Java EE 7 and HTML5: Developing for the Cloud

Arun Gupta, Java EE & GlassFish Guy
blogs.oracle.com/arungupta, @arungupta
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.
Java EE 6 – Key Statistics

• 40+ Million Java EE 6 Component Downloads
• #1 Choice for Enterprise Developers
• #1 Application Development Platform
• Fastest implementation of a Java EE release
Top Ten Features in Java EE 6

1. EJB packaging in a WAR
2. Type-safe dependency injection
3. Optional web.xml
4. CDI Events
5. JSF standardizing on Facelets
6. EJBContainer API
7. @Schedule
8. EJB No Interface View
9. Servlet and CDI extension points
10. Web Profile
Today’s Cloud Offerings are all Proprietary
We’re moving Java EE into the Cloud!
Java EE 7 and 8 – Focus Areas

**Cloud**
- Provisioning
- Elastic & Autonomic Scalability
- Multi-Tenancy

**Modularity**
- Building on Jigsaw
- Focus on OSGi interop
- Supporting Profiles & Modular Applications

**HTML5**
- Emerging Web Standards require a programming model
- JSON, WebSockets, off-line, APIs & DOM
Roles

- Developer
- PaaS Provider
- Customer/Tenant
- Deployer
- Application Submitter
- Application Administrator
- Tenant 1
- Tenant 2
- Tenant 3
- PaaS Account Manager
- PaaS Administrator

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.
Existing Java EE model

- Configure Java EE resources – JDBC, JMS etc
- Deploy Application EAR

1. Provision and Initialize
2. Provision and Initialize
3. Provision and Initialize
4. Provision and Initialize
5. Deploy Application (EAR/GAR/SAR …)
Java EE 7 Model: Auto-Provision Services from Application Dependencies

• Provision and deploy application resources (e.g. LDAP stripe, data source instantiation and connection …)

• Extensible Deployment Models Supporting Multiple Frameworks
  • Spring, Seam, Play …
Services

• Cloud apps consume services
  – Persistence, queueing, mail, caching, …

• Service metadata facilitates ease of use when deploying into the cloud

```java
@DataSourceDefinition(
    name="java:app/jdbc/myDB",
    className="oracle.jdbc.pool.OracleDataSource",
    isolationLevel=TRANSACTION_REPEATABLE_READ,
    initialPoolSize=5
)
```
Services

• Cloud apps consume services
  – Persistence, queueing, mail, caching, …

• Service metadata facilitates ease of use when deploying into the cloud

  @JMSConnectionFactoryDefinition(
      name="java:app/myJMSConnectionFactory",
      className="javax.jms.QueueConnectionFactory",
      resourceAdapterName="myJMSRA")

  @JMSDestinationDefinition(
      name="java:app/myQueue",
      className="javax.jms.Queue",
      destinationName="myQueue1")
Elasticity

- Service Levels
- Minimum and Maximum Instances
- Futures – Self Adjustment, Capacity On Demand
Demo

PaaSing a Java EE Application in the Cloud

glassfish.org/javaone2011
Java EE Application

JSF

JPA

EJB

Services Metadata

Conference Planning in the Cloud

<glassfish-services>
  <service-description init-type="LR" name="ConferencePlanner-1b">
    <template id="LBNative"/>
    <configurations>
      <configuration name="https-port" value="50443"/>
      <configuration name="ssl-enabled" value="false"/>
      <configuration name="http-port" value="50080"/>
    </configurations></service-description>
  <service-description init-type="JavaEE" name="ConferencePlanner">
    <characteristics>
      <characteristic name="service-type" value="JavaEE"/>
    </characteristics>
    <configurations>
      <configuration name="max.clustersize" value="4"/>
      <configuration name="min.clustersize" value="2"/>
    </configurations>
  </service-description>
</glassfish-services>
Service Provisioning
Java EE 7 – Candidate JSRs

