```java
@Entity
@Table(name = "categories")
public class Category implements java.io.Serializable {

	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue
	private Long id;

	private String categoryName;

	@OneToOne(mappedBy = "productCategory")
	private Product categoryProduct;

	public Category() {
		// Used by JPA.
	}

	public void setId(Long id) {
		this.id = id;
	}

	public Long getId() {
		return id;
	}

	public String getCategoryName() {
		return this.categoryName;
	}

	public void setCategoryName(String categoryName) {
		this.categoryName = categoryName;
	}

	public Product getCategoryProduct() {
		return this.categoryProduct;
	}

	public void setCategoryProduct(Product categoryProduct) {
		this.categoryProduct = categoryProduct;
	}
}
```

1

```java
public class CategoryRepository implements ServletContextListener {
```

```
45
46      @PersistenceUnit(unitName = "store-pu")
47      private EntityManagerFactory emf;
48
49      private EntityManager em;
50
51      public CategoryRepository() {
52              emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
53              System.out.println("[SERVLET-TEST-INFO]: CategoryRepository Constructor");
54      }
55
56      @Override
57      public void contextDestroyed(ServletContextEvent sce) {
58              if (emf.isOpen()) {
59                      emf.close();
60              }
61      }
62
63      @Override
64      public void contextInitialized(ServletContextEvent sce) {
65              ServletContext context = sce.getServletContext();
66              context.setAttribute("categoryRepository", this);
67              System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado!
CategoryRepository");
68      }
69
70      public final EntityManager entityManager() {
71              if (em == null || !em.isOpen()) {
72                      em = emf.createEntityManager();
73              }
74              return em;
75      }
76
77      public void persistOrMerge(Serializable entity, Serializable id) {
78              em = entityManager();
79
80              if (entity == null) {
81                      throw new IllegalArgumentException("entity");
82              }
83              try {
84                      em.getTransaction().begin();
85                      if (id == null) {
86                              em.persist(entity);
87                      } else {
88                              em.merge(entity);
89                      }
90                      em.getTransaction().commit();
91              } finally {
92                      em.close();
```

```
93              }
94      }
95
96      public Category findCategoryById(Long id) {
97              em = entityManager();
98              return em.find(Category.class, id);
99      }
100
101     public List findAllCategories() {
102             em = entityManager();
103             Query q = em.createQuery("select c from Category c");
104             return q.getResultList();
105     }
106
107     public List findAllDistinctCategories() {
108             em = entityManager();
109             Query q = em.createQuery("select DISTINCT (c.categoryName) from Category c");
110             return q.getResultList();
111     }
112
113     public Category newCategory() {
114             Category c = new Category();
115             persistOrMerge(c, c.getId());
116             return c;
117     }
118 }
```

2

```
119 public interface ControllerAction {
120
121     void execute(HttpServletRequest request, HttpServletResponse response);
122 }
```

3

```
123 public class ControllerServlet extends HttpServlet {
124
125     private static final long serialVersionUID = 1L;
126     private Map<String, ControllerAction> actions = new HashMap<String, ControllerAction>();
127     private static final String ACTION_IDENTIFIER = "action";
128
129     public void init() {
130             actions.put("goToHome", new GoToAction("home.jsp"));
131             actions.put("goToCatalog", new GoToAction("catalog.jsp"));
132             actions.put("goToCheckout", new GoToAction("checkout.jsp"));
133             //#if defined(DisplayByCategory)
134             actions.put("goToCatalogShowByCategory", new GoToAction("catalogByCategory.jsp"));
```

```
135                 //#endif
136                 //#if defined(DisplayWhatIsNew)
137                 actions.put("goToCatalogShowByNearestDate", new
GoToAction("catalogByNearestDate.jsp"));
138                 //#endif
139                 actions.put("goToSeller", new GoToAction("seller.jsp"));
140                 //#if defined(Paypal)
141                 actions.put("goToPaypal", new GoToAction("paypal.jsp"));
142                 //#endif
143                 actions.put("goToPayment", new GoToAction("payment.jsp"));
144                 //#if defined(Bankslip)
145                 actions.put("goToBankSlip", new GoToAction("bankslip.jsp"));
146                 //#endif
147                 actions.put("goToResponse", new GoToAction("response.jsp"));
148
149                 actions.put("verifyCatalogForm", new VerifyCatalogFormAction());
150                 actions.put("verifySellerForm", new VerifySellerFormAction());
151                 actions.put("verifyCheckoutForm", new VerifyCheckoutFormAction());
152
153                 actions.put("processSellerForm", new ProcessSellerFormAction());
154                 actions.put("processCheckoutForm", new ProcessCheckoutFormAction());
155         }
156
157         public void processAction(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
158
159                 String actionRequestParameter = request.getParameter(ACTION_IDENTIFIER);
160
161                 if (null == actionRequestParameter) {
162                         actionRequestParameter = "goToHome";
163                 }
164
165                 ControllerAction command = (ControllerAction) actions.get(actionRequestParameter);
166
167                 if (null == command) {
168                         throw new IllegalArgumentException("No command for form action: " +
actionRequestParameter);
169                 }
170                 command.execute(request, response);
171         }
172
173         public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
174                 processAction(request, response);
175         }
176
177         public void doGet(HttpServletRequest request, HttpServletResponse response)      throws
ServletException, IOException {
178                 processAction(request, response);
179         }
180 }
```

```java
@Entity
@Table(name = "customers")
public class Customer implements Serializable {

	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue
	private Long id;

	private String name;

	public void setName(String name) {
		this.name = name;
	}

	private String address;

	private String email;

	@OneToMany(targetEntity = Order.class, mappedBy = "customer", fetch = FetchType.LAZY)
	private List<Order> orders;

	public Customer() {
		// JPA
	}

	public Long getId() {
		return id;
	}

	public String getName() {
		return name;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	public String getAddress() {
		return address;
	}

	public void setOrders(List<Order> orders) {
		this.orders = orders;
	}

