.websphere.nd.doc/info/ae/ae/tejb_ssb.html

The following example shows a basic singleton session bean:

Table of contents

```
@Singleton
public class ConfigurationBean implements Configuration {
        private Map<String, Object> settings = new HashMap<String, Object>();

        @Lock(READ)
        public Object get(String name) {
                return settings.get(name);
        }

        public void set(String name, Object value) {
                settings.put(name, value);
        }
}
```

Singleton session beans can have business local, business remote, and web service client views; they cannot have EJB 2.1
remote client views. This singleton session bean support replaces the proprietary startup bean functionality, which has bee

https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.0.0/com.ibm.websphere.nd.doc/info/ae/ae/tejb_ssb.html

The following example shows a basic singleton session bean:

public interface Configuration {

Contact U

@Singleton as façade

DZone / Integration Zone

Thanks for visiting DZone today, Jose Maria

REFCARDZ GUIDES ZONES JOBS | AGILE AI BIG DATA CLOUD DATABASE DEVOPS INTEGRATION IOT JAVA MOBILE MORE

How to Connect to MongoDB from a Java EE Stateless Application

by Adrian Matei MVB · Oct. 20, 14 · Integration Zone

Like (1) Comment (1) Save Tweet 10.83k Views

In this post I will present how to connect to MongoDB from a stateless Java EE application, to take advantage of the built-in pool of connections to the database offered by the MongoDB Java Driver. This might be the case if you develop a REST API, that executes operations against a MongoDB.

Get the Java MongoDb Driver

https://dzone.com/articles/how-connect-mongodb-java-ee

To connect from Java to MongoDB, you can use the Java MongoDB Driver. If you are building your application with Maven, you can add the dependency to the pom.xml

```
package org.codingpedia.demo.mongoconnection;

import java.net.UnknownHostException;

import javax.annotation.PostConstruct;
import javax.ejb.ConcurrencyManagement;
import javax.ejb.ConcurrencyManagementType;
import javax.ejb.Lock;
import javax.ejb.LockType;
import javax.ejb.Singleton;

import com.mongodb.MongoClient;

@Singleton
@ConcurrencyManagement(ConcurrencyManagementType.CONTAINER)
public class MongoClientProvider {

private MongoClient mongoClient = null;

@Lock(LockType.READ)
public MongoClient getMongoClient(){
return mongoClient;
}

@PostConstruct
public void init() {
String mongoIpAddress = "x.x.x.x";
Integer mongoPort = 11000;
try {
mongoClient = new MongoClient(mongoIpAddress, mongoPort);
} catch (UnknownHostException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

}
```

building your application with Maven, you can add the dependency to the pom.xml

```java
package org.codingpedia.demo.mongoconnection;

import java.net.UnknownHostException;

import javax.annotation.PostConstr
import javax.ejb.ConcurrencyManage
import javax.ejb.ConcurrencyManage
import javax.ejb.Lock;
import javax.ejb.LockType;
import javax.ejb.Singleton;

import com.mongodb.MongoClient;

@Singleton
@ConcurrencyManagement(Concurrency
public class MongoClientProvider {

private MongoClient mongoClient =

@Lock(LockType.READ)
public MongoClient getMongoClient(
return mongoClient;
}

@PostConstruct
public void init() {
String mongoIpAddress = "x.x.x.x";
Integer mongoPort = 11000;
try {
mongoClient = new MongoClient(mong
} catch (UnknownHostException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

}
```

```java
package org.codingpedia.demo.mongoconnection;

import java.util.Set;

import javax.ejb.EJB;
import javax.ejb.Stateless;

import com.mongodb.BasicDBObject;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.DBObject;
import com.mongodb.MongoClient;
import com.mongodb.util.JSON;

@Stateless
public class TestMongoClientProvider {

@EJB
MongoClientProvider mongoClientProvider;

public Set<String> getCollectionNames(){

MongoClient mongoClient = mongoClientProvider.getMongoClient();

DB db = mongoClient.getDB("myMongoDB");
Set<String> colls = db.getCollectionNames();

for (String s : colls) {
    System.out.println(s);
}

return colls;
}

}
```

