ORACLE

Java EE 7 in Practice: Blueprints Reborn

Ed Burns
JSF Spec Lead
edward.burns@oracle.com
@edburns

DEVNEXUS

My Plan for Your Time Investment

- Java EE 7 Overview
- Cargo Tracker Overview
- Java EE 7 Feature Traversal
- Resources

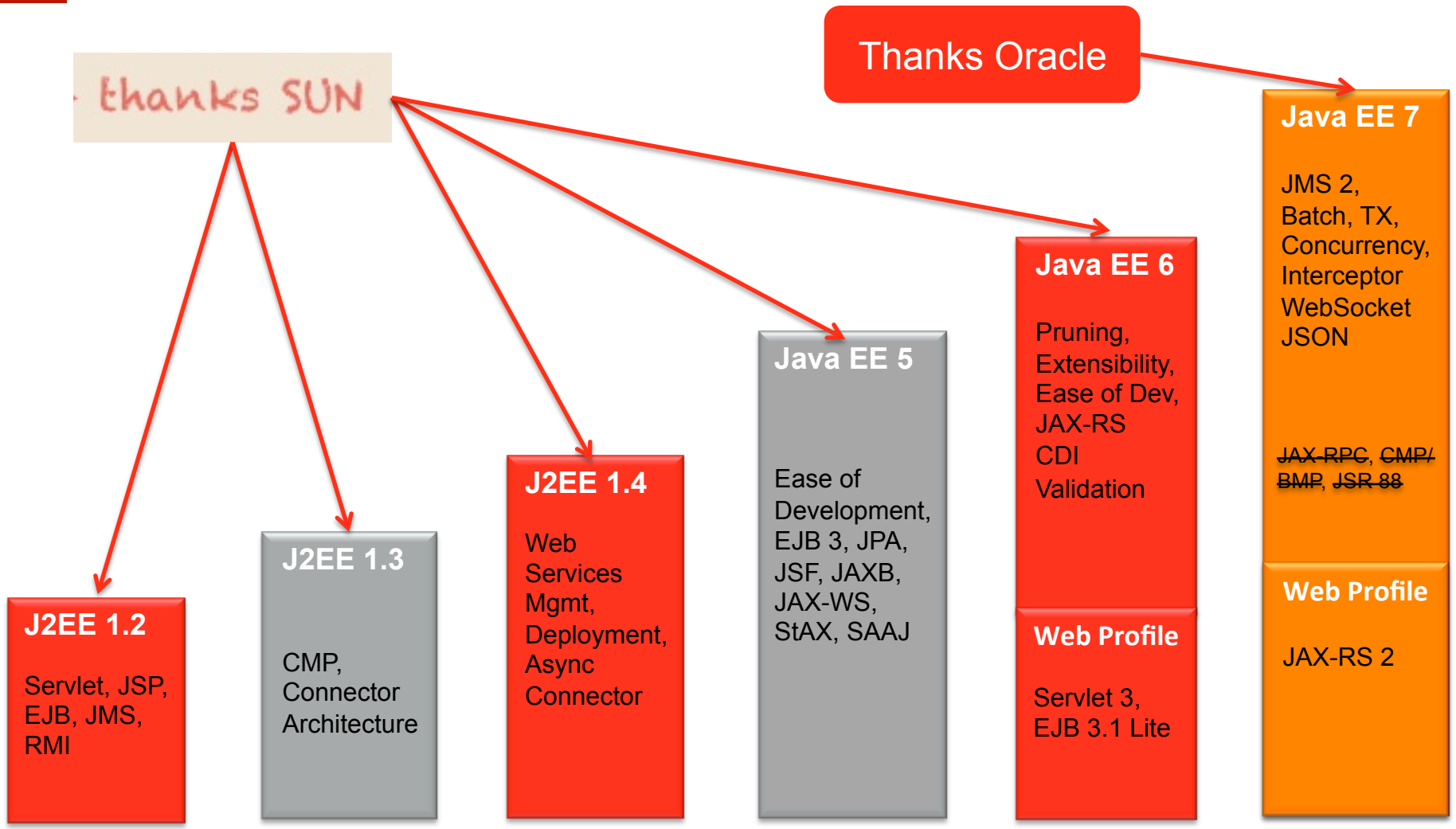
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Speaker Qualifications – Ed Burns

