JUDCon

JBoss Users & Developers Conference

Boston : 2010

Next Gen. Web Apps with GWT & JBoss

Mike Brock (cbrock@redhat.com)

The Browser is a Platform!

Beyond Hypertext

- Web browsers now have very fast and very usable JavaScript engines where you can get real work done.
- The emergence of HTML5 makes the browser a full platform.

The Thick-Thin Client

- Rich browser apps aren't really thin apps.
- Most rich browser apps are actually larger bytewise than many "thick" apps were when the term "thin client' was coined.
- HTML5 makes the browser a distribution platform for thick apps.

Rich Browser Apps

- Hypertext templating languages made UI programming quite easy and accessible. But static HTML is not very interactive.
- The complexity that rich clients brings means better user experience, but a nightmare for programmers.

GWT

GWT

- **Java to JavaScript Compiler**
  - You program your UI code in Java. You almost never need to touch JavaScript, HTML or any other web technology.
- **A JRE Emulation Library**
  - Supports the java.lang.*, java.util.* java.text.*.
  - Does not support reflection.
- **Works with your favourite IDE and tools (Ant, Maven, etc.)**

GWT

- GWT is a UI framework, isn't it? Why do you need a "GWT Framework"?
  - GWT provides very basic nuts and bolts capabilities out of the box.
  - GWT lends itself very well to framework development, and it's screaming for a good framework.

What is Errai?

- An end-to-end client/server framework.
- Transparent push messaging into the browser.
- Shared clientnt & server APIs
- An optional RAD framework.
- Integration with JMS, and Seam CDI.

Why do we need another web framework?

Another Framework?

- Rich client browser frameworks are still in their infancy.
- There's plenty of gaps to be filled to make developers more productive leveraging the capabilities of modern browsers.
- Errai fills many of these gaps and more.

Problems solved by Errai

- GWT is statically linked. Errai's bus technology can overcome this and provide runtime federation inside the browser.
- Moving beyond request/response. Sending messages to and from the server is a single unified API.
- Massively asynchronous applications can be hard to balance in the browser. Errai manages all conversations through a common transport.

Introducing ErraiBus

ErraiBus

- The "backbone" of the Errai Framework.
- Designed around the idea of a peered-architecture (or "flat hierarchy") instead of client-server.
- Very low learning curve. Straight forward Pub/Sub approach. Automatic resource management/ cleanup.
- Uses declarative APIs (annotations) and fluent interfaces.
- Object serialization.
- Simple conversations.
- Simplified RPC mechanism for type-safe calls from client-to-server. (Server -to-client coming soon)

Why a proprietary transport? Why not Bayeaux?

- A standardized wire protocol offers **no advantage** when the server provides the client code for every session.
- The Bayeaux specification is very complex and overly thought out.
- Errai can generate a custom protocol for handling model at compile-time for higher efficiency.

ErraiBus

Sending a message

```
MessageBuilder.createMessage()
        .toSubject("MySubject")
        .command("SayHello")
        .noErrorHandling().sendNowWith(dispatcher);
```

Receiving a message

```
bus.subscribe("MySubject", new MessageCallback() {
    public void callback(Message message) {
        if ("SayHello".equals(message.getCommandType())) {
            System.out.println("Hello!");
        }
    }
});
```

