InterConnect 2016
The Premier Cloud & Mobile Conference

Don’t Wait!

Develop responsive applications with Java EE7 instead!

Erin Schnabel
schnabel@us.ibm.com
@ebullientworks

IBM

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada

Focus on the server side

- “Responsive” application
- Maximize resource utilization
- Technologies in EE7 can help

JAX-RS 2.0 async processing

Concurrency Utilities (JSR-236)

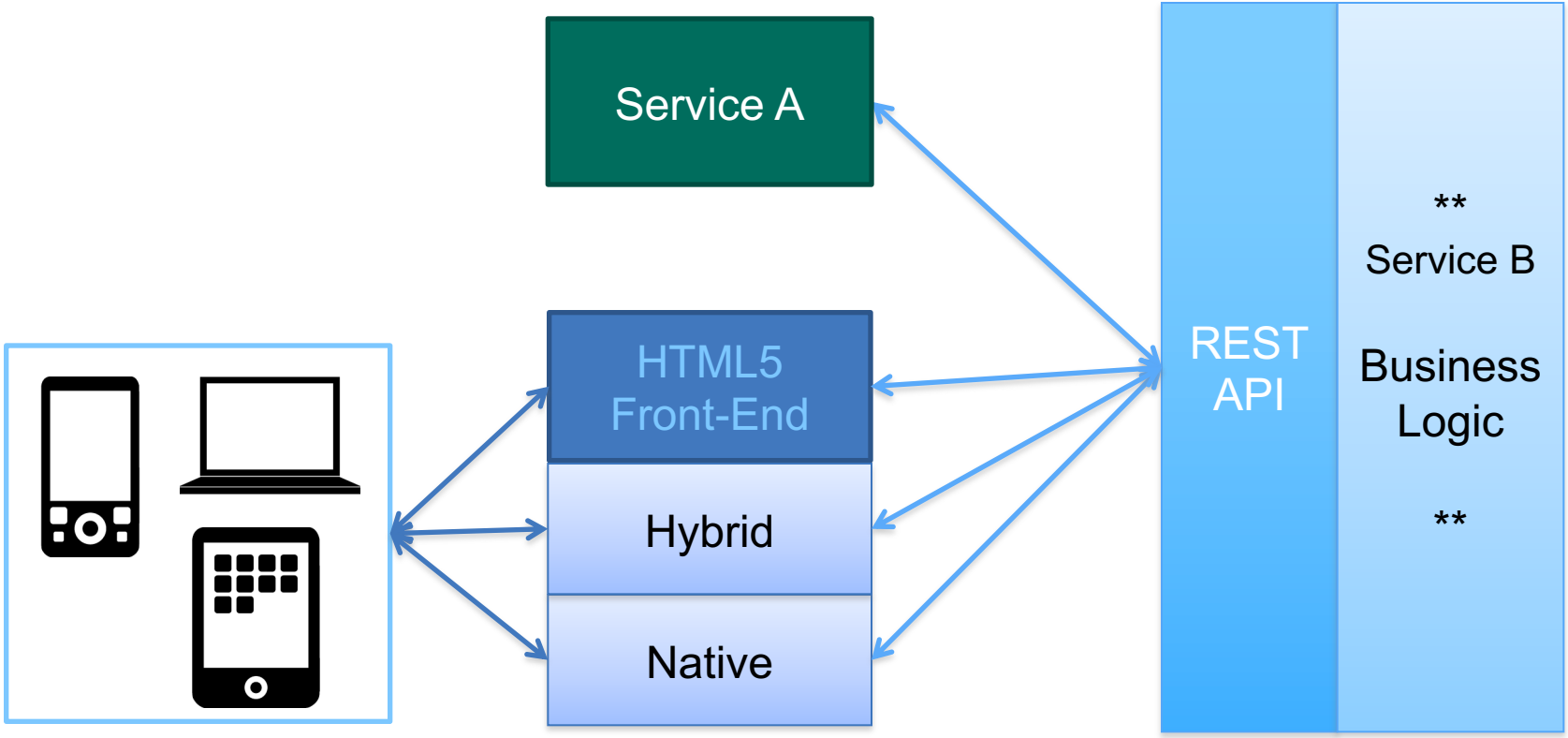
WebSocket API (JSR-356)