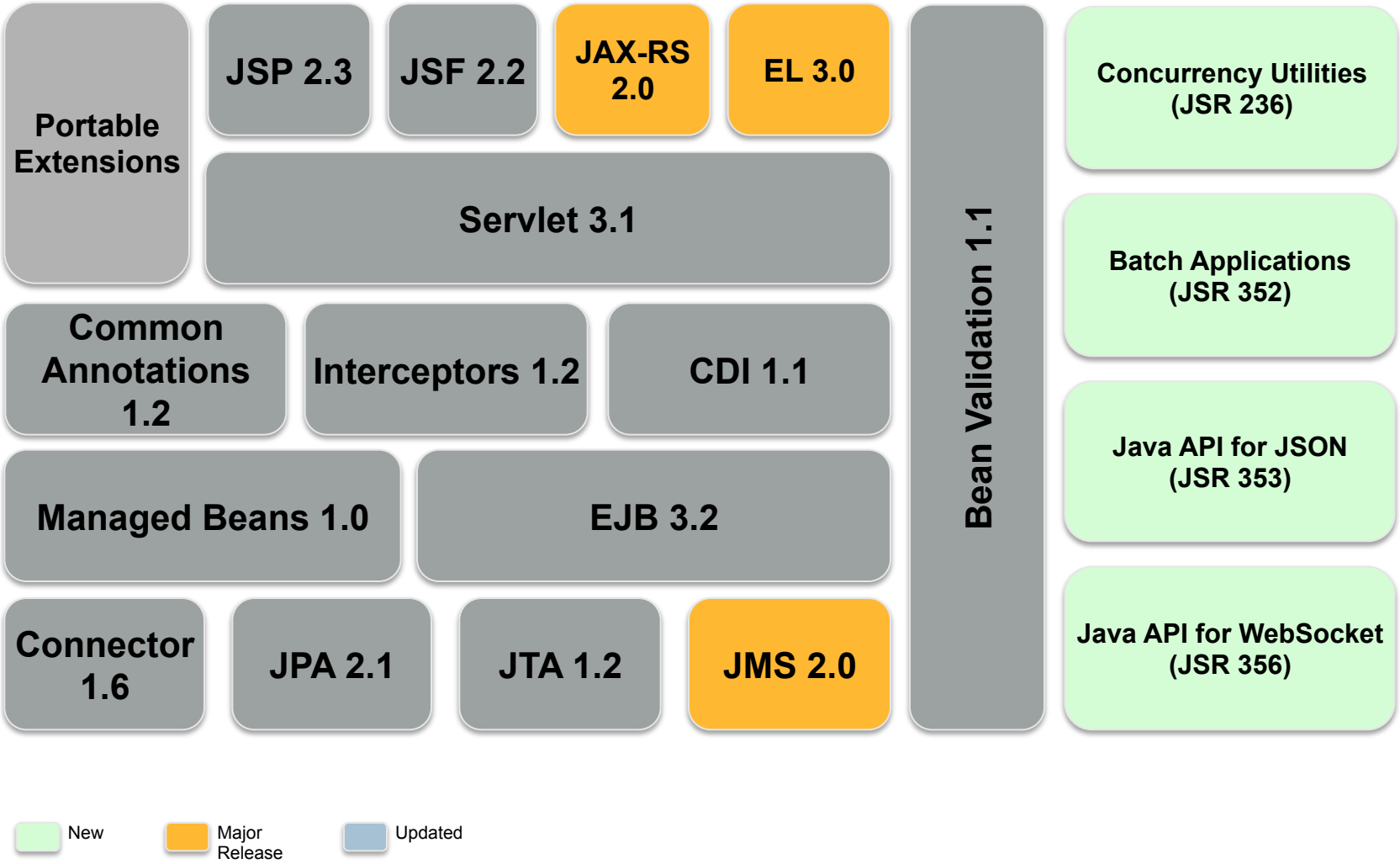
And non-qualifications

- Involved with JSF since 2002
- Spec lead since 2003
  - Most fun part of the job: cleanly integrating other people's great ideas into JSF (and hopefully improving on the in the process)
  - Not an expert in applying JSF in practice
- Author of four books for McGraw-Hill