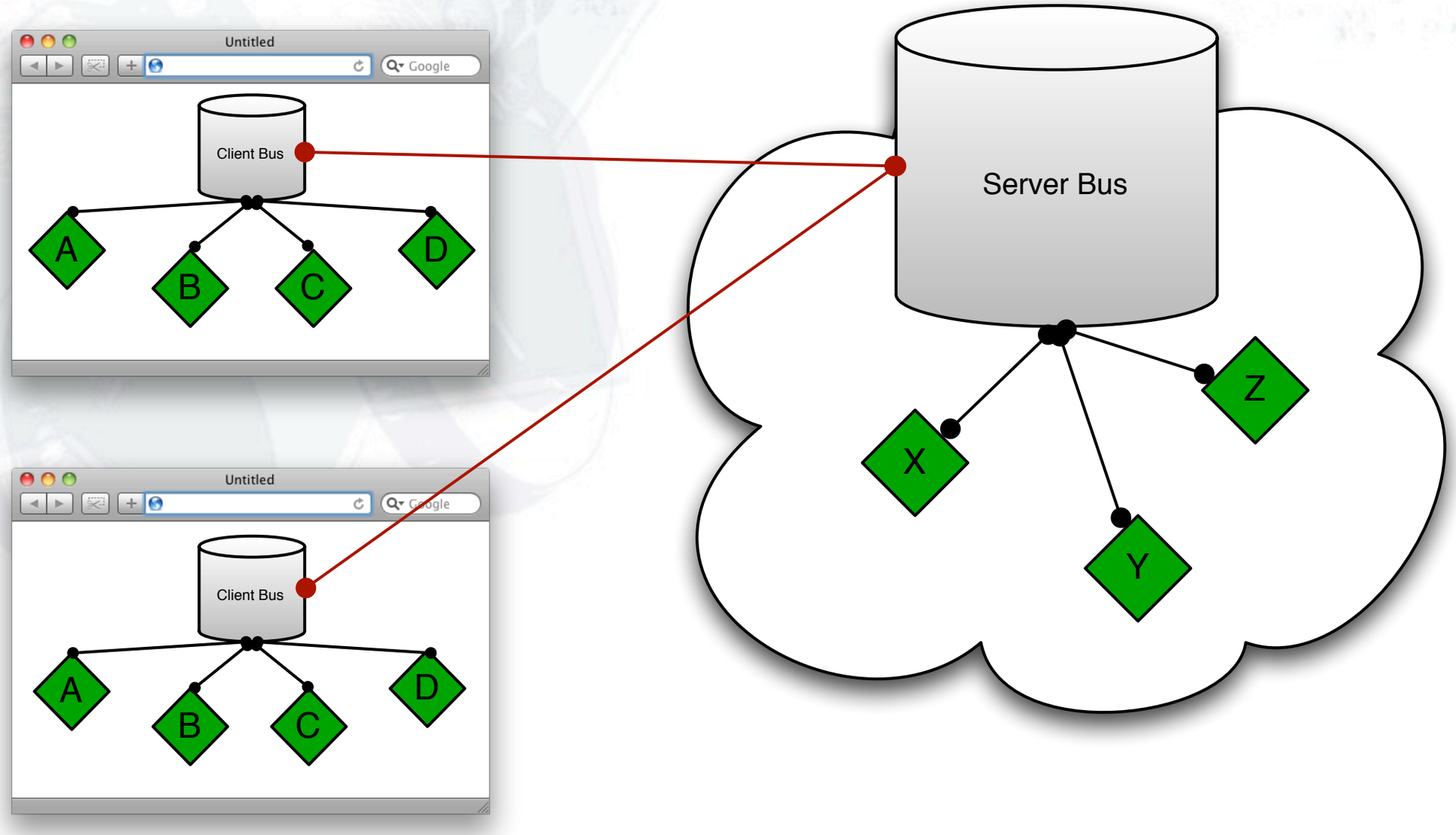
ErraiBus

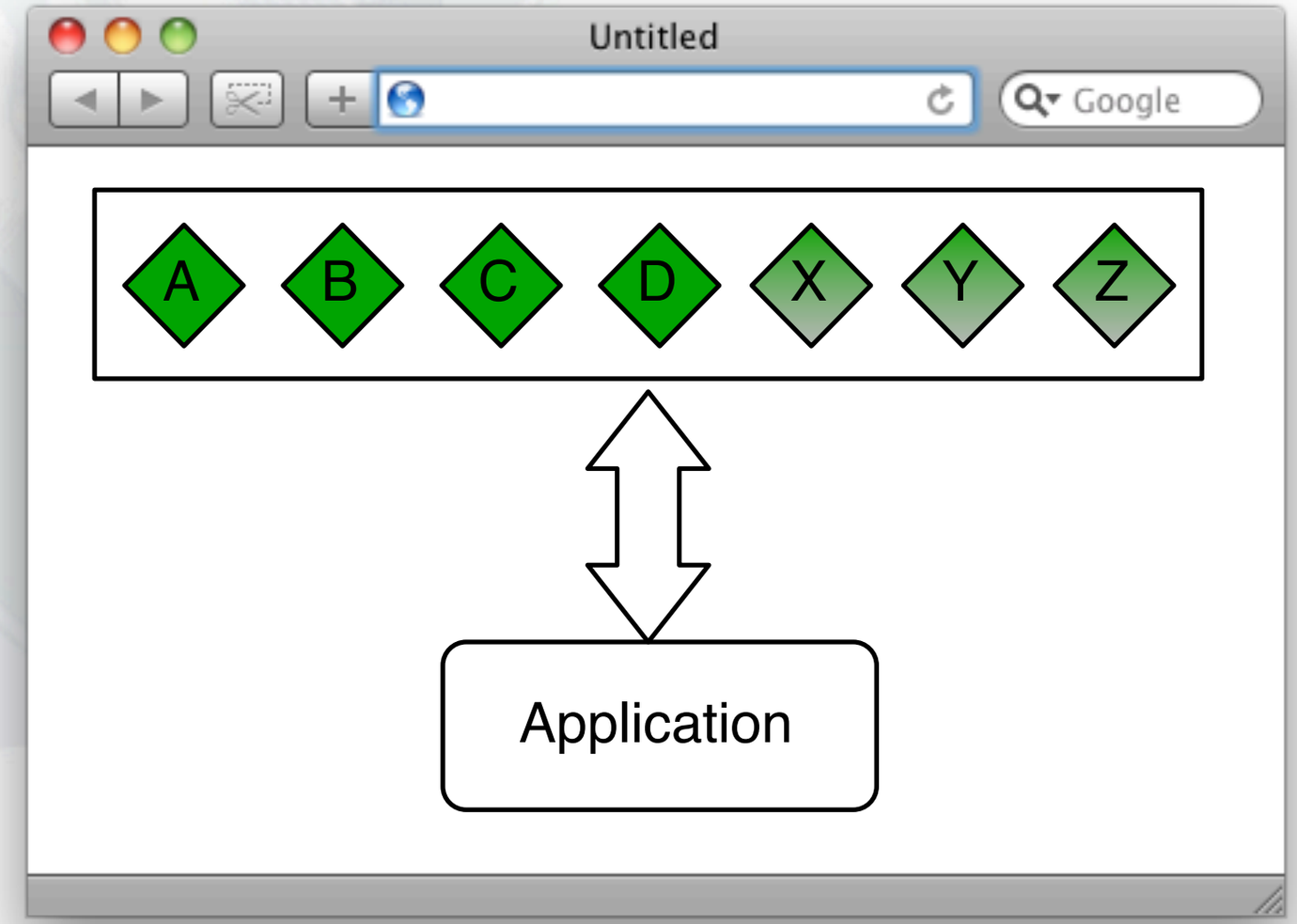
Declarative Approach

```
@Service
public class MySubject implements MessageCallback {
    public void callback(Message message) {
        if ("SayHello".equals(message.getCommandType())) {
            System.out.println("Hello!");
        }
    }
}
```

Demo!

"Hello, World!"


Untitled
Google
Client Bus
A
B
C
D
Untitled
Google
Client Bus
A
B
C
D
Server Bus
X
Y
Z


Untitled
Google
A
B
C
D
X
Y
Z
Application

- Single, flat, endpoint namespace across the browser and the server.
- Conversations provide implicit session isolation.
- No fuss in-client, client-to-server, server-to-client messaging.

Naturally, this has implications ...

Advantages

- Single client and server-side programming model. (Server code, calling client code is the same as vice-versa -- almost).
- No need to wire-up XML or other config files to make it work. Everything is auto-discovered at compile and runtime.
- Push is free!

Demo!

Browser Push!

Java EE

Weld Integration

- Integrates with CDI Event Subsystem
- Call CDI Managed Beans directly via Errai's RPC Mechanism

Weld Integration

- Most low level use cases (Errai Bus, Pub/Sub)
- Similar to MessageDriven Bean (actually maps to it)
- Service implementation fully managed by CDI container
- Message Bus instance injected

GWT Client