```

```
	public List<Order> getOrders() {
		return orders;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getEmail() {
		return email;
	}
}
```

5

```
public class CustomerRepository implements ServletContextListener {

	@PersistenceUnit(unitName = "store-pu")
	private EntityManagerFactory emf;

	private EntityManager em;

	public CustomerRepository() {
		emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
		System.out.println("[SERVLET-TEST-INFO]: CustomerRepository Constructor");
	}

	@Override
	public void contextDestroyed(ServletContextEvent sce) {
		if (emf.isOpen()) {
			emf.close();
		}
	}

	@Override
	public void contextInitialized(ServletContextEvent sce) {
		ServletContext context = sce.getServletContext();
		context.setAttribute("customerRepository", this);
		System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! CustomerRepository");
	}

	public final EntityManager entityManager() {
		if (em == null || !em.isOpen()) {
			em = emf.createEntityManager();
		}
		return em;
	}

```

```
273         public void persistOrMerge(Serializable entity, Serializable id) {
274                 em = entityManager();
275
276                 if (entity == null) {
277                         throw new IllegalArgumentException("entity");
278                 }
279                 try {
280                         em.getTransaction().begin();
281                         if (id == null) {
282                                 em.persist(entity);
283                         } else {
284                                 em.merge(entity);
285                         }
286                         em.getTransaction().commit();
287                 } finally {
288                         em.close();
289                 }
290         }
291
292         public Customer findCustomerById(Long id) {
293                 em = entityManager();
294                 return em.find(Customer.class, id);
295         }
296
297         public List findAllCustomers() {
298                 em = entityManager();
299                 Query q = em.createQuery("select c from Customer c");
300                 return q.getResultList();
301         }
302
303         public Customer newCustomer() {
304                 Customer c = new Customer();
305                 persistOrMerge(c, c.getId());
306                 return c;
307         }
308 }
```

6

```
309 public class GoToAction implements ControllerAction {
310
311         private String jspFileName;
312
313         public GoToAction(String jspFileName) {
314                 this.setJspFileName(jspFileName);
315         }
316
317         @Override
```

```
318         public void execute(HttpServletRequest request, HttpServletResponse response) {
319                 ServletContext context = request.getSession().getServletContext();
320
321                 try {
322
323                         context.getRequestDispatcher("/" + jspFileName).forward(request, response);
324                 } catch (Exception e) {
325                         e.printStackTrace();
326                 }
327         }
328
329         public void setJspFileName(String jspFileName) {
330                 this.jspFileName = jspFileName;
331         }
332
333         public String getJspFileName() {
334                 return jspFileName;
335         }
336 }
```

7

```
337 public abstract class ModelAction {
338
339         public Object execute(HttpServletRequest request, HttpServletResponse response) {
340                 setPersistenceRepository(request);
341                 getRequestParameters(request);
342                 return newEntity();
343         }
344
345         Object newEntity() {
346                 Object o = createEntity();
347                 setEntityAssociations(o);
348
349                 return o;
350         }
351
352         abstract Object createEntity();
353
354         abstract void getRequestParameters(HttpServletRequest request);
355
356         abstract void setPersistenceRepository(HttpServletRequest request);
357
358         abstract void setEntityAssociations(Object o);
359 }
```

