WebSockets for Java EE

Session AAD-1602

Erin Schnabel, Liberty Profile Development Lead, IBM
Bill Wigger, WebSphere Development & JSR 356 expert group, IBM

Impact2014

Be First.

April 27 – May 1 | The Venetian – Las Vegas, NV

#ibmimpact

Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

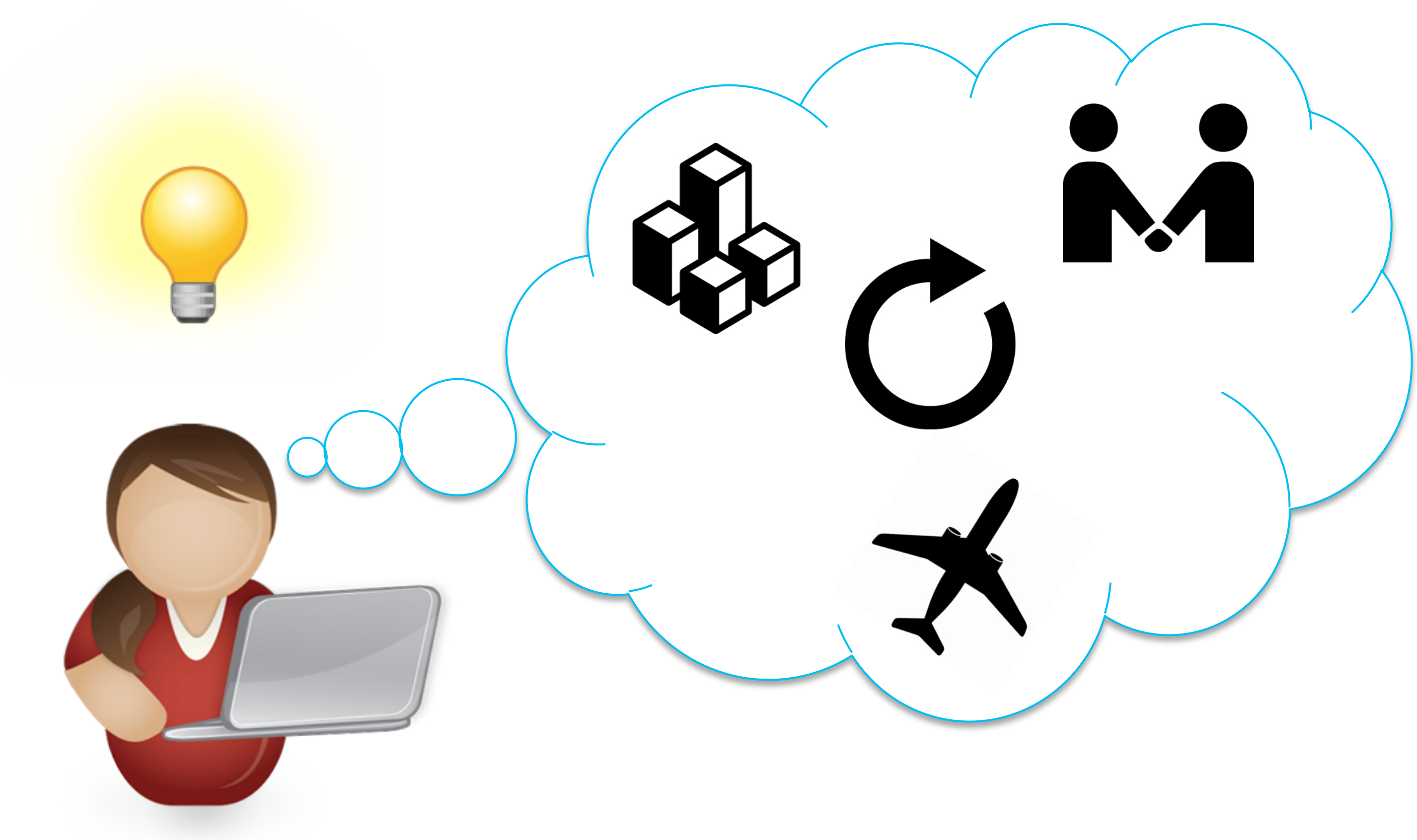
The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Using WebSockets in a Java EE Environment to Improve Web Application Development

- Programming model for the Java EE environment
- WebSocket API
  - Endpoint configuration
  - Session open/close
  - Message read/write
  - Error handling
  - Annotations
- WebSocket network protocol
  - Data format
- Network architecture
  - Proxies / load balancers / routers…

We're going to start with Judy…



Judy

… and her killer app

The app shares information in real time…

- Real time notifications (server -> client)
  - Flight status
  - Alternate flights
- Location updates (client -> server)
  - GPS data
- Social / conversation (broadcast)
  - Are you there yet?

… so how are they going to build it?

<tripDownMemoryLane>
Before WebSockets...

Impact2014

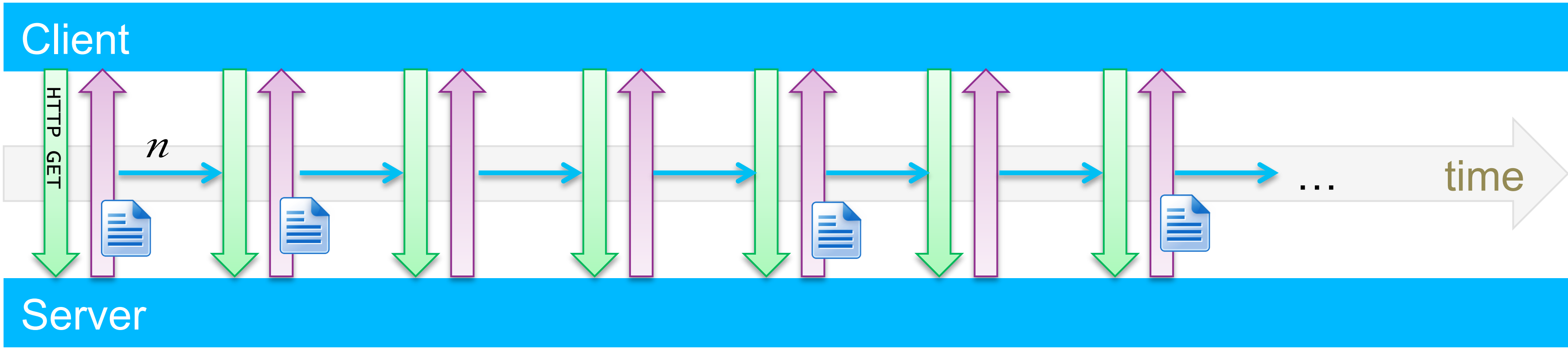
Be First.

#ibmimpact

Options for two-way communication

- Polling
- Long polling
- Streaming / forever response
- Multiple connections

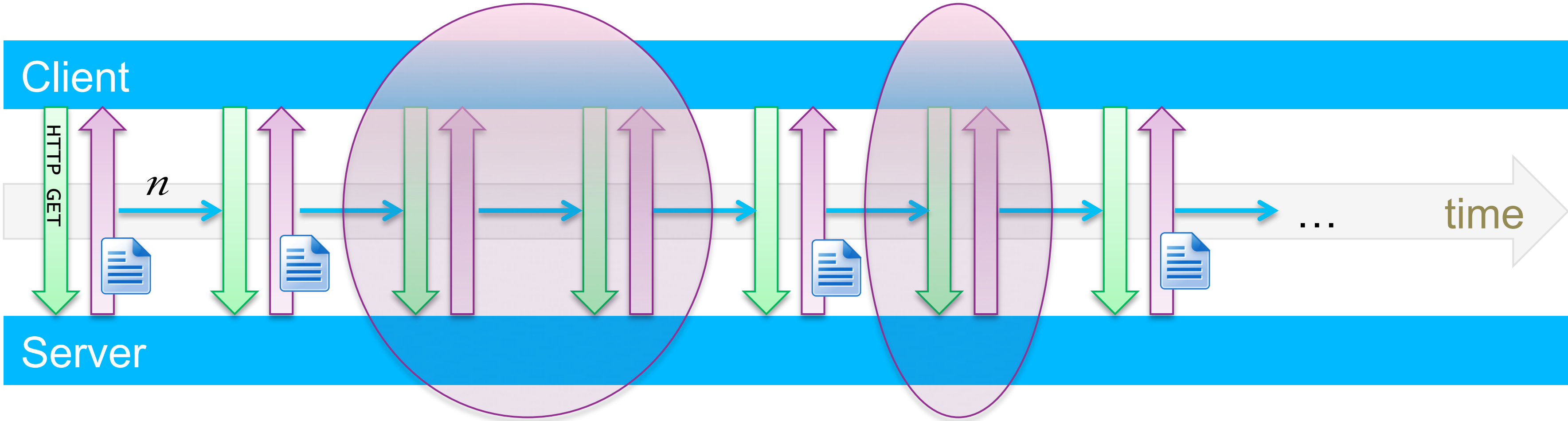
(1): Polling



- Pure JavaScript: client polls the server every n …
- Server always immediately responds (with or without data)
- Might work for periodic data where the period is known/constant

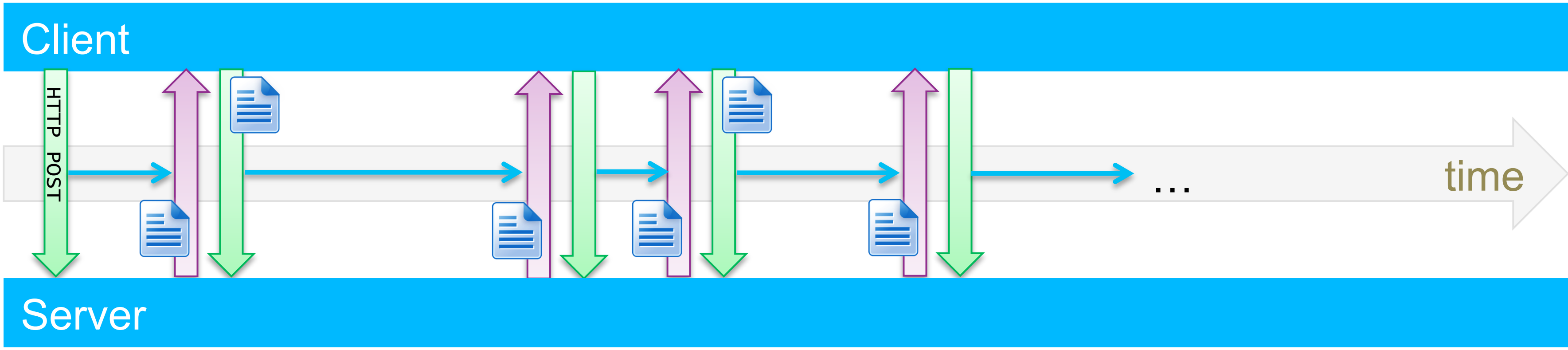
BUT…

(1): Polling



- Pure JavaScript: client polls the server every n …
- Server always immediately responds (with or without data)
- Might work for periodic data where the period is known/constant
- *Obvious waste (CPU and bandwidth) when there is no data*

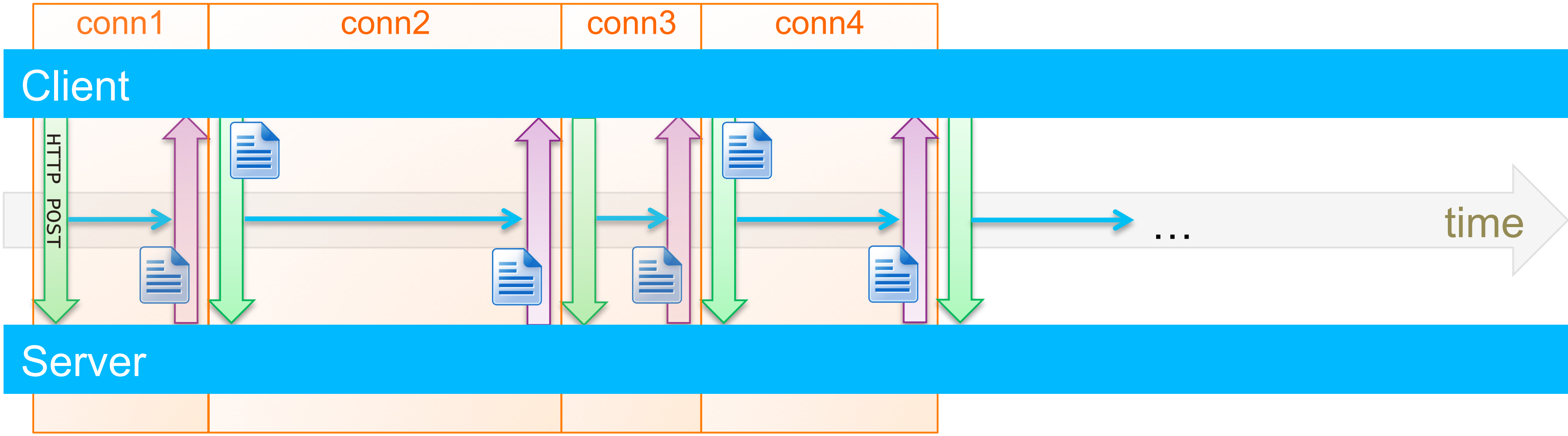
(2): Long Polling



- Client sends initial request
- Server waits until it has data to respond
- Client receives response, and immediately creates new request
- Obvious improvement over plain polling

