

Code 1 - Vogella tutorial on JPA

```
1  --Persistence.xml
2
3  <?xml version="1.0" encoding="UTF-8" ?>
4  <persistence (...) >
5    <persistence-unit name="todos"
6      transaction-type="RESOURCE_LOCAL">
7      <class>Todo</class>
8      <properties> (...) </properties>
9    </persistence-unit>
10   <persistence-unit name="people"
11     transaction-type="RESOURCE_LOCAL">
12     <class>Person</class>
13     <class>Family</class>
14   </persistence-unit>
15 </persistence>
16
17  --Family.java
18  (...)

19  @Entity
20  public class Family {
21    @Id @GeneratedValue(strategy =
22 GenerationType.TABLE)
23    private int id;  private String
24    description;
25    @OneToMany
26    private final List<Person> members;
27    // getters & setters omitted
28  }
29
30  --Person.java
31  (...)

32  @Entity
33  public class Person {
34    @Id
35    @GeneratedValue(strategy =
36 GenerationType.TABLE)
37    private String id;  private String
38    firstName;
39    private String lastName;
40    // non essential getters & setters omitted
41  }
42
43  --Job.java
44  (...)

45  @Entity
46  public class Job {
47    @Id
48    @GeneratedValue(
49      strategy = GenerationType.TABLE)
50    private String id;  private String
51    function;
52    // non essential getters & setters omitted
53  }
54
55  --JpaTest.java
56  (...)

57  public class JpaTest {
58    static final String UNIT_NAME = "people";
59    private EntityManagerFactory factory;
60    EntityManager em = factory
61      .createEntityManager();
62
63    @Before
64    public void setUp() throws Exception {
65      factory = Persistence
66        .createEntityManagerFactory(UNIT_NAME);
67      this.em = factory.createEntityManager();
68      em.getTransaction().begin();
69
70      Query q=em.createQuery("select m from
71      Person m");
72      boolean createNewEntries =
73        (q.getResultList().size() == 0);
74      if (createNewEntries) {
75        assertTrue(
76          q.getResultList().size() == 0
77        );
78        Family family = new Family();
79        family.setDescription("F Knopfs");
80        em.persist(family);
81        for (int i = 0; i < 40; i++) {
82          Person person = new Person();
83          person.setFirstName("Jim_" + i);
84          person.setLastName("Knopf_" + i);
85          em.persist(person);
86          family.getMembers().add(person);
87          em.persist(person);
88          em.persist(family);
89        }
90      }
91      em.getTransaction().commit();
92      em.close();
93    }
94    @Test
95    public void checkAvailablePeople() {
96      Query q=em
97        .createQuery("select m from Person m");
98      assertTrue(
99        q.getResultList().size() == 40
100      );
101      em.close();
102    }
103    @Test
104    public void checkFamily() {
105      Query q=
106      em.createQuery("select f from Family f");
107      assertTrue(
108        q.getResultList().size()==1);
109      assertTrue(
110        ((Family) q.getSingleResult())
111        .getMembers().size()==40
112      );
113      em.close();
114    }
115
116    @Test(expected =
117 javax.persistence.NoResultException.class)
118    public void deletePerson() {
119      em.getTransaction().begin();
120      Query q = em.createQuery(
121        "SELECT p FROM Person p
122        WHERE p.firstName = :firstName
123        AND p.lastName = :lastName");
124
125      q.setParameter("firstName", "Jim_1");
126      q.setParameter("lastName", "Knopf_!");
127      Person user =
128        (Person) q.getSingleResult();
129
130      em.remove(user);
131      em.getTransaction().commit();
132      Person person =
133        (Person) q.getSingleResult();
134      em.close();
135    }}
```

Code 2 - Netbeans' tutorial on News

```
1 --NewsEntityFacade.java
2 @Stateless
3 public class NewsEntityFacade {
4     @PersistenceContext(unitName = "NewsApp")
5     private EntityManager em;
6     public void create(NewsEntity newsEntity) {
7         em.persist(newsEntity);    }
8     public void edit(NewsEntity newsEntity) {
9         em.merge(newsEntity);    }
10    public void remove(NewsEntity newsEntity) {