Non-blocking I/O added in Servlet 3.1

JAX-RS 2.0 Async Support

Concurrency Utilities

REST application (client-server or server-server)



A (very) simple JAX-RS example

```
@Path(”items")
public class ItemResource {

     // various ways to get this guy, play nice and assume we have one
     protected ItemService itemService;

     @GET
     @Produces(MediaType.APPLICATION_JSON)
     public Collection<Item> listItems() {
          return itemService.listItems();
     }

     @POST
     @Consumes(MediaType.APPLICATION_JSON)
     public void addItem(Item item) {
          itemService.addItem(item);
     }

    ...
}
```

A (very) simple JAX-RS example

```
@Path(”items")
public class ItemResource {

     // various ways to get this guy, play nice and ass
     protected ItemService itemService;

     @GET
     @Produces(MediaType.APPLICATION_JSON)
     public Collection<Item> listItems() {
          return itemService.listItems();
     }

     @POST
     @Consumes(MediaType.APPLICATION_JSON)
     public void addItem(Item item) {
          itemService.addItem(item);
     }

    ...
}
```

What if this is an encapsulation of a remote service?

JAX-RS 2.0 Asynchronous processing

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public void getItems(@Suspended final AsyncResponse ar) {

    Collection<Item> result = itemService.listItems();
    ar.resume(result );
}
```

- @Suspended annotation with AsyncResponse parameter
- void return type (to allow this method to return immediately)
    - “suspend” the connection -- NOT DISCONNECTED!
- AsyncResponse.resume(…) to send the response to the client.

JAX-RS 2.0 Asynchronous processing

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public void getItems(@Suspended final AsyncResponse ar) {

    Collection<Item> result = itemService.listItems();
    ar.resume(result );
}
```

Hmm…I don't see any threading stuff. How is this asynchronous?

- @Suspended annotation
- void return type (to allow th y)
  - "suspend" the connection -- NOT DISCONNECTED!
- AsyncResponse.resume(…) to send the response to the client.

JAX-RS 2.0 and …

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public void getItems(@Suspended final AsyncResponse ar) {
    Runnable r = () -> {
        Collection<Item> result = itemService.listItems();
        Response resp = Response.ok(result).build();
        ar.resume(resp);
    };
    Executors.newSingleThreadExecutor().submit(r);
}
```

This would totally work:
The long-running operation is definitely on a different thread.

Hopefully we also know that this is a horrible idea!

#notcontainerfriendly

JAX-RS 2.0 and EJB

- For a local EJB, use the @Asynchronous method annotation.

```
@Stateless
@Path("ejbitems")
public class ItemEJBResource {
    ...
    @GET
    @Asynchronous
    @Produces(MediaType.APPLICATION_JSON)
    public void getItems(@Suspended final AsyncResponse ar) {
        Collection<Item> result = itemService.listItems();
        Response resp = Response.ok(result).build();
        ar.resume(resp);
    }
    ...
}
```

The EJB Container will dispatch to a different thread before invoking the method.

JAX-RS 2.0 and Concurrency Utilities

```
@Path("execitems")
public class ItemsExecutorResource {

    private ExecutorService getExecutor() throws NamingException {
        return (ExecutorService) new InitialContext()
                .lookup("java:comp/DefaultManagedExecutorService");
    }

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public void getItems(@Suspended final AsyncResponse ar) {
        Runnable r = () -> {...};

        try {
            ExecutorService executor = getExecutor();
            executor.submit(r);
        } catch (NamingException e) {
            r.run();
        }
    }
    ...
}
```

JNDI lookup for managed executor #oldschool

JAX-RS 2.0, CDI, and Concurrency Utilities

- Enable CDI (beans.xml), and have an Executor provided

```
@Path("cdiexecitems")
public class ItemsCDIExecutorResource {
    ...
    @Resource
    ManagedExecutorService executor;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public void getItems(@Suspended final AsyncResponse ar) {
        Runnable r = () -> {
            Collection<Item> result = itemService.listItems();
            Response resp = Response.ok(result).build();
            ar.resume(resp);
        };

        executor.submit(r);
    }
}
```

JAX-RS 2.0: Time out!