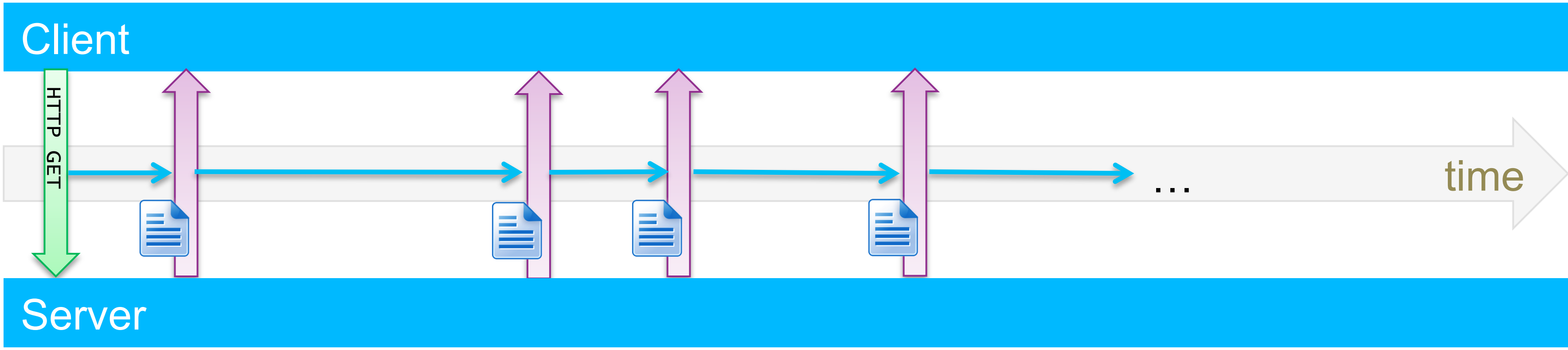
BUT…

(2): Long Polling



- Client sends initial request
- Server waits until it has data to respond
- Client receives response, and immediately creates new request
- Obvious improvement over plain polling
- *Each request/response creates and closes a connection*
- *Client has to wait to send new data until the server responds*

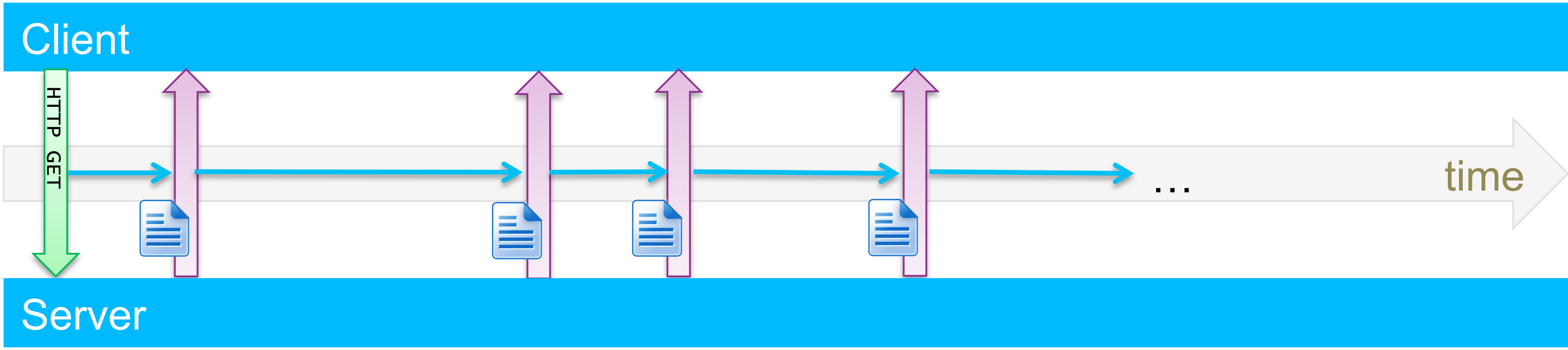
(3): Streaming / forever response



- Client sends initial request
- Server waits until it has data to respond
- Server responds by streaming data
    - Server has an open connection to *push* updates
- Connection is maintained

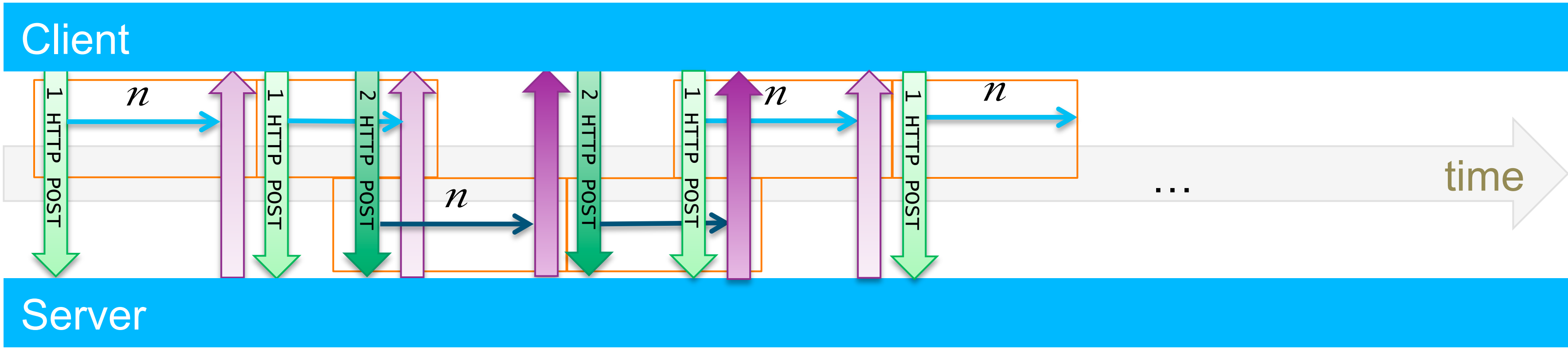
BUT…

(3): Streaming / forever response



- Client sends initial request
- Server waits until it has data to respond
- Server responds by streaming data
  - Server has an open connection to *push* updates
- Connection is maintained
- *It is half-duplex: only server to client*
- *User agents and proxies might not like partial responses*

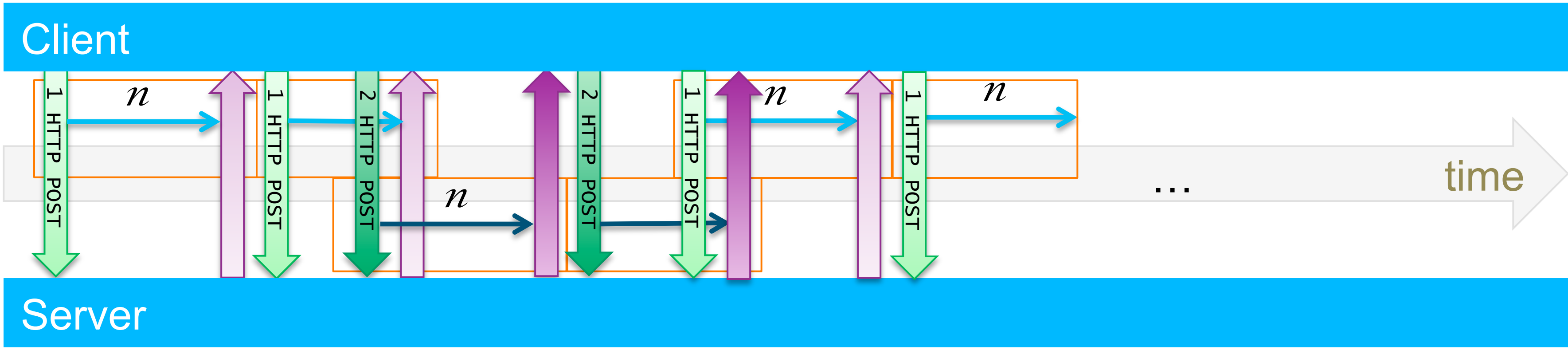
(4): Multiple connections



- Long polling over two separate HTTP connections
  - Approximation of bi-directional connection
  - Two connections are used (HTTP recommended max)
    - long polling
    - second connection allows client to send data to the server

BUT...

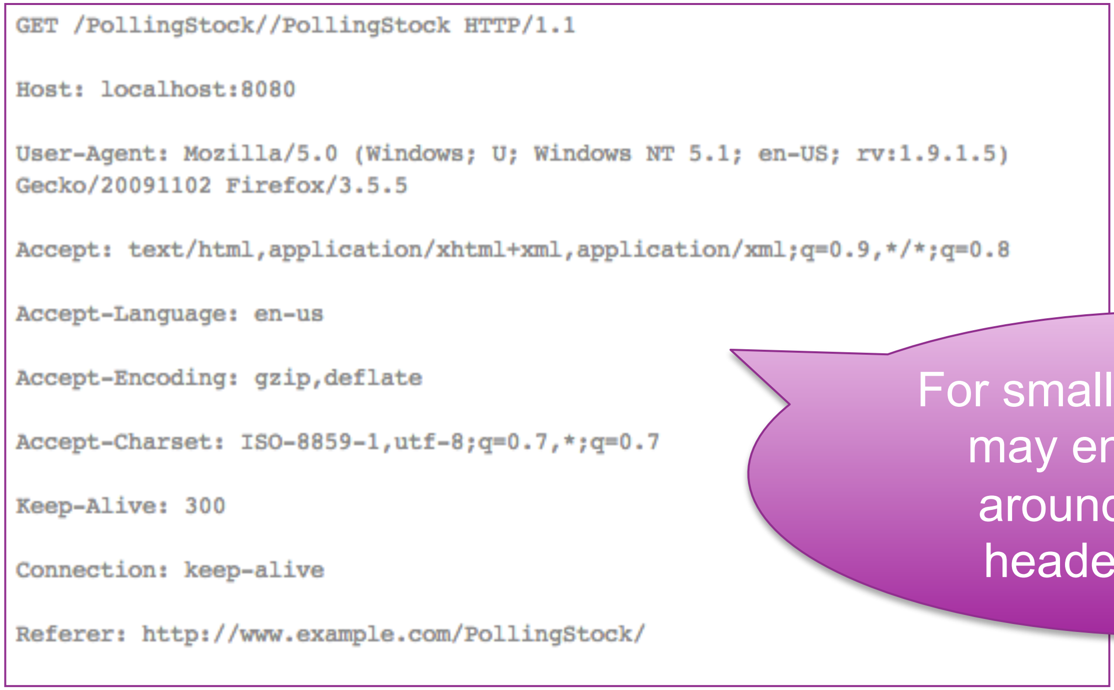
(4): Multiple connections



- Long polling over two separate HTTP connections
  - Approximation of bi-directional connection
  - Two connections are used (HTTP recommended max)
    - long polling
    - second connection allows client to send data to the server
- *Non-trivial connection coordination and management*
- *Two connections for every client*

Hidden cost of HTTP…

- TCP handshake when establishing new connection
  - Even worse for SSL...
- HTTP headers on every message
  - Always present, can vary in size and quantity



```
GET /PollingStock//PollingStock HTTP/1.1

Host: localhost:8080

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5)
Gecko/20091102 Firefox/3.5.5

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-us

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Referer: http://www.example.com/PollingStock/
```

http://www.websocket.org/quantum.html

For small messages, you may end up pushing around more HTTP headers than data!

</tripDownMemoryLane>

Impact2014

Be First.