Java EE Past, Present and Future



Java EE 7



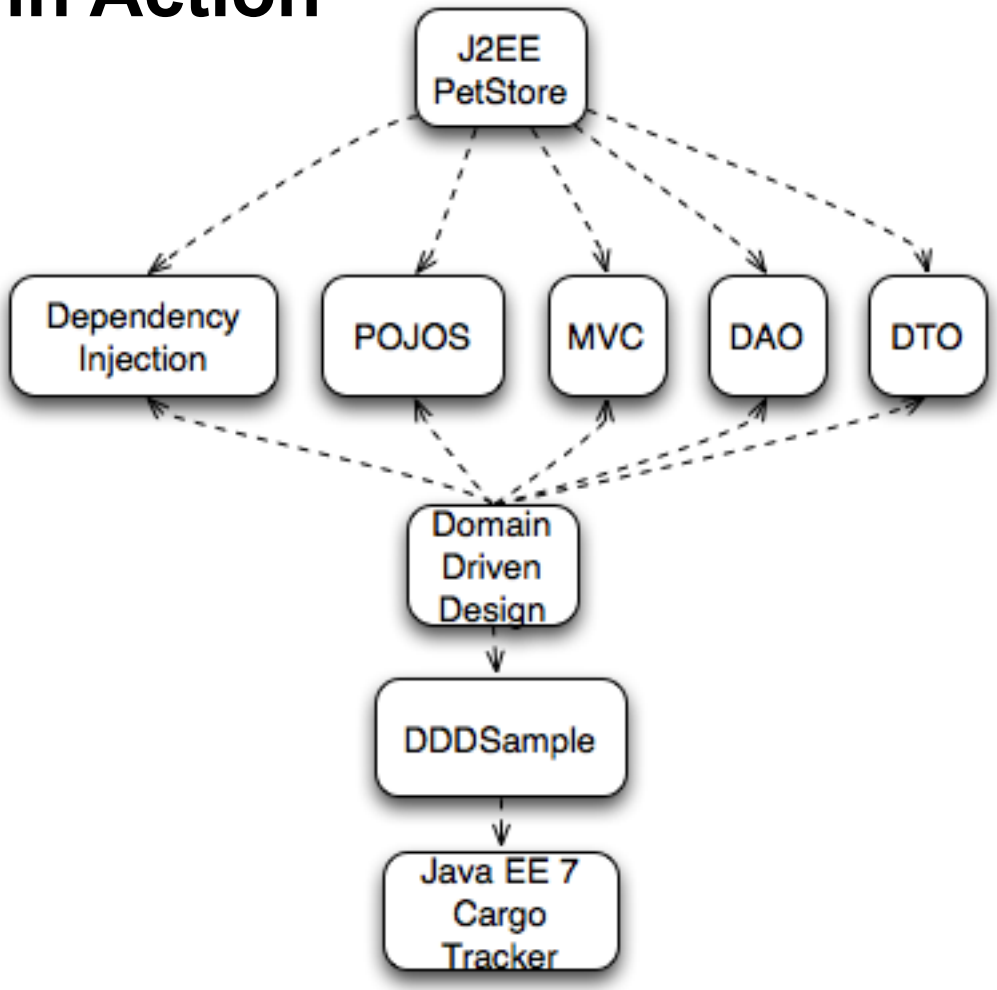
Java EE in Action

Cargo Tracker

Applied domain-driven design blueprints for Java EE

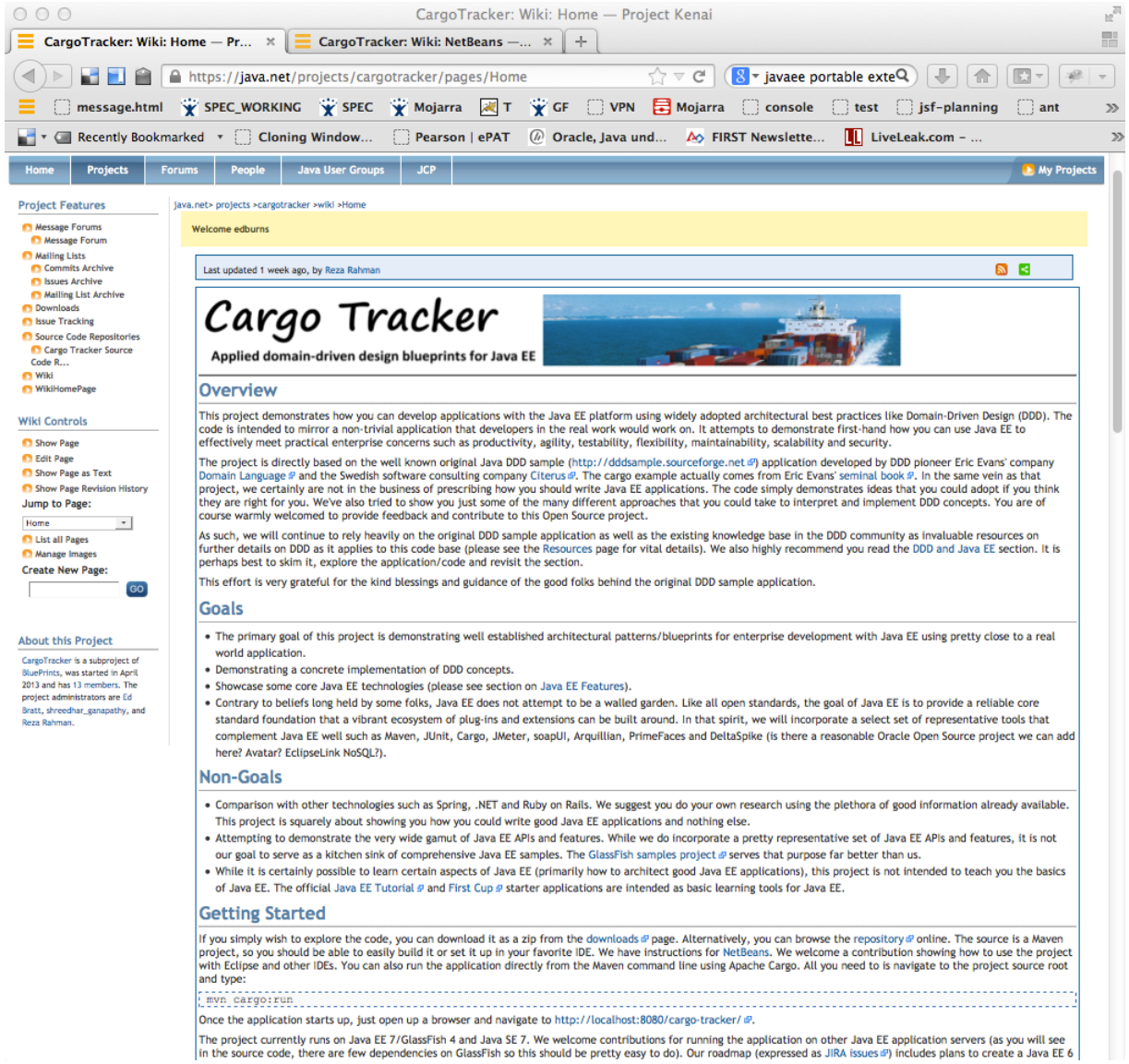
http://cargotracker.java.net

Java EE in Action



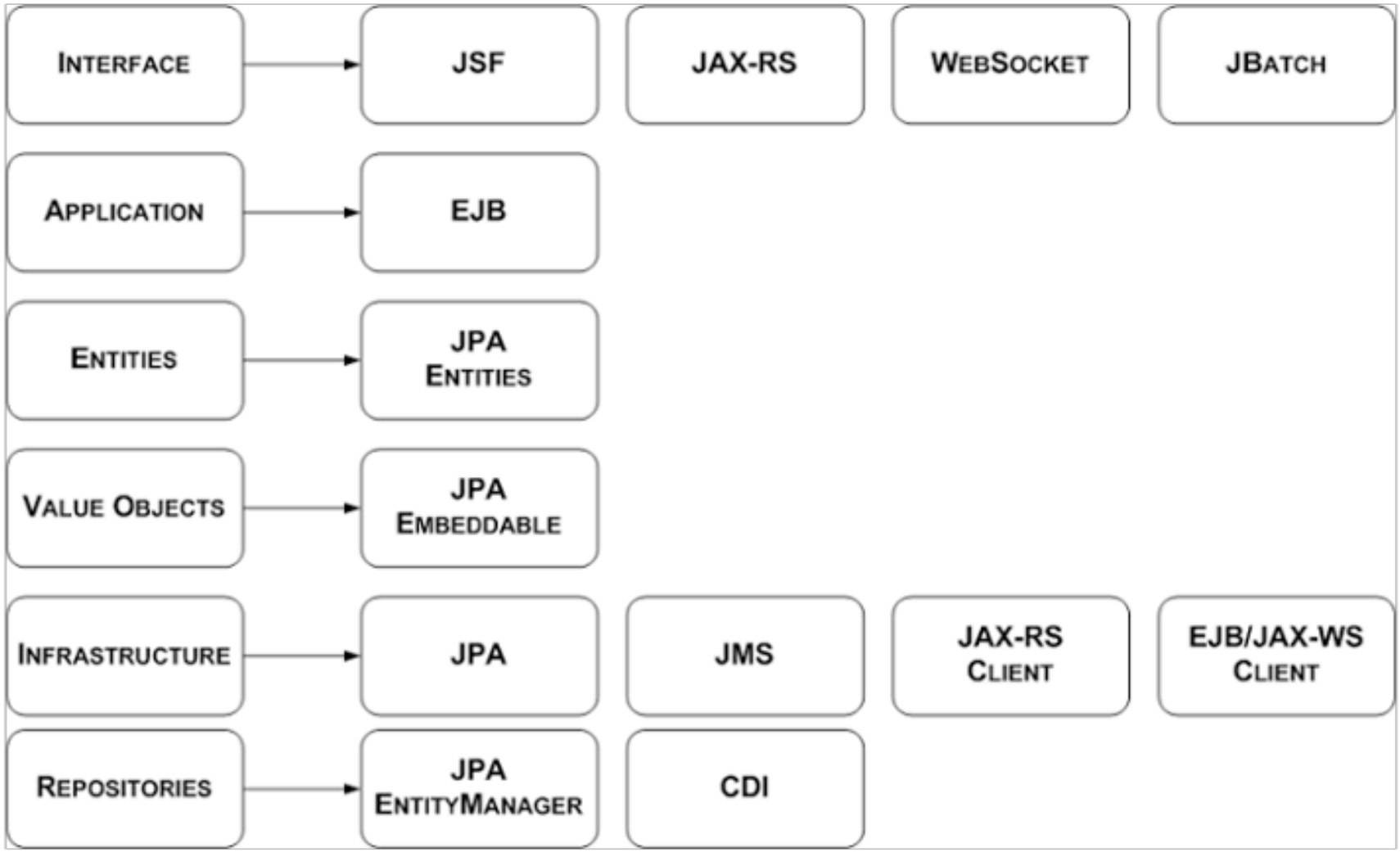
Cargo Tracker

Applied domain-driven design blueprints for Java EE



http://cargotracker.java.net

Java EE 7 in Action



Java EE 7 in Action

Components in this talk

- JSF 2.2
- Bean Validation 1.1
- JAX-RS 2.0
- WebSocket 1.0
- JSON-P 1.0

JSF 2.2 HTML5 Friendly Markup

The best part of Wicket comes to JSF

- This is a JSF page

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:myNS="http://xmlns.jcp.org/jsf”>
<form myNS:id="form">
    <input name="textField" type="text" myNS:value="#{bean.text1}" />
    <input type="submit" myNS:id="submitButton" value="submit" />
    <p>submitted text: #{bean.text1}.</p>
</form>
</html>
```

1090 HTML5 Friendly Markup

Let's get back to basics

- JSF Views are written in a View Declaration Language (VDL).
- The standard Facelet VDL is an XML application with two kinds of elements
  - HTML Markup
  - JSF Components
- HTML Markup is passed through straight to the browser
- JSF Components take some action on the server, during the lifecycle

1090 HTML5 Friendly Markup

Let the elegance of HTML shine through

- Before JSF 2.2
  - JSF tags hide complexity of underlying HTML+script+css+images
  - JSF “Renderer”:
    - encode: markup to browser
    - decode: name=value from browser

```
<html>…
  <my:colorPicker value="#{colorBean.color2}" />
  <my:calendar value="#{calendarBean.date1}" />
</html>
```

- Context: Missing feature in browser? Write a JSF component.

1090 HTML5 Friendly Markup

Let the elegance of HTML shine through

- With JSF 2.2
  - Pure HTML+script+css+images in the JSF page
  - JSF Renderer handles decode from browser
    - Leverage the strength of the JSF lifecycle
    - Leverage the expressiveness of HTML5

