Code 1 - Vogella tutorial on JPA

--Persistence.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence (…) >
  <persistence-unit name="todos"
transaction-type="RESOURCE_LOCAL">
  <class>Todo</class>
  <properties>     (…)  </properties>
 </persistence-unit>
 <persistence-unit name="people"
transaction-type="RESOURCE_LOCAL">
  <class>Person</class>
  <class>Family</class>
</persistence-unit>
</persistence>
```

--Family.java

```
@Entity
public class Family {
  @Id  @GeneratedValue(strategy =
GenerationType.TABLE)
  private int id;   private String
description;
  @OneToMany
  private final List<Person> members;
  // getters & setters omitted
}
```

--Person.java

```
@Entity
public class Person {
  @Id
  @GeneratedValue(strategy =
GenerationType.TABLE)
  private String id;   private String
firstName;
  private String lastName;
  // getters & setters omitted
}
```

--Job.java

```
@Entity
public class Job {
  @Id
  @GeneratedValue(
    strategy = GenerationType.TABLE)
  private String id;   private String
function;
  // getters & setters omitted
}
```

--JpaTest.java

```
public class JpaTest {
  static final String UNIT_NAME =
"people";
  private EntityManagerFactory factory;
  EntityManager em = factory
   .createEntityManager();
  @Before
  public void setUp() throws Exception
{
   factory = Persistence

.createEntityManagerFactory(UNIT_NAME);
   this.em =
factory.createEntityManager();
   em.getTransaction().begin();
   Query q=em.createQuery("select m
from Person m");
   boolean createNewEntries =
    (q.getResultList().size() == 0);
   if (createNewEntries) {
    assertTrue(
      q.getResultList().size() == 0
    );
    Family family = new Family();
    family.setDescription("F Knopfs");
    em.persist(family);
    for (int i = 0; i < 40; i++) {
      Person person = new Person();
      person.setFirstName("Jim_" + i);
      person.setLastName("Knopf_" + i);
      em.persist(person);
      family.getMembers().add(person);
      em.persist(person);
      em.persist(family);
    }
   }
   em.getTransaction().commit();
   em.close();
  }
  @Test
  public void checkAvailablePeople() {
   Query q=em
     .createQuery("select m from Person
m");
   assertTrue(
     q.getResultList().size() == 40
   );
   em.close();
  }
  @Test
  public void checkFamily() {
   Query q=
   em.createQuery("select f from Family
f");
   assertTrue(
     q.getResultList().size()==1);
   assertTrue(
    ((Family) q.getSingleResult())
      .getMembers().size()==40
   );
   em.close();
  }
  @Test(expected =
javax.persistence.NoResultException.class)
  public void deletePerson() {
   em.getTransaction().begin();
   Query q = em.createQuery(
    "SELECT p FROM Person p
     WHERE p.firstName = :firstName
     AND p.lastName = :lastName");

   q.setParameter("firstName",
"Jim_1");
   q.setParameter("lastName",
"Knopf_!");
   Person user =
     (Person) q.getSingleResult();
     em.remove(user);
   em.getTransaction().commit();
   Person person =
     (Person) q.getSingleResult();
   em.close();
  }}
```

Code 2 - Netbeans' tutorial on News

--NewsEntityFacade.java

```
@Stateless
public class NewsEntityFacade {
  @PersistenceContext(unitName = "NewsApp")
  private EntityManager em;
  public void create(NewsEntity newsEntity)
{
   em.persist(newsEntity);    }
  public void edit(NewsEntity newsEntity) {
    em.merge(newsEntity);    }
  public void remove(NewsEntity newsEntity)
{
    em.remove(em.merge(newsEntity));    }
  public NewsEntity find(Object id) {
   return em.find(NewsEntity.class, id);
}
  public List<NewsEntity> findAll() {(…) }
  public int count() { (…)  }
}
```

--NewsEntity.java

```
@Entity
public class NewsEntity implements
Serializable {
  private static final long
serialVersionUID = 1L;
  @Id  @GeneratedValue(strategy =
GenerationType.AUTO)
  private Long id;
  private String title;
  private String body;
  // getters & setters omitted
}
```

--NewMessage.java

```
@MessageDriven(mappedName = "jms/NewMessage",
activationConfig =
{ @ActivationConfigProperty(propertyName =
"acknowledgeMode", propertyValue = "Auto-
acknowledge"),
  @ActivationConfigProperty(propertyName =
"destinationType", propertyValue =
"javax.jms.Queue")  })
public class NewMessage implements
MessageListener {
  @Resource
  private MessageDrivenContext mdc;
  @PersistenceContext(unitName = "NewsApp-
ejbPU")
  private EntityManager em;
  public NewMessage() { (…)  }
  public void onMessage(Message message) {
   ObjectMessage msg = null;
   if (message instanceof ObjectMessage) {
     msg = (ObjectMessage) message;
     NewsEntity e =
       (NewsEntity) msg.getObject();
     save(e);
   }
  }
  public void save(Object object) {
   em.persist(object);  }
}
```

--Message1.java

```
@WebServlet(name="Message1",
urlPatterns={"/Message1"})
public class Message1 extends HttpServlet {
  @EJB  private SessionManagerBean
sessionManagerBean;
  @EJB  private NewsEntityFacade
newsEntityFacade;
  protected void
processRequest(HttpServletRequest request,
HttpServletResponse response)  throws (…) {
    request.getSession(true);
    PrintWriter out = response.getWriter();
    (…)
      out.println("<body>");
      for (NewsEntity elem :
newsEntityFacade.findAll() ){
        out.println("
<b>"+elem.getTitle()+" </b><br />");
        out.println(elem.getBody()+"<br />
");
      }
     (…)
  }
}
```

--Message2.java

```
@WebServlet(name = "Message2", urlPatterns =
{"/Message2"})
public class Message2 extends HttpServlet {
  @Resource(mappedName =
"jms/NewMessageFactory")
  private ConnectionFactory cFactory;
  @Resource(mappedName = "jms/NewMessage")
  private Queue que;

  protected void
processRequest(HttpServletRequest request,
HttpServletResponse response)  throws (…) {
   String title =
     request.getParameter("title");
   String body =
request.getParameter("body");
   if ((title != null) && (body != null)){
     Connection conn =
       cFactory.createConnection();
     Session sess =
conn.createSession((...) );
     MessageProducer mProducer=
       sess.createProducer(que);
     ObjectMessage msg =
       sess.createObjectMessage();
     NewsEntity e = new NewsEntity();
     e.setTitle(title);  e.setBody(body);
     msg.setObject(e);
     mProducer.send(msg);
     mProducer.close();
     conn.close();
  }
}
```

--persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence (...) >
  <persistence-unit name="NewsApp-ejbPU"
transaction-type="JTA">
<provider>org.eclipse.persistence.jpa.Persistenc
eProvider</provider>
  <jta-data-source>jdbc/sample</jta-data-source>
  <class>ejb.NewsEntity</class>
  <exclude-unlisted-classes>true</exclude-
unlisted-classes>
  <properties>
    <property name="eclipselink.ddl-generation"
value="create-tables"/>
  </properties>
  </persistence-unit>
</persistence>
```

Code 3 - REST with Java

--Hello.java

```
@Path("/hello")
public class Hello {
  @GET   @Produces(MediaType.TEXT_PLAIN)
  public String sayPlainTextHello() {
   return "Hello Jersey";
  }
  @GET  @Produces(MediaType.TEXT_XML)
  public String sayXMLHello() {
   return "<?xml version=\"1.0\"?>"
    + "<hello> Hello Jersey" + "</hello>";
  }
  @GET  @Produces(MediaType.TEXT_HTML)
  public String sayHtmlHello() {
   return "<html> " + "<title>"
    + "Hello Jersey" + "</title>"
    + "<body><h1>" + "Hello Jersey"
    + "</body></h1>" + "</html> ";
  }
}
```

-- Test.java

```
public class Test {
  public static void main(String[] args) {
    ClientConfig config = new
ClientConfig();
    Client client =
ClientBuilder.newClient(config);
    WebTarget target = client
      .target(getBaseURI());
   String response = target.path("rest")
     .path("hello").request()
     .accept(MediaType.TEXT_PLAIN)
     .get(Response.class).toString();
   String plainAnswer =
target.path("rest")
    .path("hello")

.request().accept(MediaType.TEXT_PLAIN)
    .get(String.class);
   String xmlAnswer = target.path("rest")
    .path("hello").request()
     .accept(MediaType.TEXT_XML)
     .get(String.class);
   String htmlAnswer= target.path("rest")
     .path("hello").request()
     .accept(MediaType.TEXT_HTML)
     .get(String.class);
    (...)
  }
  private static URI getBaseURI() {
    return
UriBuilder.fromUri("http://localhost:8080/
com.vogella.jersey.first").build();
  }
}
```

-- web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app (...)>
  <display-
name>com.vogella.jersey.first</display-
name>
 <servlet>
   <servlet-name>Jersey REST
Service</servlet-name>
   <servlet-
class>org.glassfish.jersey.servlet.Servlet
Container</servlet-class>
    <init-param> <param-name>
jersey.config.server.provider.packages</pa
ram-name>
      <param-value>
com.vogella.jersey.first</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Jersey REST
Service</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Code 4 - A TRUE RESTFUL Example

--Todo.java

```
@XmlRootElement
public class Todo {
  private String summary;
  private String description;
  // getters & setters omitted
}
```

--TodoResource.java

```
@Path("/todo")
public class TodoResource {

  @GET
Produces({MediaType.APPLICATION_XML})
  public Todo getXML() {
    Todo todo = new Todo();
    todo.setSummary("Todo Summary");
    todo.setDescription("Todo
Description");
    return todo;
  }

  @GET
Produces({MediaType.APPLICATION_JSON})
  public Todo getJSON() {
    Todo todo = new Todo();
    todo.setSummary("Todo Summary");
    todo.setDescription("Todo
Description");
    return todo;
  }
  @GET    @Produces({ MediaType.TEXT_XML
})
  public Todo getHTML() {
    Todo todo = new Todo();
    todo.setSummary("XML Todo Summary");
    todo.setDescription("Todo
Description");
    return todo;
  }
}
```

-- TodoTest.java

```
public class TodoTest {
  public static void main(String[] args)
{
   ClientConfig config = new
ClientConfig();
   Client client =
      ClientBuilder.newClient(config);
   WebTarget target =
     client.target(getBaseURI());
   String xmlResponse =
target.path("rest")
      .path("todo").request()
      .accept(MediaType.TEXT_XML)
      .get(String.class);
   String xmlAppResponse
=target.path("rest")
      .path("todo").request()
      .accept(
         MediaType.APPLICATION_XML)
      .get(String.class);
   String jsonResponse =
target.path("rest")
       .path("todo").request()
       .accept(
          MediaType.APPLICATION_JSON)
       .get(String.class);
    (...)
  }
  private static URI getBaseURI() {
    return
UriBuilder.fromUri("http://localhost:8080
/com.vogella.jersey.jaxb").build();
  }
}
```

Code 5 - Building REST services

```
-- Account.java
@Entity
public class Account {
    @Id          @GeneratedValue
    Long id;
    String username;
    @JsonIgnore
    private String password;
    @OneToMany(mappedBy = "account")
    private Set<Bookmark> bookmarks = new
HashSet<>();

   // constructors, getters & setters
omitted
}
--AccountRepository.java
public interface AccountRepository extends
JpaRepository<Account, Long> {
    Optional<Account> findByUsername(String
username);
}
-- Bookmark.java
@Entity
public class Bookmark {
  @Id    @GeneratedValue
  private Long id;

  @JsonIgnore    @ManyToOne
  private Account account;

  private String uri;
  private String description;
  private Bookmark() { } // JPA only

  // constructors , getters & setters
omitted
}
```

```
-- BookmarkRestController.java
@RestController  @RequestMapping("/{userId}/bookmarks")
class BookmarkRestController {
  private final BookmarkRepository bookmarkRepository;
  private final AccountRepository accountRepository;
  @Autowired
  BookmarkRestController(BookmarkRepository bookmarkRepository,AccountRepository
accountRepository) {
   this.bookmarkRepository = bookmarkRepository;
   this.accountRepository = accountRepository;
  }
  @RequestMapping(method = RequestMethod.GET)
  Collection<Bookmark> readBookmarks(@PathVariable String userId) {
   this.validateUser(userId);
   return this.bookmarkRepository.findByAccountUsername(userId);
  }
  @RequestMapping(method = RequestMethod.POST)
  ResponseEntity<?> add(@PathVariable String userId, @RequestBody Bookmark input) {
   this.validateUser(userId);
   return this.accountRepository.findByUsername(userId)
    .map(account -> {
      Bookmark result = bookmarkRepository.save(new Bookmark(account,
         input.getUri(), input.getDescription()));
      URI location = ServletUriComponentsBuilder
        .fromCurrentRequest().path("/{id}")
        .buildAndExpand(result.getId()).toUri();
      return ResponseEntity.created(location).build();
     }).orElse(ResponseEntity.noContent().build());
  }
  @RequestMapping(method = RequestMethod.GET, value = "/{bookmarkId}")
  Bookmark readBookmark(@PathVariable String userId, @PathVariable Long bookmarkId) {
    this.validateUser(userId);
    return this.bookmarkRepository.findOne(bookmarkId);
  }
  private void validateUser(String userId) {
   this.accountRepository.findByUsername(userId).orElseThrow(
    () -> new UserNotFoundException(userId));
  } }
```

Code 6 - Consume Kafka

-- ProducerTrigger.java

```
@Singleton
@Startup
public class ProducerTrigger {
  @Resource  ManagedExecutorService mes;

  @PostConstruct
  public void trigger() {
   mes.execute(new Producer());
   System.out.println("Producer
triggered....");
  }
}
```

--KafkaMDB.java

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(
     propertyName = "clientId",
     propertyValue = "testClient"),
    @ActivationConfigProperty(
      propertyName = "groupIdConfig",
      propertyValue = "testGroup"),
    @ActivationConfigProperty(
      propertyName = "topics",
      propertyValue =
"${ENV=TOPIC_NAME}"),
    @ActivationConfigProperty(
      propertyName =
"bootstrapServersConfig",
      propertyValue =
"${ENV=KAFKA_BROKER}"),
    @ActivationConfigProperty(
      propertyName = "keyDeserializer",
      propertyValue =
"...StringDeserializer"),
    @ActivationConfigProperty(
      propertyName = "valueDeserializer",
      propertyValue = "
...StringDeserializer")

})
public class KafkaMDB implements
KafkaListener {
  public KafkaMDB() {   }

  @OnRecord( topics={"payara-kafka-mdb-
topic"})
  public void
test(ConsumerRecord<Object,Object> record)
{
   System.out.println("Payara Kafka MDB
record " + record );
  }

}
```

--Producer.java

```
public class Producer implements Runnable
{
  KafkaProducer<String, String> produc;
  String topic = null;
  public Producer() {
   Properties consumerProps = new
Properties();
   consumerProps.put(

ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,

System.getenv().getOrDefault("KAFKA_BROKER
", "192.168.99.100:9092"));
   consumerProps.put(
     ProducerConfig.KEY_SERIALIZER_CLASSG,
     " ... StringSerializer");
   consumerProps.put(