REFCARD

Mode
virtua

In th
appl
data
REST

To c
building your application with Maven, yo

Websocket in JavaEE: singleton

```
@Singleton
@ServerEndpoint("/singleton/")
public class SingletonlEndpoint {

    @OnOpen
    public void onOpen(Session s) throws IOException {
        s.getBasicRemote().sendText(String.valueOf(hashCode()));
    }

    @PreDestroy
    public void onDestroy() {
        System.out.println("Singleton bean " + hashCode() + " will be destroyed");
    }

    @OnClose
    public void onClose(Session session, CloseReason closeReason) {
        System.out.println("Closed " + session.getId() + " due to " + closeReason.getCloseCode());
    }
}
```

java websocket api handbook
https://abhirockzz.gitbooks.io/java-websocket-api-handbook/content/

Websocket in JavaEE: stateful

```
@Singleton
@ServerEndpoint("/singleto
public class SingletonlEnd

    @OnOpen
    public void onOpen(Ses
        s.getBasicRemote()
    }

    @PreDestroy
    public void onDestroy(
        System.out.println
    }

    @OnClose
    public void onClose(Se
        System.out.println
    }
}
```

```
@Stateful
@ServerEndpoint("/chat/{user}")
public class StatefulChat {
    private transient Session s;
    private String userID;
    private List<History> history;

    @OnOpen
    public void onOpen(@PathParam("user") String user, Session s) throws IOException {
        this.userID= user;
        this.s = s;
    ....
    }

    @OnMessage
    public void chat(String msg) {
        history.add(msg);
       //route message to intended recipient(s)
    }
```

java websocket api handbook
https://abhirockzz.gitbooks.io/java-websocket-api-handbook/content/

Thinking in Java EE (at least trying to!)

Write Once, Read Forever

Me Java EE 7 Books, articles etc.. Java Magazine Hub

← New release: Java WebSocket API handbook

Quick tip: managing Stateful EJBs in WebSocket endpoints →

WebSocket endpoint as Singleton EJB

Posted on February 14, 2017

By default ...

... a WebSocket implementation creates a **new** (server) endpoint instance per client. In case you need a **single** instance, you can implement this using a custom `ServerEndpointConfig.Configurator` (by overriding the `getEndpointInstance` method)

The catch: you might have to sacrifice some of the (Java EE) platform related services like dependency injection and interceptors

Java EE eBooks on Leanpub

JAX-RS, EJB & WebSocket bundle

DZone Java Caching Refcard

WebSocket endpoint as Singleton EJB
https://abhirockzz.wordpress.com/2017/02/14/websocket-endpoint-as-singleton-ejb/

Thinking in Java EE (at least trying to!)

```
@Singleton
@ServerEndpoint("/singleton/")
public class SingletonEndpoint {
    @OnOpen
    public void onopen(Session s) throws IOException {
        s.getBasicRemote().sendText(String.valueOf(hashCode()));
    }
    @PreDestroy
    public void onDestroy() {
        System.out.println("Singleton bean " + hashCode() + " will
be destroyed");
    }
    @OnClose
    public void onClose(Session session, CloseReason closeReason) {
        System.out.println("Closed " + session.getId() + " due to "
+ closeReason.getCloseCode());
    }
}
```

The catch: you might have to sacrifice some of the (Java EE) platform related services like dependency injection and interceptors

WebSocket endpoint as Singleton EJB
https://abhirockzz.wordpress.com/2017/02/14/websocket-endpoint-as-singleton-ejb/

Thinking in Java EE (at least trying to!)

```
@Singleton
@ServerEndpoint("/singleton/")
public class SingletonEndpoint {
    @OnOpen
    public void onopen(Session s) throws IOException {
        s.getBasicRemote().sendText(String.valueOf(hashCode()));
    }
    @PreDestroy
    public void onDestroy() {
        System.out.println("Singleton bean " + hashCode() + " will
be destroyed");
    }
    @OnClose
    public void onClose(Session session, CloseReason closeReason) {
        System.out.println("Closed " + session.getId() + " due to "
+ closeReason.getCloseCode());
    }
}
```

The catch: you might have to sacrifice some of the (Java EE) platform related services like dependency injection and interceptors

WebSocket endpoint as Singleton EJB
https://abhirockzz.wordpress.com/2017/02/14/websocket-endpoint-as-singleton-ejb/

Thinking in Java EE (at least trying to!)