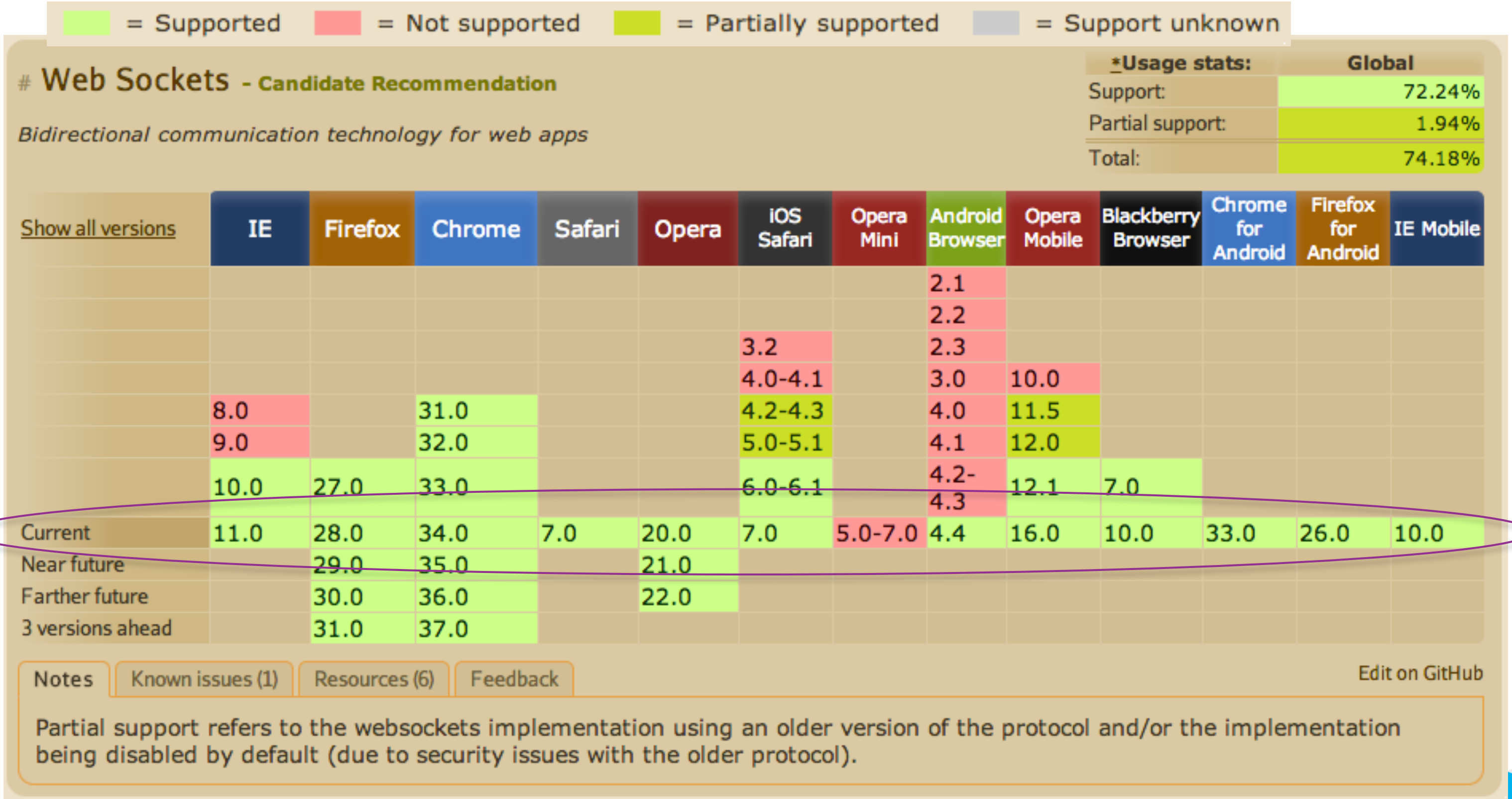
#ibmimpact

There is a better way: WebSockets

- Bi-directional
  - Client and server can send messages at any time
  - ~~Long-polling, one way streaming~~ (hooray!)
- Full duplex
  - Client and server can send updates at the same time
  - No requirement for request/response pair or message ordering
- Single long running connection with established context
  - No connection management/coordination
- Connection upgraded from HTTP
  - No new connection protocol to build infrastructure for
- Efficient use of bandwidth and CPU
  - **Messages can focus on application data**

WebSockets have been standardized

- IETF RFC-6455: WebSocket Protocol Specification, 2011
- JSR 356: WebSocket API Specification, 2013
  - Part of Java EE 7
- Fairly broad adoption for newer browsers and clients



🟩 = Supported 🟥 = Not supported 🟨 = Partially supported ⬜ = Support unknown

Web Sockets - Candidate Recommendation

Bidirectional communication technology for web apps

*Usage stats:	Global
Support:	72.24%
Partial support:	1.94%
Total:	74.18%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Opera Mobile	Blackberry Browser	Chrome for Android	Firefox for Android	IE Mobile
								2.1					
								2.2					
						3.2		2.3					
						4.0-4.1		3.0	10.0				
	8.0		31.0			4.2-4.3		4.0	11.5				
	9.0		32.0			5.0-5.1		4.1	12.0				
	10.0	27.0	33.0			6.0-6.1		4.2-4.3	12.1	7.0			
Current	11.0	28.0	34.0	7.0	20.0	7.0	5.0-7.0	4.4	16.0	10.0	33.0	26.0	10.0
Near future		29.0	35.0		21.0								
Farther future		30.0	36.0		22.0								
3 versions ahead		31.0	37.0										

Partial support refers to the websockets implementation using an older version of the protocol and/or the implementation being disabled by default (due to security issues with the older protocol).

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

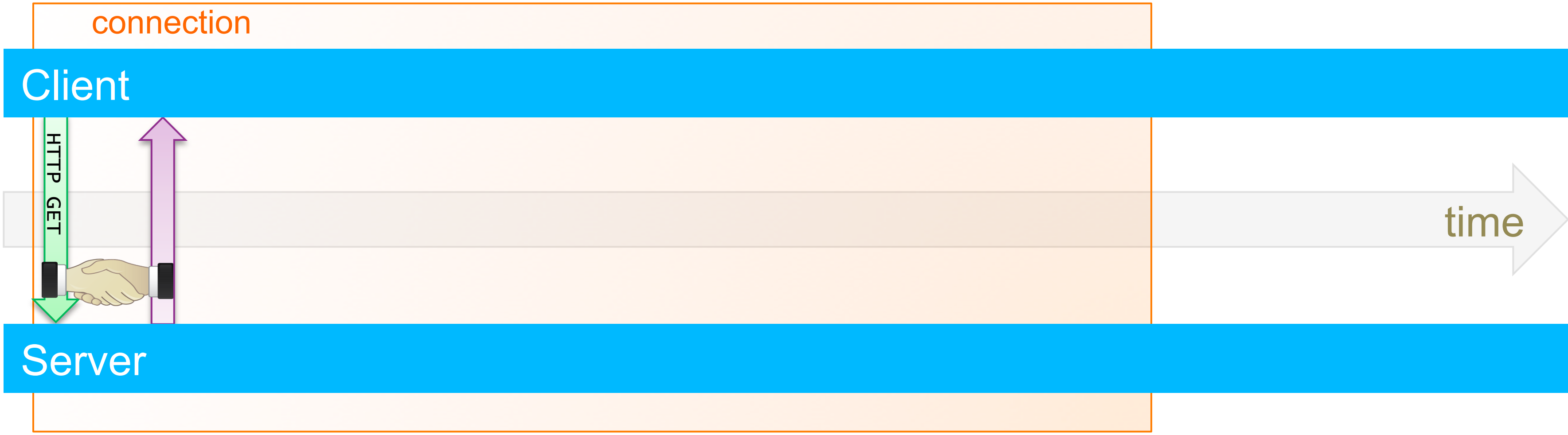
<background>
How WebSockets work..

Impact2014

Be First.

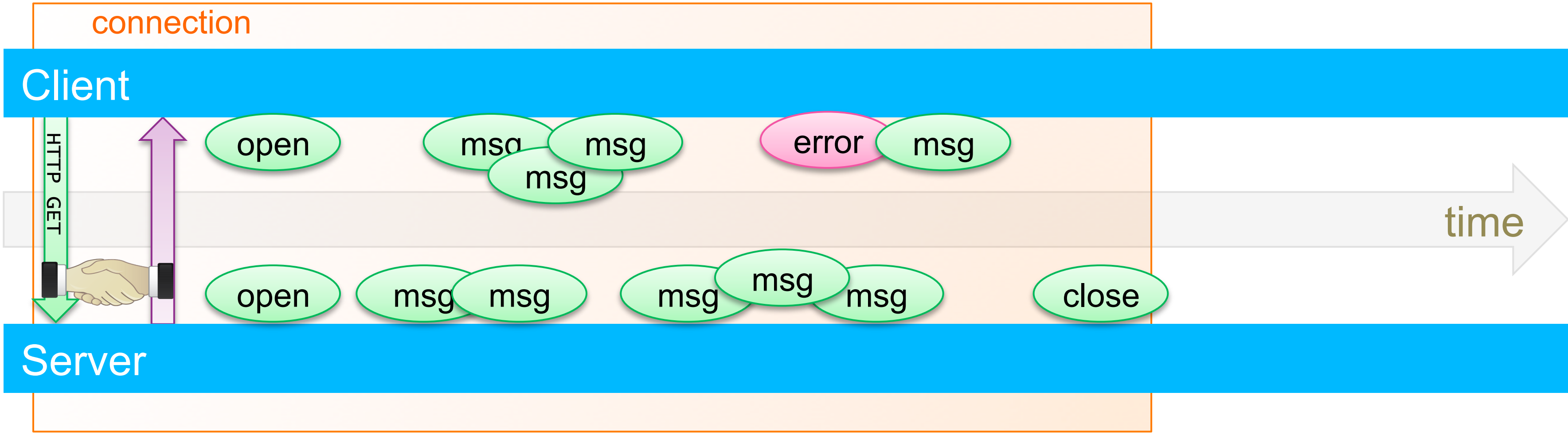
#ibmimpact

WebSocket connection



- Handshake:
  - Client initiates connection
  - Server responds (accepts the upgrade)

WebSocket connection



- Handshake:
  - Client initiates connection
  - Server responds (accepts the upgrade)
- Once the WebSocket is established
  - both sides notified that socket is open
  - either side can send messages at any time
  - either side can close the socket

WebSocket Protocol: it starts with a handshake...

request

```
GET /myapp HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: GhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
Sec-WebSocket-Protocol: custom
Sec-WebSocket-Extensions: compress
Origin: http://example.com
...
```

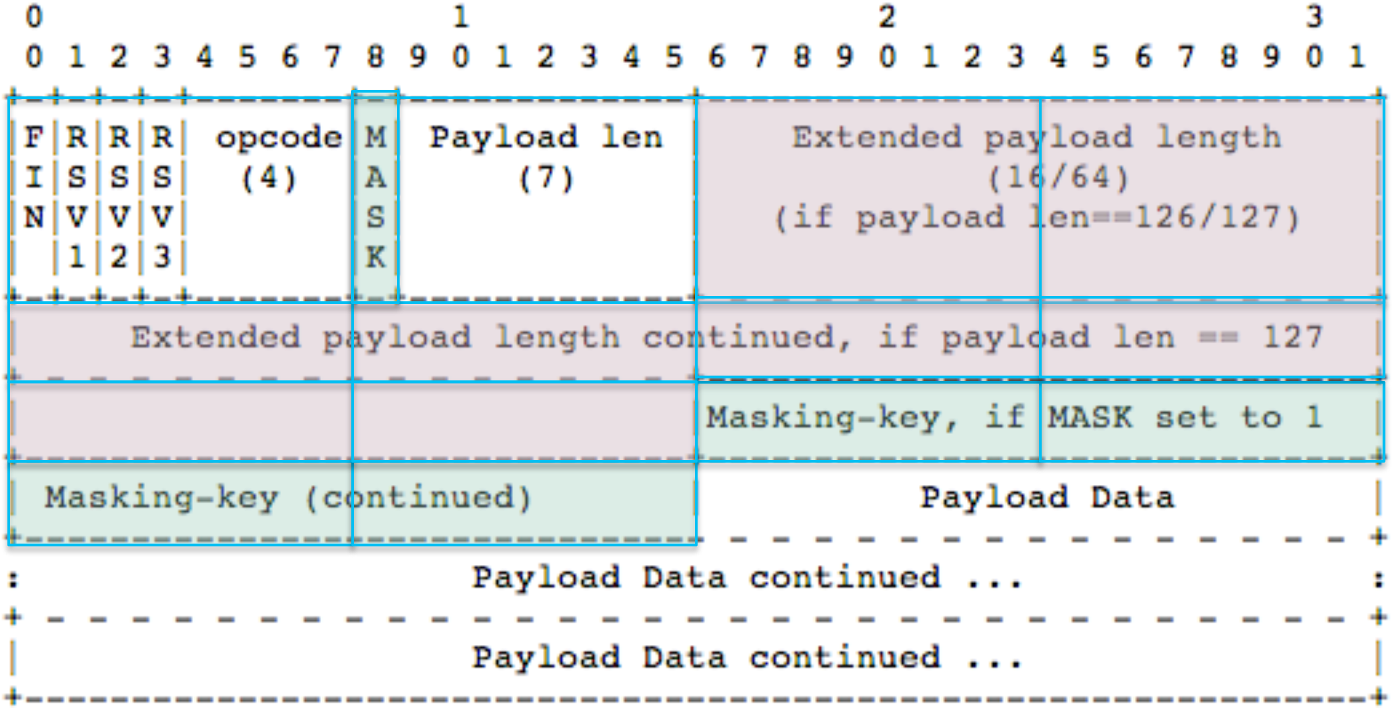
Request upgrade to WebSocket connection

WebSocket handshake headers

response

```
HTTP/1.1 101 Switching Protocols
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: custom
Sec-WebSocket-Extensions: compress
```