ProducerConfig.VALUE_SERIALIZER_CLASS,
     "...StringSerializer");

   produc = new
KafkaProducer<>(consumerProps);
   topic = System.getenv()
      .getOrDefault("TOPIC_NAME", "test-
topic");
 }

  static Random rnd = new Random();

  @Override
  public void run() {
   (…) // "Producing to topic
   while (true) {
    String key = "";
    produc.send(
      new ProducerRecord(topic, "key-" +
rnd.nextInt(10), "val-" +
rnd.nextInt(10)),
      new Callback() {
       public void
onCompletion(RecordMetadata rm, Exception
excptn) {
         (…) // sent data
       }
      });
      try {
        Thread.sleep(10000); //take it
easy!
      } catch (InterruptedException ex) {
       (…) // log exception      }
    }
  }
}
```

Code 7 - A store catalogue

```
-- StockApplication.java
@ApplicationPath("/api")
@Path("/items")
@Stateless
@TransactionAttribute(TransactionAttributeType.
NEVER)
public class StockApplication extends
Application {
  @EJB CatalogueBean catalogue;

  @POST @Consumes(MediaType.APPLICATION_JSON)
  @Produces(MediaType.APPLICATION_JSON)
  public Item
createItem(StockCreateRequest item) throws
IOException   {
    return catalogue
     .addItem(item.getName(),
item.getStock()); }
  @GET    @Path("/{id}")
  @Produces(MediaType.APPLICATION_JSON)
  public Item getItem(@PathParam("id") int
id) throws IOException  {
    return catalogue.getItem(id);    }
  @PUT    @Path("/{id}")
  @Consumes(MediaType.APPLICATION_JSON)
  @Produces(MediaType.APPLICATION_JSON)
  public Item updateStock(@PathParam("id")
int id, StockUpdateRequest update) throws
IOException    {
    return catalogue
      .addStock(id, update.getAmmount());
}
}
```

```
--Cart.java
@ManagedBean
@SessionScoped
public class Cart {
  @EJB  CartBean cart;

  public Collection<Item> getItems() {
   return cart.getItems();
  }
  public void add(Item item) {
   if (!cart.add(item)) //  Could not add
  }
  public void remove(Item item) {
   if (!cart.remove(item))// Could not
remove
  }
  public void purchase() {
    try  {
      if (cart.purchase()) // successful
      else //.No items in cart
    } catch (IOException e) {
       // Purchase failed
    }
  }
}
```

```

--CatalogueBean.java
@Stateless
public class CatalogueBean {
  (…)
  public List<Item> getCatalogue() { (…) }
  public Item getItem(int id) { (…) }
  public Item addItem(String name, int
stock)
    { (…)   }
  public Item addStock(int id, int amount)
    { (…) }
  public void purchase(int id) { (…)  }
}
```

```
-- Catalogue.java
@ManagedBean
public class Catalogue {
  @EJB  CatalogueBean catalogue;
  public Collection<Item> getItems()
throws IOException    {
    return this.catalogue.getCatalogue();
 }
}
```

```
-- CartBean.java
@Stateful
public class CartBean{
  List<Item> items = new ArrayList<>();
  public boolean add(Item item) {
   return this.items.add(item); }
  public boolean remove(Item item) {
   return this.items.remove(item);   }
  public List<Item> getItems()   {
   return this.items;   }
@TransactionAttribute(TransactionAttributeType.
REQUIRED)
  public boolean purchase() throws
IOException {
   if (items.isEmpty()) { return false; }
   for (Item item : items)
     item.purchase();
   clear();
   return true;
  }
  @Remove
  public void clear() {
this.items.clear();}
}
```

```
-- Item.java
public class Item{
  private CatalogueBean catalogue;
  private int id;
  private String name;
  private int stock;
  // constructors, getter setter omitted
}
```