```
<html>…
  <input type="color" jsf:value="#{colorBean.color2}"/>
  <input type="date" jsf:value="#{calendarBean.date1}" />
</html>
```

- Context: New feature in browser? Use “pass through elements”

JSF 2.2 HTML5 Friendly Markup

Let the elegance of HTML shine through

- DEMO

Java EE 7 in Action

Components in this talk

- JSF 2.2
- Bean Validation 1.1
- JAX-RS 2.0
- WebSocket 1.0
- JSON-P 1.0

Bean Validation – History Lesson

- Just as JSF had managed beans and inversion of control years before CDI, JSF also had validation years before Bean Validation
- Task for JSF 2.0 was to seamlessly integrate Bean Validation into JSF
- Biggest hurdle: conceptual mismatch
  - When does validation occur?
    - JSF: before values are in the model
    - Bean Validation: after values are in the model (but before they are persisted via JPA)

Bean Validation – History Lesson

- Answer: define a standard JSF validator

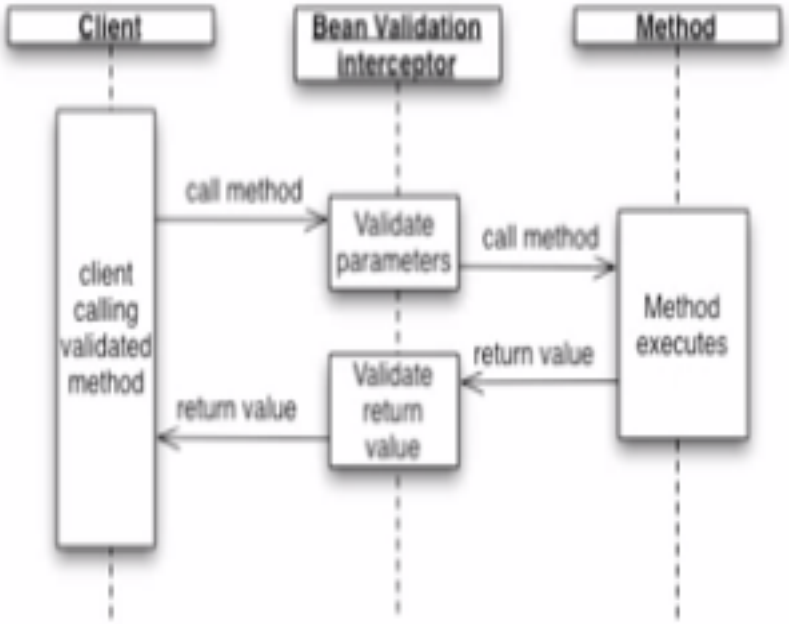
  `javax.faces.Bean`
- Its `validate()` method calls the a `validateValue()` method added to the Bean Validator API at the request of JSF.
- This allows the JSF validation reporting system to take action, while allowing Bean Validation be effective as well.

Bean Validation

- Write once, validate anywhere
- Defines validation behavior for use in all tiers of Java EE (and SE)
- Works well with JSF, JPA, and even Servlet

Bean Validation 1.1: New Features

- Approaches Programing By Contract (PBC)
  - Validates incoming parameters and outgoing return value
- Can inject with CDI into Validator implementations
- Better message interpolation



BeanValidation 1.1 PostCondition validation

Programming by Contract comes to Java EE

- DEMO

Java EE 7 in Action

Components in this talk

- JSF 2.2
- Bean Validation 1.1
- JAX-RS 2.0
- WebSocket 1.0
- JSON-P 1.0

JAX-RS 2.0

```
@Stateless // TODO Make this a stateless bean for better scalability.
@Path("/handling")
public class HandlingReportService {

    public static final String ISO_8601_FORMAT = "yyyy-MM-dd HH:mm";
    @Inject
    private ApplicationEvents applicationEvents;


    @POST
    @Path("/reports")
    @Consumes(MediaType.APPLICATION_JSON)
    // TODO Better exception handling.
    public void submitReport(@NotNull @Valid HandlingReport handlingReport) {
        try {
```