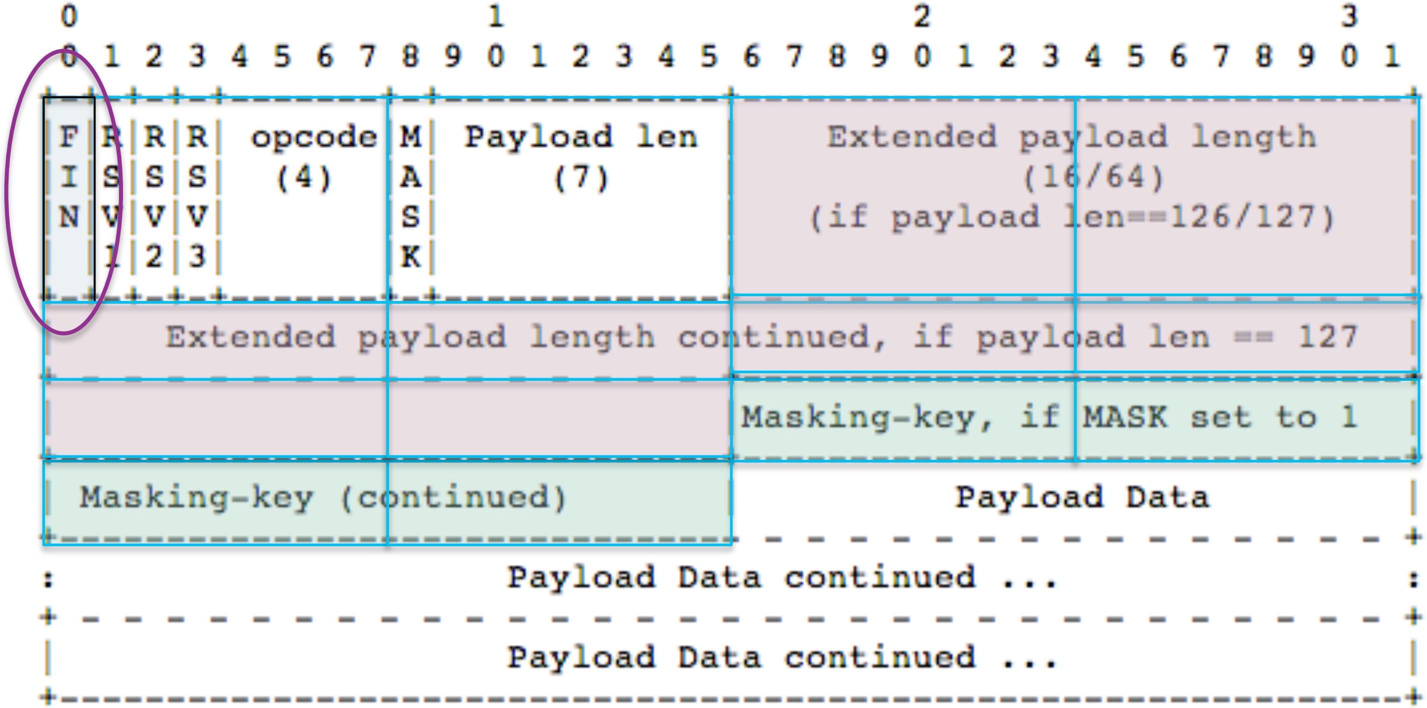
... and then transitions to frames



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

- Data or text is transmitted in frames
  - Minimally framed: small header, then payload

Messages can be fragmented across frames



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

- Message can be in one or more frames
  - Continue until FIN
  - A frame contains data for only one message
  - Extensions can be used to multiplex connections

Op Codes: identifying messages

- Control frames
  - Ping – 0xA
  - Pong – 0x9
  - Close – 0x8
- Data frames
  - Text – 0x1
    - UTF-8
  - Binary – 0x2
    - Arbitrary content: up to the application layer to determine
- Additional op codes are defined by negotiated extensions
  - Use reserved flags in the header

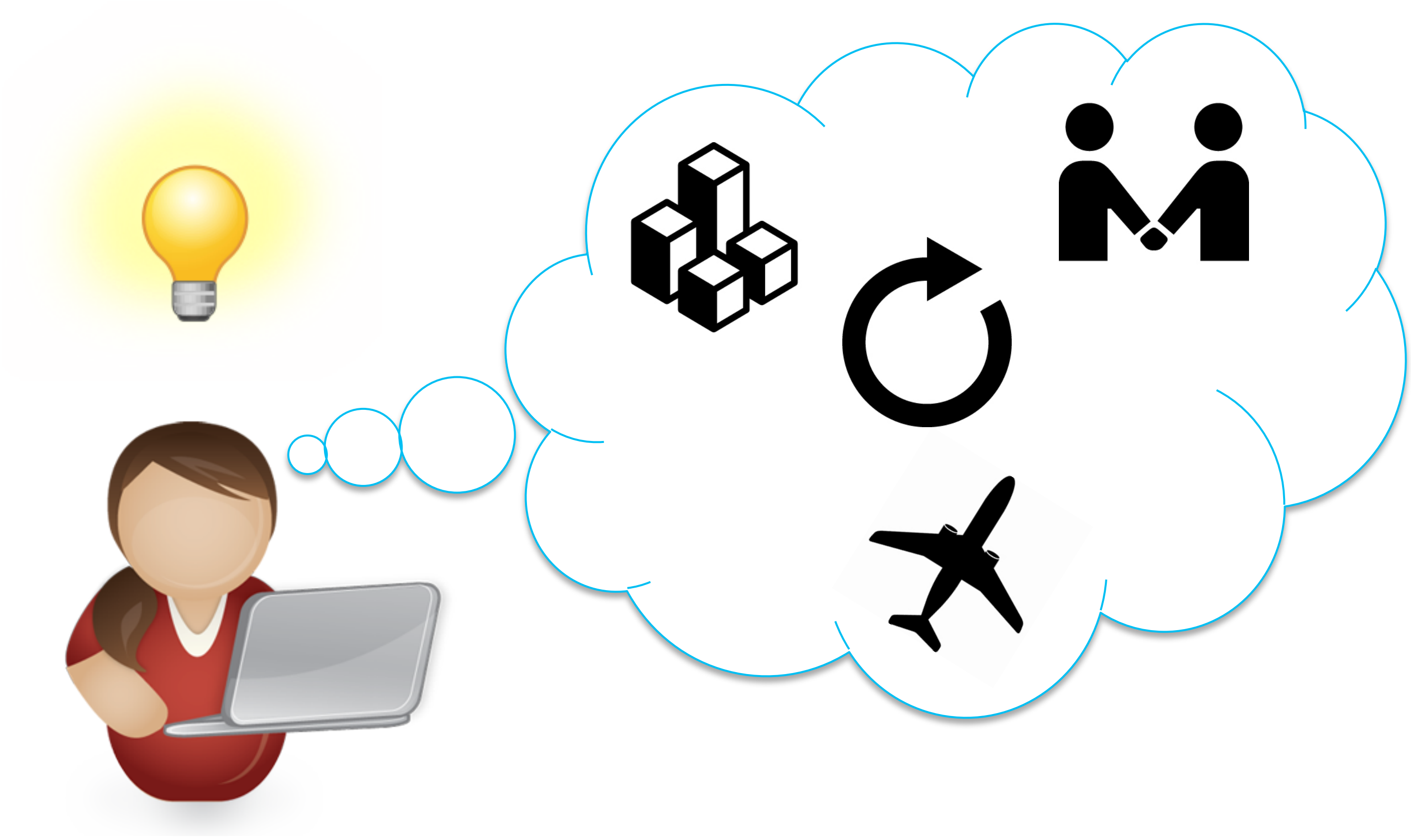
</background>

Impact2014

Be First.

#ibmimpact

How do we use WebSockets in an application?



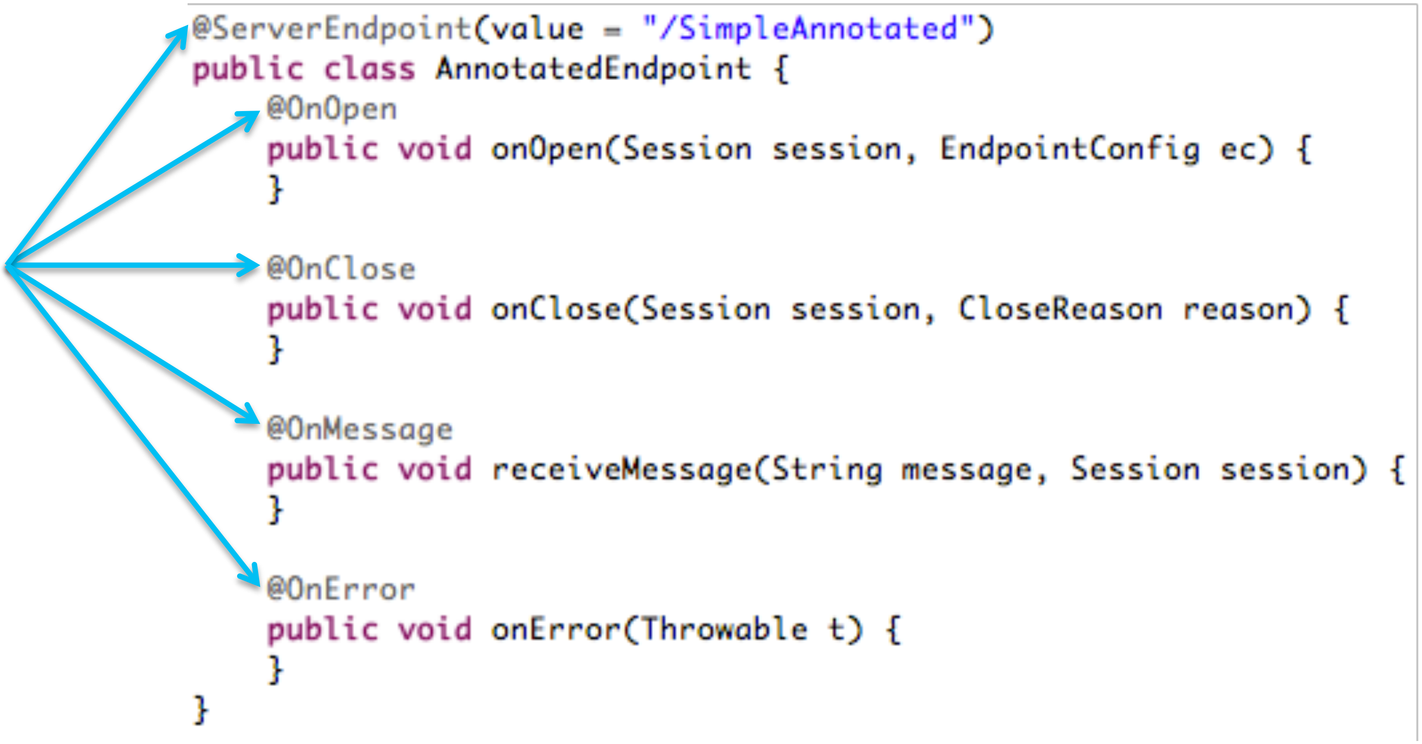
Judy

WebSockets API

- Programmatic or annotation-based approach
- Client and Server Endpoints
  - Have a lifecycle
    - onOpen
    - onClose
    - onError
  - Communicate using Messages
    - onMessage
    - send
  - Use sessions
- Encoders and Decoders deal with data formatting
  - Messages ←→Java Objects
- SPI: extensions and data frames

