

#ibminterconnect

Don't Wait! Develop Responsive Applications with Java EE7 Instead

Session 3085

Erin Schnabel
Liberty Profile Development Lead, IBM
@ebullientworks



InterConnect2015

The Premier Cloud & Mobile Conference

February 22 – 26

MGM Grand & Mandalay Bay | Las Vegas, Nevada



Focus on the server side



- “Responsive” application
 - Throughput is king (or queen)
 - Should be able to handle as many requests as we can!
- Maximize resource utilization
 - Make the best use of allocated resources
 - Take advantage of tiered services
- 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

#ibminterconnect

JAX-RS 2.0
Async support
Concurrency Utilities

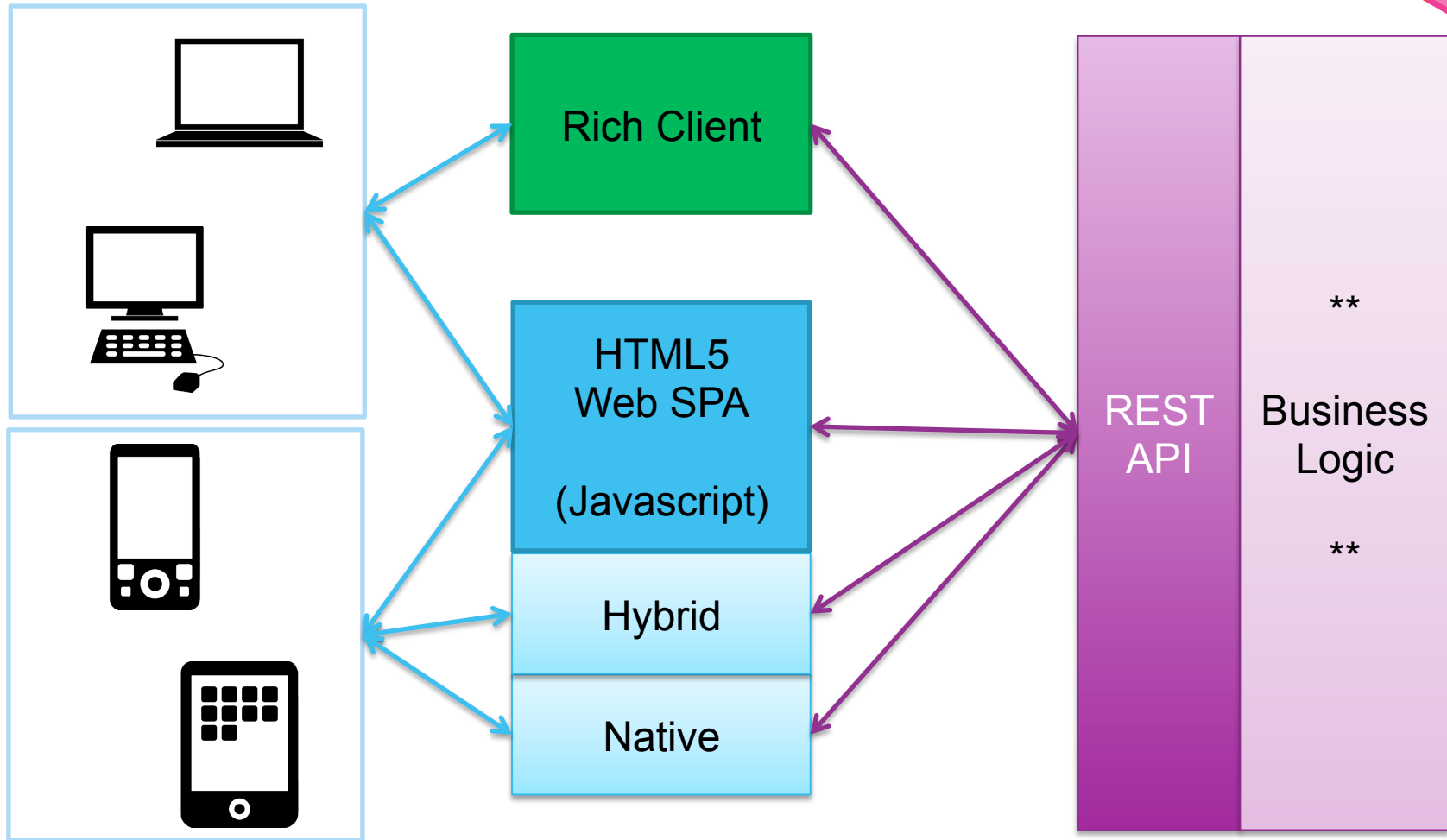


InterConnect2015

The Premier Cloud & Mobile Conference



“Modern Web Application” -- REST





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
    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?