```
xhr.open("POST", "http://localhost:8080/cargo-tracker/rest/handling/reports", true);
xhr.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');
xhr.send(JSON.stringify(data));
xhr.onreadystatechange = update;
```

Java EE 7 in Action

Components in this talk

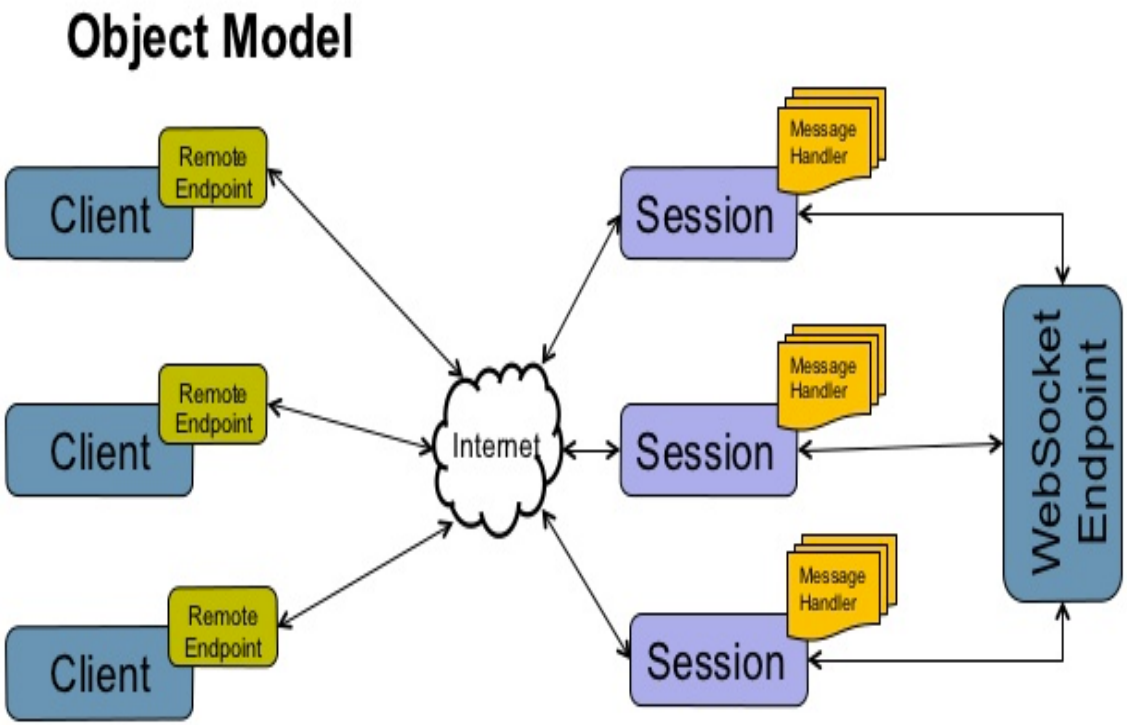
- JSF 2.2
- Bean Validation 1.1
- JAX-RS 2.0
- WebSocket 1.0
- JSON-P 1.0

WebSockets

What's In a name?

- Several moving parts, each with its own standard!
  - Client: W3C JavaScript API http://dev.w3.org/html5/websockets/
  - Transport: IETF RFC 6455 http://www.ietf.org/rfc/rfc6455.txt
  - Server: JSR-356: http://jcp.org/en/jsr/detail?id=356
- Even with all these parts, it's still very understandable
  - Client: 17 page downs
  - Transport: 86 page downs (about a third of which you can skip)
  - Server: 43 pages

WebSockets



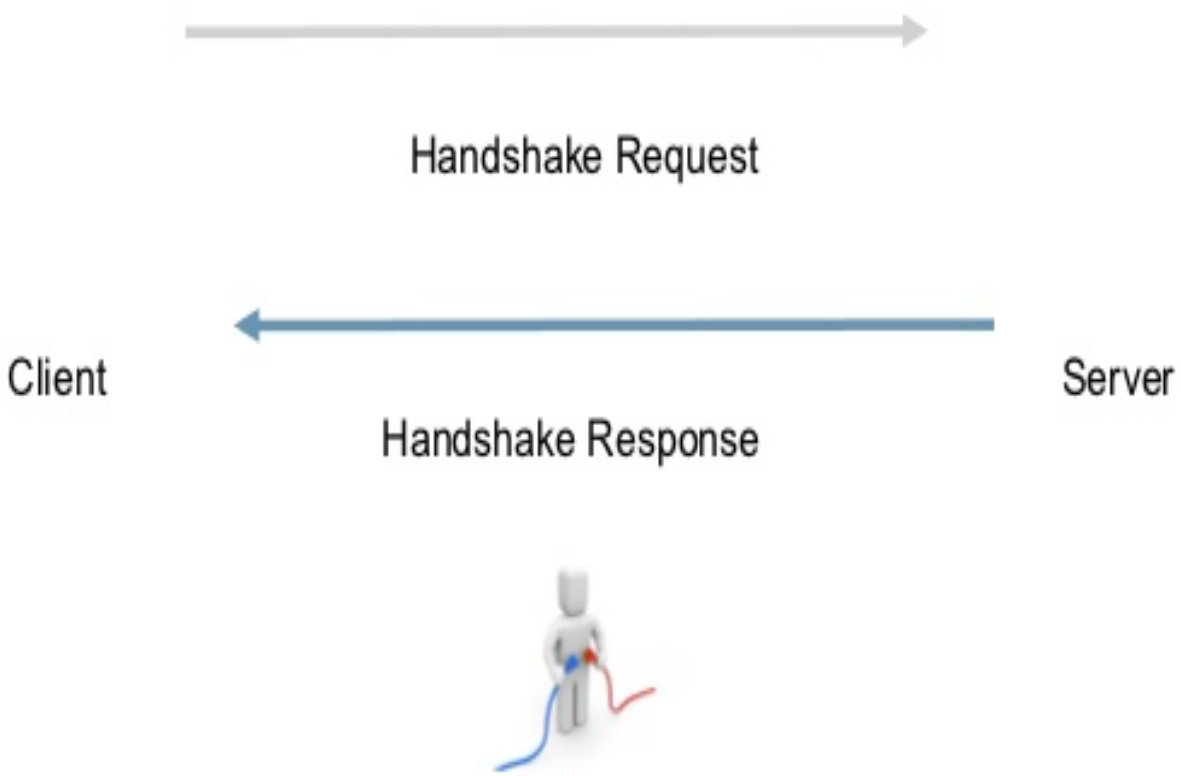
WebSocket

Big Picture

- It really is a Socket for the Web
  - It just works over proxies
- Lets you do TCP easily in a web app
  - Establish the connection
  - Send messages both ways
  - Send messages independent of timing
  - Close the connection
- Two basic types: text and binary

WebSocket

Establish the connection



WebSocket

TCP over HTTP, use the Upgrade: header