```
@Path("cdiexecitemsTimeout")
public class ItemsCDIExecutorResourceTimeout {
    ...
    public void getItems(@Suspended final AsyncResponse ar) {

        ar.setTimeout(500, TimeUnit.MILLISECONDS);
        ar.setTimeoutHandler(new TimeoutHandler() {
            public void handleTimeout(AsyncResponse arg0) {
                ar.resume(Response.ok("Backup plan!").build());
            }
        });

        Runnable r = () -> {
            Collection<Item> result = itemService.listItems();
            Response resp = Response.ok(result).build();
            ar.resume(resp);
        };

        executor.submit(r);
    }
}
```

JAX-RS 2.0: Callbacks

- Register callbacks on AsyncResponse:

```
ar.register(new CompletionCallback() {
        @Override
        public void onComplete(Throwable t) {
            System.out.println("DONE! ");
            if ( t != null ) {
                t.printStackTrace();
            }
        }
    });

    ar.register(new ConnectionCallback() {
        @Override
        public void onDisconnect(AsyncResponse ar) {
            System.out.println("Disconnected: " + ar);
        }
    });
```

Support for ConnectionCallback is optional. #YMMV

Concurrency Utilities

- Extension of java.util.concurrent (familiar API)
  - ManagedExecutorService
    `java:comp/DefaultManagedExecutorService`
  - ManagedScheduledExecutorService
    `java:comp/DefaultManagedScheduledExecutorService`
  - ManagedThreadFactory
    `java:comp/DefaultManagedThreadFactory`
  - ContextService
    `java:comp/DefaultContextService`
- Container-friendly mechanisms to queue and manage work
  - Java EE Context propagation
    - JNDI, classloader, security, etc.
  - Allows container to manage async work, too!

InterConnect 2016
The Premier Cloud & Mobile Conference

WebSocket APIs

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada

Server Endpoint: Annotated

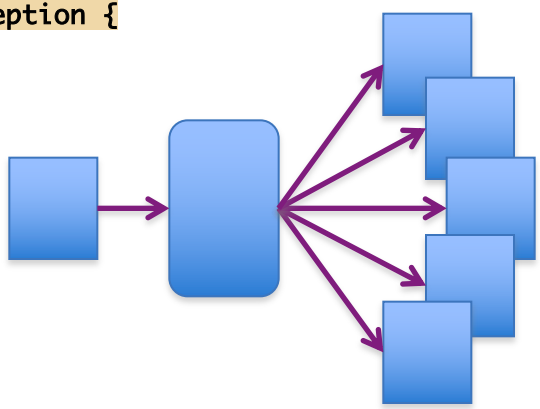
- Simple POJO with `@ServerEndpoint` annotation
- Annotations for notifications: lifecycle and messages

```
@ServerEndpoint(value = "/SimpleAnnotated")
public class SimpleEndpoint {

    @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) {
    }
}
```

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 message, Session session) throws IOException {
    if ( "stop".equals(message) ) {
        session.close();
    } else {
        int id = count++;
        for (Session s : session.getOpenSessions() ) {
            s.getBasicRemote().sendText("Echo " + id + ":  " + message);
        }
    }
}
```



- Broadcast – iterate over open sessions

InterConnect 2016
The Premier Cloud & Mobile Conference

Non-blocking I/O

Servlet 3.1

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada

Non-blocking I/O in Servlet 3.1

- Builds on Asynchronous processing from Servlet 3.0
  - Only works for async-enabled apps
- Enables reading/writing data without blocking a thread
  - ReadListener on ServletInputStream
  - WriteListener on ServletOutputStream
- Best used with slow clients
  - For clients that are slow to read data, use a WriteListener
  - For clients that are slow to write data, use a ReadListener

