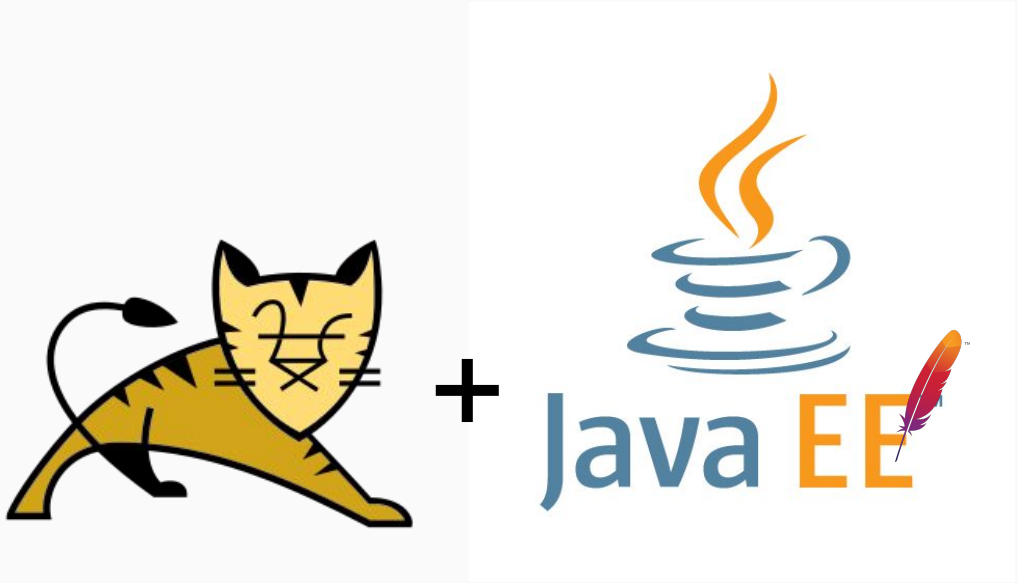
ASF ❤ Tomcat and EE:
→ from Meecrowave to TomEE

Romain Manni-Bucau

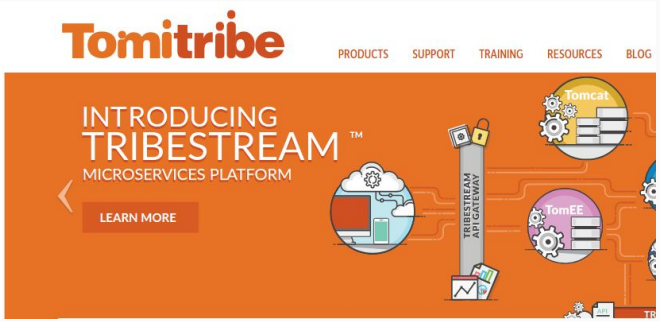
ApacheCon NA 2017

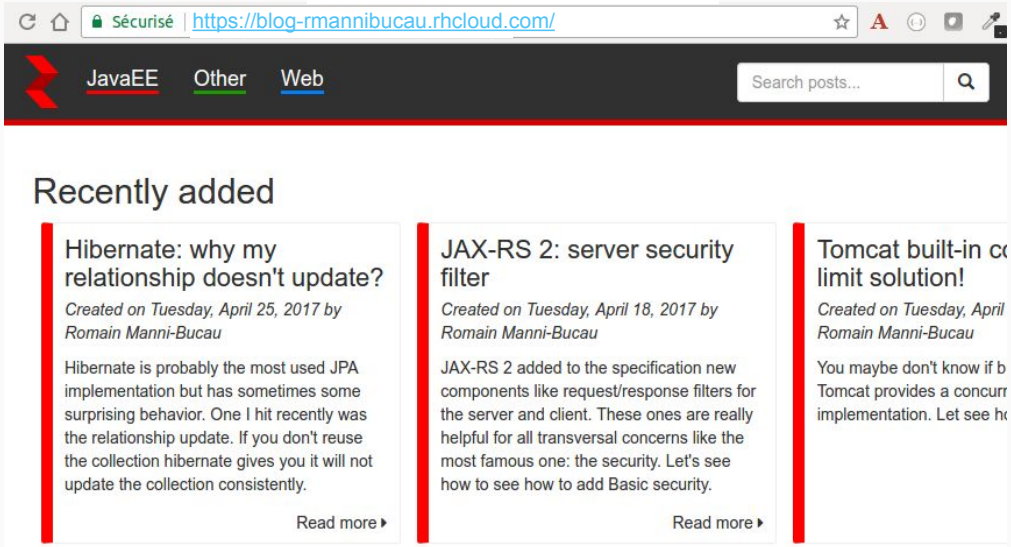
Agenda



Who am I?

Romain Manni-Bucau
@rmannibucau





ASF Projects around EE

APACHE BVal™

CXF

Meecrowave

TomEE

OpenWebBeans

OpenJPA

APACHE JOHNZON

ActiveMQ Artemis

APACHE MyFaces

BatchEE

DELTASPIKE

Reminder: today “EE” ecosystem

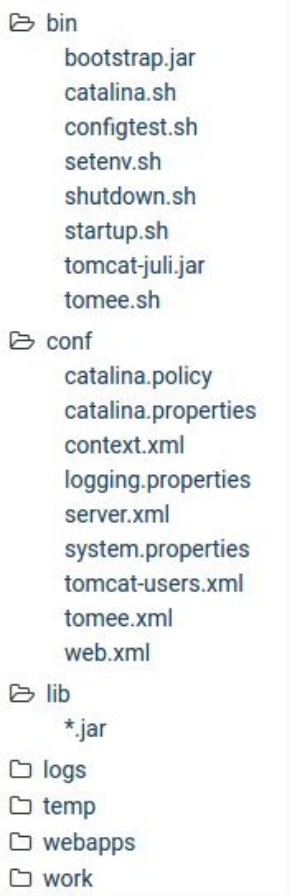
- EE 6: full/web profile
- Microprofile
- Spring boot

Part I > TomEE

What TomEE is

- EE 6 certified
- EE 7
- Tomcat based
- Light!
- Usage oriented

What does TomEE look like?



From 26 to ~117 or 184 jars

Some TomEE resource

```
<Resource id="MySQL" aliases="myAppDataSourceName" type="DataSource">
   JdbcDriver = com.mysql.jdbc.Driver
   JdbcUrl = jdbc:mysql://${OPENSHIFT_MYSQL_DB_HOST}:${OPENSHIFT_MYSQL_DB_PORT}/rmannibucau?tcpKeepAlive=true
   UserName = ${OPENSHIFT_MYSQL_DB_USERNAME}
   Password = cipher:Static3DES:ULkcoVik7DM=
   ValidationQuery = SELECT 1
   ValidationInterval = 30000
   NumTestsPerEvictionRun = 5
   TimeBetweenEvictionRuns = 30 seconds
   TestWhileIdle = true
   MaxActive = 200
 </Resource>
```

```
┌( rmannibucau @ alienware )─( 11:20 -:- 05/12 )
└( «he-tomee-webprofile-7.0.3 ) -> ./bin/tomee.sh cipher -c Static3DES test
ULkcoVik7DM=
```

TomEE challenge : integration*s*

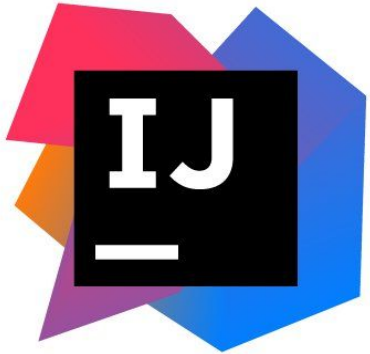
From 1 integration between 2
To N integrations between M

TomEE challenge : integration*s*

CDI
+
Servlet
+
JAX-RS
+
JSON-P/JSON-B
+
…
≠
It works
≠
EE

TomEE : distributions

TomEE : tooling



TomEE : maven example

```
<plugin>
  <groupId>org.apache.tomee.maven</groupId>
  <artifactId>tomee-maven-plugin</artifactId>
  <version>${tomee7.version}</version>
  <configuration>
    <tomeeClassifier>plus</tomeeClassifier>
    <debug>false</debug>
    <debugPort>5005</debugPort>
    <args>-Dfoo=bar</args>
    <config>${project.basedir}/src/test/tomee/conf</config>
    <libs>
      <lib>mysql:mysql-connector-java:5.1.20</lib>
    </libs>
    <webapps>
       <webapp>org.superbiz:myapp:4.3?name=ROOT</webapp>
       <webapp>org.superbiz:api:1.1</webapp>
    </webapps>
    <apps>
        <app>org.superbiz:mybugapp:3.2:ear</app>
    </apps>
    <libs>
        <lib>mysql:mysql-connector-java:5.1.21</lib>
        <lib>unzip:org.superbiz:hibernate-bundle:4.1.0.Final:zip</lib>
        <lib>remove:openjpa-</lib>
    </libs>
  </configuration>
</plugin>
```

```
$ mvn tomee:run
$ mvn tomee:build
```

TomEE : Testing

JUnit

TomEE : Testing / ApplicationComposer

```
@EnableServices("jaxrs")
@JaxrsProviders(JohnzonProvider.class)
@Classes(cdi = true, value = GenericClientService.class, ConfigurationProducer.class, OAuth2Mock.class)
public class GenericClientServiceTest {
  @Rule
  public final TestRule app = RuleChain.outerRule(new SftpServer())
                                       .around(new CassandraRule(this))
                                       .around(new ApplicationComposerRule(this));

  @RandomPort("http")
  private URL base;

  @Inject
  private SomeService service;

  @Test
  public void test() {
    assertEquals("xxx", service.someCall("test"));
    // or jaxrs client api using base
  }

  @Path("mock/oauth2")
  public static class OAuth2Mock {
    @POST @Path("token")
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
    public String token(final MultivaluedMap<String, String> form) {
      return "{\"access_token\":\"test-token\",\"token_type\":\"bearer\"}";
    }
  }
}
```

TomEE : Testing / Arquillian

```java
@RunWith(Arquillian.class)
public class ArquillianTest {
  @Deployment
  public static Archive<?> orange() {
     return ShrinkWrap.create(WebArchive.class, "orange.war")
        .addClasses(MyService.class)
        .addAsWebInfResource(EmptyAsset.INSTANCE, "beans.xml");
  }

  @Inject
  private MyService service;

  @Test
  public void test() {
     assertEquals("xxx", service.someCall("test"));
  }
}
```

```xml
<?xml version="1.0"?>
<arquillian xmlns="http://jboss.org/schema/arquillian"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
          http://jboss.org/schema/arquillian
          http://jboss.org/schema/arquillian/arquillian_1_0.xsd">
 <container qualifier="tomee" default="true">
  <configuration>
    <property name="conf">src/test/conf</property>
    <property name="classifier">plus</property>
    <property name="httpPort">-1</property>
    <property name="stopPort">-1</property>
    <property name="simpleLog">true</property>
    <property name="cleanOnStartUp">true</property>
    <property name="dir">target/tomee</property>
    <property name="appWorkingDir">target/arquillian</property>
    <property name="properties">
     openejb.deploymentId.format={appId}/{ejbJarId}/{ejbName}
    </property>
    <property name="additionalLibs">
     mvn:org.apache.commons:commons-rock:1.0
    </property>
    <property name="catalina_opts">
      -Duser.language=en -Duser.region=en_US
    </property>
  </configuration>
 </container>
</arquillian>
```

TomEE : Testing / Others

```java
@RunWith(TomEEEmbeddedSingleRunner.class)
public class NoScannerSingleRunnerTest {
  @Application
  private ScanApp app;

  @TomEEEmbeddedApplicationRunner.Args
  private String[] args;

  @Test
  public void run() {
    app.check();
  }

  @Application
  @Classes(value = ScanMe.class)
  @ContainerProperties({
    @ContainerProperties.Property(
      name = "app.database.url", value = "jdbc:h2:mem:test")
  })
  public static class ScanApp {
    @Inject
    private ScanMe ok;

    @Inject
    private Instance<NotScanned> ko;

    public void check() {
      assertNotNull(ok);
      assertTrue(ko.isUnsatisfied());
    }
  }
}
```


```java
@Properties({
  @Property(
    key = DeploymentFilterable.CLASSPATH_INCLUDE,
    value = ".*app.*")
})
@RunWith(EJBContainerRunner.class)
public class TestWithCdiBean {
  @Inject
  private CdiBean cdi;

  @Inject
  private EjbBean ejb;

  @Test
  public void test() {
    ...
  }
}
```

TomEE : Some more goodness

- System properties
  - Container/Resource definition
  - Overriding (MDB, resource placehodlers)
- Container events
- Resource templates/properties provider
- Reloadable PersistenceUnit
- @MBean
- Lazy tomcat components (valve, realm) to impl them from the webapp classloader
- ...

Part II > Meecrowave

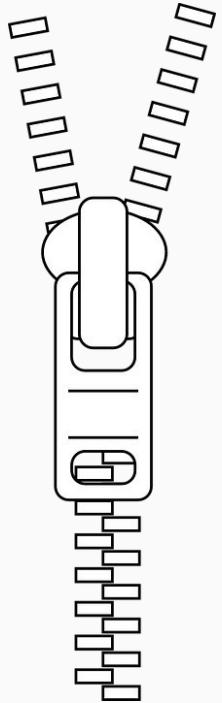
Did you mean Microwave?

Meecrowave

Meecrowave: scope

- Server (tomcat based)
- Test solutions
- Tooling/dev

Meecrowave: the server



Meecrowave: programmatic flavor

```
new Meecrowave().bake().await()
```

```
final Meecrowave.Builder builder = new Meecrowave.Builder();
builder.setHttpPort(8080);
builder.setScanningPackageIncludes("com.company.app");
builder.setScanningPackageIncludes("app-");
builder.setConf("app/conf");
builder.setJaxrsMapping("/api/*");
builder.setJsonpPrettify(true);
builder.setRealm(new RealmBase() {
  @Override
  protected String getPassword(final String s) {
    return "test";
  }

  @Override
  protected Principal getPrincipal(final String s) {
    return new GenericPrincipal(s, "test", singletonList("admin"));
  }
});
builder.instanceCustomizer(t -> t.getHost().getPipeline().addValve(new ValveBase() {
  @Override
  public void invoke(final Request request, final Response response) throws IOException, ServletException {
    response.getWriter().write("custom");
  }
}));
// ...

try (final Meecrowave meecrowave = new Meecrowave(builder).bake().deployWebApp(new File("app.war"))) {
  meecrowave.await(); // or any command handling/CLI fitting your soft
}
```

Meecrowave: CLI flavor

```
$ java -jar meecrowave-bundle.jar \
>> --http 8080 \
>> --tomcat-access-log-pattern ‘%h %l %u %t "%r" %s %b’ \
>> --web-resource-cached true \
>> --meecrowave-properties /home/meecrowave/defaults.properties
```

Meecrowave: build tools/dev

```
apply plugin: 'org.apache.meecrowave.meecrowave'

meecrowave { // ~builder
  httpPort = 9090
}
```

```
$ gradle meecrowave
```

```
<plugin>
 <groupId>org.apache.meecrowave</groupId>
 <artifactId>meecrowave-maven-plugin</artifactId>
 <version>${meecrowave.version}</version>
</plugin>
```

```
$ mvn meecrowave:run
$ mvn meecrowave:bundle
```

```
.
├── bin
│   └── meecrowave.sh
├── conf
│   ├── log4j2.xml
│   └── meecrowave.properties
├── lib
│   └── *.jar
├── logs
│   └── meecrowave.log
└── temp
```

Meecrowave: testing

```
public class MyResourceTest {
  private final MonoMeecrowave.Rule container = new MonoMeecrowave.Rule();

  @Rule
  public final TestRule rule = RuleChain.outerRule(MySQLRule.instance()).around(container);

  @Test
  public void test() {
     final Client client = ClientBuilder.newClient();
     try {
        final WebTarget base = client.target("http://localhost:" + container.getConfiguration().getHttpPort() + "/api");
        //...
     } finally {
        client.close();
     }
  }
```

Meecrowave: extensions

@Unit

Qhawtio

JTA

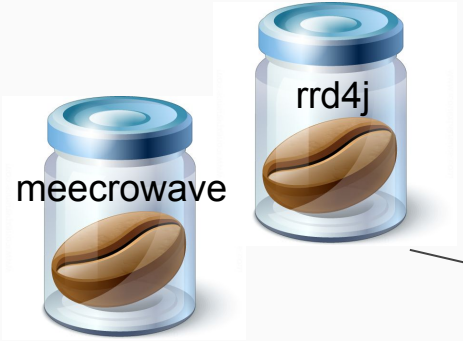
Meecrowave: new CDI paradigm (JPA ex)

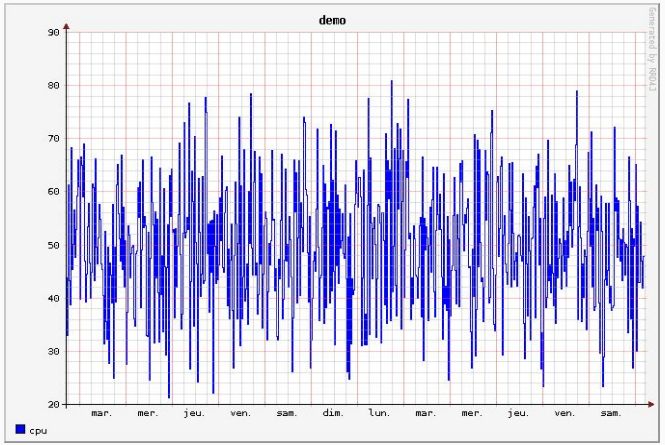
```
@ApplicationScoped
public static class DataSourceConfig {
  @Produces
  @ApplicationScoped
  public DataSource dataSource() {
    final BasicDataSource source = new BasicDataSource();
    source.setDriver(new Driver());
    source.setUrl("jdbc:h2:mem:apachecon");
    return source;
  }
}

@ApplicationScoped
public class JpaConfig {
  @Produces
  public PersistenceUnitInfoBuilder unit(final DataSource ds) {
    return new PersistenceUnitInfoBuilder()
            .setUnitName("apachecon")
            .setDataSource(ds)
            .setExcludeUnlistedClasses(true)
            .addManagedClazz(User.class)
            .addProperty("openjpa.jdbc.SynchronizeMappings", "buildSchema");
  }
}
```

```
@ApplicationScoped
public class JPADao {
  @Inject
  @Unit(name = "apachecon")
  private EntityManager em;

  // tx by default
  public User save(final User user) {
    em.persist(user);
    return user;
  }

  @Jpa(transactional = false) // no tx
  public User find(final long id) {
    return em.find(User.class, id);
  }
}
```

Meecrowave ex.: rrd server

Goal: build a quick RRD server





Meecrowave ex.: rrd server / code

Defaults

Build a rrd4j graph spec

Fill graph data

Respond with the picture

```
@GET
@Path("{db}/graph")
public Response graphs(@PathParam("db") String name,
                       @QueryParam("fun") @DefaultValue("AVERAGE") String fun,
                       @QueryParam("from") long from,
                       @QueryParam("to") long to) {
    final RrdDb rrdDb = getDb(name, true);
    if (to == 0) to = TimeUnit.MILLISECONDS.toSeconds(System.currentTimeMillis());
    if (from == 0) from = to - TimeUnit.DAYS.toSeconds(7);

    final RrdGraphDef gDef = new RrdGraphDef();
    gDef.setWidth(600);
    gDef.setHeight(400);
    gDef.setStartTime(from);
    gDef.setEndTime(to);
    gDef.setTitle(name);
    gDef.setImageFormat("png");

    final long[] interval = new long[]{from, to};
    IntStream.range(0, rrdDb.getDsCount()).forEach(it -> {
        final Datasource ds = rrdDb.getDatasource(it);
        try {
            gDef.datasource(ds.getName(), rrdDb.createFetchRequest(ConsolFun.AVERAGE, interval[0], interval[1]).fetchData());
            gDef.line(ds.getName(), Color.BLUE, ds.getName());
        } catch (final IOException e) {
            throw new WebApplicationException(Response.Status.INTERNAL_SERVER_ERROR);
        }
    });

    try {
        final byte[] payload = Optional.ofNullable(new RrdGraph(gDef).getRrdGraphInfo())
                .map(RrdGraphInfo::getBytes)
                .orElseGet(() -> "No data".getBytes(StandardCharsets.UTF_8));
        return Response.ok(payload)
                .header("Content-Type", "image/png")
                .build();
    } catch (final IOException e) {
        throw new WebApplicationException(Response.Status.INTERNAL_SERVER_ERROR);
    }
}
```

Meecrowave for batchs/daemons?

```java
@Named
@Dependent
public class Task extends AbstractBatchlet {
  @Override
  public String process() throws Exception {
    return "demo :)";
  }
}
```

```xml
<job xmlns="http://xmlns.jcp.org/xml/ns/javaee"
     version="1.0" id="demo">
 <step id="start">
  <batchlet ref="task" />
 </step>
</job>
```

```xml
<dependencies>
 <dependency>
  <groupId>org.apache.geronimo.specs</groupId>
  <artifactId>geronimo-jbatch_1.0_spec</artifactId>
  <version>1.0</version>
 </dependency>
 <dependency>
  <groupId>org.apache.batchee</groupId>
  <artifactId>batchee-jbatch</artifactId>
  <version>0.4-incubating</version>
 </dependency>
 <dependency>
  <groupId>org.apache.meecrowave</groupId>
  <artifactId>meecrowave-core</artifactId>
  <version>0.3.1</version>
 </dependency>
</dependencies>
```

Meecrowave short life softs, jbatch ex

```
@ApplicationScoped
public class BatchExecutor {
  private long taskId;
  private CountDownLatch latch = new CountDownLatch(1);
  private JobOperator jobOperator;

  public void launchAndWait(@Observes @Initialized(ApplicationScoped.class) final ServletContext context) {
    jobOperator = BatchRuntime.getJobOperator();
    taskId = jobOperator.start("demo", new Properties());
    latch.countDown();
  }

  public void waitBatchResult() throws Exception {
    latch.await(5, TimeUnit.MINUTES);

    // portable wait
    while (!ofNullable(jobOperator.getJobExecution(taskId))
          .map(JobExecution::getBatchStatus)
          .map(Enum::name)
          .filter(s -> s.endsWith("ED")).isPresent()) {
      sleep(1000);
    }

    /* batchee wait
    JobExecutionCallbackService callbackService = ServicesManager.find().service(JobExecutionCallbackService.class);
    callbackService.waitFor(taskId);
    */

    dumpExecution(jobOperator.getJobExecution(taskId));
    jobOperator.getStepExecutions(taskId).forEach(this::dump);
  }
}
```

```
public static void main(String[] args) throws Exception {
  new Meecrowave(new Meecrowave.Builder() {{ setSkipHttp(true); }}).bake();
  lookup(CDI.current().getBeanManager()).waitBatchResult();
}
```

DeltaSpike the Meecrowave friend

```java
@ApplicationScoped
public class SomeService {
  @Inject
  private EntityManager entityManager;

  public DataSourcePayload findById(final String id) {
    return entityManager.find(SomeEntity.class, id);
  }

  @Transactional
  public SomeEntity create(final SomeEntity entity) {
    entityManager.persist(entity);
    entityManager.flush();
    return entity;
  }
```

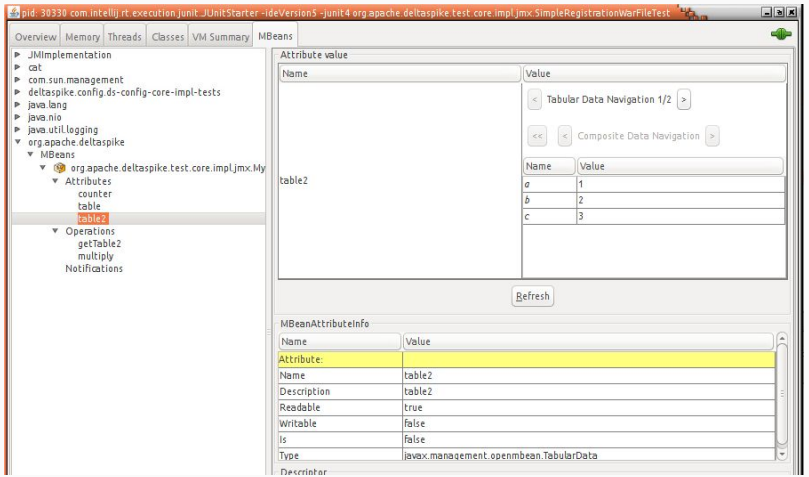
```java
@Repository // @MappingConfig
public interface MyRepository extends
    EntityRepository<MyEntity, String> {
}

@ApplicationScoped
public class MyService {
  @Inject
  private MyRepository repository;

  public MyEntity find(String id) {
    return repository.findBy(id);
  }

  public List<MyEntity> findAll(int start, String id) {
    return repository.findAll(start, 10);
  }
}
```

DeltaSpike the Meecrowave friend

```
@Configuration(prefix = "app.database.")
public interface AppConfiguration {
  @ConfigProperty(name = "url", defaultValue = "jdbc:mysql://localhost:3306/app")
  String databaseUrl();

  @ConfigProperty(name = "driver", defaultValue = "com.mysql.cj.jdbc.Driver")
  String databaseDriver();

  @ConfigProperty(name = "validation.query")
  String databaseValidationQuery();

  @ConfigProperty(name = "validation.interval", defaultValue = "-1")
  long databaseValidationInterval();

  @ConfigProperty(name = "username", defaultValue = "user")
  String databaseUser();

  @ConfigProperty(name = "password", defaultValue = "pass")
  String databasePassword();

  @ConfigProperty(name = "idle.min", defaultValue = "5")
  int databaseMinIdle();

  @ConfigProperty(name = "idle.max", defaultValue = "256")
  int databaseMaxTotal();
}
```

DeltaSpike the Meecrowave friend

```
@ApplicationScoped
@MBean(description = "my mbean")
public class MyMBean {
  @Inject JmxBroadcaster broadcaster;

  @JmxManaged(description = "get counter") int counter = 0; // getter/setter
  @JmxManaged Table table2;

  @JmxManaged public Table getTable2() {
     return new Table()
        .withColumns("a", "b", "c")
        .withLine("1", "2", "3")
        .withLine("alpha", "beta", "gamma");
  }

  @JmxManaged(description = "multiply counter")
  public int multiply(@JmxParameter(name = "multiplier", description = "the
multiplier") final int n) {
     return counter * n;
  }

  public void broadcast() {
    broadcaster.send(new Notification(String.class.getName(), this, 10L));
  }
}
```



TomEE or Meecrowave?

Which server?

Criteria	TomEE	Meecrowave
Stack	*Standard* (JPA, JTA, JAX-*S, …)	*Web, DS* or *Custom* (NoSQL, ...)
Front	JSF, any	Angular/React, Vaadin, ...
Library support	full EE	Servlet, JAX-RS, JSON*, CDI
Transactions (XA)	Yes	~No*
Resource definition	External (dev vs ops)	Internal (dev = ops)
Customization	8/10	9/10
Team responsibilities	9/10	7/10

So which one?

The beast or the beauty?

Microprofile?

And Microprofile/Spring boot?

spring boot

THANK YOU

@rmannibucau | https://blog-rmannibucau.rhcloud.com/