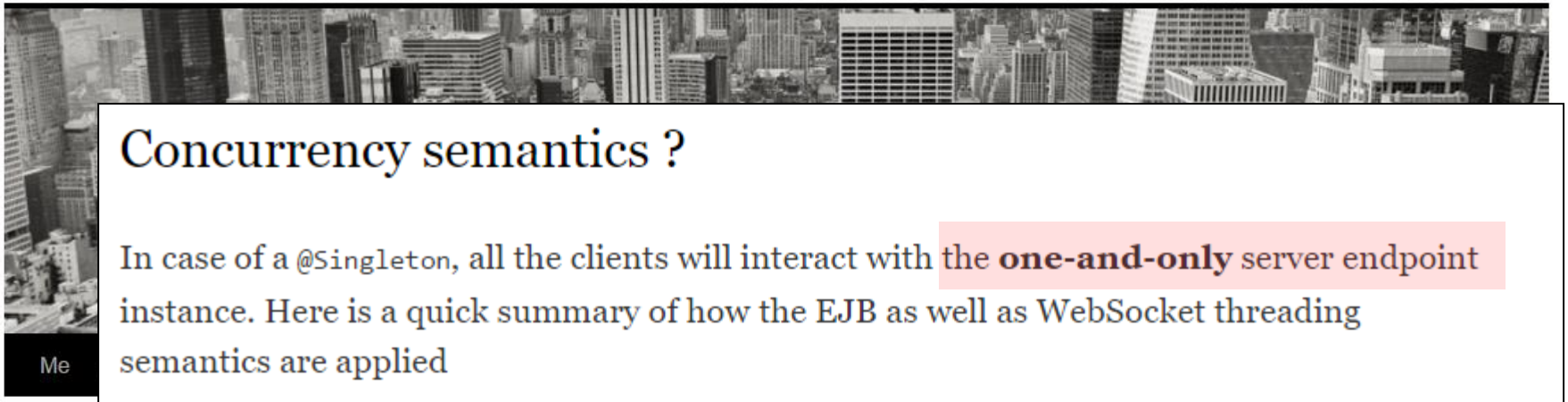
```
@Singleton
@ServerEndpoint("/singleton/")
public class SingletonEndpoint {
    @OnOpen
    public void onopen(Session s) throws IOException {
        s.getBasicRemote().sendText(String.valueOf(hashCode()));
    }
    @PreDestroy
    public void onDestroy() {
        System.out.println("Singleton bean " + hashCode() + " will
be destroyed");
    }
    @OnClose
    public void onClose(Session session, CloseReason closeReason) {
        System.out.println("Closed " + session.getId() + " due to "
+ closeReason.getCloseCode());
    }
}
```

The catch: you might have to sacrifice some of the (Java EE) platform related services like dependency injection and interceptors

WebSocket endpoint as Singleton EJB
https://abhirockzz.wordpress.com/2017/02/14/websocket-endpoint-as-singleton-ejb/

Thinking in Java EE (at least trying to!)

Write Once, Read Forever



Me

← New

Web

Posted

By

... a W

case y

custom `ServerEndpointConfig.Configurator` (by overriding the `getEndpointInstance` method)

The catch: you might have to sacrifice some of the (Java EE) platform related services like dependency injection and interceptors

Concurrency semantics ?

In case of a `@Singleton`, all the clients will interact with the **one-and-only** server endpoint instance. Here is a quick summary of how the EJB as well as WebSocket threading semantics are applied

- The `Singleton` bean default approach WRITE lock ensures single threaded access across all connected clients
- If thread-safety is not a concern (e.g. in case where you do not deal with client specific data/state in your logic) and in case the single-threaded access model proves to be a bottleneck, override the default behavior by switching to a READ lock which allows concurrent threads to access the methods (unless of course a WRITE lock is not already in effect)

WebSocket endpoint as Singleton EJB
https://abhirockzz.wordpress.com/2017/02/14/websocket-endpoint-as-singleton-ejb/

ADAM BIEN'S WEBLOG

« Kickstarting Real... | Main | @Singleton - The... »