Server Endpoint: Annotated

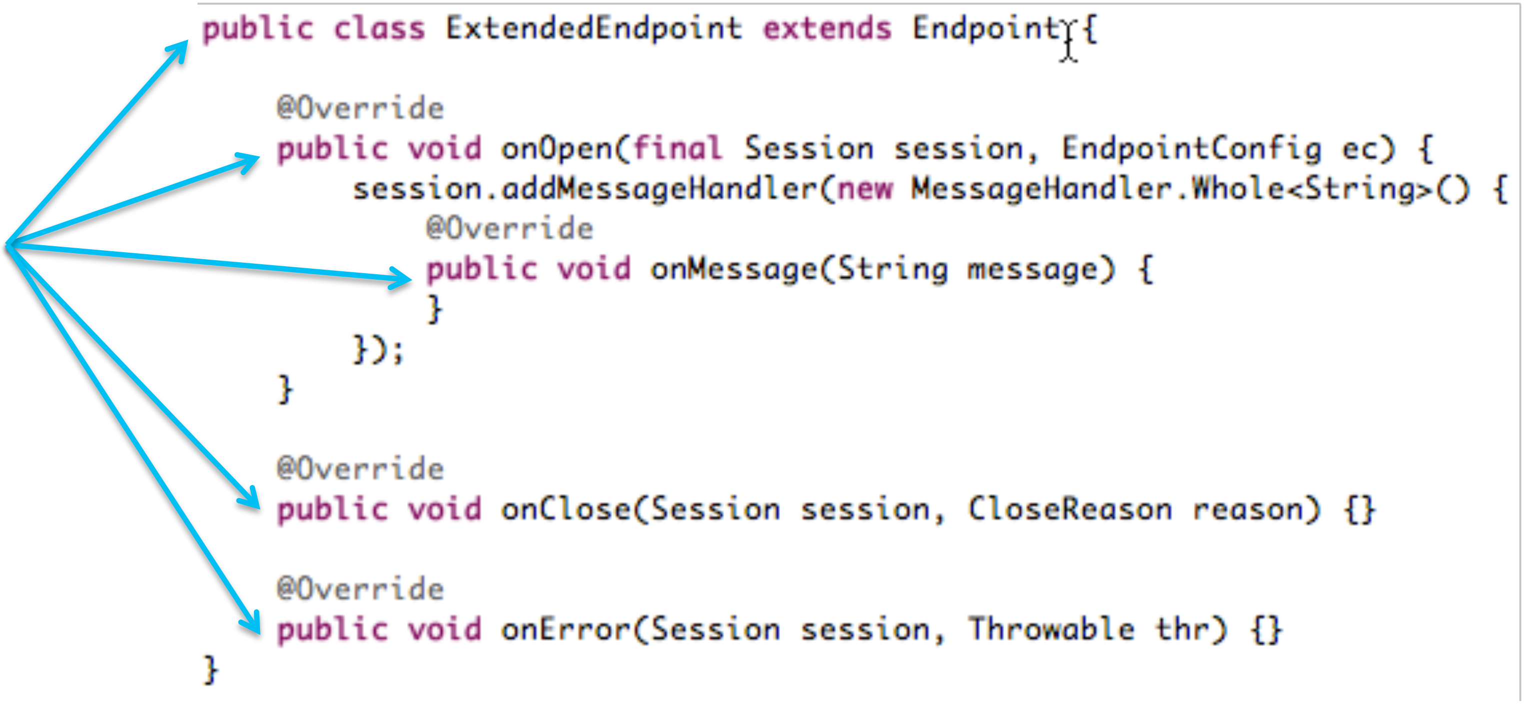
- Simple POJO with `@ServerEndpoint` annotation
  - value is the URI relative to your app's context root, e.g. ws://localhost/myapp/SimpleAnnotated
- Annotations for notifications: lifecycle and messages



```
@ServerEndpoint(value = "/SimpleAnnotated")
public class AnnotatedEndpoint {
    @OnOpen
    public void onOpen(Session session, EndpointConfig ec) {
    }

    @OnClose
    public void onClose(Session session, CloseReason reason) {
    }

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

    @OnError
    public void onError(Throwable t) {
    }
}
```

Server Endpoint: Programmatic

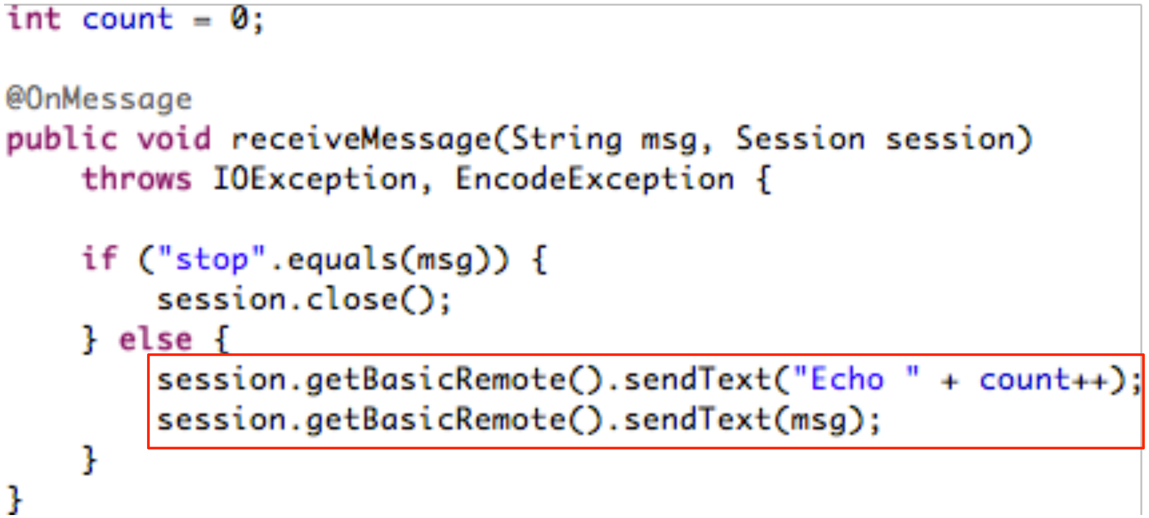
- Class extends `Endpoint`
- Callback methods for lifecycle event notifications
- Message notifications require a `MessageHandler`



```
public class ExtendedEndpoint extends Endpoint {

    @Override
    public void onOpen(final Session session, EndpointConfig ec) {
        session.addMessageHandler(new MessageHandler.Whole<String>() {
            @Override
            public void onMessage(String message) {
            }
        });
    }

    @Override
    public void onClose(Session session, CloseReason reason) {}

    @Override
    public void onError(Session session, Throwable thr) {}
}
```

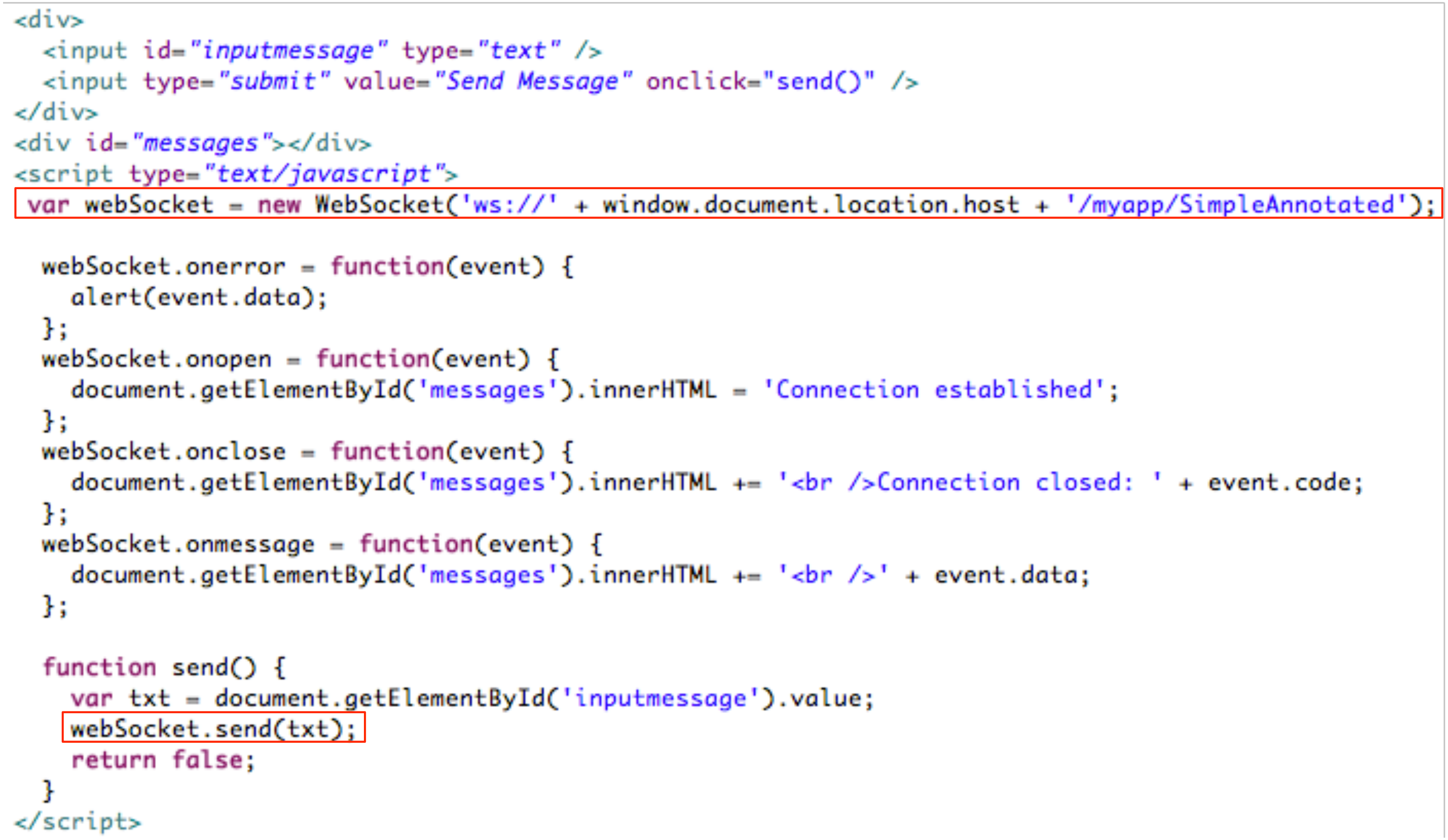
Simple echo + server provided data (using annotations)

- `@OnMessage` method is called when a message is received
  - If message is ‘stop’: close the session
  - Otherwise, echo the message along with a hit count



```
int count = 0;

@OnMessage
public void receiveMessage(String msg, Session session)
    throws IOException, EncodeException {

    if ("stop".equals(msg)) {
        session.close();
    } else {
        session.getBasicRemote().sendText("Echo " + count++);
        session.getBasicRemote().sendText(msg);
    }
}
```

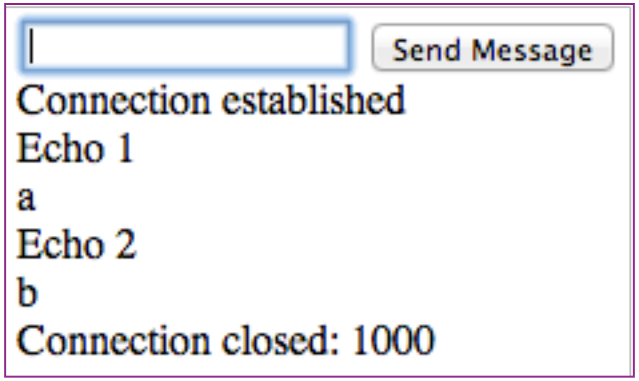
JavaScript client invocation...



```
<div>
  <input id="inputmessage" type="text" />
  <input type="submit" value="Send Message" onclick="send()" />
</div>
<div id="messages"></div>
<script type="text/javascript">
 var webSocket = new WebSocket('ws://' + window.document.location.host + '/myapp/SimpleAnnotated');

  webSocket.onerror = function(event) {
    alert(event.data);
  };
  webSocket.onopen = function(event) {
    document.getElementById('messages').innerHTML = 'Connection established';
  };
  webSocket.onclose = function(event) {
    document.getElementById('messages').innerHTML += '<br />Connection closed: ' + event.code;
  };
  webSocket.onmessage = function(event) {
    document.getElementById('messages').innerHTML += '<br />' + event.data;
  };

  function send() {
    var txt = document.getElementById('inputmessage').value;
    webSocket.send(txt);
    return false;
  }
</script>
```

Invocation.. what happens?