Code Samples – Aync Servlet Listener

```
// start async
AsyncContext ac = request.startAsync();

// set up async listener
ac.addListener(new AsyncListener() {
    @Override
    public void onComplete(AsyncEvent event) throws IOException {
        System.out.println("AsyncServlet onComplete() called");
    }

    @Override
    public void onError(AsyncEvent event) {
        System.out.println("AsyncServlet onError() " + event.getThrowable());
    }

    @Override
    public void onStartAsync(AsyncEvent event) {
        System.out.println("AsyncServlet onStartAsync()");
    }

    @Override
    public void onTimeout(AsyncEvent event) {
        System.out.println("AsyncServlet onTimeout()");
    }
}, request, response);
```

Code Samples – Non-Blocking I/O: Input

```
final ServletInputStream is = request.getInputStream();

// Start NIO Mode!! May only be called once…
// Can not use regular servlet input stream read/write after this
is.setReadListener(new AsyncReadListener(...));


public class AsyncReadListener implements ReadListener {
    ...
    @Override
    public void onDataAvailable() throws IOException {
        System.out.println("AsyncReadListener: data available ");
    }

    @Override
    public void onAllDataRead() throws IOException {
        System.out.println("AsyncReadListener: All data read.. ");
    }

    @Override
    public void onError(Throwable t) {
        System.out.println("AsyncReadListener onError() " + t);
        t.printStackTrace();
        ac.complete();
    }
}
```

Code Samples – Non-Blocking I/O: Output

```
final ServletOutputStream os = response.getOutputStream();

// Start NIO Mode!! May only be called once…
// Can not use regular servlet input stream read/write after this
os.setWriteListener(new AsyncWriteListener(...));

public class AsyncWriteListener implements WriteListener {

    @Override
    public void onWritePossible() throws IOException {
        System.out.println("AsyncWriteListener: onWritePossible.. ");

        // At some point, you know you're all done..
        ac.complete();
    }

    @Override
    public void onError(Throwable t) {
        System.out.println("AsyncWriteListener onError() " + t);
        t.printStackTrace();
        ac.complete();
    }
}
```

InterConnect 2016
The Premier Cloud & Mobile Conference

RxJava + EE7

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada

Reactive Programming

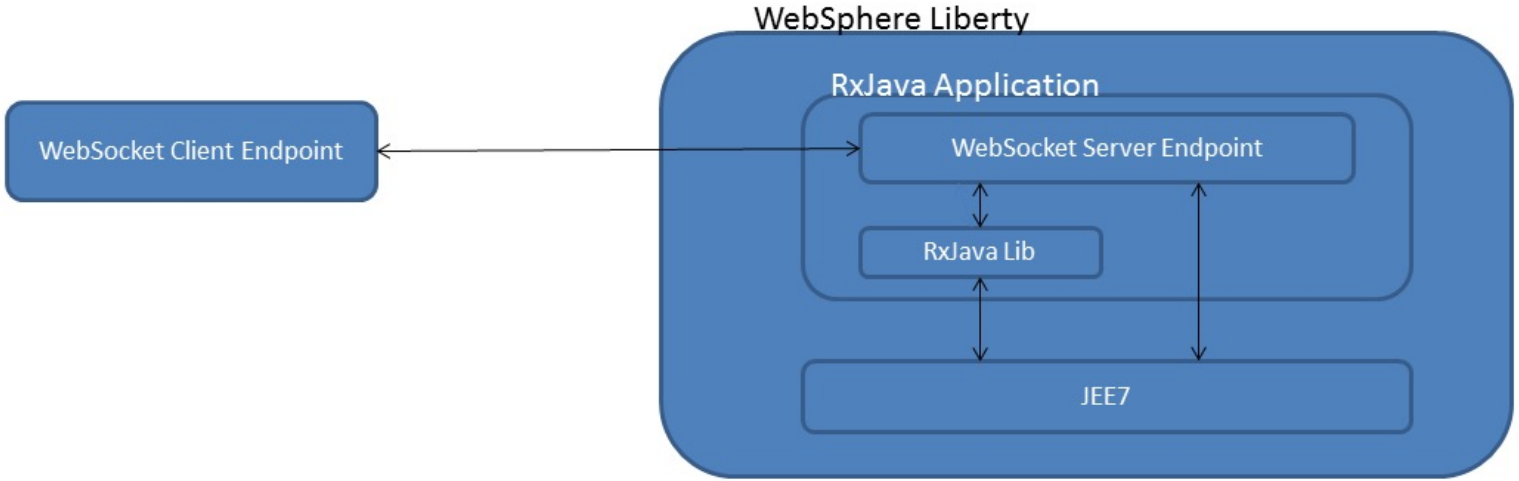
- Reactive programming focuses on data flows and propagation of change.
- Java 8 has introduced lambdas and a new *java.util.stream* package, which help reactive programmers get started.
- For highly concurrent stream processing and more advanced scenarios, ReactiveX is the library of choice.
- RxJava is the Java implementation of ReactiveX.