- CDI Extensions
- Web Container Extensions
- JAX-RS 2.0
- JSF 2.2
- JSP 2.2
- EL 3.0
- Servlet 3.1
- CDI 1.1 / Interceptors 1.1 / JSR 250 1.1
- Managed Beans 1.0
- EJB 3.2
- JPA 2.1
- JTA 1.1
- JMS 2.0
- Bean Validation 1.1
- Jcache 1.0 (JSR 107)
- Concurrency Utilities 1.0 (JSR 236)
- State Management 1.0 (JSR 350)
- Batch Processing 1.0 (JSR 352)
- JSON 1.0 (JSR 353)
• Requires Java SE 7
• Added new roles
• Resource definition metadata
  – DataSourceDefinition, JMSCoreConnectionFactoryDefinition, JMSDestinationDefinition, MailSessionDefinition, ConnectorResourceDefinition
  – Pre-provisioned or shared resource
Java EE 7 Early Draft

• Default Data Source java:comp/defaultDataSource
• Default JMS Connection Factory
  java:comp/defaultJMSConnectionFactory
• Tenant Identifier: java:comp/tenantId
• Made optional
  – EJB Entity Beans, EJB QL, JAX-RPC, Deployment API, JAXR
Stored Procedures

```java
@NamedStoredProcedureQuery(name="topGiftsStoredProcedure",
procedureName="Top10Gifts")
public class Product {
  ...
}

EntityManager.createQuery("topGiftsStoredProcedure");
query.setParameter(1, "top10");
query.setParameter(2, 100);
// there are other setParameter methods for defining the temporal type
...
query.execute();
String response = query.getOutputParameterValue(1);
```
JAX-RS 2.0

Client-side API

```java
URL url = new URL("http://.../atm/balance");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setDoInput(true);
conn.setDoOutput(false);
BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
String line;
while ((line = br.readLine()) != null) {
    out.println(line);
}
Client client = ClientFactory.newClient();
String balance = client.target("http://.../atm/balance")
    .request()
    .get(String.class);
```
@Provider
class LoggingFilter implements RequestFilter, ResponseFilter {
    @Override
    public FilterAction preFilter(FilterContext ctx) throws IOException {
        logRequest(ctx.getRequest());
        return FilterAction.NEXT;
    }

    @Override
    public FilterAction postFilter(FilterContext ctx) throws IOException {
        logResponse(ctx.getResponse());
        return FilterAction.NEXT;
    }
}
JAX-RS 2.0
Bean Validation Integration

@Path("/")
class ProductResource {
   @POST
   @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
   public void addProduct(@NotNull
                           @FormParam("productName") String name,
                           @NotNull
                           @Category
                           @FormParam("category") String category) {
      ...
   }
}
@Resource(lookup = "jms/connectionFactory ")
ConnectionFactory connectionFactory;

@Resource(lookup="jms/inboundQueue")
Queue inboundQueue;

public void sendMessageOld (String payload) throws JMSException {
try (Connection connection = connectionFactory.createConnection()) {
    Session session = connection.createSession();
    MessageProducer messageProducer = session.createProducer(inboundQueue);
    TextMessage textMessage = session.createTextMessage(payload);
    messageProducer.send(textMessage);
}
}
JMS 2.0 – Sending a Message (Java EE)
New Simplified API (with Injection)

```java
@Inject
@JMSConnectionFactory("jms/contextFactory")
JMSSystemContext context;

@Resource(mappedName="jms/inboundQueue")
Queue inboundQueue;

public void sendMessage(String payload) {
    context.send(inboundQueue,payload);
}
```
JSON 1.0 (JSR 353)

javax.json.*

- DOM-based APIs (javax.json.* package)
  - JSONBuilder – Builds a JSON object
  - JSONReader – Reads a JSON object or array from the stream
  - JSONWriter – Writes a JSON object or array to the stream

- Streaming APIs (javax.json.stream.* package)
  - JsonGenerator – Streaming JSON generator
  - JsonParser – Allows forward, read-only access to JSON
JsonObject value = new JsonBuilder()
    .beginObject()
    .add("firstName", "John")
    .add("lastName", "Smith")
    .add("age", 25)
    .beginObject("address")
    .add("streetAddress", "21 2nd Street")
    .add("city", "New York")
    .add("state", "NY")
    .add("postalCode", "10021")
    .endObject()
    .beginArray("phoneNumber")
    .beginObject()
    .add("type", "home")
    .add("number", "212 555-1234")
    .endObject()
    .beginObject()
    .add("type", "fax")
    .add("number", "646 555-4567")
    .endObject()
    .endArray()
    .endObject()
    .build();