A (very) simple JAX-RS example



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

    // various ways to get this guy, play nice
    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?

Not much changes from the client point of view: they will make a request and wait for the response either way...



A (very) simple JAX-RS example

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

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

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

    ...
}
```

What if this is an encapsulation of a remote service?

Not much changes from the client point of view: they will make a request and wait for the response either way...

If the remote service operation takes awhile, this server thread...

... just sits there...

... waiting...

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 );
}
```

Good catch!
So far, all we've achieved is parity
with what we had before...

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

Choices, choices.

How to queue long-running
work onto a different thread?

- `AsyncResponse.resume(...)`

ately)

client.

JAX-RS 2.0 and....



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

    Executors.newSingleThreadExecutor().submit(new Runnable() {
        Collection<Item> result = itemService.listItems();
        Response resp = Response.ok(result).build();
        ar.resume(resp);
    });
}
```

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 an 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 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) {
        executor.submit(new Runnable() {
            public void run() {
                Collection<Item> result = itemService.listItems();
                Response resp = Response.ok(result).build();
                ar.resume(resp);
            }
        });
    }
}
```

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");
    }
}
```

JNDI lookup for
managed executor

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

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

```
...
}
```

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());
            }
        });

        executor.submit(new Runnable() {
            public void run() {
                Collection<Item> result = itemService.listItems();
                Response resp = Response.ok(result).build();
                ar.resume(resp);
            }
        });
    }
}
```

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);
    }
});
```

- ConnectionCallback is optionally supported, 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!

#ibminterconnect

Non-blocking I/O Servlet 3.1



InterConnect2015

The Premier Cloud & Mobile Conference



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 – Async 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

```
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



```
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();
    }
}
```

#ibminterconnect

WebSocket API



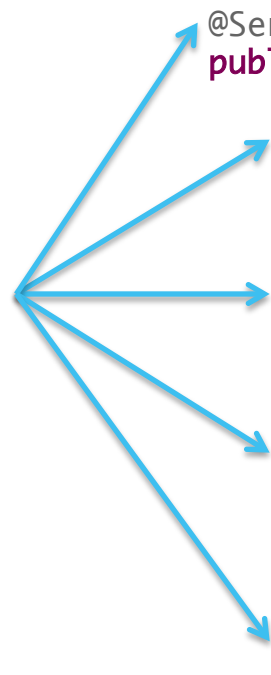
InterConnect2015

The Premier Cloud & Mobile Conference



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 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) {
    }
}
```




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(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 t) {  
    }  
}
```

Simple echo + server provided data

(using annotations)



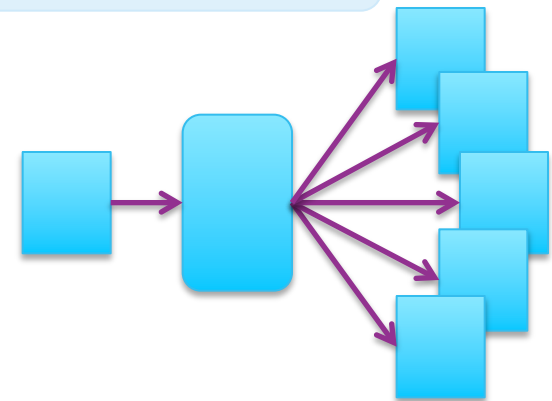
- `@OnMessage` method is called when a message is received
 - If message is 'stop': close the session
 - Otherwise, echo the message along with a hit count

```
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



Simple echo + delay...



```
@OnMessage
public void receiveMessage(final String message, final Session session)
throws IOException {
    if ( "stop".equals(message) ) {
        session.close();
    } else {
        System.out.println(message + ", " + executor);
        final int id = count++;
        broadcast(session, "Echo " + id + ": " + message);

        executor.submit(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                }
                broadcast(session, "Delayed " + id + ": " + message);
                System.out.println("executor -- send " + message);
            }
        });
    }
}
```

Notices and Disclaimers



Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

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.



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 herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

- IBM, the IBM logo, ibm.com, Bluemix, Blueworks Live, CICS, Clearcase, DOORS®, Enterprise Document Management System™, Global Business Services®, Global Technology Services®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, SoDA, SPSS, StoredIQ, Tivoli®, Trusteer®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

#ibminterconnect

Thank You

Your Feedback is
Important!

Access the InterConnect 2015
Conference CONNECT Attendee
Portal to complete your session
surveys from your smartphone,
laptop or conference kiosk.



InterConnect2015

The Premier Cloud & Mobile Conference