- Connection established with page load
  - Browser starts the handshake
  - Server completes
  - “onopen” invoked on client/server
    - Client prints “Connection established”

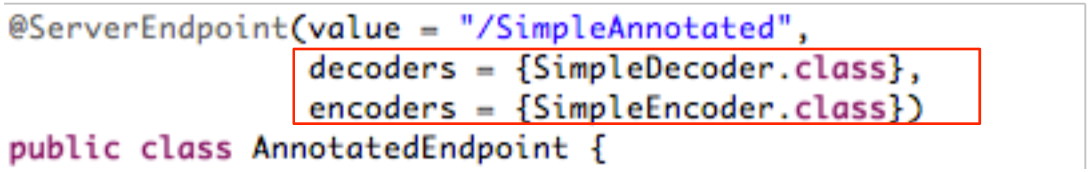


```
Connection established
Echo 1
a
Echo 2
b
Connection closed: 1000
```

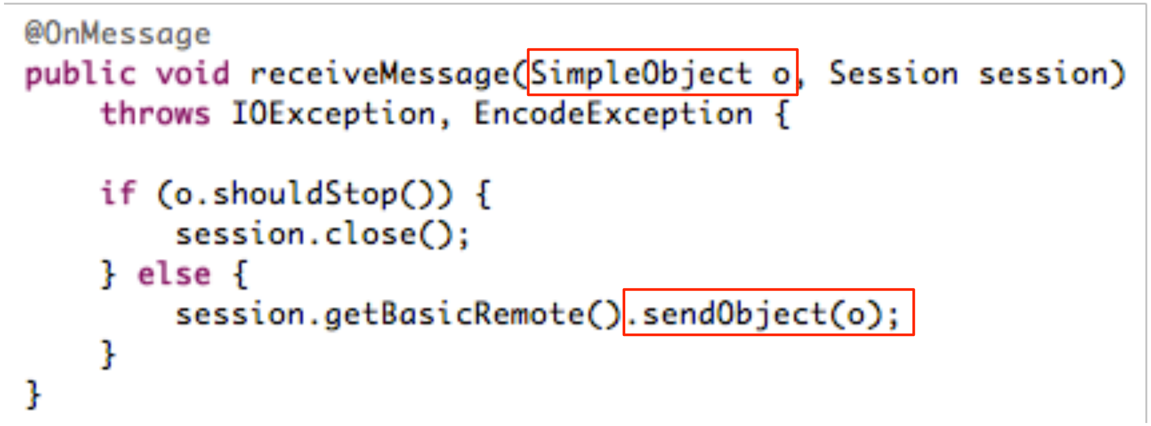
- Client sends ‘a’, receiveMessage method called on the server
  - Server returns “Echo 1” and “a” (two messages)
- Client sends ‘b’,
  - Server returns “Echo 2” and “b”
- Client sends ‘stop’
  - Server closes the session
  - Client’s onclose invoked, prints “Connection closed: 1000”

Encoder/Decoder: dealing with data

- Messages can be in text or binary format
- Encoders and Decoders translate between data on the socket and Java Objects

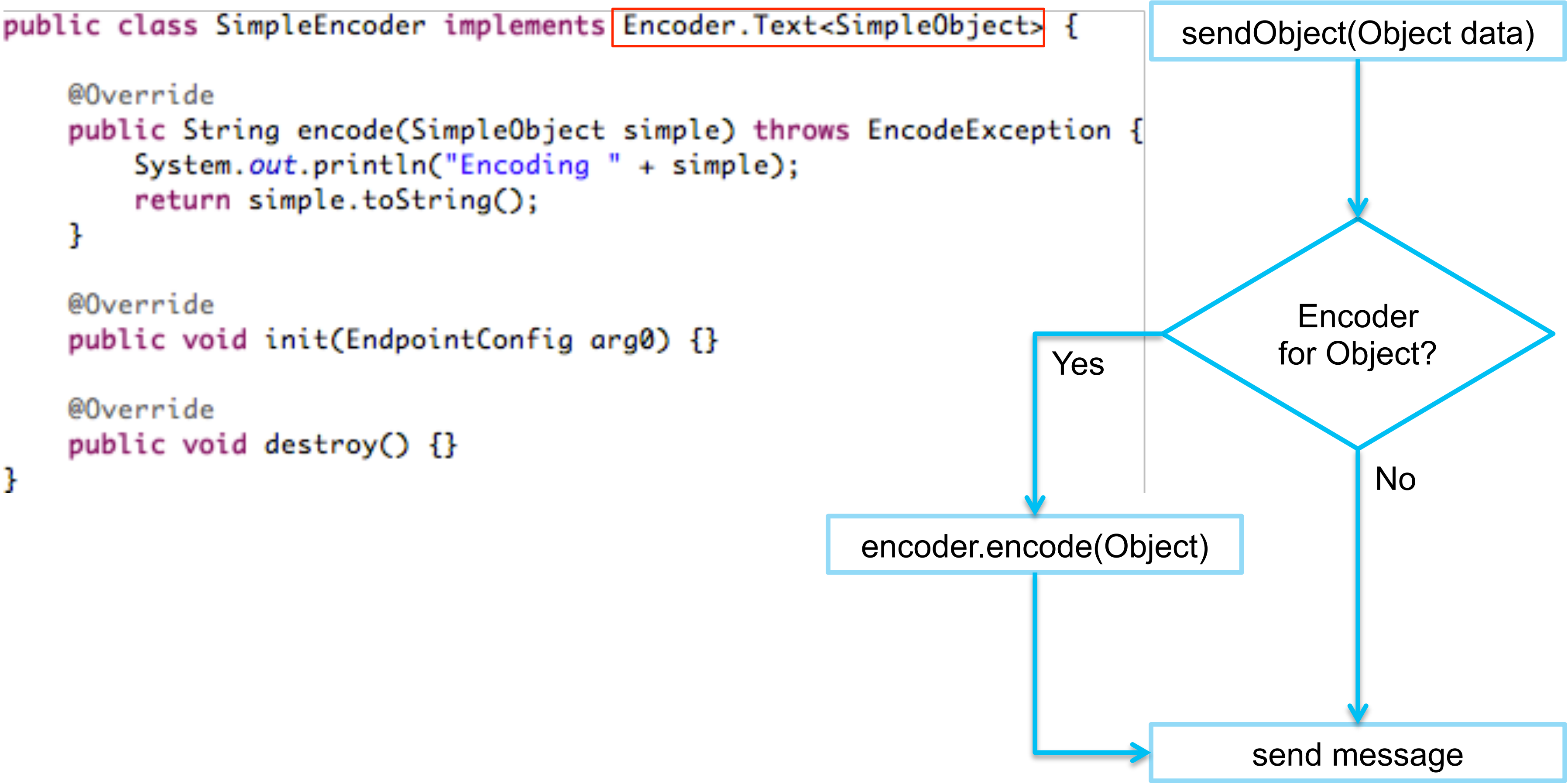


```
@ServerEndpoint(value = "/SimpleAnnotated",
                decoders = {SimpleDecoder.class},
                encoders = {SimpleEncoder.class})
public class AnnotatedEndpoint {
```



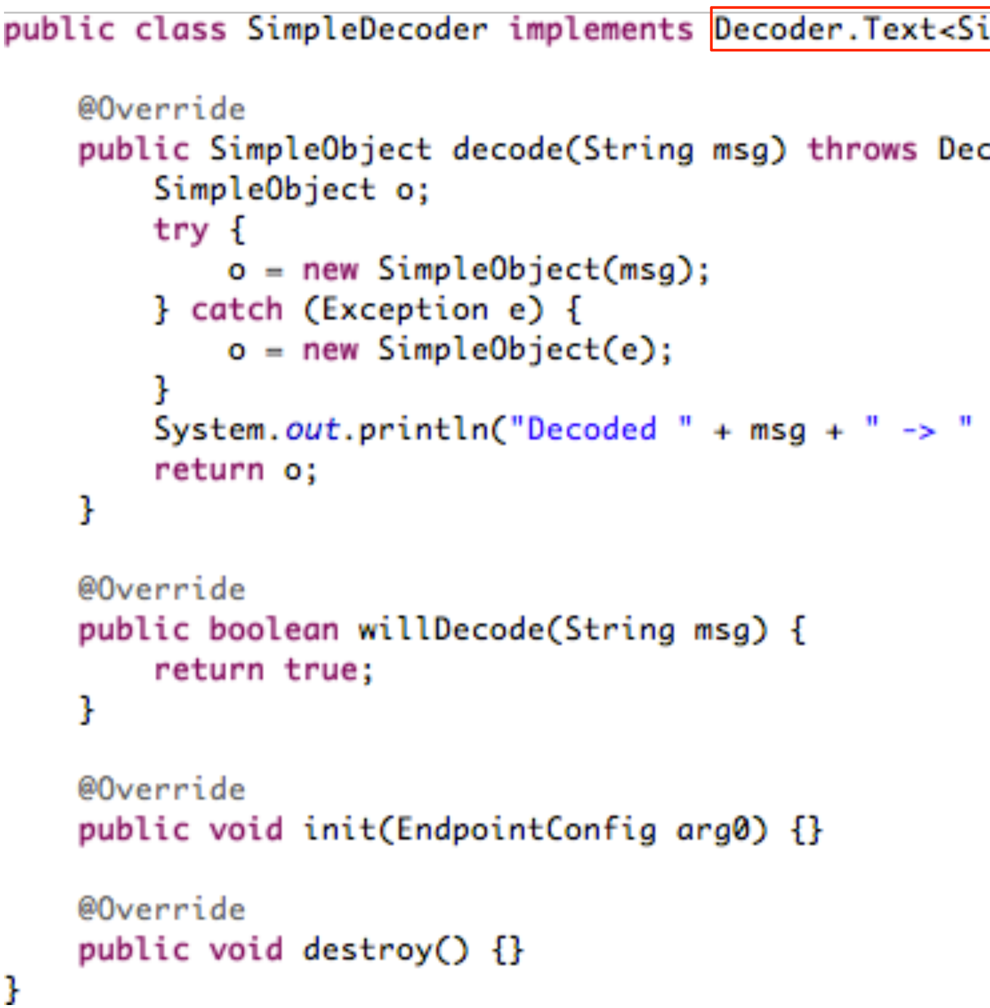
```
@OnMessage
public void receiveMessage(SimpleObject o, Session session)
    throws IOException, EncodeException {

    if (o.shouldStop()) {
        session.close();
    } else {
        session.getBasicRemote().sendObject(o);
    }
}
```