11        em.remove(em.merge(newsEntity));    }
12    public NewsEntity find(Object id) {
13        return em.find(NewsEntity.class, id);    }
14    public List<NewsEntity> findAll() {(...)}    }
15    public int count() {(...)}    }
16 }
17 --NewsEntity.java
18 @Entity
19 public class NewsEntity implements Serializable
20 {
21     private static final long serialVersionUID =
22 1L;
23     @Id @GeneratedValue(strategy =
24 GenerationType.AUTO)
25     private Long id;
26     private String title;
27     private String body;
28     // getters & setters omitted
29 }
30
31 --NewMessage.java
32 @MessageDriven(mappedName = "jms/NewMessage",
33 activationConfig =
34 { @ActivationConfigProperty(propertyName =
35 "acknowledgeMode", propertyValue = "Auto-
36 acknowledge"),
37     @ActivationConfigProperty(propertyName =
38 "destinationType", propertyValue =
39 "javax.jms.Queue") })
40
41 public class NewMessage implements
42 MessageListener {
43     @Resource
44     private MessageDrivenContext mdc;
45     @PersistenceContext(unitName = "NewsApp-
46 ejbPU")
47     private EntityManager em;
48     public NewMessage() {(...)}    }
49     public void onMessage(Message message) {
50         ObjectMessage msg = null;
51         if (message instanceof ObjectMessage) {
52             msg = (ObjectMessage) message;
53             NewsEntity e =
54                 (NewsEntity) msg.getObject();
55             save(e);
56         }
57     }
58     public void save(Object object) {
59         em.persist(object);    }
60 }
61
62 --Message1.java
63 @WebServlet(name="Message1",
64 urlPatterns={"/Message1"})
65 public class Message1 extends HttpServlet {
66     @EJB private SessionManagerBean
67     sessionManagerBean;
68     @EJB private NewsEntityFacade
69     newsEntityFacade;
70     protected void
71     processRequest(HttpServletRequest request,
72     HttpServletResponse response) throws (...) {
73         request.getSession(true);
74         PrintWriter out = response.getWriter();
75         (...)
76         out.println("<body>");
77         for (NewsEntity elem :
78             newsEntityFacade.findAll() ){
79             out.println(" <b>" + elem.getTitle() +
80             "</b><br />");
81             out.println(elem.getBody() + "<br /> ");
82         }
83         (...)
84     }
85 }
86 --Message2.java
87 @WebServlet(name = "Message2", urlPatterns =
88 {"/Message2"})
89 public class Message2 extends HttpServlet {
90     @Resource(mappedName =
91 "jms/NewMessageFactory")
92     private ConnectionFactory cFactory;
93     @Resource(mappedName = "jms/NewMessage")
94     private Queue que;
95
96     protected void
97     processRequest(HttpServletRequest request,
98     HttpServletResponse response) throws (...) {
99
100    String title =
101        request.getParameter("title");
102
103    String body = request.getParameter("body");
104    if ((title != null) && (body != null)) {
105        Connection conn =
106            cFactory.createConnection();
107        Session sess = conn.createSession((..));
108        MessageProducer mProducer=
109            sess.createProducer(que);
110        ObjectMessage msg =
111            sess.createObjectMessage();
112        NewsEntity e = new NewsEntity();
113        e.setTitle(title); e.setBody(body);
114        msg.setObject(e);
115        mProducer.send(msg);
116        mProducer.close();
117        conn.close();
118    }
119 }
120 --persistence.xml
121 <?xml version="1.0" encoding="UTF-8"?>
122 <persistence (...) >
123     <persistence-unit name="NewsApp-ejbPU"
124     transaction-type="JTA">
125     <provider>org.eclipse.persistence.jpa.Persistence
126     ceProvider</provider>
127     <jta-data-source>jdbc/sample</jta-data-
128     source>
129     <class>ejb.NewsEntity</class>
130     <exclude-unlisted-classes>true</exclude-
131     unlisted-classes>
132     <properties>
133         <property name="eclipselink.ddl-generation"
134         value="create-tables"/>
135     </properties>
136     </persistence-unit>
137 </persistence>
138 }
```

Code 3 - REST with Java

```
1 --Hello.java
2
3 (...)

4 @Path("/hello")
5 public class Hello {
6
7     @GET    @Produces(MediaType.TEXT_PLAIN)
8     public String sayPlainTextHello() {
9         return "Hello Jersey";
10    }
11
12    @GET    @Produces(MediaType.TEXT_XML)
13    public String sayXMLHello() {
14        return "<?xml version=\"1.0\"?>" +
15            + "<hello> Hello Jersey" + "</hello>";
16    }
17
18    @GET    @Produces(MediaType.TEXT_HTML)
19    public String sayHtmlHello() {
20        return "<html> " + "<title>" +
21            + "Hello Jersey" + "</title>" +
22            + "<body><h1>" + "Hello Jersey" +
23            + "</body></h1>" + "</html> ";
24    }
25 }

26
27 -- Test.java
28
29 (...)

30 public class Test {
31     public static void main(String[] args) {
32         ClientConfig config = new ClientConfig();
33         Client client =
34     ClientBuilder.newClient(config);
35         WebTarget target = client
36             .target(getBaseURI());
37         String response = target.path("rest")
38             .path("hello").request()
39             .accept(MediaType.TEXT_PLAIN)
40             .get(Response.class).toString();
41
42         String plainAnswer = target.path("rest")
43             .path("hello")
44             .request().accept(MediaType.TEXT_PLAIN)
45             .get(String.class);
46
47         String xmlAnswer = target.path("rest")
48             .path("hello").request()
49             .accept(MediaType.TEXT_XML)
50             .get(String.class);
51
52         String htmlAnswer= target.path("rest")
53             .path("hello").request()
54             .accept(MediaType.TEXT_HTML)
55             .get(String.class);
56
57     (...)

58
59     private static URI getBaseURI() {
60         return
61     UriBuilder.fromUri("http://localhost:8080/com
62     .vogella.jersey.first").build();
63     }
64
65
66
67 -- web.xml
68
69 <?xml version="1.0" encoding="UTF-8"?>
70 <web-app (...)>
71     <display-
72     name>com.vogella.jersey.first</display-name>
73     <servlet>
74         <servlet-name>Jersey REST
75     Service</servlet-name>
76     <servlet-
77     class>org.glassfish.jersey.servlet.ServletCon
78     tainer</servlet-class>
79         <!-- Register resources and providers
80     under com.vogella.jersey.first package. -->
81         <init-param>
82             <param-
83     name>jersey.config.server.provider.packages</
84     param-name>
85             <param-
86     value>com.vogella.jersey.first</param-value>
87             </init-param>
88             <load-on-startup>1</load-on-startup>
89         </servlet>
90         <servlet-mapping>
91             <servlet-name>Jersey REST
92     Service</servlet-name>
93             <url-pattern>/rest/*</url-pattern>
94         </servlet-mapping>
95     </web-app>
96
97
98
99
100
101
102
```

Code 4 - A TRUE RESTFUL Example

```
1 --Todo.java
2
3 (...)

4 @XmlRootElement
5 public class Todo {
6     private String summary;
7     private String description;
8     // getters & setters omitted
9 }
10
11 --TodoResource.java
12
13 (...)

14 @Path("/todo")
15 public class TodoResource {
16
17     @GET
18     @Produces({MediaType.APPLICATION_XML})
19     public Todo getXML() {
20         Todo todo = new Todo();
21         todo.setSummary("Todo Summary");
22         todo.setDescription("Todo Description");
23         return todo;
24     }
25
26     @GET
27     @Produces({MediaType.APPLICATION_JSON})
28     public Todo getJSON() {
29         Todo todo = new Todo();
30         todo.setSummary("Todo Summary");
31         todo.setDescription("Todo Description");
32         return todo;
33     }
34
35     @GET      @Produces({ MediaType.TEXT_XML })
36     public Todo getHTML() {
37         Todo todo = new Todo();
38         todo.setSummary("XML Todo Summary");
39         todo.setDescription("Todo Description");
40         return todo;
41     }
42     }
43
44 -- TodoTest.java
45
46 (...)

47 public class TodoTest {
48     public static void main(String[] args) {
49         ClientConfig config = new ClientConfig();
50         Client client =
51             ClientBuilder.newClient(config);
52         WebTarget target =
53             client.target(getBaseURI());
54
55         String xmlResponse = target.path("rest")
56             .path("todo").request()
57             .accept(MediaType.TEXT_XML)
58             .get(String.class);
59
60         String xmlAppResponse = target.path("rest")
61             .path("todo").request()
62             .accept(MediaType.APPLICATION_XML)
63             .get(String.class);
64
65         String jsonResponse = target.path("rest")
66             .path("todo").request()
67             .accept(MediaType.APPLICATION_JSON)
68             .get(String.class);
69
70     }
71
72     private static URI getBaseURI() {
73         return
74             UriBuilder.fromUri("http://localhost:8080/com
75             .vogella.jersey.jaxb").build();
76     }
77 }
78
79
80
81
82 }
```

Code 5 - Building REST services

```
1  -- Account.java
2 (...) 26
3 @Entity 27
4 public class Account { 28 -- Bookmark.java
5     @Id @GeneratedValue 29
6     Long id; 30 (...)

7     String username; 31
8     @JsonIgnore 32 @Entity
9     private String password; 33 public class Bookmark {
10    @OneToMany(mappedBy = "account") 34     @Id @GeneratedValue
11    private Set<Bookmark> bookmarks = new 35     private Long id;
12    HashSet<>(); 36
13 } // constructors, getters & setters omitted 37     @JsonIgnore @ManyToOne
14 } 38     private Account account;
15 --AccountRepository.java 39
16 (...) 40     private String uri;
17 public interface AccountRepository extends 41     private String description;
18 JpaRepository<Account, Long> { 42     private Bookmark() { } // JPA only
19     Optional<Account> findByUsername(String 43     // constructors , getters & setters omitted
20     username); 44 }
21 } 45 }
```

```
1 -- BookmarkRestController.java
2
3 (...) 26
4 @RestController @RequestMapping("/{userId}/bookmarks") 27
5 class BookmarkRestController { 28
6     private final BookmarkRepository bookmarkRepository; 29
7     private final AccountRepository accountRepository; 30
8     @Autowired 31
9     BookmarkRestController(BookmarkRepository bookmarkRepository, AccountRepository accountRepository) { 32
10         this.bookmarkRepository = bookmarkRepository; 33
11         this.accountRepository = accountRepository; 34
12     } 35
13     @RequestMapping(method = RequestMethod.GET) 36
14     Collection<Bookmark> readBookmarks(@PathVariable String userId) { 37
15         this.validateUser(userId); 38
16         return this.bookmarkRepository.findByAccountUsername(userId); 39
17     } 40
18     @RequestMapping(method = RequestMethod.POST) 41
19     ResponseEntity<?> add(@PathVariable String userId, @RequestBody Bookmark input) { 42
20         this.validateUser(userId); 43
21         return this.accountRepository.findByUsername(userId) 44
22             .map(account -> { 45
23                 Bookmark result = bookmarkRepository.save(new Bookmark(account, 46
24                     input.getUri(), input.getDescription())); 47
25                 URI location = ServletUriComponentsBuilder 48
26                     .fromCurrentRequest().path("/{id}") 49
27                     .buildAndExpand(result.getId()).toUri(); 50
28                     return ResponseEntity.created(location).build(); 51
29             }).orElse(ResponseEntity.noContent().build()); 52
30     } 53
31     @RequestMapping(method = RequestMethod.GET, value = "/{bookmarkId}") 54
32     Bookmark readBookmark(@PathVariable String userId, @PathVariable Long bookmarkId) { 55
33         this.validateUser(userId); 56
34         return this.bookmarkRepository.findOne(bookmarkId); 57
35     } 58
36     private void validateUser(String userId) { 59
37         this.accountRepository.findByUsername(userId).orElseThrow( 60
38             () -> new UserNotFoundException(userId)); 61
39     } 62
40 } 63
```

Code 6 - Consume Kafka

```
1 //note classe packages and some default
2 configuration keys simplified for space
3 constraints
4
5
6 -- ProducerTrigger.java
7
8 (...)