8

```
360 public class NewCategoryAction extends ModelAction {
361
362         Object params[];
363         private CategoryRepository categoryRepository;
364
365         @Override
366         public void setPersistenceRepository(HttpServletRequest request) {
367                 ServletContext context = request.getSession().getServletContext();
368                 categoryRepository = (CategoryRepository) context.getAttribute("categoryRepository");
369         }
370
371         @Override
372         public Object createEntity() {
373                 Category c = categoryRepository.newCategory();
374                 setEntityAssociations(c);
375                 categoryRepository.persistOrMerge(c, c.getId());
376                 return c;
377         }
378
379         @Override
380         public void setEntityAssociations(Object o) {
381                 ((Category) o).setCategoryName((String) params[0]);
382         }
383
384         @Override
385         public void getRequestParameters(HttpServletRequest request) {
386                 params = new Object[1];
387                 params[0] = request.getParameter("CategoryName");
388         }
389 }
```

9

```
390 public class NewCustomerAction extends ModelAction {
391
392         Object params[];
393         private CustomerRepository customerRepository;
394
395         @Override
396         public void setPersistenceRepository(HttpServletRequest request) {
397                 ServletContext context = request.getSession().getServletContext();
398                 customerRepository = (CustomerRepository)
    context.getAttribute("customerRepository");
399         }
400
401         @Override
402         public Object createEntity() {
403                 Customer c = customerRepository.newCustomer();
404                 setEntityAssociations(c);
```

```
405                 return c;
406         }
407
408         @Override
409         public void setEntityAssociations(Object o) {
410                 ((Customer) o).setName((String) params[0]);
411                 ((Customer) o).setAddress((String) params[1]);
412                 ((Customer) o).setEmail((String) params[2]);
413         }
414
415         @Override
416         public void getRequestParameters(HttpServletRequest request) {
417                 params = new Object[3];
418                 params[0] = request.getParameter("CustomerName");
419                 params[1] = request.getParameter("CustomerAddress");
420                 params[2] = request.getParameter("CustomerMail");
421         }
422 }
```

10

```
423 public class NewOrderAction extends ModelAction {
424
425         Object params[];
426         private OrderRepository orderRepository;
427
428         @Override
429         public void setPersistenceRepository(HttpServletRequest request) {
430                 ServletContext context = request.getSession().getServletContext();
431                 orderRepository = (OrderRepository) context.getAttribute("orderRepository");
432         }
433
434         @Override
435         public Object createEntity() {
436                 Order o = orderRepository.newOrder();
437                 setEntityAssociations(o);
438                 orderRepository.persistOrMerge(o, o.getId());
439                 return o;
440         }
441
442         @Override
443         public void setEntityAssociations(Object o) {
444                 ((Order) o).setCustomer((Customer) params[0]);
445                 ((Order) o).setProducts((List<Product>) params[1]);
446         }
447
448         @Override
449         public void getRequestParameters(HttpServletRequest request) {
```

```
450                 params = new Object[2];
451                 params[0] = request.getAttribute("customer");
452                 params[1] = request.getAttribute("products");
453         }
454 }
```

11

```
455 public class NewPaymentAction extends ModelAction {
456
457         Object params[];
458         private PaymentRepository paymentRepository;
459
460         @Override
461         public void setPersistenceRepository(HttpServletRequest request) {
462                 ServletContext context = request.getSession().getServletContext();
463                 paymentRepository = (PaymentRepository) context.getAttribute("paymentRepository");
464         }
465
466         @Override
467         public Object createEntity() {
468                 Payment object = paymentRepository.newPayment();
469                 setEntityAssociations(object);
470                 paymentRepository.persistOrMerge(object, object.getId());
471                 return object;
472         }
473
474         @Override
475         public void setEntityAssociations(Object payment) {
476                 ((Payment) payment).setPaymentStatus("NOT YET PAID");
477                 ((Payment) payment).setPaymentOrder((Order) params[0]);
478                 ((Payment) payment).setPaymentType((String) params[1]);
479         }
480
481         @Override
482         public void getRequestParameters(HttpServletRequest request) {
483                 params = new Object[2];
484                 params[0] = request.getAttribute("order");
485                 params[1] = request.getParameter("paymentType");
486         }
487 }
```