@SINGLETON - THE PERFECT CACHE FACADE

@Singleton in the default configuration is the perfect bottleneck. Only one thread a time can access a @Singleton instance.
The annotation `@ConcurrencyManagement(ConcurrencyManagementType.BEAN)` deactivates this limitation and makes a Singleton instance accessible from multiple threads:

```
import javax.ejb.Singleton;
import javax.annotation.PostConstruct;
import javax.ejb.ConcurrencyManagement;
import java.util.concurrent.ConcurrentHashMap;
import javax.ejb.ConcurrencyManagementType;

@Singleton
@ConcurrencyManagement(ConcurrencyManagementType.BEAN)
public class Hits {

    private ConcurrentHashMap hits = null;

        @PostConstruct
    public void initialize() {
        this.hits = new ConcurrentHashMap();
        }
//cache accessors omitted
}
```

http://www.adam-bien.com/roller/abien/entry/singleton_the_perfect_cache_facade

A `java.util.concurrent.ConcurrentHashMap` can be accessed by multiple threads consistently without any locking or synchronization required.

Number of posts: 1680
Number of comments: 596
Yesterday's hits: 23976
Today's hits: 3918
Post reads / hour: 1153
Top posts:

1. Adam Bien's Weblog
   322285
2. Java 8 FlatMap Example : Adam Bien's Weblog
   47509
3. Search Results for " : Adam Bie Weblog
   44799
4. Java 8: From a for-loop to forEa statement : Adam Bien's Weblo
   19703

Trending (last hour):

1. Why Not One Application Per S / Domain? : Adam Bien's Weblo
   3
2. Java EE 8 Security API 1.0-m0 Available For Testing : Adam Bi Weblog
   2
3. The Return Of JSPs In HTML 5 Adam Bien's Weblog
   2
4. Star 7 feat. James Gosling: The Origins of Java ...and iPhone? : Adam Bien'[...]
   2
5. Java EE 8: Serializing POJOs w JSON-B : Adam Bien's Weblog
   2

about.adam-bien.com
blog archives

REALWORLDPATTERNS.COM

ADAM BIEN'S WEBLOG

```
import javax.ejb.Singleton;
import javax.annotation.PostConstruct;
import javax.ejb.ConcurrencyManagement;
import java.util.concurrent.ConcurrentHashMap;
import javax.ejb.ConcurrencyManagementType;

@Singleton
@ConcurrencyManagement(ConcurrencyManagementType.BEAN)
public class Hits {

    private ConcurrentHashMap hits = null;

        @PostConstruct
    public void initialize() {
        this.hits = new ConcurrentHashMap();
        }
//cache accessors omitted
}
```

http://www.adam-bien.com/roller/abien/entry/singleton_the_perfect_cache_facade

A `java.util.concurrent.ConcurrentHashMap` can be accessed by multiple threads consistently without any locking or synchronization required.

Singleton Pattern: The Good, the Bad, and the Ugly

The singleton pattern is one of the most well-known design patterns... wait, or is it an anti-pattern? See how to implement the singleton in Java EE, and look at some of the pros and cons.

by Alex Theedom MVB · May. 25, 16 · Java Zone

OK, let's have a look at our first pattern.

The singleton pattern is one of the most well-known design patterns and is used extensively by frameworks such as Spring, pool managers, and loggers.

If you read the GoF book, it's about having a single instance of an object in the JVM which is only ever instantiated once within the life-cycle of the application. Once created it is not normally destroyed until the application terminates. Why do you need this? The GoF motivation was that heavy-weight objects you would not want to create because they are expensive to have around, are created by the singleton.

https://dzone.com/articles/singleton-pattern-1 implementing the singleton pattern is quite non-trivial. You have to start with some pretty unnatural constructs like private constructors, double locking and the like, and in the end, you still didn't get thread safety. You need to think about

Singleton Pattern: The Good, the Bad, a

The Good

Java's EE manner of implementing the Singleton pattern has reduced substantially the lines of boilerplate code required to achieve a tread-safe Singleton. Not only that but it adds features to our singleton that it allows to be instantiated on application start-up or on first invocation. We can make its existence depend on the successful construction of another bean.

By convention, the singleton has container-managed concurrency but we can take back control and manage it ourselves we can even specify an access timeout value.

The Bad

Overuse of singletons can cause problems with your application, as can the overuse of anything. Lazy loading will cause delays on first use while eager loading may cause memory problems. An instance might be created on start-up but never used or garbage collected using up memory resources.