9 @Singleton
10 @Startup
11 public class ProducerTrigger {
12     @Resource ManagedExecutorService mes;
13
14     @PostConstruct
15     public void trigger() {
16         mes.execute(new Producer());
17         System.out.println("Producer
18 triggered....");
19     }
20 }

21
22 --KafkaMDB.java
23 (...)

24 @MessageDriven(activationConfig = {
25     @ActivationConfigProperty(
26         propertyName = "clientId",
27         propertyValue = "testClient"),
28     @ActivationConfigProperty(
29         propertyName = "groupIdConfig",
30         propertyValue = "testGroup"),
31     @ActivationConfigProperty(
32         propertyName = "topics",
33         propertyValue = "${ENV=TOPIC_NAME}"),
34     @ActivationConfigProperty(
35         propertyName =
36 "bootstrapServersConfig",
37         propertyValue = "${ENV=KAFKA_BROKER}"),
38     @ActivationConfigProperty(
39         propertyName = "keyDeserializer",
40         propertyValue =
41 "...StringDeserializer"),
42     @ActivationConfigProperty(
43         propertyName = "valueDeserializer",
44         propertyValue =
45 ...StringDeserializer")
46 })
47 public class KafkaMDB implements
48 KafkaListener {
49     public KafkaMDB() {    }

50     @OnRecord( topics={"payara-kafka-mdb-
51 topic"})
52     public void
53     test(ConsumerRecord<Object, Object> record) {
54         System.out.println("Payara Kafka MDB
55 record " + record );
56     }
57 }

58 }

59 }

60 }
```

```
61
62 --Producer.java
63
64 (...)

65 public class Producer implements Runnable {
66     KafkaProducer<String, String> produc;
67     String topic = null;
68
69     public Producer() {
70         Properties consumerProps = new
71         Properties();
72         consumerProps.put(
73             ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
74             System.getenv().getOrDefault("KAFKA_BROKER",
75             "192.168.99.100:9092"));
76         consumerProps.put(
77             ProducerConfig.KEY_SERIALIZER_CLASSG,
78             "... StringSerializer");
79         consumerProps.put(
80             ProducerConfig.VALUE_SERIALIZER_CLASS,
81             "...StringSerializer");
82
83         produc = new
84         KafkaProducer<>(consumerProps);
85         topic = System.getenv()
86             .getOrDefault("TOPIC_NAME", "test-
87 topic");
88     }
89
90     static Random rnd = new Random();
91
92     @Override
93     public void run() {
94         System.out.println("Producing to topic " +
95             topic);
96         while (true) {
97             String key = "";
98             produc.send(
99                 new ProducerRecord(topic, "key-" +
100 rnd.nextInt(10), "val-" + rnd.nextInt(10)),
101                 new Callback() {
102                     public void
103                     onCompletion(RecordMetadata rm, Exception
104                     excptn) {
105                         System.out.println("Sent data....");
106                     }
107                 });
108             try {
109                 Thread.sleep(10000); //take it easy!
110             } catch (InterruptedException ex) {
111             Logger.getLogger(Producer.class.getName()).lo
112             g(Level.SEVERE, null, ex);
113             }
114             }
115             }
116             }
117             }
118     }
119 }
```

Code 7 - A store catalogue

```
1 //some code omitted for sake of clarity and
2 space constraints
3
4 -- StockApplication.java
5 (...)