```
MessageBuilder.createMessage()
    .toSubject("calculator")
    .signalling()
    .with(MessageParts.ReplyTo, "calculationResult")
    .with("a", Double.valueOf(a.getText()))
    .with("b", Double.valueOf(b.getText()))
    .noErrorHandling()
    .sendNowWith(bus);
```

CDI Component

```
@Service("calculator")
@ApplicationScoped
public class CalculatorService implements MessageCallback
{

  private static final Logger log =
      LoggerFactory.getLogger(CalculatorService.class);

  @Inject
  MessageBus bus;

  @Inject
  Calculator calculator;

  public void callback(Message message)
  {...}
}
```

Weld Integration

Client

```
CDI.handleEvent(Fraud.class, new EventHandler<Fraud>() {
    @Override
    public void handleEvent(Fraud event) {
        responsePanel.setText("Fraud detected: " + event.getTimestamp());
    }
});
```

Server

```
@ApplicationScoped
public class AccountService {
    @Inject @Any
    Event<Outbound> event;

    public void watchActivity(@Observes @Inbound AccountActivity activity) {
        Fraud payload = new Fraud(System.currentTimeMillis());
        event.fire(new Outbound(payload));
    }
}
```

Demo!

CDI Events

JMS

JMS Integration

- Directly subscribe to a JMS queues over the bus.

Demo!

Sending and Receiving a Message to and from JMS

Thanks!

Questions?

Mike Brock (cbrock@redhat.com)

http://www.jboss.org/errai/