The Ugly

The singleton pattern is often considered an anti-pattern and should be used for niche use-case scenarios. It would be smarter to use a stateless session bean.

https://dzone.com/articles/singleton-pattern-1

implementing the singleton pattern is quite non-trivial. You have to start with some pretty unnatural constructs like private constructors, double locking and the like, and in the end, you still didn't get thread safety. You need to think about

Baeldung

Start Here | Courses | Guides | About

A Guide to the Java API for WebSocket

Last modified: March 16, 2017

by baeldung

Java

If you're new here, you may want to check out the "API Discoverability with Spring and Spring HATEOAS" live Webinar. Thanks for visiting!

The Master Class of "*Learn Spring Security*" is live:

>> CHECK OUT THE COURSE

1. Overview

WebSocket provides an alternative to the limitation of efficient communication between the server and the web browser by providing bi-directional, full-duplex, real-time client/server communications. The server can send data to the client at any time. Because it runs over TCP, it also provides a low-latency low-level communication and reduces the overhead of each message.

In this tutorial, we'll take a look at the Java API for WebSocket by creating a chat-like application.

2. JSR 356

JSR 356 – Java API for WebSocket specifies an API [illegible]

A Guide to the Java API for WebSocket (very fresh)

http://www.baeldung.com/java-websockets

CodingpediaOrg Bookmarks About Tags Categories Feed

Fork me on GitHub

Using The Java Api For Websocket To Create A Chat Server

July 18, 2016

Reading time ~5 minutes

Using The Java Api For Websocket To Create A Chat Server

In a previous post, I've used Server Sent Events to create a monitoring dashboard. SSE are a one way messaging format form server to clients in contrast to Web Sockets where communication is bidirectional. In this post, I'll use Web sockets to create a tiny chat server using Tyrus, the reference implementation of the Java API for WebSocket (JSR 356). A great introduction to this API can be found on Oracle Network here.

In order to keep the tutorial simple, the server and clients will be command line apps, no GUIs here, it is a serious blog :smile: So let's get started!

A quick introduction to the Java API for WebSocket

If you don't have time to check all details of this API, here are the most important points to know. JSR 356 provides:

- both a programmatic and a declarative way to define client and server endpoints. In this post, I'll use the declarative way, through annotations.
- lifecyle events to which we can listen: open/close a session, send/receive a message, etc
- message encoder/decoder to marshal/unmarshall (binary or character) messages between clients and servers. Encoders/Decorders are important since they allow us to use Java objects as messages instead of dealing with raw data that transit over the wire. We will see an example in a couple of minutes

That's all we need to know for now in order to implement the chat application.

Using The Java Api For Websocket To Create A Chat Server
http://www.codingpedia.org/benas/using-the-java-api-for-webSocket-to-create-a-chat-server.html

COPY

```
public class Message {
```

CodingpediaOrg Bookmarks About Tags Categories Feed

Fork me on GitHub

A kind of singleton…

```
"/chat", encoders = MessageEncoder.class, decoders = MessageDecoder.class)
public class ServerEndpoint {

    static Set<Session> peers = Collections.synchronizedSet(new HashSet<Session>());

    @OnOpen
    public void onOpen(Session session) {
        System.out.println(format("%s joined the chat room.", session.getId()));
        peers.add(session);
    }
```

Usir

In a prev

contrast to Web Sockets where comm Web sockets to create a tiny chat server using Tyrus, the reference implementation of the Java API for W this API can be found on Oracle Network here.

A kind of broker…

In order to keep the tutorial simple, the server and clients will be command line apps, no GUIs here, it is a serious blog :smile: So let's get started!