{ "firstName": "John", "lastName": "Smith", "age": 25, 
  "address": {
    "streetAddress": "21 2nd Street", 
    "city": "New York", "state": "NY", 
    "postalCode": "10021"
  }, 
  "phoneNumber": [
    { "type": "home", 
      "number": "212 555-1234" }, 
    { "type": "fax", 
      "number": "646 555-4567" } 
  ] 
}
Transparency

- Oracle’s Java EE 7 JSRs are run in the open on java.net
  - http://javaee-spec.java.net
  - One project per spec – e.g., jpa-spec, jax-rs-spec, jms-spec…
- Publicly viewable Expert Group mail archive
  - Users observer list gets copies of all Expert Group emails
- Publicly viewable download area
- Publicly viewable issue tracker
- Commitment to update to JCP 2.8 Process
Status and Schedule

• All JSRs up and running

• Early Drafts
  – Java EE 7 Platform, JSF 2.2, JAX-RS 2.0, JPA 2.1, CDI 1.1, EJB 3.2, JMS 2.0, Bean Validation 1.1, Expression Language 3.0

• Final release target: Q2 2013
Java EE: Investments for HTML5

- WebSocket and Server Sent Events Infrastructure
- WebSocket SDK and JSR-356
  - Simplify the programming model for Java EE
WebSocket

• Stand-alone Specification
  – Earlier part of HTML 5 Specification
• Bi-directional
• Full-duplex
• Over a single TCP connection
• Examples
  – Real-time online games
  – Collaborative platforms
  – Social Networking
WebSocket Protocol – Handshake

GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMBDL1EzLkh9GBhXDW==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
<EMPTY LINE>

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMIYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
<EMPTY LINE>
WebSocket API

dev.w3.org/html5/websockets/

```javascript
[Constructor(DOMString url, optional (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
    readonly attribute DOMString url;

    // ready state
    const unsigned short CONNECTING = 0;
    const unsigned short OPEN = 1;
    const unsigned short CLOSING = 2;
    const unsigned short CLOSED = 3;
    readonly attribute unsigned short readyState;
    readonly attribute unsigned long bufferedAmount;

    // networking
    [TreatNonCallableAsNull] attribute Function? onopen;
    [TreatNonCallableAsNull] attribute Function? onerror;
    [TreatNonCallableAsNull] attribute Function? onclose;
    readonly attribute DOMString extensions;
    readonly attribute DOMString protocol;
    void close([Clamp] optional unsigned short code, optional DOMString reason);

    // messaging
    [TreatNonCallableAsNull] attribute Function? onmessage;
        attribute DOMString binaryType;
    void send(DOMString data);
    void send(ArrayBufferView data);
    void send(Blob data);
}
```
WebSocket SDK
websocket-sdk.java.net

- Annotation Driven Model
  - Allows you to turn POJOs into WebSocket Endpoints
  - Support for Server Sent Events
  - WebLogic 12.1.2 support
  - Runs on GlassFish 4.0 Early Builds Now!

- WebSocket SDK and JSR-356
  - Simplify the programming model for Java EE
  - Integrated with Java EE Platform e.g. CDI
WebSocket Example

```java
import org.glassfish.websocket.api.annotations.WebSocketClose;
import org.glassfish.websocket.api.annotations.WebSocket;
import org.glassfish.websocket.api.annotations.WebSocketMessageType;

@WebSocket(path="/HelloWorld")
public class HelloWorld{

    @WebSocketMessage
    public String sayHelloWorld(String caller) {
        return "hello " + caller + "!";
    }

    @WebSocketClose
    public void goodbye() {
        System.out.println("Adios");
    }
}
```

https://blogs.oracle.com/arungupta/entry/websockets_and_java_ee_7
Server-Sent Events