12

```
488 public class NewProductAction extends ModelAction {
489
490         private ProductRepository productRepository;
491         Object params[];
```

```
492
493         @Override
494         public void setPersistenceRepository(HttpServletRequest request) {
495                 ServletContext context = request.getSession().getServletContext();
496                 productRepository = (ProductRepository) context.getAttribute("productRepository");
497         }
498
499         @Override
500         public Object createEntity() {
501                 Product p = productRepository.newProduct();
502                 setEntityAssociations(p);
503                 productRepository.persistOrMerge(p, p.getId());
504                 return p;
505         }
506
507         @Override
508         public void setEntityAssociations(Object product) {
509                 ((Product) product).setProductName((String) params[0]);
510                 ((Product) product).setProductDescription((String) params[1]);
511
512                 Double price;
513                 if (params[2] == null) {
514                         price = Double.valueOf("0");
515                 } else {
516                         price = Double.valueOf((String) params[2]);
517                 }
518                 ((Product) product).setProductPrice(price);
519                 //#if defined(DisplayWhatIsNew)
520                 ((Product) product).setProductInsertDate(new Date());
521                 //#endif
522         }
523
524         @Override
525         public void getRequestParameters(HttpServletRequest request) {
526                 params = new Object[3];
527                 params[0] = request.getParameter("ProductName");
528                 params[1] = request.getParameter("ProductDescription");
529                 params[2] = request.getParameter("ProductPrice");
530         }
531 }
```

13

```
532 public class NewSellerAction extends ModelAction {
533
534         SellerRepository sellerRepository;
535         Object params[];
536
```

```
	@Override
	public Object createEntity() {
		Seller s = sellerRepository.newSeller();
		setEntityAssociations(s);
		sellerRepository.persistOrMerge(s, s.getId());
		return s;
	}

	@Override
	public void setEntityAssociations(Object seller) {
		((Seller) seller).setName((String) params[0]);
		((Seller) seller).setAddress((String) params[1]);
		((Seller) seller).setEmail((String) params[2]);
	}

	@Override
	public void setPersistenceRepository(HttpServletRequest request) {
		ServletContext context = request.getSession().getServletContext();
		sellerRepository = (SellerRepository) context.getAttribute("sellerRepository");
	}

	@Override
	public void getRequestParameters(HttpServletRequest request) {
		params = new Object[3];
		params[0] = request.getParameter("SellerName");
		params[1] = request.getParameter("SellerAddress");
		params[2] = request.getParameter("SellerMail");
	}
}
```

14

```
@Entity
@Table(name = "orders")
public class Order implements Serializable {

	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue
	private Long id;

	@ManyToOne(targetEntity = Customer.class, fetch = FetchType.LAZY)
	private Customer customer;

	@OneToMany(targetEntity = Product.class, mappedBy = "productOrder", fetch = FetchType.LAZY)
	@Column(name = "orderProducts")
	private List<Product> products;
```

```
582
583         @OneToOne(mappedBy = "paymentOrder")
584         private Payment orderPayment;
585
586         public Order() {
587                 // JPA
588         }
589
590         public Order(Customer c) {
591                 customer = c;
592         }
593
594         public void setId(Long id) {
595                 this.id = id;
596         }
597
598         public Long getId() {
599                 return id;
600         }
601
602         public void setCustomer(Customer client) {
603                 this.customer = client;
604         }
605
606         public Customer getClient() {
607                 return customer;
608         }
609
610         public void setProducts(List<Product> products) {
611                 this.products = products;
612         }
613
614         public List<Product> getProducts() {
615                 return products;
616         }
617
618         public Payment getOrderPayment() {
619                 return this.orderPayment;
620         }
621
622         public void setOrderPayment(Payment orderPayment) {
623                 this.orderPayment = orderPayment;
624         }
625 }
```



```
626 public class OrderRepository implements ServletContextListener {
```

```
627
628         @PersistenceUnit(unitName = "store-pu")
629         private EntityManagerFactory emf;
630
631         private EntityManager em;
632
633         public OrderRepository() {
634                 emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
635                 System.out.println("[SERVLET-TEST-INFO]: OrderRepository Constructor");
636         }
637
638         @Override
639         public void contextDestroyed(ServletContextEvent sce) {
640                 if (emf.isOpen()) {
641                         emf.close();
642                 }
643         }
644
645         @Override
646         public void contextInitialized(ServletContextEvent sce) {
647                 ServletContext context = sce.getServletContext();
648                 context.setAttribute("orderRepository", this);
649                 System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! OrderRepository");
650         }
651
652         public final EntityManager entityManager() {
653                 if (em == null || !em.isOpen()) {
654                         em = emf.createEntityManager();
655                 }
656                 return em;
657         }
658
659         public void persistOrMerge(Serializable entity, Serializable id) {
660                 em = entityManager();
661
662                 if (entity == null) {
663                         throw new IllegalArgumentException("entity");
664                 }
665                 try {
666                         em.getTransaction().begin();
667                         if (id == null) {
668                                 em.persist(entity);
669                         } else {
670                                 em.merge(entity);
671                         }
672                         em.getTransaction().commit();
673                 } finally {
674                         em.close();
675                 }
```

```
        }

        public Order findOrderById(Long id) {
                em = entityManager();
                return em.find(Order.class, id);
        }

        public List<Order> findAllOrders() {
                em = entityManager();
                Query q = em.createQuery("select o from Order o");
                return q.getResultList();
        }

        public Order newOrder() {
                Order o = new Order();
                persistOrMerge(o, o.getId());

                return o;
        }
}
```

16

```
@Entity
@Table(name = "payments")
public class Payment implements java.io.Serializable {

        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue
        private Long id;

        private String paymentType;

        @OneToOne
        private Order paymentOrder;

        private String paymentStatus;

        public Payment() {
                // Used by JPA.
        }

        public void setId(Long id) {
                this.id = id;
        }

```

```
721         public Long getId() {
722                 return id;
723         }
724
725         public String getPaymentStatus() {
726                 return this.paymentStatus;
727         }
728
729         public void setPaymentStatus(String paymentStatus) {
730                 this.paymentStatus = paymentStatus;
731         }
732
733         public Order getPaymentOrder() {
734                 return this.paymentOrder;
735         }
736
737         public void setPaymentOrder(Order paymentOrder) {
738                 this.paymentOrder = paymentOrder;
739         }
740
741         public void setPaymentType(String paymentType) {
742                 this.paymentType = paymentType;
743         }
744
745         public String getPaymentType() {
746                 return paymentType;
747         }
748 }
```

17

```
749 public class PaymentRepository implements ServletContextListener {
750
751         @PersistenceUnit(unitName = "store-pu")
752         private EntityManagerFactory emf;
753
754         private EntityManager em;
755
756         public PaymentRepository() {
757                 emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
758                 System.out.println("[SERVLET-TEST-INFO]: PaymentRepository Constructor");
759         }
760
761         @Override
762         public void contextDestroyed(ServletContextEvent sce) {
763                 if (emf.isOpen()) {
764                         emf.close();
765                 }
```

```
766         }
767
768         @Override
769         public void contextInitialized(ServletContextEvent sce) {
770                 ServletContext context = sce.getServletContext();
771                 context.setAttribute("paymentRepository", this);
772                 System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! PaymentRepository");
773         }
774
775         public final EntityManager entityManager() {
776                 if (em == null || !em.isOpen()) {
777                         em = emf.createEntityManager();
778                 }
779                 return em;
780         }
781
782         public void persistOrMerge(Serializable entity, Serializable id) {
783                 em = entityManager();
784
785                 if (entity == null) {
786                         throw new IllegalArgumentException("entity");
787                 }
788                 try {
789                         em.getTransaction().begin();
790                         if (id == null) {
791                                 em.persist(entity);
792                         } else {
793                                 em.merge(entity);
794                         }
795                         em.getTransaction().commit();
796                 } finally {
797                         em.close();
798                 }
799         }
800
801         public Payment findPaymentById(Long id) {
802                 em = entityManager();
803                 return em.find(Payment.class, id);
804         }
805
806         public List findAllPayments() {
807                 em = entityManager();
808                 Query q = em.createQuery("select obj from Payment obj");
809                 return q.getResultList();
810         }
811
812         public Payment newPayment() {
813                 Payment object = new Payment();
814
```

```
815                 persistOrMerge(object, object.getId());
816                 return object;
817         }
818 }
```

18

```
819 public class ProcessCheckoutFormAction implements ControllerAction {
820
821         @Override
822         public void execute(HttpServletRequest request, HttpServletResponse response) {
823                 ControllerAction paymentAction = null;
824                 createEntities(request, response);
825
826                 String paymentType = request.getParameter("PaymentType");
827
828                 paymentAction = selectPaymentMethod(paymentAction, paymentType);
829
830                 paymentAction.execute(request, response);
831         }
832
833         private ControllerAction selectPaymentMethod(ControllerAction paymentAction, String
    paymentType) {
834                 if (paymentType.equals("Default")) {
835                         paymentAction = new GoToAction("payment.jsp");
836                 }
837                 //#if defined(Bankslip)
838                 if (paymentType.equals("Bankslip")) {
839                         paymentAction = new GoToAction("bankslip.jsp");
840                 }
841                 //#endif
842                 //#if defined(Paypal)
843                 if (paymentType.equals("Paypal")) {
844                         paymentAction = new GoToAction("paypal.jsp");
845                 }
846                 //#endif
847                 return paymentAction;
848         }
849
850         private void createEntities(HttpServletRequest request,     HttpServletResponse response) {
851                 NewCustomerAction newCustomerAction = new NewCustomerAction();
852                 Customer c = (Customer) newCustomerAction.execute(request, response);
853
854                 request.setAttribute("customer", c);
855                 request.setAttribute("products", request.getSession().getAttribute("products"));
856
857                 NewOrderAction newOrderAction = new NewOrderAction();
858                 Order o = (Order) newOrderAction.execute(request, response);
859                 request.setAttribute("order", o);
```

```
860
861             NewPaymentAction newPaymentAction = new NewPaymentAction();
862             newPaymentAction.execute(request, response);
863       }
864 }
```

19

```
865 public class ProcessSellerFormAction implements ControllerAction {
866
867       @Override
868       public void execute(HttpServletRequest request, HttpServletResponse response) {
869             try {
870
871                   Product p = createProduct(request, response);
872                   setProductAssociations(request, response, p);
873                   updateEntity(request, p);
874                   forward(request, response);
875
876             } catch (Exception e) {
877                   e.printStackTrace();
878             }
879       }
880
881       private Product createProduct(HttpServletRequest request, HttpServletResponse response) {
882             NewProductAction newProductAction = new NewProductAction();
883             Product p = (Product) newProductAction.execute(request, response);
884             return p;
885       }
886
887       private void forward(HttpServletRequest request, HttpServletResponse response) {
888             try {
889                   request.setAttribute("message",   "<br /> Product inserted with success");
890
891                   GoToAction responseAction = new GoToAction("response.jsp");
892                   responseAction.execute(request, response);
893             } catch (Exception e) {
894                   e.printStackTrace();
895             }
896       }
897
898       private void updateEntity(HttpServletRequest request, Product p) {
899             ServletContext context = request.getSession().getServletContext();
900             ProductRepository productRepository = (ProductRepository)
context.getAttribute("productRepository");
901             productRepository.persistOrMerge(p, p.getId());
902       }
903
904       private void setProductAssociations(HttpServletRequest request,    HttpServletResponse
response, Product p) {
```

```
                setProductSeller(request, response, p);
                setProductCategory(request, response, p);
        }

        private void setProductCategory(HttpServletRequest request, HttpServletResponse response,
Product p) {
                NewCategoryAction newCategoryAction = new NewCategoryAction();
                Category c = (Category) newCategoryAction.execute(request, response);
                p.setProductCategory(c);
        }

        private void setProductSeller(HttpServletRequest request, HttpServletResponse response,
Product p) {
                NewSellerAction newSellerAction = new NewSellerAction();
                Seller s = (Seller) newSellerAction.execute(request, response);
                p.setProductSeller(s);
        }
}
```

20

```
@Entity
@Table(name = "products")
public class Product implements Serializable {

        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue
        private Long id;

        private String productName;

        private String productDescription;

        private Double productPrice;

        @ManyToOne(targetEntity = Order.class, fetch = FetchType.LAZY)
        private Order productOrder;

        @OneToOne
        private Seller productSeller;

        @OneToOne
        private Category productCategory;
        //#if defined(DisplayWhatIsNew)
        private Date productInsertDate;

        //#endif

```

```
950         public Product() {
951                 // JPA
952         }
953
954         public Long getId() {
955                 return id;
956         }
957
958         public void setId(Long id) {
959                 this.id = id;
960         }
961
962         public void setProductName(String productName) {
963                 this.productName = productName;
964         }
965
966         public String getProductName() {
967                 return productName;
968         }
969
970         public void setProductDescription(String productDescription) {
971                 this.productDescription = productDescription;
972         }
973
974         public String getProductDescription() {
975                 return productDescription;
976         }
977
978         public void setProductPrice(Double productPrice) {
979                 this.productPrice = productPrice;
980         }
981
982         public Double getProductPrice() {
983                 return productPrice;
984         }
985
986         public void setProductOrder(Order productOrder) {
987                 this.productOrder = productOrder;
988         }
989
990         public Order getProductOrder() {
991                 return productOrder;
992         }
993
994         public void setProductSeller(Seller productSeller) {
995                 this.productSeller = productSeller;
996         }
997
998         public Seller getProductSeller() {
```

```
999             return productSeller;
100     }
100
100     //#if defined(DisplayWhatIsNew)
100     public void setProductInsertDate(Date productInsertDate) {
100             this.productInsertDate = productInsertDate;
100     }
100
100     public Date getProductInsertDate() {
100             return productInsertDate;
100     }
101
101     //#endif
101     public void setProductCategory(Category productCategory) {
101             this.productCategory = productCategory;
101     }
101
101     public Category getProductCategory() {
101             return productCategory;
101     }
101 }
21
```

```
102 public class ProductRepository implements ServletContextListener {
102
102     @PersistenceUnit(unitName = "store-pu")
102     private EntityManagerFactory emf;
102
102     private EntityManager em;
102
102     public ProductRepository() {
102             emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
102             System.out.println("[SERVLET-TEST-INFO]: ProductRepository Constructor");
103     }
103
103     @Override
103     public void contextDestroyed(ServletContextEvent sce) {
103             if (emf.isOpen()) {
103                     emf.close();
103             }
103     }
103
103     @Override
104     public void contextInitialized(ServletContextEvent sce) {
104             ServletContext context = sce.getServletContext();
104             context.setAttribute("productRepository", this);
104             System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado!
ProductRepository");
104     }
104
```

```
104         public final EntityManager entityManager() {
104                 if (em == null || !em.isOpen()) {
104                         em = emf.createEntityManager();
104                 }
105                 return em;
105         }
105
105         public void persistOrMerge(Serializable entity, Serializable id) {
105                 em = entityManager();
105
105                 if (entity == null) {
105                         throw new IllegalArgumentException("entity");
105                 }
105                 try {
106                         em.getTransaction().begin();
106                         if (id == null) {
106                                 em.persist(entity);
106                         } else {
106                                 em.merge(entity);
106                         }
106                         em.getTransaction().commit();
106                 } finally {
106                         em.close();
106                 }
107         }
107
107         public Product newProduct() {
107                 Product p = new Product();
107                 persistOrMerge(p, p.getId());
107                 return p;
107         }
107
107         public Product findProductById(Long id) {
107                 em = entityManager();
108                 return em.find(Product.class, id);
108         }
108
108         public List findAllProducts() {
108                 em = entityManager();
108                 Query q = em.createQuery("select p from Product p");
108                 return q.getResultList();
108         }
108
108         public List findAllProductsAvailable() {
109                 em = entityManager();
109                 Query q = em.createQuery("select p from Product p where p.productOrder = NULL");
109                 return q.getResultList();
109         }
109
```

```
109         //#if defined(DisplayWhatIsNew)
109         public List findAllProductsAvailableByNearestDate() {
109                 em = entityManager();
109                 Query q = em.createQuery("select p from Product p where p.productOrder = NULL
ORDER BY p.productInsertDate DESC");
109                 return q.getResultList();
110         }
110         //#endif
110
110         //#if defined(DisplayByCategory)
110         public List findAllProductsByCategory(String category) {
110                 em = entityManager();
110                 Query q = em.createQuery("select p from Product p where p.productOrder = NULL AND
p.productCategory.categoryName=:cat");
110                 q.setParameter("cat", category);
110                 return q.getResultList();
110         }
111         //#endif
111 }
```

22

```
111 @Entity
111 @Table(name = "sellers")
111 public class Seller implements Serializable {
111
111         private static final long serialVersionUID = 1L;
111
111         @Id
111         @GeneratedValue
112         private Long id;
112
112         private String name;
112
112         private String address;
112
112         private String email;
112
112         @OneToOne(mappedBy = "productSeller")
112         private Product product;
113
113         public Seller() {
113                 // JPA
113         }
113
113         public Long getId() {
113                 return id;
113         }
113
113         public void setId(Long id) {
```

```
114                 this.id = id;
114         }
114
114         public String getName() {
114                 return name;
114         }
114
114         public void setName(String name) {
114                 this.name = name;
114         }
115
115         public void setEmail(String email) {
115                 this.email = email;
115         }
115
115         public String getEmail() {
115                 return email;
115         }
115
115         public void setAddress(String address) {
116                 this.address = address;
116         }
116
116         public String getAddress() {
116                 return address;
116         }
116
116         public void setProduct(Product product) {
116                 this.product = product;
116         }
117
117         public Product getProduct() {
117                 return product;
117         }
117 }
```

23

```
117 public class SellerRepository implements ServletContextListener {
117
117         @PersistenceUnit(unitName = "store-pu")
117         private EntityManagerFactory emf;
117
118         private EntityManager em;
118
118         public SellerRepository() {
118                 emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
118                 System.out.println("[SERVLET-TEST-INFO]: SellerRepository Constructor");
```

```
	}

	@Override
	public void contextDestroyed(ServletContextEvent sce) {
		if (emf.isOpen()) {
			emf.close();
		}
	}

	@Override
	public void contextInitialized(ServletContextEvent sce) {
		ServletContext context = sce.getServletContext();
		context.setAttribute("sellerRepository", this);
		System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! SellerRepository");
	}

	public final EntityManager entityManager() {
		if (em == null || !em.isOpen()) {
			em = emf.createEntityManager();
		}
		return em;
	}

	public void persistOrMerge(Serializable entity, Serializable id) {
		em = entityManager();

		if (entity == null) {
			throw new IllegalArgumentException("entity");
		}
		try {
			em.getTransaction().begin();
			if (id == null) {
				em.persist(entity);
			} else {
				em.merge(entity);
			}
			em.getTransaction().commit();
		} finally {
			em.close();
		}
	}

	public Seller findSellerById(Long id) {
		em = entityManager();
		return em.find(Seller.class, id);
	}

	public Seller findSellerByEmail(String sellerEmail) {
		em = entityManager();
```

```
123             return (Seller) em.createQuery("select s from Seller s where s.email = '" +
sellerEmail + "'").getSingleResult();
123         }
123
123         public List<Seller> findAllSellers() {
123             em = entityManager();
123             Query q = em.createQuery("select s from Seller s");
124             return q.getResultList();
124         }
124
124         public Seller newSeller() {
124             Seller s = new Seller();
124             persistOrMerge(s, s.getId());
124             return s;
124         }
124 }
```

24

```
124 public class VerifyCatalogFormAction implements ControllerAction {
125
125         @Override
125         public void execute(HttpServletRequest request, HttpServletResponse response) {
125             String[] productsIds = request.getParameterValues("productsIds");
125
125             if (productsIds == null) {
125                 request.setAttribute("message",   "<br/ > ERROR: You must select at least a
product!");
125
125                 GoToAction responseAction = new GoToAction("response.jsp");
125                 responseAction.execute(request, response);
126             }
126
126             else {
126                 getProductList(request, productsIds);
126
126                 GoToAction checkoutAction = new GoToAction("checkout.jsp");
126                 checkoutAction.execute(request, response);
126             }
126         }
126
127         private void getProductList(HttpServletRequest request, String[] productsIds) {
127             ServletContext context = request.getSession().getServletContext();
127             ProductRepository productRepository = (ProductRepository)
context.getAttribute("productRepository");
127
127             List<Product> products = new ArrayList();
127
127             for (String productId : productsIds) {
127                 products.add(productRepository.findProductById((Long.valueOf(productId))));
```

```
127                 }
127
128                 HttpSession session = request.getSession();
128                 session.setAttribute("products", products);
128         }
128 }
```

25

```
128 public class VerifyCheckoutFormAction implements ControllerAction {
128
128         @Override
128         public void execute(HttpServletRequest request, HttpServletResponse response) {
128                 List params = getRequestParameters(request);
128                 Boolean emptyFields = false;
129
129                 for (Object param : params) {
129                         if (param == null || param.equals("")) {
129                                 emptyFields = true;
129                         }
129                 }
129
129                 if (emptyFields) {
129                         request.setAttribute("message",   "Form with blank fields! Complete your
form first!");
129                         GoToAction responseAction = new GoToAction("response.jsp");
130                         responseAction.execute(request, response);
130                 } else {
130                         ProcessCheckoutFormAction checkoutFormAction = new
ProcessCheckoutFormAction();
130                         checkoutFormAction.execute(request, response);
130                 }
130         }
130
130         private List getRequestParameters(HttpServletRequest request) {
130                 List params = new ArrayList();
130                 addParameters(request, params);
131                 return params;
131         }
131
131         private void addParameters(HttpServletRequest request, List params) {
131                 params.add(request.getParameter("CustomerName"));
131                 params.add(request.getParameter("CustomerAddress"));
131                 params.add(request.getParameter("CustomerMail"));
131                 params.add(request.getParameter("paymentType"));
131         }
131 }
```

26

```
public class VerifySellerFormAction implements ControllerAction {

	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		List params = getRequestParameters(request);
		Boolean emptyFields = false;

		for (Object param : params) {
			if (param == null || param.equals("")) {
				emptyFields = true;
			}
		}

		if (emptyFields) {
			request.setAttribute("message",  "Form with blank fields! Complete your form first!");
			GoToAction responseAction = new GoToAction("response.jsp");
			responseAction.execute(request, response);
		} else {
			ProcessSellerFormAction sellerFormAction = new ProcessSellerFormAction();
			sellerFormAction.execute(request, response);
		}
	}

	private List getRequestParameters(HttpServletRequest request) {
		List params = new ArrayList();
		addParameters(request, params);
		return params;
	}

	private void addParameters(HttpServletRequest request, List params) {
		params.add(request.getParameter("CategoryName"));
		params.add(request.getParameter("SellerName"));
		params.add(request.getParameter("SellerAddress"));
		params.add(request.getParameter("SellerMail"));
		params.add(request.getParameter("ProductName"));
		params.add(request.getParameter("ProductDescription"));
		params.add(request.getParameter("ProductPrice"));
	}
}
```