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

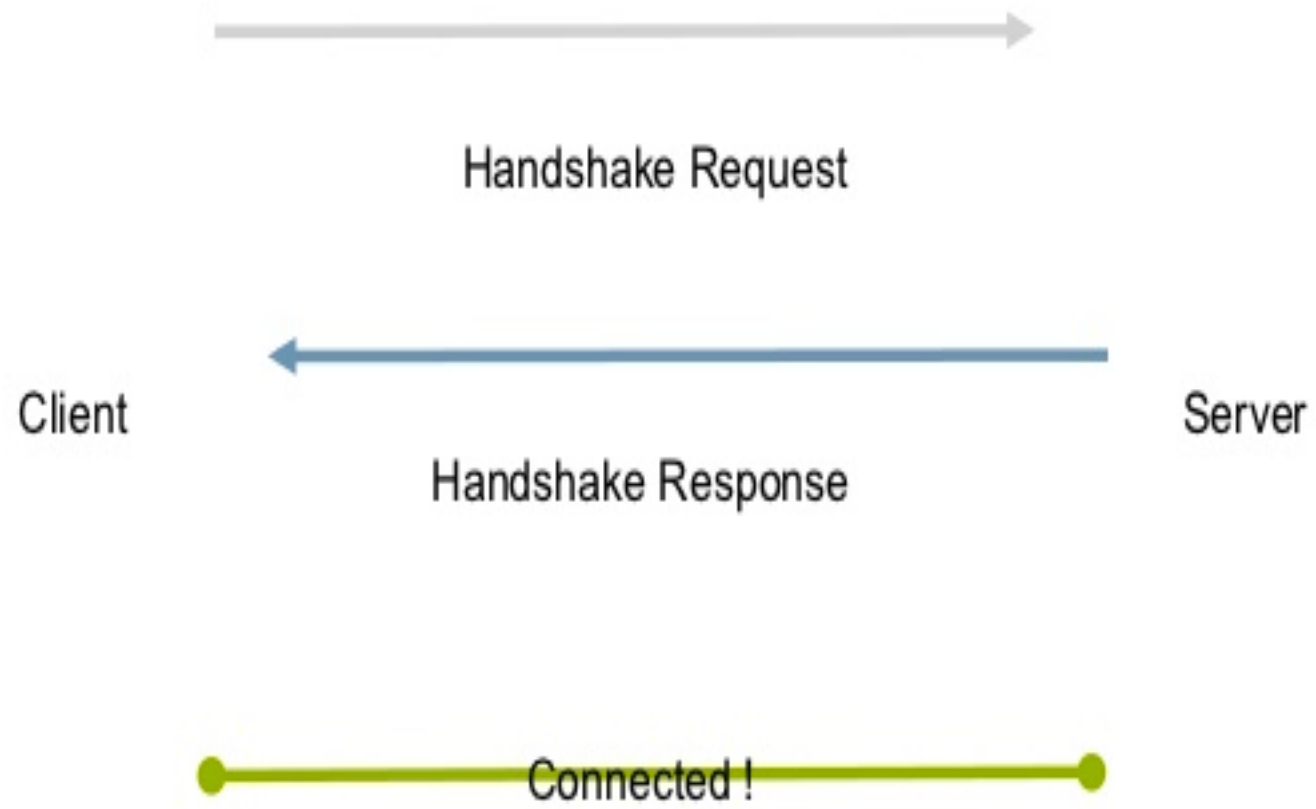
WebSocket

TCP over HTTP, use the Upgrade: header

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

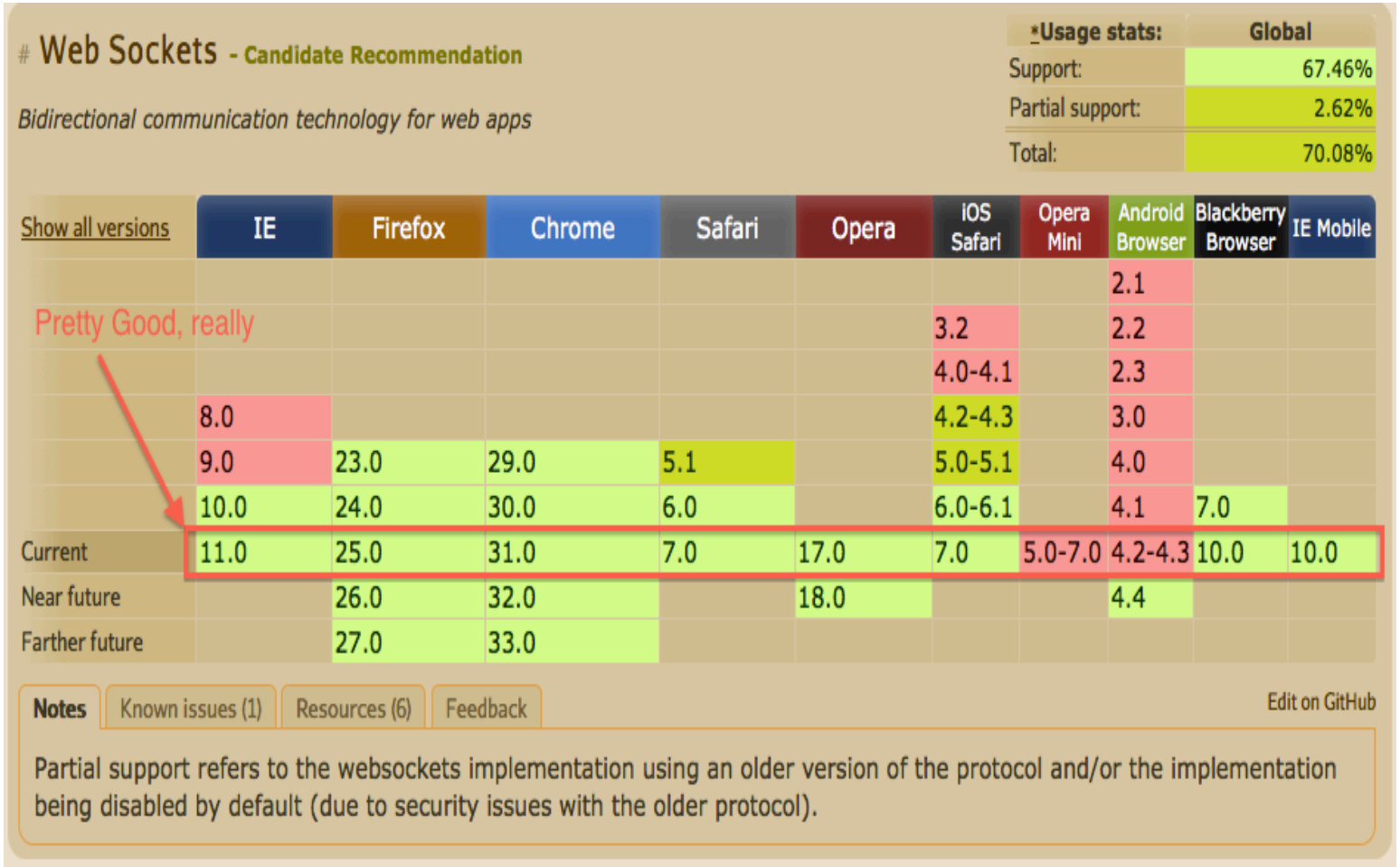
WebSocket

TCP over HTTP, use the Upgrade: header



WebSocket

Browser Support – caniuse.com



Web Sockets - Candidate Recommendation

Bidirectional communication technology for web apps

*Usage stats:	Global
Support:	67.46%
Partial support:	2.62%
Total:	70.08%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
Pretty Good, really						3.2		2.2		
						4.0-4.1		2.3		
	8.0					4.2-4.3		3.0		
	9.0	23.0	29.0	5.1		5.0-5.1		4.0		
	10.0	24.0	30.0	6.0		6.0-6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		
Farther future		27.0	33.0							

Notes | Known issues (1) | Resources (6) | Feedback

Edit on GitHub

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).

JavaScript/Browser Client

JavaScript WebSocket Object

- Minimal Lifecycle
  - `new WebSocket(url)`
  - pass in some functions
    - `onopen`
    - `onmessage`
  - call `send()`
  - call `close()`
- Can connect to arbitrary servers, other than the page origin
- Server may enforce use of `Origin` header

WebSocket