Encoder: dealing with data

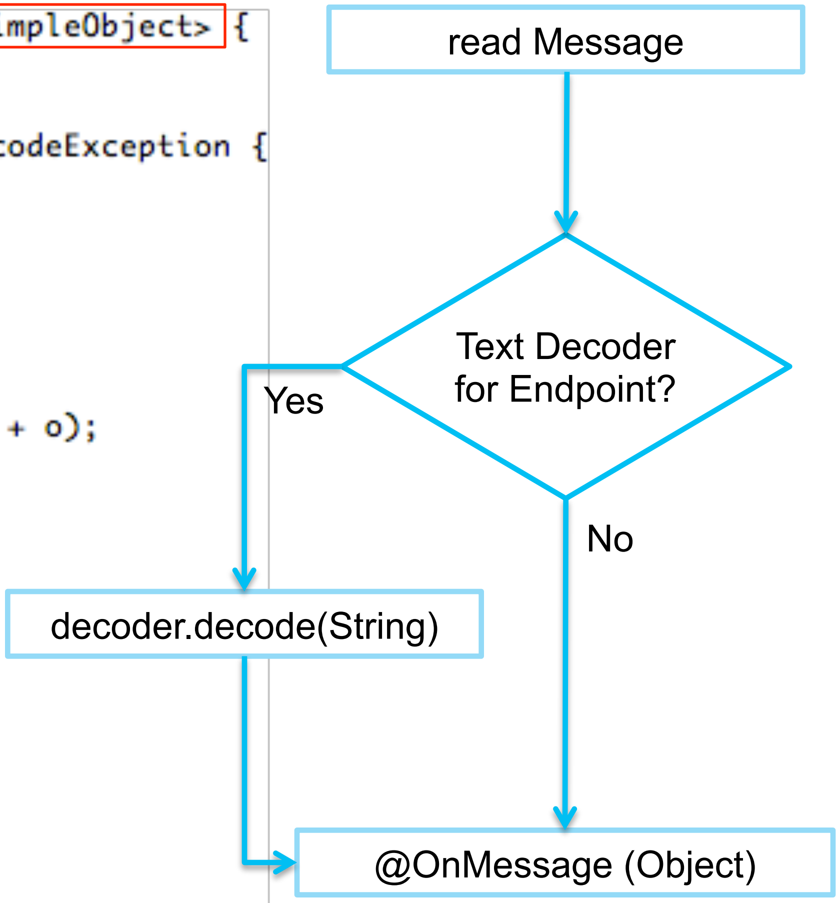


```java
public class SimpleEncoder implements Encoder.Text<SimpleObject> {

    @Override
    public String encode(SimpleObject simple) throws EncodeException {
        System.out.println("Encoding " + simple);
        return simple.toString();
    }

    @Override
    public void init(EndpointConfig arg0) {}

    @Override
    public void destroy() {}
}
```

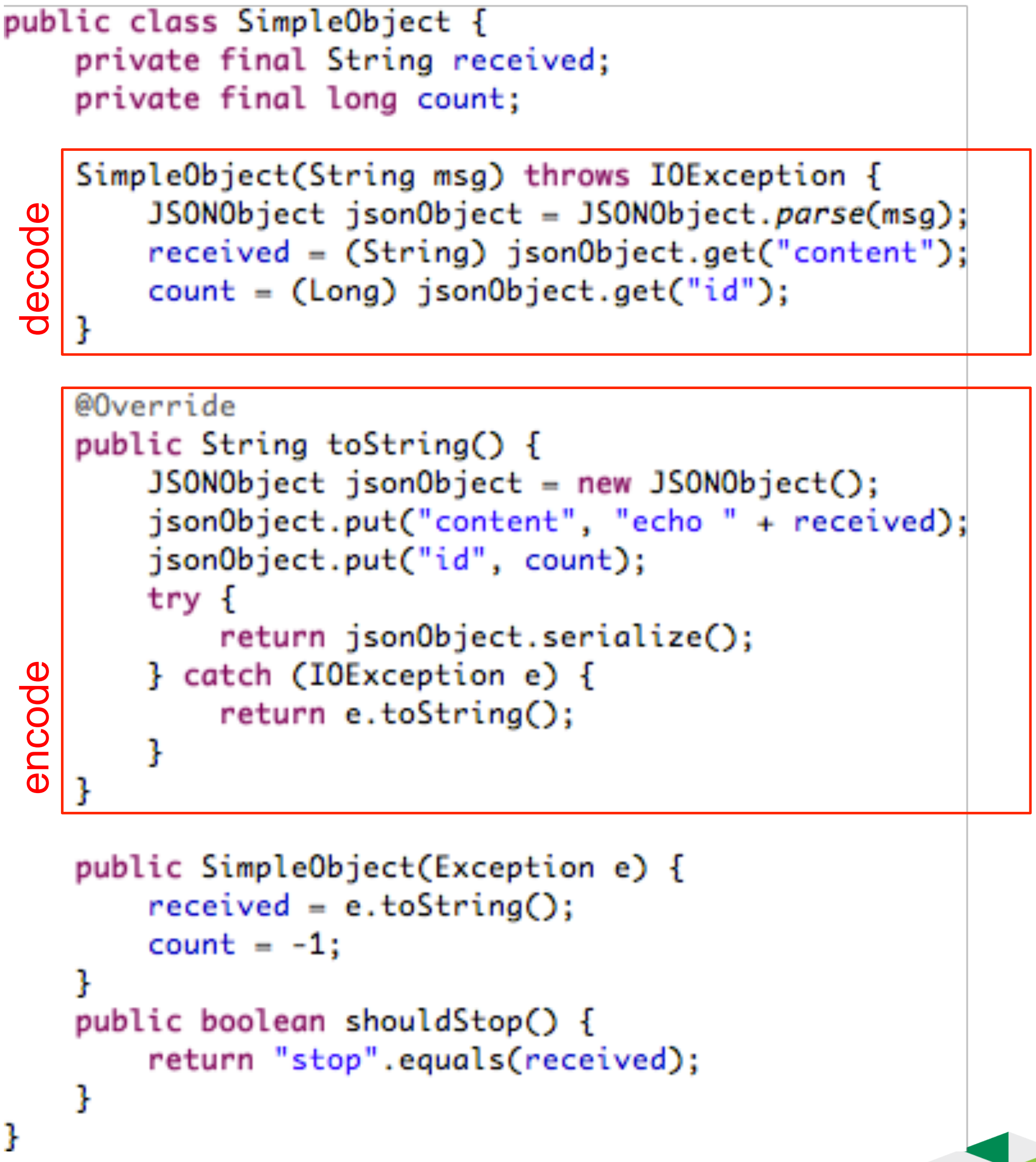
Decoder: dealing with data



```
public class SimpleDecoder implements Decoder.Text<SimpleObject> {

    @Override
    public SimpleObject decode(String msg) throws DecodeException {
        SimpleObject o;
        try {
            o = new SimpleObject(msg);
        } catch (Exception e) {
            o = new SimpleObject(e);
        }
        System.out.println("Decoded " + msg + " -> " + o);
        return o;
    }

    @Override
    public boolean willDecode(String msg) {
        return true;
    }

    @Override
    public void init(EndpointConfig arg0) {}

    @Override
    public void destroy() {}
}
```

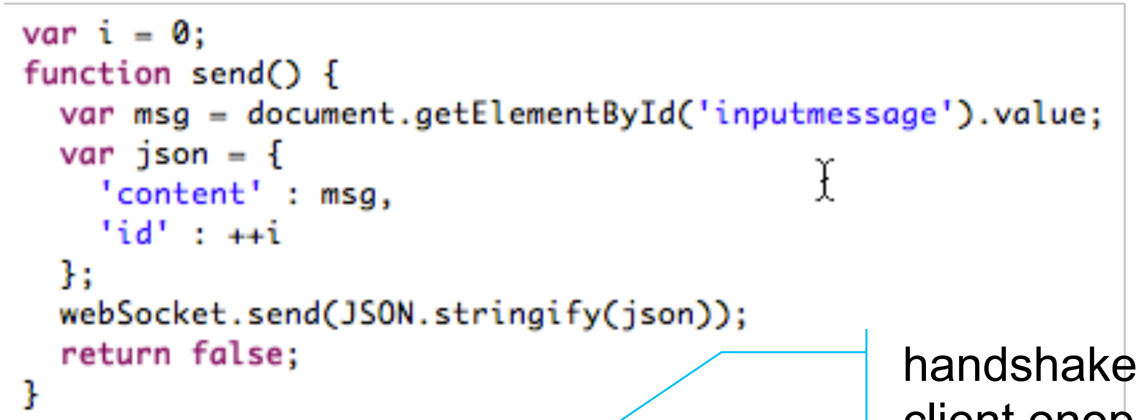


SimpleObject

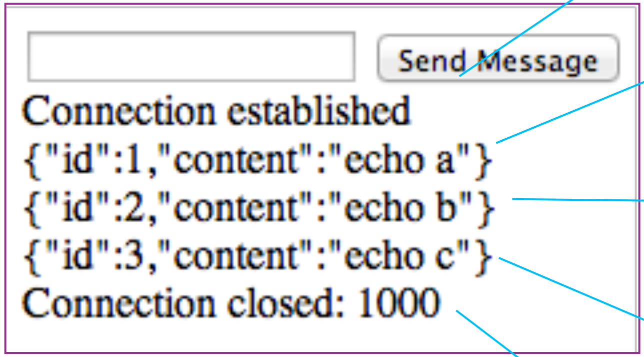


```
public class SimpleObject {
    private final String received;
    private final long count;

    SimpleObject(String msg) throws IOException {
        JSONObject jsonObject = JSONObject.parse(msg);
        received = (String) jsonObject.get("content");
        count = (Long) jsonObject.get("id");
    }

    @Override
    public String toString() {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("content", "echo " + received);
        jsonObject.put("id", count);
        try {
            return jsonObject.serialize();
        } catch (IOException e) {
            return e.toString();
        }
    }

    public SimpleObject(Exception e) {
        received = e.toString();
        count = -1;
    }
    public boolean shouldStop() {
        return "stop".equals(received);
    }
}
```

JavaScript and invocation



```
var i = 0;
function send() {
  var msg = document.getElementById('inputmessage').value;
  var json = {
    'content' : msg,
    'id' : ++i
  };
  webSocket.send(JSON.stringify(json));
  return false;
}
```



```
Connection established
{"id":1,"content":"echo a"}
{"id":2,"content":"echo b"}
{"id":3,"content":"echo c"}
Connection closed: 1000
```

handshake
client onopen method invoked

client sent {"content":"a","id":1}
server returned {"id":1,"content":"echo a"}

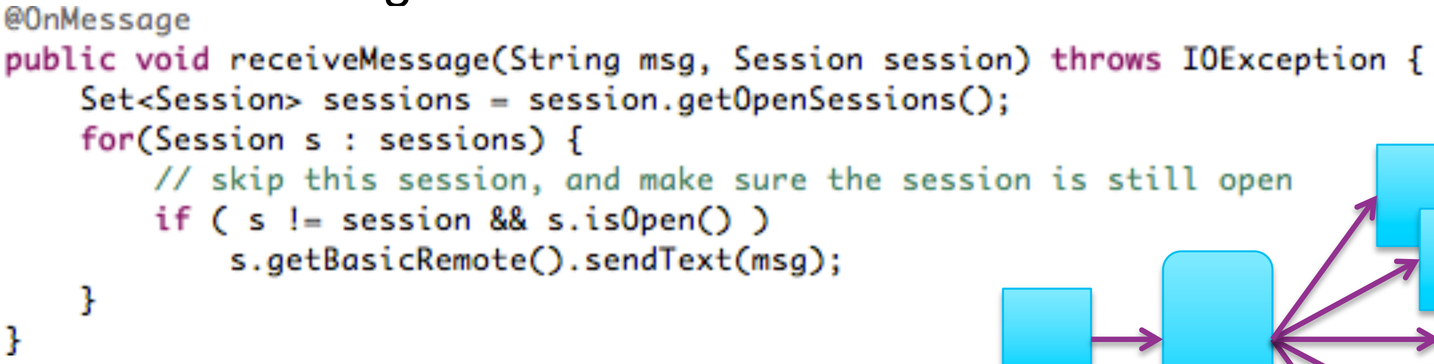
client sent {"content":”b","id”:2}
server returned {"id”:2,"content":"echo b"}

client sent {"content":”c","id”:3}
server returned {"id”:3,"content":"echo c"}

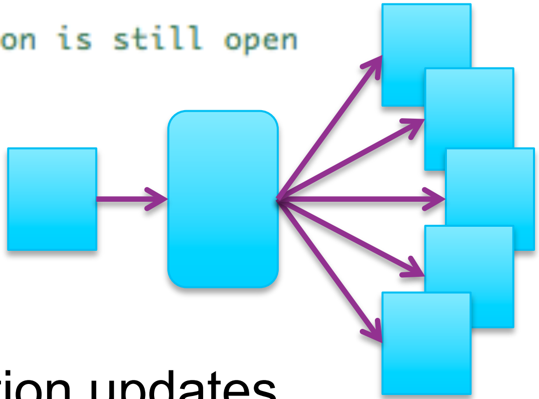
client sent {"content":”stop","id”:4}
server closed connection
client onclose method invoked

Message broadcasting

- Broadcast a message to all the sessions connected to a given Server Endpoint
  - Receive a message
  - Find all current sessions associated with the current endpoint
  - Send the message to all of the other WebSocket sessions



```
@OnMessage
public void receiveMessage(String msg, Session session) throws IOException {
    Set<Session> sessions = session.getOpenSessions();
    for(Session s : sessions) {
        // skip this session, and make sure the session is still open
        if ( s != session && s.isOpen() )
            s.getBasicRemote().sendText(msg);
    }
}
```



- Uses:
  - Multiplayer video games
  - Social networking: chats/tweets/GPS location updates
  - Whiteboard collaborations/meetings
  - ...