A qui

If you do

- bo
- life
- me im in

That's al

```
@OnMessage
public void onMessage(Message message, Session session) throws IOException, EncodeException {
    //broadcast the message
    for (Session peer : peers) {
        if (!session.getId().equals(peer.getId())) { // do not resend the message to its sender
            peer.getBasicRemote().sendObject(message);
        }
    }
}
```

The data model

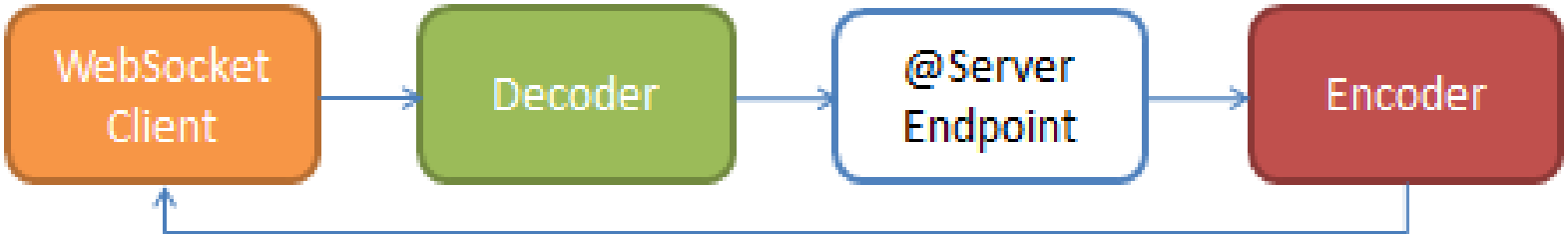
Since we will be creating a chat server, let's first model messages with a POJO:

```
public class Message {
```

COPY

Using The Java Api For Websocket To Create A Chat Server
http://www.codingpedia.org/benas/using-the-java-api-for-webSocket-to-create-a-chat-server.html

Decoders and encoders



Decoders and encoders

```
@ServerEndpoint(value = "/hello",
        decoders = {
            MessageDecoder.class,},
        encoders = {
            MessageEncoder.class
        })
public class HelloWorldEndpoint {

    @OnMessage
    public Person hello(Person person, Session session) {
        if (person.getName().equals("john")) {
            person.setName("Mr. John");
        }
        try {
            session.getBasicRemote().sendObject(person);
            System.out.println("sent ");
        } catch (Exception ex) {
            Logger.getLogger(HelloWorldEndpoint.class.getName()).log(Lev
        }
        return person;

    }

    @OnOpen
    public void myOnOpen(Session session) {
    }

}
```

Websockets: using Encoders and Decoders
http://www.mastertheboss.com/javaee/websockets/websockets-using-encoders-and-decoders

Decoders and encoders

```java
@ServerEndpoint(value = "/hello",
        decoders = {
            MessageDecoder.class,},
        encoders = {
            MessageEncoder.class
        })
public class HelloWorldEndpoint {

    @OnMessage
    public Person hello(Person per
        if (person.getName().equal
            person.setName("Mr. Jo
        }
        try {
            session.getBasicRemote
            System.out.println("se
        } catch (Exception ex) {
            Logger.getLogger(Hello
        }
        return person;

    }

    @OnOpen
    public void myOnOpen(Session s
    }

}
```

```java
public class MessageDecoder implements Decoder.Text<Person> {

    @Override
    public Person decode(String s) {
        System.out.println("Incoming XML " + s);
        Person person = null;
        JAXBCo
        try {
            ja

            Un

            St
            pe
        } catc
            ex
        }
        return
    }

    @Override
    public boo

        return
    }

    @Override
    public voi
        // do
    }
```

```java
public class MessageEncoder implements Encoder.Text<Person> {

    @Override
    public String encode(Person object) throws EncodeException {

        JAXBContext jaxbContext = null;
        StringWriter st = null;
        try {
            jaxbContext = JAXBContext.newInstance(Person.class);

            Marshaller marshaller = jaxbContext.createMarshaller();
            st = new StringWriter();
            marshaller.marshal(object, st);
            System.out.println("OutGoing XML " + st.toString());

        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return st.toString();
    }

    @Override
    public void init(EndpointConfig endpointConfig) {
        // do nothing.
    }
```

Websockets: using Encoders and Decoders
http://www.mastertheboss.com/javaee/websockets/websockets-using-encoders-and-decoders

Using the WebSocket API in a Web Application

PRINTABLE VERSION

This tutorial demonstrates how to create a simple web application that enables collaboration between client browsers that are connected to a single server application. When a user draws a graphic element on a canvas in the client browser the element appears on the canvas of all connected clients. How does it work? When the browser loads the web page a client-side script sends a WebSocket handshake request to the application server. The application can accept JSON and binary messages from the clients connected in the session and broadcast the messages to all the connected clients.