• Part of HTML5 Specification
• Server-push notifications
• Cross-browser JavaScript API: EventSource
• Message callbacks
• MIME type: text/eventstream
EventSource API

dev.w3.org/html5/eventsource/

```
[Constructor(DOMString url, optional EventSourceInit eventSourceInitDict)]
interface EventSource : EventTarget {
    readonly attribute DOMString url;
    readonly attribute boolean withCredentials;

    // ready state
    const unsigned short CONNECTING = 0;
    const unsigned short OPEN = 1;
    const unsigned short CLOSED = 2;
    readonly attribute unsigned short readyState;

    // networking
    [TreatNonCallableAsNull] attribute Function? onopen;
    [TreatNonCallableAsNull] attribute Function? onmessage;
    [TreatNonCallableAsNull] attribute Function? onerror;
    void close();
};

dictionary EventSourceInit {
    boolean withCredentials = false;
};
```
Server-Sent Events Example
Client-side

```javascript
var url = 'http://' + document.location.host + '/glassfish-sse/simple';
eventSource = new EventSource(url);
eventSource.onmessage = function (event) {
    var theParagraph = document.createElement('p');
    theParagraph.innerHTML = event.data.toString();
    document.body.appendChild(theParagraph);
}
```

https://blogs.oracle.com/arungupta/entry/server_sent_events_using_glassfish
Server-Sent Events Example
Server-side Handler

```java
@ServerSentEvent("/simple")
public class MySimpleHandler extends ServerSentEventHandler {

    public void sendMessage(String data) {
        try {
            connection.sendMessage(data);
        } catch (IOException ex) {
            . . .
        }
    }
}
```

https://blogs.oracle.com/arungupta/entry/server_sent_events_using_glassfish
Server-Sent Events Example

Server-side Business Logic

```java
@Stateless
class SimpleEvent {

    @Inject @ServerSentEventContext("/simple")
    ServerSentEventHandlerContext<MySimpleHandler> simpleHandlers;

    @Schedule(hour="*", minute="*", second="*/10")
    public void sendDate() {
        for(MySimpleHandler handler : simpleHandlers.getHandlers()) {
            handler.sendMessage(new Date().toString());
        }
    }
}
```

https://blogs.oracle.com/arungupta/entry/server_sent_events_using_glassfish
Project Avatar
Complete Solution for Dynamic Rich Clients

HTML 5 browser

HTML & Java hybrid application

Java application

JSON over WebSocket

Java EE Cloud
JPA-RS: Building Block of Project Avatar

• Exposes JPA mapped entities over REST via JAX-RS
• HTTP Methods
  – GET, PUT, POST, DELETE
• Content-Type and Accept-based content negotiation
  – XML or JSON
• Client
  – HTML5 with JavaScript (primary focus)
  – JavaFX
JPA-RS: Current Programming Model

GET http://…/order/4

Customer  Product  Order

Shop Persistence Unit

JAX-RS

JPA
JPA-RS: Thin Server Architecture

GET http://…/<pu-name>/<entity>/4
JPA-RS: Building Block of Project Avatar

Server-side Business Logic

• Persistence Unit Operations
  – /<root-uri>/<pu-name>/entity
  – /<root-uri>/<pu-name>/query
  – /<root-uri>/<pu-name>/metadata

• Supports invocation of @NamedQueries via HTTP

• Server-caching – EclipseLink clustered cache

• Dynamic Persistence also supported
  – Entities defined via metadata – no Java classes required
  – Enables persistence for HTML5/JavaScript apps
NoSQL Stores

• No standards
• Differing APIs and feature sets
• Some offer query language/API – some not
EclipseLink NoSQL

• Support JPA access to NoSQL databases
• Define annotations and XML to identify NoSQL stored entities (e.g., @NoSQL)
• Support JPQL subset for each
  – Key principal: leverage what’s available
• Initial support for MongoDB and Oracle NoSQL.
• Support mixing relational and non-relational data in single composite persistence unit
GlassFish Roadmap

GlassFish v3
- Java EE 6 support
- Single instance
- GlassFish Enterprise Mgr

GlassFish Server 3.0.1
- Oracle branding
- Oracle platform support
- Oracle interoperability

2009

GlassFish Server 3.1
- Centralized administration
- Clustering / HA
- GlassFish Server Control

2010

GlassFish Server 3.1.1
- Bug fixes
- Updated components
- Incremental features

2011

GlassFish Server 3.1.2
- Bug Fixes
- Incremental features

2012

GlassFish Server 4
- Java EE 7
- Multitenancy
- PaaS-enablement

2013
Call to Action

• Java EE 7 Expert Group Project
  – http://javaee-spec.java.net

• Java EE 7 Reference Implementation
  – http://glassfish.org

• The Aquarium
  – http://blogs.oracle.com/theaquarium