WebSockets have everything this application needs..

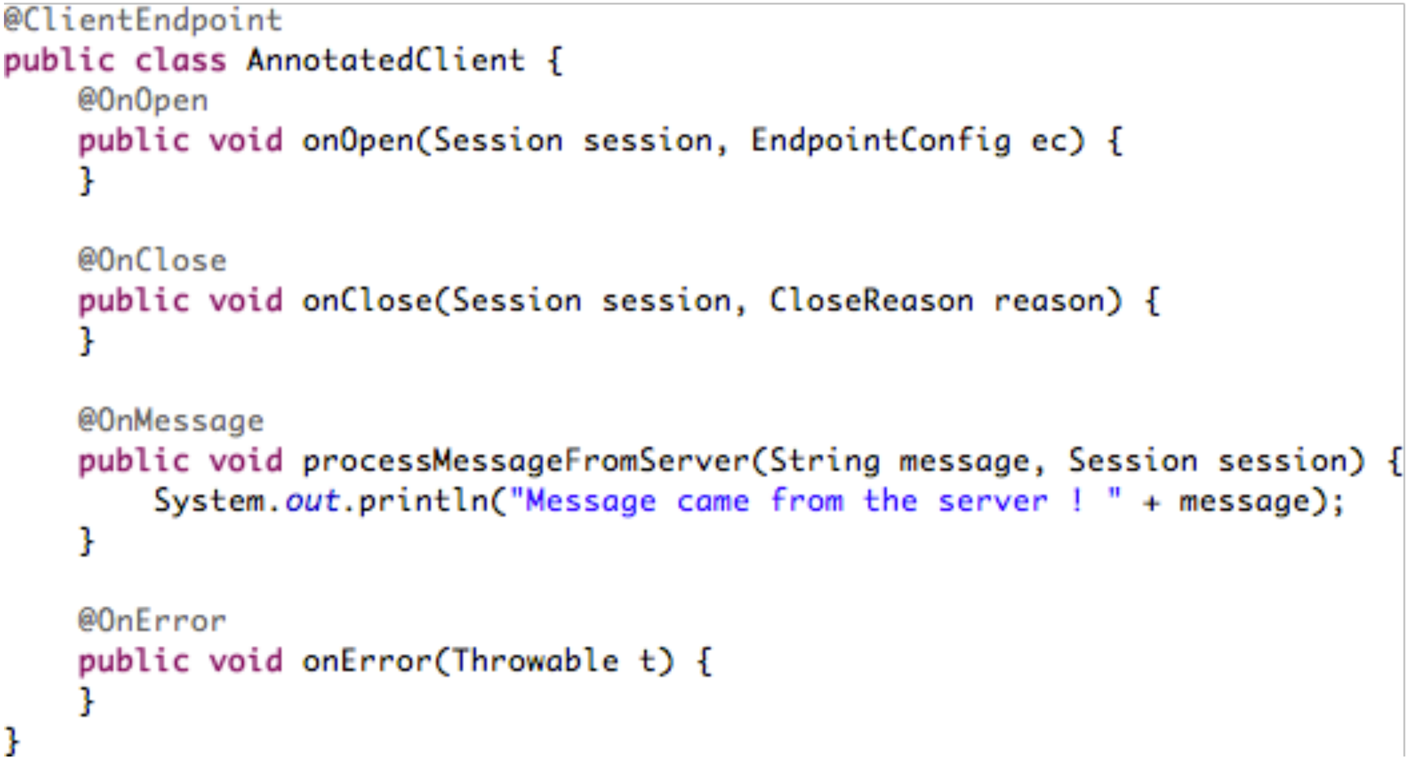
- Why wouldn’t you use them?
  - Older devices / browsers don’t support WebSockets
  - May be a challenge to degrade gracefully so older devices still have a decent experience
- Trouble with proxies..
  - wss:// recommended over ws://
    - Some proxy servers do not inspect encrypted traffic: just pass through

WebSockets and Proxy Servers

- WebSocket protocol is unaware of proxies / firewalls
- Proxy servers required to strip certain headers when forwarding
  - *Some proxies adjusted to leave Connection / Upgrade headers*
- Hop-by-Hop Upgrade
  - Proxy server sends the request to the next hop
  - Upgrade header is only good for one link
    - *Proxy server updated to propagate the Upgrade headers...*
- HTTP CONNECT
  - Proxy to forward the TCP Connection to the destination
    - SSL tunneling
  - *Some proxies still analyze traffic: would choke on websocket frame*
  - *Some proxies restrict CONNECT to SSL*
  - *SSL termination*

WebSockets for Rich Clients

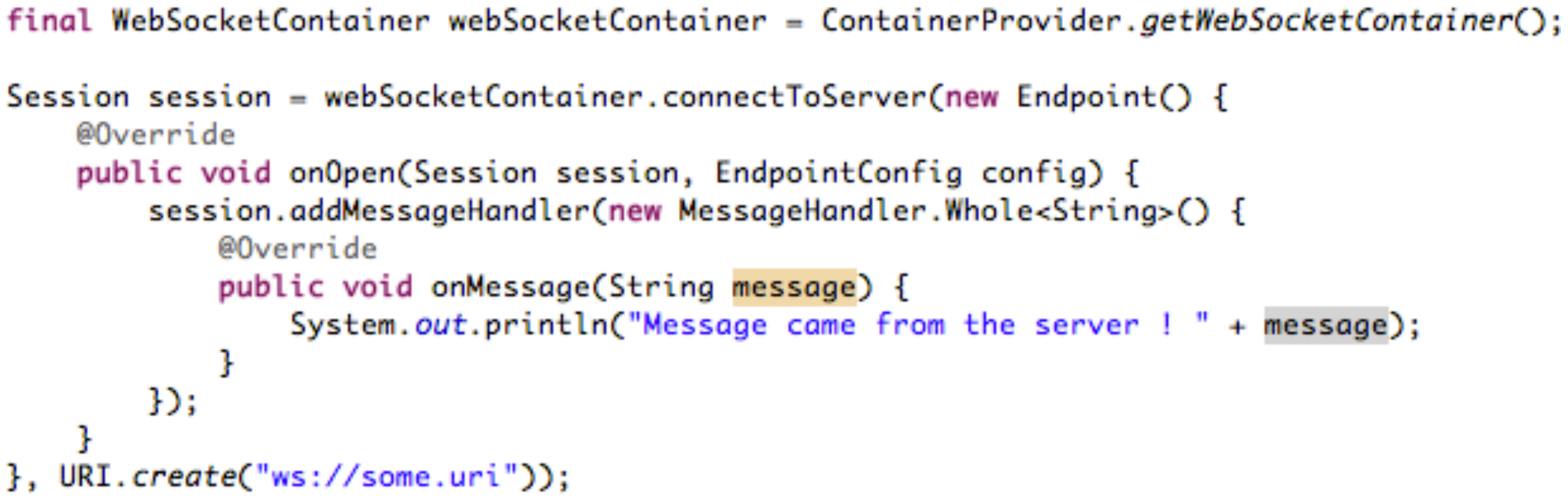
- Java API for a rich Client is similar to API for Server
- Annotations:



```java
@ClientEndpoint
public class AnnotatedClient {
    @OnOpen
    public void onOpen(Session session, EndpointConfig ec) {
    }

    @OnClose
    public void onClose(Session session, CloseReason reason) {
    }

    @OnMessage
    public void processMessageFromServer(String message, Session session) {
        System.out.println("Message came from the server ! " + message);
    }

    @OnError
    public void onError(Throwable t) {
    }
}
```

WebSockets for Rich Clients

- Java API for a rich Client is similar to API for Server
- Programmatic:



```
final WebSocketContainer webSocketContainer = ContainerProvider.getWebSocketContainer();

Session session = webSocketContainer.connectToServer(new Endpoint() {
    @Override
    public void onOpen(Session session, EndpointConfig config) {
        session.addMessageHandler(new MessageHandler.Whole<String>() {
            @Override
            public void onMessage(String message) {
                System.out.println("Message came from the server ! " + message);
            }
        });
    }
}, URI.create("ws://some.uri"));
```

Questions?

Impact2014

Be First. ►► ►

#ibmimpact

We Value Your Feedback

- Don' t forget to submit your Impact session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Use the Conference Mobile App or the online Agenda Builder to quickly submit your survey
  - Navigate to "Surveys" to see a view of surveys for sessions you' ve attended

Other sessions you might like

AAD-2522	MongoDB & IBM WebSphere Mon, 04:00 PM - 05:00 PM, Venetian Delfino 4101 B
AAD-2488	Getting the Most out of Modular OSGi Applications in IBM WebSphere Application Server Mon, 05:15 PM - 06:15 PM, Venetian Delfino 4105
AAD-1627	Using CDI Portable Extensions on Liberty Profile V8.5.5 Tue, 10:30 AM - 11:30 AM, Venetian Palazzo P
AAD-2524	Java EE 7 Introduction Tue, 03:45 PM - 04:45 PM, Venetian Palazzo O
AAD-1345	What's New in Java 8 SE Tue, 05:00 PM - 06:00 PM, Venetian Palazzo O
AAD-1278	HTML5 for Mobile Technical Deep Dive Wed, 01:00 PM - 02:00 PM, Venetian Palazzo N

Other sessions you might like

AAD-2818	Agile Development with IBM WebSphere
	Mon, 05:15 PM - 06:15 PM, Venetian Delfino 4103
AAI-1522	Technical Introduction to the IBM WebSphere Liberty Profile
	Tue, 10:30 AM - 11:30 AM, Venetian Palazzo O
AAI-1526	Deployment Topologies
	Tue, 02:15 PM - 03:15 PM, Venetian Marcello 4503
AAI-1524	Rapidly Moving Applications to the IBM WebSphere Liberty Profile
	Wed, 02:15 PM - 03:15 PM, Venetian Palazzo N
AAD-1520	IBM WebSphere Application Server Liberty Profile
	Wed, 10:30 AM - 11:30 AM, Venetian San Polo 3501 B

Thank You

Impact2014

#ibmimpact

Be First. ►► ►

Legal Disclaimer

-
- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- If the text contains performance statistics or references to benchmarks, insert the following language; otherwise delete:
  Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
- If the text includes any customer examples, please confirm we have prior written approval from such customer and insert the following language; otherwise delete:
  All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.
- Please review text for proper trademark attribution of IBM products. At first use, each product name must be the full name and include appropriate trademark symbols (e.g., IBM Lotus® Sametime® Unyte™). Subsequent references can drop "IBM" but should include the proper branding (e.g., Lotus Sametime Gateway, or WebSphere Application Server). Please refer to http://www.ibm.com/legal/copytrade.shtml for guidance on which trademarks require the ® or ™ symbol. Do not use abbreviations for IBM product names in your presentation. All product names must be used as adjectives rather than nouns. Please list all of the trademarks that you use in your presentation as follows; delete any not included in your presentation. IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino, Quickr, Sametime, WebSphere, UC2, PartnerWorld and Lotusphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. Unyte is a trademark of WebDialogs, Inc., in the United States, other countries, or both.
- If you reference Adobe® in the text, please mark the first use and include the following; otherwise delete:
  Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- If you reference Java™ in the text, please mark the first use and include the following; otherwise delete:
  Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- If you reference Microsoft® and/or Windows® in the text, please mark the first use and include the following, as applicable; otherwise delete:
  Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- If you reference Intel® and/or any of the following Intel products in the text, please mark the first use and include those that you use as follows; otherwise delete:
  Intel, Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- If you reference UNIX® in the text, please mark the first use and include the following; otherwise delete:
  UNIX is a registered trademark of The Open Group in the United States and other countries.
- If you reference Linux® in your presentation, please mark the first use and include the following; otherwise delete:
  Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.
- If the text/graphics include screenshots, no actual IBM employee names may be used (even your own), if your screenshots include fictitious company names (e.g., Renovations, Zeta Bank, Acme) please update and insert the following; otherwise delete: All references to [insert fictitious company name] refer to a fictitious company and are used for illustration purposes only.