```
@Singleton
@ServerEndpoint("/tracking")
public class RealtimeCargoTrackingService {

    private static final Logger logger = Logger.getLogger(
            RealtimeCargoTrackingService.class.getName());

    private final Set<Session> sessions = new HashSet<>();

    @OnOpen
    public void onOpen(final Session session) {
        sessions.add(session);
    }

    @OnClose
    public void onClose(final Session session) {
        sessions.remove(session);
    }

    public void onCargoInspected(@Observes @CargoInspected Cargo cargo) {
        Writer writer = new StringWriter();
```

WebSocket and JSON

```
public void onCargoInspected(@Observes @CargoInspected Cargo cargo) {
    Writer writer = new StringWriter();

    try (JsonGenerator generator = Json.createGenerator(writer)) {
        generator
                .writeStartObject()
                .write("trackingId", cargo.getTrackingId().getIdString())
                .write("origin", cargo.getOrigin().getName())
                .write("destination", cargo.getRouteSpecification().getDestination().getName())
                .write("lastKnownLocation", cargo.getDelivery().getLastKnownLocation().getName())
                .write("transportStatus", cargo.getDelivery().getTransportStatus().toString())
                .writeEnd();
    }

    String jsonValue = writer.toString();

    for (Session session : sessions) {
        try {
            session.getBasicRemote().sendText(jsonValue);
        } catch (IOException ex) {
            logger.log(Level.WARNING, "Unable to publish WebSocket message", ex);
        }
    }
```

WebSocket

```
(function() {

    var ws = new WebSocket("ws://localhost:8080/cargo-tracker/tracking");
    ws.onopen = function(event) {
        onConnect(event)
    };
    ws.onmessage = function(event) {
        onMessage(event)
    };

    onConnect = function(event) {
    }

    onMessage = function(event) {
        populateListRouted(event);
    }

    populateListRouted = function(event) {

        var jsonObject = JSON.parse(event.data);
        var table = document.getElementById("listRoutedTab");

        for (var count = 1, row; row = table.rows[count]; count++) {
            if (row.id === jsonObject.trackingId) {
                row.cells[1].innerHTML = jsonObject.origin;
                row.cells[2].innerHTML = jsonObject.destination;
                row.cells[3].innerHTML = jsonObject.lastKnownLocation;
                row.cells[4].innerHTML = jsonObject.transportStatus;
                break;
            }
        }
    }
```

Java EE 7 in Action

Components in this talk

- JSF 2.2
- Bean Validation 1.1
- JAX-RS 2.0
- WebSocket 1.0
- JSON-P 1.0

There is much more in CargoTracker, to learn more, visit

http://cargotracker.java.net/

Java EE 8

- JSON-B
- JCache
- CDI 2
- More CDI/EJB alignment
- JMS.next()?
- Security
- Testability
- Cloud, PaaS, multitenancy/SaaS
- Action-oriented Web framework/HTML 5 alignment?
- NoSQL?
- Modularity?

http://glassfish.org/survey

Try it Out!

http://download.java.net/glassfish/4.0/release/glassfish-4.0.zip

Come to JavaLand in Brühl, Germany

http://javaland.eu/
~ 50% English Talks

Resources

- Java EE Tutorials
  - http://docs.oracle.com/javaee/7/tutorial/doc/home.htm
- Digging Deeper
  - http://docs.oracle.com/javaee/7/firstcup/doc/home.htm
  - https://glassfish.java.net/hol/
  - https://java.net/projects/cargotracker/
- Java EE 7 Transparent Expert Groups
  - http://javaee-spec.java.net
- Java EE 7 Reference Implementation
  - http://glassfish.org
- The Aquarium
  - http://blogs.oracle.com/theaquarium