6 @ApplicationPath("/api")
7 @Path("/items")
8 @Stateless
9 @TransactionAttribute(TransactionAttributeTyp
10 e.NEVER)
11 public class StockApplication extends
12 Application {
13     @EJB CatalogueBean catalogue;
14
15     @POST
16     @Consumes(MediaType.APPLICATION_JSON)
17     @Produces(MediaType.APPLICATION_JSON)
18     public Item createItem(StockCreateRequest
19 item) throws IOException {
20         return catalogue
21             .addItem(item.getName(),
22 item.getStock()); }
23
24     @GET      @Path("/{id}")
25     @Produces(MediaType.APPLICATION_JSON)
26     public Item getItem(@PathParam("id") int
27 id) throws IOException {
28         return catalogue.getItem(id); }
29
30     @PUT      @Path("/{id}")
31     @Consumes(MediaType.APPLICATION_JSON)
32     @Produces(MediaType.APPLICATION_JSON)
33     public Item updateStock(@PathParam("id")
34 int id, StockUpdateRequest update) throws
35 IOException {
36         return catalogue
37             .addStock(id, update.getAmmount()); }
38 }

39
40 --Cart.java
41 (...)

42 @ManagedBean
43 @SessionScoped
44 public class Cart {
45     @EJB CartBean cart;
46
47     public Collection<Item> getItems() {
48         return cart.getItems();
49     }
50
51     public void add(Item item) {
52         if (!cart.add(item)) // Could not add
53     }
54
55     public void remove(Item item) {
56         if (!cart.remove(item))// Could not remove
57     }
58
59     public void purchase() {
60         try {
61             if (cart.purchase()) // successful
62                 else //No items in cart
63             } catch (IOException e) {
64                 // Purchase failed
65             }
66     }
67 }
68 (...)

69 --CatalogueBean.java
70 (...)

71 @Stateless
72 public class CatalogueBean {
73     (...)

74     public List<Item> getCatalogue() { (...) }
75     public Item getItem(int id) { (...) }
76
77     public Item addItem(String name, int stock)
78         { (...) }
79
80     public Item addStock(int id, int amount)
81         { (...) }
82
83     public void purchase(int id) { (...) }
84
85 }
86
87 -- Catalogue.java
88 (...)

89 @ManagedBean
90 public class Catalogue {
91     @EJB CatalogueBean catalogue;
92
93     public Collection<Item> getItems() throws
94 IOException {
95         return this.catalogue.getCatalogue(); }
96     }
97 }

98
99 -- CartBean.java
100 (...)

101 @Stateful
102 public class CartBean{
103     List<Item> items = new ArrayList<>();
104
105     public boolean add(Item item) {
106         return this.items.add(item); }
107
108     public boolean remove(Item item) {
109         return this.items.remove(item); }
110
111     public List<Item> getItems() {
112         return this.items; }
113
114     @TransactionAttribute(TransactionAttributeTyp
115 e.REQUIRED)
116     public boolean purchase() throws
117 IOException {
118         if (items.isEmpty()) { return false; }
119         for (Item item : items)
120             item.purchase();
121         clear();
122         return true;
123     }
124     @Remove
125     public void clear() { this.items.clear(); }
126 }

127
128 -- Item.java
129 public class Item{
130     private CatalogueBean catalogue;
131     private int id;
132     private String name;
133     private int stock;
134
135     // constructors, getter and setter omitted
136 }
```