In this tutorial you will create a web application that uses the Java API for WebSocket (JSR 356) to enable bi-directional communication between browser clients and the application server. The Java API for WebSocket provides support for creating WebSocket Java components, initiating and intercepting WebSocket events and creating and consuming WebSocket text and binary messages. The tutorial will also demonstrate how you can use the Java API for JSON Processing (JSR 353) to produce and consume JSON. The Java API for WebSocket and the Java API for JSON Processing are part of the Java EE 7 platform (JSR 342).

The application contains a WebSocket endpoint and decoder and encoder interfaces, a web page and some JavaScript files that are run in the client browser when the page is loaded or when invoked from a form in the web page. You will deploy the application to GlassFish Server Open Source Edition 4, the reference implementation of Java EE 7 technology.

Note. This tutorial is based on the Collaborative Whiteboard using WebSocket in GlassFish 4 - Text/JSON and Binary/ArrayBuffer Data Transfer (TOTD #189) blog post and other blog entries which can be found on Arun Gupta's blog. Be sure to visit the blog and see many other excellent entries on working with the WebSocket API and GlassFish 4.

You can also watch the Video of Using the WebSocket API in a Web Application.

Tutorial Exercises

- Creating the Web Application Project
- Creating the WebSocket Endpoint
  - Create the Endpoint
  - Initiate the WebSocket Session
  - Test the Endpoint
- Creating the Whiteboard
  - Add the Canvas
  - Create the POJO
  - Create a Coordinates Class
  - Generate the JSON String
  - Implement the Encoder and Decoder Interfaces
  - Run the Application
- Sending Binary Data to the Endpoint

REQUIRES NetBeans 8.0/7.4/7.3

To follow this tutorial, you need the following software and resources.

Software or Resource	Version Required
NetBeans IDE	7.3.1, 7.4, 8.0, Java EE version
Java Development Kit (JDK)	version 7 or 8
GlassFish Server Open Source Edition	4

Note. GlassFish 4 is bundled with the Java EE download bundle of NetBeans IDE.

Prerequisites

Download NetBeans IDE

Training
- Java Programming Language

Support
- Oracle Development Tools Support Offering for NetBeans IDE

Documentation
- General Java Development
- External Tools and Services
- Java GUI Applications
- Java EE & Java Web Development
- Web Services Applications
- NetBeans Platform (RCP) and Module Development
- PHP and HTML5 Applications
- C/C++ Applications
- Mobile Applications
- Sample Applications
- Demos and Screencasts

More
- FAQs
- Contribute Documentation!
- Docs for Earlier Releases

Using the WebSocket API in a Web Application
https://netbeans.org/kb/docs/javaee/maven-websocketapi.html

ADAM BIEN'S WEBLOG

« Java 8 partitioningB... | Main | Importance of Java... »

JAVA 8 BASE64 ENCODING / DECODING

Java 8 comes with the Base64 class which supports Base64 encoding / decoding out-of-the-box:

```
import java.util.Base64;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.MatcherAssert.assertThat;
import org.junit.Test;

public class Base64Test {

    @Test
    public void encodeAndDecode() {
        final String rawString = "duke";
        Base64.Encoder encoder = Base64.getEncoder();
        byte[] encodedContent = encoder.encode(rawString.getBytes());

        Base64.Decoder decoder = Base64.getDecoder();
        byte[] decodedContent = decoder.decode(encodedContent);

        String decodedString = new String(decodedCor
        assertThat(decodedString, is(rawString));
    }
```

Apache Johnzon
https://johnzon.apache.org/

JAVA 8 BASE64 ENCODING / DECODING
http://www.adam-bien.com/roller/abien/entry/java_8_base64_encoding_decoding

ADAM BIEN'S WEB

« Java 8 partitioningB... | Main | In

JAVA 8 BASE64 ENCODING / DECO

Java 8 comes with the Base64 class w
encoding / decoding out-of-the-box:

```
import java.util.Base64;
import static org.hamcrest.CoreMatcher
import static org.hamcrest.MatcherAsse
import org.junit.Test;

public class Base64Test {

    @Test
    public void encodeAndDecode() {
        final String rawString = "duke
        Base64.Encoder encoder = Base
        byte[] encodedContent = encode

        Base64.Decoder decoder = Base
        byte[] decodedContent = decode

        String decodedString = new Str
        assertThat(decodedString, is(r
```

JAVA 8 BASE64 ENCODING / DECODING
http://www.adam-bien.com/roller/abien/entry/java_8_base64_encoding_decoding

Apache Johnzon

Last Published: 2017-01-16 | Version: 1.0.1-SNAPSHOT

USER GUIDE
- Home
- Download
- Javadoc
- Source Code
- Changelog
- Mailing Lists

OLD RELEASES
- Johnzon 0.7-incubating
- Johnzon 0.2-incubating
- Johnzon 0.1-incubating

PROJECT DOCUMENTATION
- Project Information
  - CI Management
  - Dependencies
  - Dependency Convergence
  - Dependency Information

Apache Johnzon

Apache Johnzon is a project providing an implementation of JsonProcessing (aka jsr-353) and a set of useful extension for this specification like an Object mapper, some JAX-RS providers and a WebSocket module provides a basic integration with Java WebSocket API (JSR 356).

Status

Apache Johnzon is a Top Level Project at the Apache Software Foundation (ASF). It fully implements the JSON-P_1.0 specification (JSR-353). Johnzon also targets the upcoming JSON-P_1.1 and JSON-B_1.0 specifications.

The current version is Apache Johnzon 1.0.0.

Get started

Johnzon comes with four main modules.

Core (stable)

```
1. <dependency>
2.   <groupId>org.apache.johnzon</groupId>
3.   <artifactId>johnzon-core</artifactId>
4.   <version>${johnzon.version}</version>
5. </dependency>
```

This is the implementation of the specification. You'll surely want to add the API as dependency too:

```
1. <dependency>
```

Apache Johnzon
https://johnzon.apache.org/

ADAM BIEN'S WEB

« Java 8 partitioningB... | Main | In

JAVA 8 BASE64 ENCODING / DECO

Java 8 comes with the Base64 class w
encoding / decoding out-of-the-box:

```
import java.util.Base64;
import static org.hamcrest.CoreMatcher
import static org.hamcrest.MatcherAsse
import org.junit.Test;

public class Base64Test {

    @Test
    public void encodeAndDecode() {
        final String rawString = "duke
        Base64.Encoder encoder = Base
        byte[] encodedContent = encode

        Base64.Decoder decoder = Base
        byte[] decodedContent = decode

        String decodedString = new Str
        assertThat(decodedString, is(
```

Apache Johnzon

Last Published: 2017-01-16 | Version: 1.0.1-SNAPSHOT

USER GUIDE
- Home
- Download
- Javadoc
- Source Code
- Changelog
- Mailing Lists

OLD RELEASES
- Johnzon 0.7-incubating
- Johnzon 0.2-incubating
- Johnzon 0.1-incubating

PROJECT DOCUMENTATION
- Project Information
  - CI Management
  - Dependencies
  - Dependency Convergence
  - Dependency Information

Apac

Apache Johnz
useful extensio
module provide

Status

Apache Johnz
the JSON-P_
B_1.0 specific

The current ve

Get st

Johnzon com

Core (st

```
1. <depend
2.   <grou
3.   <arti
4.   <vers
5. </depen
```

This is the imp

```
1. <dependency>
```

- The ones based on JSON-P (JsonObject, JsonArray, JsonStructure)
- The ones based on Johnzon Mapper

JSON-P integration

Encoders:

- `org.apache.johnzon.websocket.jsr.JsrObjectEncoder`
- `org.apache.johnzon.websocket.jsr.JsrArrayEncoder`
- `org.apache.johnzon.websocket.jsr.JsrStructureEncoder`

Decoders:

- `org.apache.johnzon.websocket.jsr.JsrObjectDecoder`
- `org.apache.johnzon.websocket.jsr.JsrArrayDecoder`
- `org.apache.johnzon.websocket.jsr.JsrStructureDecoder`

Mapper integration

Encoder:

- `org.apache.johnzon.websocket.mapper.JohnzonTextEncoder`

Decoder:

- `org.apache.johnzon.websocket.mapper.JohnzonTextDecoder`

Apache Johnzon
https://johnzon.apache.org/

JAVA 8 BASE64 ENCODING / DECODING
http://www.adam-bien.com/roller/abien/entry/java_8_base64_encoding_decoding

The END