RxJava and JEE7

- RxJava and EE7 go hand-in-hand:
  - Inject (CDI) and ExecutorServices from Concurrency Utilities into your app
  - Encapsulate these concurrent artifacts with a RxJava Scheduler
  - Use Websockets to provide real-time updates as observables are completed

RxJava and Liberty

- Applications deployed in Liberty can include the RxJava library and seamlessly tap into its EE7 framework

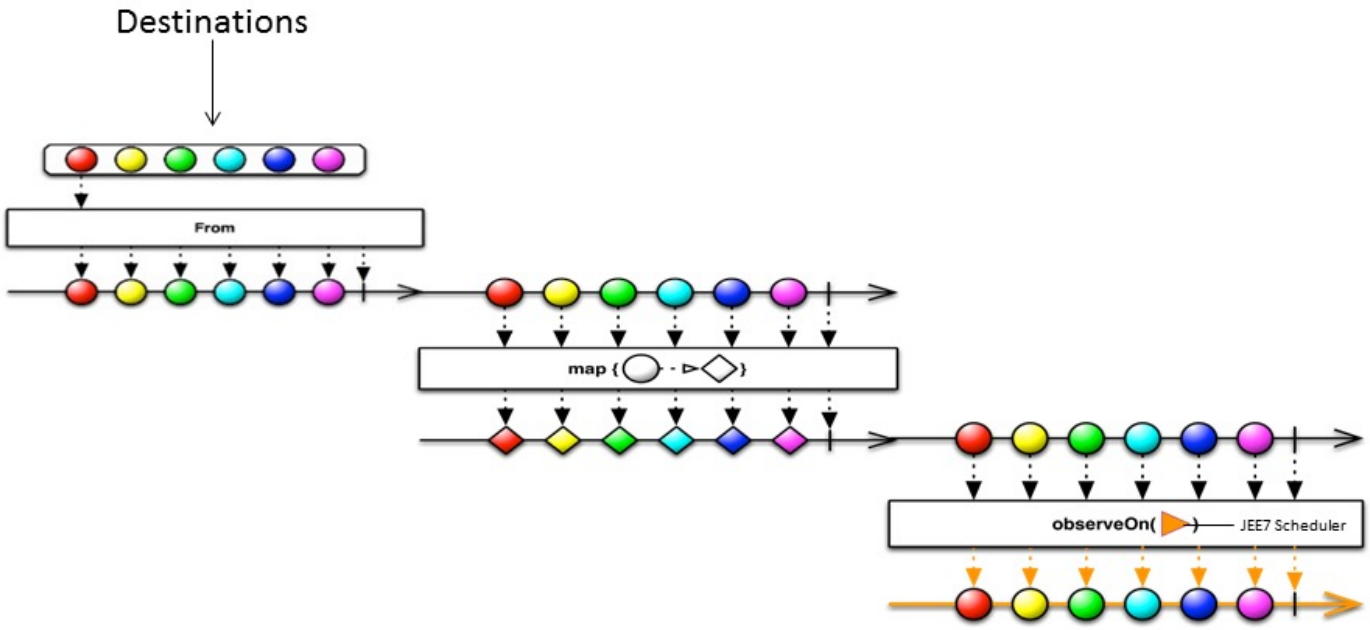


RxJava and Liberty

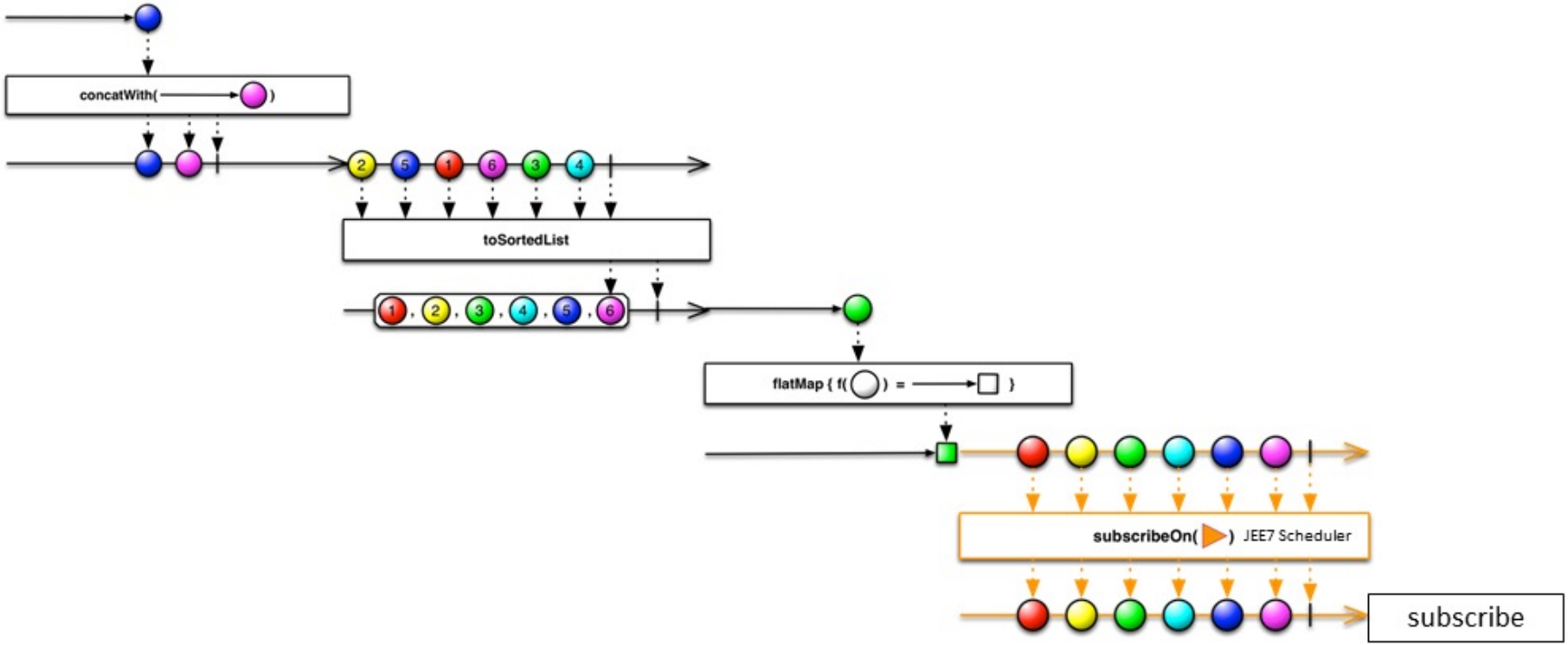
- GitHub sample: https://github.com/WASdev/sample.rxjava

- App provides reviews and weather information about vacation destinations
  - concurrent-1.0
  - websocket-1.1
  - cdi-1.2

RxJava and Liberty



RxJava and Liberty



Notices and Disclaimers

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law

Notices and Disclaimers Con't.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained h erein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

InterConnect 2016
The Premier Cloud & Mobile Conference

Thank You

Your Feedback is Important!

Access the InterConnect 2016 Conference Attendee Portal to complete your session surveys from your smartphone,

laptop or conference kiosk.

IBM

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada