JavaEE examples java-ee-examples (0.0.1) [Build Status]

Maksim Kostromin

Version 0.0.1, 2019-05-02 04:10:18 UTC

Table of Contents

Introduction

This documentation contains some help to examples from java-ee-examples repository. It's contains some JavaEE playground projects

build run and test

```
./gradlew clean build composeUp

http :8080/app/
http :8080/app/dwr/index.html

./gradlew composeDown

# or just
./gradlew build ; ./gradlew composeDown ; ./gradlew composeUp ; docker-compose -f
docker-compose-gradle.yaml logs -f -t
```

Chapter 1. jboss-eap-postgres

jboss-eap-6.4/standalone/configuration/standalone.xml

```
        <!-- ... -->
    </extensions>
    <!-- TODO: CHANGE ME: https://access.redhat.com/solutions/3442891 -->
    <system-properties>
        <property name="jackson.deserialization.whitelist.packages"
                  value="org.kie,org.drools,daggerok"/>
    </system-properties>
    <!-- TODO: CHANGE ME END -->
    <management>
        <!-- ... -->
        <!-- ... -->
        <subsystem xmlns="urn:jboss:domain:datasources:1.2">
            <datasources>
                <!-- TODO:
                        CHANGE ME: https://access.redhat.com/documentation/en-
us/red_hat_jboss_enterprise_application_platform/6.4/html-
single/administration_and_configuration_guide/#sect-Example_Datasources
                                    https://www.redhat.com/cms/managed-
files/eap_msa_ose3_0.pdf -->
                <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name=
"ExampleDS" enabled="true" jta="true" use-java-context="true" use-ccm="false">
                    <connection-url>jdbc:postgresql://${env.POSTGRES_HOST:postgres.my-
app.com}:${env.POSTGRES_PORT:5432}/${env.POSTGRES_DB:db}</connection-url>
                    <driver>postgresql</driver>
                    <security>
                        <user-name>${env.POSTGRES_USER:user}</user-name>
                        <password>${env.POSTGRES_PASSWORD:password}</password>
                    </security>
                </datasource>
                <drivers>
                    <driver name="postgresql" module="org.postgresql">
                        <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-
datasource-class>
                    </driver>
                </drivers>
                <!-- TODO: CHANGE ME END -->
            </datasources>
        </subsystem>
        <!-- ... -->
```

jboss-eap-6.4/modules/org/postgresql/main/module.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.4-1206-jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
    <module name="javax.servlet.api" optional="true"/>
  </dependencies>
</module>
```

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
             xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="" transaction-type="JTA">
    <jta-data-source>java:jboss/datasources/ExampleDS</jta-data-source>
    <properties>
      <property name="hibernate.hbm2ddl.auto" value="create"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.format_sql" value="false"/>
      <!-- optionals: -->
      <!--<property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQL82Dialect"/>-->
    </properties>
    <!--<provider>org.hibernate.ejb.HibernatePersistence</provider>-->
  </persistence-unit>
</persistence>
```

Entity

```
@Data
@Entity
@Table(name = "my_entities")
@NoArgsConstructor(access = PROTECTED)
@RequiredArgsConstructor(staticName = "of")
@NamedQueries({
    @NamedQuery(name = FIND_ANY, query = "SELECT me FROM MyEntity me WHERE
LOWER(me.data) LIKE LOWER(CONCAT('%',:q,'%'))"),
    @NamedQuery(name = FIND_ALL, query = "SELECT me FROM MyEntity me ORDER BY
me.createdAt DESC"),
    @NamedQuery(name = COUNT, query = "SELECT COUNT(me) FROM MyEntity me")
})
public class MyEntity implements Serializable {

  public static final String COUNT = "MyEntity.count";
  public static final String FIND_ALL = "MyEntity.findAll";
  public static final String FIND_ANY = "MyEntity.findAny";

  @Id
  @Setter(PRIVATE)
  @GeneratedValue(generator = "UUID2")
  @GenericGenerator(name = "UUID2", strategy = "org.hibernate.id.UUIDGenerator")
  private UUID id;

  @Column
  @NonNull
  String data;

  @Temporal(TIMESTAMP)
  @Column(name = "created")
  Date createdAt;

  @Temporal(TIMESTAMP)
  @Column(name = "updated")
  Date updatedAt;

  @PrePersist
  public void onCreate() {
    createdAt = new Date();
  }

  @PreUpdate
  public void onMerge() {
    final long time = new Date().getTime();
    createdAt = new Date(time);
    updatedAt = new Date(time);
  }
}
```

JPA EntityManager CDI integration

```
/**
 * see https://stackoverflow.com/questions/51756761/jboss-eap-7-1-spring-data-jpa-cdi-extension
 */
@Slf4j
@ApplicationScoped
public class CDIEntityManagerProducer {

  @Produces
  @Dependent
  @PersistenceUnit(unitName = "")
  EntityManagerFactory emf;

  @Produces
  @RequestScoped
  public EntityManager em() {
    final EntityManager em = emf.createEntityManager();
    log.debug("hi {}", em);
    return em;
  }

  public void close(@Disposes EntityManager em) {
    log.debug("bye {}", em);
    em.close();
  }
}
```

build, run and test

```
./mvnw ; ./mvnw -Pdocker docker-compose:up

http :8080/app/ data=ololo
http :8080/app/ data=trololo
http :8080/app/

./mvnw -Pdocker docker-compose:down
```

1.1. jax-rs-hateoas-links

build

```
./gradlew clean build composeUp

http :8080/app

./gradlew composeDown
```

docker redeploy

```
/gradlew war; bash ./gradle/redeploy.sh
```

links:

- some rax-rs
- more jax-rs...

Initially generated by using generator-jvm yeoman generator (java-ee)

1.2. aop-logger

build run and test (required: docker)

```
./gradlew composeUp

http :8080/app/api/ping
http :8080/app/api/pong

# to see aop logger logs
docker logs -f aop-logger_gradle-aop-logger-app_1

./gradlew composeDown
```

JAX-RS ping-pong resource

```
@Stateless
@Path("api")
@Produces(APPLICATION_JSON)
public class HealthResource {

  static final Map<String, String> pingPongMap
      = HashMap.of("ping", "pong",
                   "pong", "ping")
              .toJavaMap();

  @Inject
  SomeBusinessLogic logic;

  @GET
  @Path("{path: (ping|pong)}")
  public Response pingPong(@PathParam("path") final String path) {
    logic.doSomething();
    return Response.ok(Json.createObjectBuilder()
                           .add("status", pingPongMap.get(path))
                           .build())
                   .build();
  }
}
```

business service

```
@Stateless
@Interceptors(LoggerInterceptor.class)
public class SomeBusinessLogic {

  public void doSomething() {}
}
```

interceptor

```
@Slf4j
public class LoggerInterceptor {

  @AroundInvoke
  public Object intercept(InvocationContext ic) throws Exception {
    log.info("intercepting: {} at: {}", ic.getContextData(), ic.getTimer());
    return ic.proceed();
  }
}
```

Initially generated by using generator-jvm yeoman generator (java-ee)

1.3. jax-rs-regex-path

example, how to handle `ping`, `pong` *and* `health` *paths by single resource method*

```
@Stateless
@Path("api")
@Produces(APPLICATION_JSON)
public class HealthResource {

  static final Map<String, String> pingPongMap
      = HashMap.of("ping", "pong",
                   "pong", "ping")
              .toJavaMap();

  @GET
  @Path("{path: (health|ping|pong)}")
  public Response health(@PathParam("path") final String path) {
    return Response.ok(Json.createObjectBuilder()
                           .add("status", pingPongMap.getOrDefault(path, "UP"))
                           .build())
                   .build();
  }
}
```

Initially generated by using generator-jvm yeoman generator (java-ee)

Chapter 2. Async

2.1. async-jax-rs-resources

async resource

```
@Stateless
@Path("items")
@Produces(APPLICATION_JSON)
class AppResource {

  @Context
  lateinit var uriInfo: UriInfo

  @Resource
  lateinit var mes: ManagedExecutorService

  @Inject
  lateinit var itemRepository: ItemRepository

  @GET
  @Path("async")
  fun getAllInAsync(@Suspended asyncResponse: AsyncResponse) = mes.execute {
    val result = itemRepository.findAll()
    asyncResponse.resume(result)
  }
}
```

RESTEeasy dto

```
@Entity
@JsonIgnoreProperties(ignoreUnknown = true)
data class Item(
    @Id @GeneratedValue var id: Long? = null,
    var value: String? = null
) : Serializable
```

Initially generated by using generator-jvm yeoman generator (kotlin-ee)

2.2. porcupine-bulkhead-jee8

build, run and test

```
./gradlew composeUp

http :8080/app/async-items/write1 value=ololo
http :8080/app/async-items/write2 value=trololo

http :8080/app/async-items/read1/1
http :8080/app/async-items/read1

http :8080/app/async-items/read2/2
http :8080/app/async-items/read2
```

entity

```
@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Accessors(chain = true)
public class Item implements Serializable {

  private static final long serialVersionUID = 1466287048756540922L;

  @Id
  @GeneratedValue
  Long id;

  String value;
}
```

repo

```
@Stateful
@TransactionAttribute(NOT_SUPPORTED)
public class ItemRepository {

  @PersistenceContext
  EntityManager em;

  @TransactionAttribute(REQUIRES_NEW)
  public Item save(final Item item) {
    em.persist(item);
    return item;
  }

  public Item findOne(final Long id) {
    return em.find(Item.class, id);
  }

  public List<Item> findAll() {
    return em.createQuery("select i from Item i", Item.class)
             .getResultList();
  }
}
```

write resource

```
@Slf4j
@Stateless
@Path("async-items")
@Produces(APPLICATION_JSON)
public class WriteItemsResource {

  @Inject
  ItemRepository itemRepository;

  @Inject
  @Dedicated("write-async-items")
  ExecutorService writeExecutor;

  @POST
  @Path("write1")
  public void post1(@Valid @NotNull final Item item, @Suspended final AsyncResponse
asyncResponse) {
    writeExecutor.execute(() -> {
      final Item result = itemRepository.save(item);
      asyncResponse.resume(result);
    });
  }

  @POST
  @Path("write2")
  public void post2(@Valid @NotNull final Item item, @Suspended final AsyncResponse
asyncResponse) {
    CompletableFuture.supplyAsync(() -> itemRepository.save(item), writeExecutor)
                     .thenAccept(asyncResponse::resume);
  }
}
```

read resource

```
@Slf4j
@Stateless
@Path("async-items")
@Produces(APPLICATION_JSON)
public class ReadItemsResource {

  @Inject
  ItemRepository itemRepository;

  @Inject
  @Dedicated("read-async-items")
  ExecutorService readExecutor;

  @GET
  @Path("read1")
  public void getAll1(@Suspended final AsyncResponse asyncResponse) {
    readExecutor.execute(() -> {
      final List<Item> result = itemRepository.findAll();
      asyncResponse.resume(result);
    });
  }

  @GET
  @Path("read2")
  public void getAll2(@Suspended final AsyncResponse asyncResponse) {
    CompletableFuture
        .supplyAsync(() -> itemRepository.findAll(), readExecutor)
        .thenAccept(asyncResponse::resume);
  }

  @GET
  @Path("read1/{id}")
  public void get1(@PathParam("id") final Long id, @Suspended final AsyncResponse
asyncResponse) {
    readExecutor.execute(() -> {
      final Item result = itemRepository.findOne(id);
      asyncResponse.resume(result);
    });
  }

  @GET
  @Path("read2/{id}")
  public void get2(@PathParam("id") final Long id, @Suspended final AsyncResponse
asyncResponse) {
    CompletableFuture.supplyAsync(() -> itemRepository.findOne(id), readExecutor)
                     .thenAccept(asyncResponse::resume);
  }
}
```

Initially generated by using generator-jvm yeoman generator (kotlin-ee)

Chapter 3. Micro Profile

MicroProfile 1.0 (CDI + JAX-RS + JSON-P)

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.wildfly.swarm</groupId>
      <artifactId>bom-all</artifactId>
      <version>${version.wildfly.swarm}</version>
      <scope>import</scope>
      <type>pom</type>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Chapter 4. JBOSS 4 | JAX-RS

build

```
./mvnw clean package com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down

./gradlew clean build composeUp
./gradlew composeDown
```

generated by daggerok-fatjar yeoman generator

in fucking progress...

1. book
2. youtube
3. Github: sdaschner/scalable-coffee-shop

kafka

```
cd /tmp
wget -O kafka.jar https://github.com/daggerok/embedded-kafka/raw/mvn-
repo/daggerok/embedded-kafka/0.0.1/embedded-kafka-0.0.1.jar
java -jar kafka.jar
```

gradle

```
./gradlew
bash build/libs/*jar

./gradlew build composeUp
./gradlew composeDown
```

Chapter 5. maven is not working properly

actually working but with hack (see src/main/webapp/WEB-INF/classes/README)

maven

```
./mvnw
java -jar target/*.jar

./mvnw; ./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down
```

generated using daggerok-fatjar yeoman generator

inside:

1. java 8 based project
2. javaee 8.0 using wildfly-swarm micro-profile
3. kotlin support
4. lombok (slf4j + logback logging)
5. vavr (javaslang)
6. support maven
7. support gradle
8. supports testing junit 4 / 5
9. docker / docker-compose support

```
./gradlew
bash build/libs/*jar

./gradlew build composeUp
./gradlew composeDown
```

```
./mvnw
java -jar target/*.jar

./mvnw; ./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down
```

generated using daggerok-fatjar yeoman generator

inside:

1. java 8 based project

2. javaee 8.0
3. lombok (slf4j + logback logging)
4. vavr (javaslang)
5. support maven
6. support gradle
7. supports testing junit 4 / 5
8. docker / docker-compose support (JBOSS EAP 7)

build

```
./mvnw clean package com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down

./gradlew clean build composeUp
./gradlew composeDown
```

Chapter 6. JavaEE Wildfly-Swarm Micro-Profile using Gradle

6.1. wildfly-swarm-gradle

This repository contains simple JavaEE JAX-RS Wildfly Swarm Micro-profile example playground application

using gradle / java microprofile

```
bash gradlew clean build
java -Djava.net.preferIPv4Stack=true -jar build/libs/wildfly-swarm-gradle-swarm.jar
http :8080
```

using docker

```
docker build --force-rm -f ./docker/src/Dockerfile -t docker-java-ee-examples .
docker run -d -p 8080:8080 --rm --name wildfly-swarm-gradle docker-java-ee-examples
http :8080
docker rm -f -v wildfly-swarm-gradle
```

using docker-compose

```
docker-compose -f ./docker/src/docker-compose.yml up -d
http :8080
docker-compose -f ./docker/src/docker-compose.yml down -v
```

using docker swarm stack deploy

```
docker swarm init
docker service create --detach=false --name registry --publish 5000:5000 registry:2
docker build -f ./docker/src/Dockerfile -t 127.0.0.1:5000/app .
docker push 127.0.0.1:5000/app
#docker-compose -f ./docker/src/stack-deploy.yml build --force-rm --no-cache --pull
#docker-compose -f ./docker/src/stack-deploy.yml push
docker stack deploy --compose-file ./docker/src/stack-deploy.yml java-ee
docker stack services --filter name="java-ee_app" --format="{{.Name}} {{.Replicas}}"
java-ee
docker service scale --detach=false java-ee_app=2

sleep 15
docker stack services java-ee

http :8080

docker swarm leave --force
docker system prune -af
```

links:

1. Eclipse MicroProfile
2. Wildfly Swarm Micro-Profile
3. Wildfly Swarm Book
4. wildfly-swarm fails on travis-ci
5. wildfly-swarm gradle plugin

Chapter 7. JavaEE Wildfly-Swarm Micro-Profile using Maven

7.1. wildfly-swarm-maven

This repository contains simple JavaEE JAX-RS Wildfly Swarm Micro-profile example playground application

using maven

```
mvn -Djava.net.preferIPv4Stack=true wildfly-swarm:run
http :8080
```

using wildfly swarm microprofile

```
mvn clean package
java -Djava.net.preferIPv4Stack=true -jar target/java-ee-examples-0.0.0-swarm.jar
http :8080
```

using docker

```
docker build --force-rm -f ./docker/src/Dockerfile -t docker-java-ee-examples .
docker run -d -p 8080:8080 --rm --name wildfly-swarm-maven docker-java-ee-examples
http :8080
docker stop wildfly-swarm-maven
```

using docker-compose

```
docker-compose -f ./docker/src/docker-compose.yml up -d
http :8080
docker-compose -f ./docker/src/docker-compose.yml down -v
```

using docker swarm stack deploy

```
docker swarm init
docker service create --detach=false --name registry --publish 5000:5000 registry:2
docker build -f ./docker/src/Dockerfile -t 127.0.0.1:5000/app .
docker push 127.0.0.1:5000/app
#docker-compose -f ./docker/src/stack-deploy.yml build --force-rm --no-cache --pull
#docker-compose -f ./docker/src/stack-deploy.yml push
docker stack deploy --compose-file ./docker/src/stack-deploy.yml java-ee
docker stack services --filter name="java-ee_app" --format="{{.Name}} {{.Replicas}}"
java-ee
docker service scale --detach=false java-ee_app=2

sleep 30
docker stack services java-ee

http :8080

docker swarm leave --force
docker system prune -af
```

links:

1. Eclipse MicroProfile
2. Wildfly Swarm Micro-Profile
3. Wildfly Swarm Book
4. wildfly-swarm fails on travis-ci

Chapter 8. Kumuluzee MicroProfile 1.0

8.1. kumuluzee-microprofile-1.0

This repository contains simple JavaEE Wildfly Swarm Micro-profile example - serving static content

using gradle / java microprofile

```
bash gradlew clean build
java -Djava.net.preferIPv4Stack=true -jar build/libs/wildfly-swarm-gradle-swarm.jar
http :8080
```

Chapter 9. JavaEE Kubernetes

9.1. java-kube-ee

This repository contains simple JavaEE JAX-RS Wildfly Swarm Micro-profile example playground application

prepare kubernetes cluster, build docker image, push into local registry

```
minikube start --cpus=4 --memory=4096
eval (minikube docker-env)
docker run -d --rm --name registry -p 5000:5000 registry:2
docker build -t 127.0.0.1:5000/app -f ./docker/src/Dockerfile .
docker push 127.0.0.1:5000/app
```

create kubernete yaml file

```
vim ./docker/k8s/app.yml

kubectl apply -f ./docker/k8s/app.yml --validate=false

kubectl get svc -o wide                                                 23:51:57
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
SELECTOR
app          NodePort    10.0.0.189   <none>        8080:32464/TCP   3m        app=app

curl 192.168.99.100:32464
# or
http (minikube service app --url)
22:21:09

[
    "one",
    "two",
    "three",
    "kubeee"
]
```

now update code and redeploy app

deploy app

```
bash gradlew clean build
docker build -t 127.0.0.1:5000/app:v2 -f ./docker/src/Dockerfile .
docker push 127.0.0.1:5000/app:v2
vim ./docker/k8s/app.yml # update image: 127.0.0.1:5000/app:v2
kubectl apply -f ./docker/k8s --validate=false
```

terminal 1

```
kubectl get pods -w                23:59:35
NAME                   READY     STATUS    RESTARTS   AGE
app-1515935557-411m1   1/1       Running   0          10m
app-1515935557-4d3gf   1/1       Terminating   0         4m
app-1515935557-4d3gf   0/1       Terminating   0         4m
app-3366916455-bw6m9   0/1       Pending   0         0s
app-3366916455-bw6m9   0/1       ContainerCreating   0         0s
app-3366916455-bw6m9   1/1       Running   0         1s
```

terminal 2

```
while true; curl (minikube service app --url); echo""; sleep 1; end
["one","two","three","kubeee"]
["one","two","three","kubeee"]
Waiting, endpoint for service is not ready yet...
["one","two","three","kubeee","2017-10-02T21:12:34.095Z"]
["one","two","three","kubeee","2017-10-02T21:12:36.182Z"]
```

second test but using ip, not minikube service

redeploy app prev version

```
vim ./docker/k8s/app.yml # update image: 127.0.0.1:5000/app
kubectl apply -f ./docker/k8s --validate=false
```

terminal 1

```
kubectl get pods -w                23:59:35
NAME                   READY     STATUS    RESTARTS   AGE
app-1515935557-411m1   1/1       Running   0          10m
...
app-1515935557-4d3gf   1/1       Terminating   0         4m
app-1515935557-4d3gf   0/1       Terminating   0         4m
...
app-3366916455-bw6m9   0/1       Pending   0         0s
app-3366916455-bw6m9   0/1       ContainerCreating   0         0s
app-3366916455-bw6m9   1/1       Running   0         1s
```

terminal 2

```
while true; curl 192.168.99.100:32161; echo""; sleep 1; end
["one","two","three","kubeee"]
["one","two","three","kubeee"]
Waiting, endpoint for service is not ready yet...
["one","two","three","kubeee","172.17.0.5"]
["one","two","three","kubeee","172.17.0.6"]
```

autoscale

```
kubectl autoscale deployment app --min=2 --max=3
```

scale down

```
kubectl scale deployment app --replicas=3
```

cleanup

```
kubectl delete all -l name=app
# or
kubectl delete service app; kubectl delete deployment app

minikube stop; minikube delete
```

9.2. old

using gradle / java microprofile

```
bash gradlew clean build
java -Djava.net.preferIPv4Stack=true -jar build/libs/java-kube-ee-swarm.jar
http :8080
```

e2e tests using bash scripts)

```
bash ./docker/bin/test-dockerfile.bash     # docker
bash ./docker/bin/test-docker-compose.bash # docker compose
bash ./docker/bin/test-stack-deploy.bash   # docker swarm cluster
```

links

1. like so

Chapter 10. JavaEE using Kotlin

10.1. java-kube-ee

This repository contains simple JavaEE JAX-RS Wildfly Swarm Micro-profile example playground application

prepare kubernetes cluster, build docker image, push into local registry

```
minikube start --cpus=4 --memory=4096
eval (minikube docker-env)
docker run -d --rm --name registry -p 5000:5000 registry:2
docker build -t 127.0.0.1:5000/app -f ./docker/src/Dockerfile .
docker push 127.0.0.1:5000/app
```

create kubernete yaml file

```
vim ./docker/k8s/app.yml

kubectl apply -f ./docker/k8s/app.yml --validate=false

kubectl get svc -o wide                                                 23:51:57
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
SELECTOR
app          NodePort    10.0.0.189   <none>        8080:32464/TCP   3m        app=app

curl 192.168.99.100:32464
# or
http (minikube service app --url)
22:21:09

[
    "one",
    "two",
    "three",
    "kubeee"
]
```

now update code and redeploy app

deploy app

```
bash gradlew clean build
docker build -t 127.0.0.1:5000/app:v2 -f ./docker/src/Dockerfile .
docker push 127.0.0.1:5000/app:v2
vim ./docker/k8s/app.yml # update image: 127.0.0.1:5000/app:v2
kubectl apply -f ./docker/k8s --validate=false
```

terminal 1

```
kubectl get pods -w                23:59:35
NAME                   READY     STATUS    RESTARTS   AGE
app-1515935557-411m1   1/1       Running   0          10m
app-1515935557-4d3gf   1/1       Terminating   0         4m
app-1515935557-4d3gf   0/1       Terminating   0         4m
app-3366916455-bw6m9   0/1       Pending   0         0s
app-3366916455-bw6m9   0/1       ContainerCreating   0         0s
app-3366916455-bw6m9   1/1       Running   0         1s
```

terminal 2

```
while true; curl (minikube service app --url); echo""; sleep 1; end
["one","two","three","kubeee"]
["one","two","three","kubeee"]
Waiting, endpoint for service is not ready yet...
["one","two","three","kubeee","2017-10-02T21:12:34.095Z"]
["one","two","three","kubeee","2017-10-02T21:12:36.182Z"]
```

second test but using ip, not minikube service

redeploy app prev version

```
vim ./docker/k8s/app.yml # update image: 127.0.0.1:5000/app
kubectl apply -f ./docker/k8s --validate=false
```

terminal 1

```
kubectl get pods -w                23:59:35
NAME                   READY     STATUS    RESTARTS   AGE
app-1515935557-411m1   1/1       Running   0          10m
...
app-1515935557-4d3gf   1/1       Terminating   0         4m
app-1515935557-4d3gf   0/1       Terminating   0         4m
...
app-3366916455-bw6m9   0/1       Pending   0         0s
app-3366916455-bw6m9   0/1       ContainerCreating   0         0s
app-3366916455-bw6m9   1/1       Running   0         1s
```

terminal 2

```
while true; curl 192.168.99.100:32161; echo""; sleep 1; end
["one","two","three","kubeee"]
["one","two","three","kubeee"]
Waiting, endpoint for service is not ready yet...
["one","two","three","kubeee","172.17.0.5"]
["one","two","three","kubeee","172.17.0.6"]
```

autoscale

```
kubectl autoscale deployment app --min=2 --max=3
```

scale down

```
kubectl scale deployment app --replicas=3
```

cleanup

```
kubectl delete all -l name=app
# or
kubectl delete service app; kubectl delete deployment app

minikube stop; minikube delete
```

10.2. old

using gradle / java microprofile

```
bash gradlew clean build
java -Djava.net.preferIPv4Stack=true -jar build/libs/java-kube-ee-swarm.jar
http :8080
```

e2e tests using bash scripts)

```
bash ./docker/bin/test-dockerfile.bash     # docker
bash ./docker/bin/test-docker-compose.bash # docker compose
bash ./docker/bin/test-stack-deploy.bash   # docker swarm cluster
```

links

1. like so

10.3. kotlin-plugins-java-ee

This repository contains simple JavaEE JAX-RS Wildfly Swarm Micro-profile example using Kotlin

```
bash gradlew clean build
bash build/libs/*-swarm.jar

http :8080
http :8080/max
```

10.4. main-swarm-rest-api

This repository contains simple JavaEE JAX-RS Wildfly Swarm Micro-profile example playground application

using gradle / java microprofile

```
bash gradlew clean build
java -Djava.net.preferIPv4Stack=true -jar build/libs/wildfly-swarm-gradle-swarm.jar
http :8080
http :8080/api
http :8080/api/max
```

10.5. main-swarm-static-content

This repository contains simple JavaEE Wildfly Swarm Micro-profile example - serving static content

using gradle / java microprofile

```
bash gradlew clean build
java -Djava.net.preferIPv4Stack=true -jar build/libs/wildfly-swarm-gradle-swarm.jar
http :8080
```

10.6. kotlin-java-ee-payara-docker

This repository contains simple JavaEE JAX-RS Wildfly Swarm Micro-profile example using Kotlin

```
./gradlew clean war
docker-compose up --force-recreate --build --remove-orphans
http :8080/payara-app/
http :8080/payara-app/max
docker-compose down -v
```

10.7. kotlin-javaee-cdi-h2

according to default ${JBOSS_HOME}/standalone/configuration/standalone.xml

```
<!-- skepped... -->
<datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS"
enabled="true" use-java-context="true">
<!-- skepped... -->
```

build and test

```
docker-compose down -v; ./gradlew; ./mvnw; docker-compose up --build --force-recreate
--remove-orphans
# gradle
http get  :8080/kotlin-ee/
http get  :8080/kotlin-ee/get-all
http post :8080/kotlin-ee/save-some
http get  :8080/kotlin-ee/get-all
# maven
http get  :8081/kotlin-ee/
http get  :8081/kotlin-ee/get-all
http post :8081/kotlin-ee/save-some
http get  :8081/kotlin-ee/get-all
```

Just JBoss EAP in Docker

Chapter 11. JBoss EAP in Docker

11.1. faces

build

```
./mvnw clean package com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down

http :8080/app/

./gradlew clean build composeUp
./gradlew composeDown
```

links:

1. facelets oracle tutorial
2. JAX-RS redirect
3. Java Platform, Enterprise Edition (Java EE) 8 The Java EE Tutorial
4. Java Platform, Enterprise Edition (Java EE) 8 Your First Cup: An Introduction to the Java EE Platform

generated by daggerok-fatjar yeoman generator === facelets-example

build

```
./mvnw clean package com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down

http :8080/app/
http :8080/app/health
http :8080/app/faces/index.xhtml

./gradlew clean build composeUp
./gradlew composeDown
```

links:

1. facelets oracle tutorial
2. JAX-RS redirect
3. Java Platform, Enterprise Edition (Java EE) 8 The Java EE Tutorial
4. Java Platform, Enterprise Edition (Java EE) 8 Your First Cup: An Introduction to the Java EE Platform

generated by daggerok-fatjar yeoman generator

11.2. jboss-eap-ext.js

packaging two WARs into jboss AS

Using JBoss EAP 6.4 in docker

```
./mvnw clean install -U -T 4; docker-compose up --build --force-recreate

open localhost:8080/ui/

docker-compose down -v
```

11.3. xmlrpc

Apache XML-RPC (org.apache.xmlrpc)

variations:

1. monolith
2. micro-server
3. micro-client

build and test

```
docker-compose down -v; ./mvnw clean package; docker-compose up --build
```

11.4. ear

build

```
./mvnw clean package -U
```

docker

```
docker-compose up --build --force-recreate --remove-orphans
docker-compose down -v
# docker rm -f -v (docker ps -a|grep -v CONTAINER|awk '{print $1}')
```

testing

```
http :8080/module-1/
http :8080/module-2/
http :8080/module-3/
http :8080/module-4/
```

lombok for java 7 (maven)

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.14.2</version>
  <scope>provided</scope>
</dependency>
```

lombok for java 7 (gradle)

```
dependencies {
  compileOnly 'org.projectlombok:lombok:1.14.2'
}
```

11.5. ejb-2

build

```
./mvnw clean package -U
```

docker

```
docker-compose up --build --force-recreate --remove-orphans
docker-compose down -v
# docker rm -f -v (docker ps -a|grep -v CONTAINER|awk '{print $1}')
```

1. Classic EJB 2.1 Local + Remote interfaces:
    1. ejb-modules/greeter-impl
    2. ejb-modules/greeter-local-api
    3. ejb-modules/greeter-remote-api
    4. client-modules/greeter-remote-client
2. Using remote EJB interface only:
    1. ejb-modules/remote-only-api
    2. ejb-modules/remote-only-impl
    3. client-modules/remote-only-client
3. Access EJB locally:
    1. ejb-modules/goodbyer-impl
    2. ejb-modules/goodbyer-local-api
    3. client-modules/goodbyer-local-client

testing (remote ejb call)

```
http :8080/greeter-remote-client/
http :8080/remote-only-client/
http :8080/goodbyer-local-client/
```

links:

1. EJB 2.1 deployment descriptor

11.6. timer

build and test

```
docker-compose down -v; ./mvnw; docker-compose up --build
http :8080/client/start
http :8080/client/stop
```

11.7. timer-async-ejb

build and test

```
docker-compose down -v; ./mvnw; docker-compose up --build
http :8080/client/start; sleep 3; http :8080/client/stop
http :8080/client/max\?name=bax
http :8080/client/max
http :8080/client
```

11.8. ejb-3-java-ee-7

build and test

```
docker-compose down -v; ./mvnw; docker-compose up --build
http :8080/client/start; sleep 3; http :8080/client/stop
http :8080/client/fax\?name=max
http :8080/client/bax
http :8080/client
```

11.9. ejb-stateful-singleton

build and test

```
docker-compose down -v; ./mvnw; docker-compose up --build
http :8080/client/
http :8080/client/get\?key=EJB
http post :8080/client/set\?key=EJB\&value=some-value
http :8080/client/get\?key=EJB
http post :8080/client/counter/increment
http post :8080/client/counter/decr
http post :8080/client/counter/incr
http :8080/client/get\?key=EJB
http post :8080/client/reset
```

11.10. jboss-eap-h2-ejb

according to default ${JBOSS_HOME}/standalone/configuration/standalone.xml

```
<!-- skepped... -->
<datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS"
enabled="true" use-java-context="true">
<!-- skepped... -->
```

build and test

```
docker-compose down -v; ./mvnw; docker-compose up --build --force-recreate --remove
-orphans
http post :8080/client/update/max
http post :8080/client/update/maxp
http post :8080/client/update/amaxp
http :8080/client/
http post :8080/client/update/max\?remove=true
http :8080/client/
```

links:

1. eap 6

11.11. plain HTTP Servlet

JBOSS EAP 7.1

build and run using docker

```
docker-compose down -v; ./mvnw clean package; ./gradlew clean build; docker-compose up
--build --force-recreate --remove-orphans

http :8081/plain-http-servlet/
http :8082/plain-http-servlet/
```

project structure

```
mkdir -p src/main/java/daggerok \
         src/main/webapp/WEB-INF

touch src/main/java/daggerok/AppServlet.java \
      src/main/webapp/WEB-INF/web.xml \
      build.gradle \
      pom.xml
```

vim src/main/java/daggerok/AppServlet.java

```
@WebServlet("/")
public class AppServlet extends HttpServlet {

  @Override
  public void service(final ServletRequest req, final ServletResponse res) throws
ServletException, IOException {
    final PrintWroter writer = res.getWriter();
    writer.println("<b>hi!</b>");
    writer.close();
  }
}
```

vim src/main/webapp/WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
</web-app>
```

vim build.gradle

```
plugins {
  id "war"
}

group "daggerok"
version "0.0.1"
description "ololo trololo"
sourceCompatibility = targetCompatibility = JavaVersion.VERSION_1_7

war {
  archiveName = "${project.name}.war"
}

repositories {
  mavenCentral()
}

dependencies {
  providedCompile "javax.servlet:javax.servlet-api:3.1.0"
}
```

vim pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns=
"http://maven.apache.org/POM/4.0.0"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>daggerok</groupId>
  <artifactId>plain-http-servlet</artifactId>
  <version>0.0.1</version>
  <packaging>war</packaging>

  <name>plain-http-servlet</name>
  <description>ololo trololo</description>

  <properties>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
      <optional>true</optional>
    </dependency>
  </dependencies>

  <build>
    <finalName>${project.artifactId}</finalName>
    <defaultGoal>clean package</defaultGoal>

    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.5</version>
        <configuration>
          <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Chapter 12. TomEE in Docker

12.1. tomee-ext.js

packaging two WARs in TomEE in Docker

Using TomEE in docker

```
./mvnw clean package -U -T 2
docker-compose up --build

open localhost:8080/ui/
open localhost:8080/rest-api/health

docker-compose down -v
```

Chapter 13. Glassfisg in Docker

13.1. glassfish-ext.js

packaging two WARs in glassfish in Docker

Using Glassfish 5.0 in docker (alpine)

```
./mvnw clean package -U -T 2
docker-compose up --build --force-recreate --remove-orphans
http :8080/ui/
docker-compose down -v
```

Chapter 14. JBoss WildFly (mvnw / gradlew) in Docker

14.1. forge-ws

init new maven project using forge cli

```
forge
project-new \
  --named forge-ws2 \
  --top-level-package daggerok \
  --final-name forge-ws \
  --type war
```

add maven wrapper

```
cd maven-forge-project/
mvn -N io.takari:maven:wrapper
```

build

```
./mvnw clean package -U -T 4
```

docker - see docker-compose and src/main/docker/Dockerfile

```
docker-compose up --build --force-recreate --remove-orphans
docker-compose down -v
# docker rm -f -v (docker ps -a|grep -v CONTAINER|awk '{print $1}')
```

testing

```
http :8080/app/v1/api
http :8080/app/UserService\?wsdl
curl -XPOST http://localhost:8080/app/User --header "content-type:text/xml" -d
@./src/test/resources/empty-request.xml | xmllint --format -
curl -XPOST http://localhost:8080/app/User --header "content-type:text/xml" -d
@src/test/resources/named-request.xml | xmllint --format -
```

lombok for java 7 (maven)

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.14.2</version>
  <scope>provided</scope>
</dependency>
```

lombok for java 7 (gradle)

```
dependencies {
  compileOnly 'org.projectlombok:lombok:1.14.2'
}
```

links:

1. jax-ws oracle tutorial
2. JBoss Forge 2, Java EE easily, so easily

Chapter 15. JBoss forge / WildFly Java EE 6 / JAX-WS

15.1. forge-javaee-6-ws

up and running

```
./mvnw clean package -U -T 2; docker-compose up --build --force-recreate --remove
-orphans

http :8080/app/health
http :8080/app/AppEndpoint\?wsdl

curl -XPOST http://localhost:8080/app/AppEndpoint --header "content-type:text/xml" -d
@request.xml | xmllint --format -
```

MicroProfile 1.1.0 (CDI + JAX-RS + JSON-P)

TODO

MicroProfile 1.2 (CDI + JAX-RS + JSON-P)

Chapter 16. Kumuluzee MicroProfile 2.0 (config.yaml)

16.1. kumuluzee-config

see rpc-app module

Chapter 17. Kumuluzee MicroProfile 2.0 (JAX-WS)

17.1. kumuluzee-mp-2.0-jax-ws

in progress...

up and running

```
./mvnw clean package -U; java -jar target/*.jar

http :8000
http :8000/api/v1/ws
http :8000/api/v1/ws?WSDL

curl -XPOST http://localhost:8000/api/v1/ws --header "content-type: text/xml" -d
@./request.xml  | xmllint --format -
```

docker / docker-compose

```
./mvnw
docker-compose up --force-recreate --remove-orphans

http :8000
http :8000/api/v1/ws
http :8000/api/v1/ws?WSDL

curl -XPOST http://localhost:8000/api/v1/ws --header "content-type: text/xml" -d
@./request.xml  | xmllint --format -

docker-compose down -v
```

Chapter 18. TODO

- JBoss / JMS / MDB

Chapter 19. links

1. Java Platform, Enterprise Edition (Java EE) 8 The Java EE Tutorial
2. Java Platform, Enterprise Edition (Java EE) 8 Your First Cup: An Introduction to the Java EE Platform
3. Reactive JavaEE (Vert.x / KumuluzEE)
4. KumuluzEE tutorials
5. KumuluzEE samples
6. Configuration beyond Java EE 8
7. Unit Testing for Java EE
8. Interfaces on Demand with CDI and EJB 3.1
9. The Java EE 6 Tutorial
10. Asciidoctor attributes

Chapter 20. Enjoy! :)