

	@Entity
	@Table(name = "categories")
1	public class Category implements java.io.Serializable {
2	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue
3	private Long id;
4	private String categoryName;
	@OneToOne(mappedBy = "productCategory")
5	private Product categoryProduct;
6	public Category() {
	// Used by JPA.
7	}
8	public void setId(Long id) {
9	this.id = id;
10	}
11	public Long getId() {
12	return id;
12	}
13	public String getCategoryName() {
14	return this.categoryName;
15	}
16	public void setCategoryName(String categoryName) {
17	this.categoryName = categoryName;
18	}
19	public Product getCategoryProduct() {
20	return this.categoryProduct;
21	}
22	public void setCategoryProduct(Product categoryProduct) {
23	this.categoryProduct = categoryProduct;
24	}
25	}

1	public class CategoryRepository implements ServletContextListener {
---	---

	@PersistenceUnit(unitName = "store-pu")
2	private EntityManagerFactory emf;
3	private EntityManager em;
4	public CategoryRepository() {
5	emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
6	System.out.println("[SERVLET-TEST-INFO]: CategoryRepository Constructor");
7	}
	@Override
8	public void contextDestroyed(ServletContextEvent sce) {
9	if (emf.isOpen()) {
10	emf.close();
11	}
12	}
	@Override
13	public void contextInitialized(ServletContextEvent sce) {
14	ServletContext context = sce.getServletContext();
15	context.setAttribute("categoryRepository", this);
16	System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! CategoryRepository");
17	}
18	public final EntityManager entityManager() {
19	if (em == null !em.isOpen()) {
20	em = emf.createEntityManager();
21	}
22	return em;
23	}
24	public void persistOrMerge(Serializable entity, Serializable id) {
25	em = entityManager();
26	if (entity == null) {
27	throw new IllegalArgumentException("entity");
28	}
29	try {
30	em.getTransaction().begin();
31	if (id == null) {
32	em.persist(entity);
33	} else {
34	em.merge(entity);
35	}
36	em.getTransaction().commit();
37	} finally {
38	em.close();

39	}
40	}
41	public Category findCategoryById(Long id) {
42	em = entityManager();
43	return em.find(Category.class, id);
44	}
45	public List findAllCategories() {
46	em = entityManager();
47	Query q = em.createQuery("select c from Category c");
48	return q.getResultList();
49	}
50	public List findAllDistinctCategories() {
51	em = entityManager();
52	Query q = em.createQuery("select DISTINCT (c.categoryName) from Category c");
53	return q.getResultList();
54	}
55	public Category newCategory() {
56	Category c = new Category();
57	persistOrMerge(c, c.getId());
58	return c;
59	}
60	}

2

1	public interface ControllerAction {
2	void execute(HttpServletRequest request, HttpServletResponse response);
3	}

3

1	public class ControllerServlet extends HttpServlet {
2	private static final long serialVersionUID = 1L;
3	private Map<String, ControllerAction> actions = new HashMap<String, ControllerAction>();
4	private static final String ACTION_IDENTIFIER = "action";
5	public void init() {
6	actions.put("goToHome", new GoToAction("home.jsp"));
7	actions.put("goToCatalog", new GoToAction("catalog.jsp"));
8	actions.put("goToCheckout", new GoToAction("checkout.jsp"));
	//#if defined(DisplayByCategory)
9	actions.put("goToCatalogShowByCategory", new GoToAction("catalogByCategory.jsp"));

	//#endif
	//#if defined(DisplayWhatIsNew)
10	actions.put("goToCatalogShowByNearestDate", new GoToAction("catalogByNearestDate.jsp"));
	//#endif
11	actions.put("goToSeller", new GoToAction("seller.jsp"));
	//#if defined(Paypal)
12	actions.put("goToPaypal", new GoToAction("paypal.jsp"));
	//#endif
13	actions.put("goToPayment", new GoToAction("payment.jsp"));
	//#if defined(Bankslip)
14	actions.put("goToBankSlip", new GoToAction("bankslip.jsp"));
	//#endif
15	actions.put("goToResponse", new GoToAction("response.jsp"));
16	actions.put("verifyCatalogForm", new VerifyCatalogFormAction());
17	actions.put("verifySellerForm", new VerifySellerFormAction());
18	actions.put("verifyCheckoutForm", new VerifyCheckoutFormAction());
19	actions.put("processSellerForm", new ProcessSellerFormAction());
20	actions.put("processCheckoutForm", new ProcessCheckoutFormAction());
21	}
22	public void processAction(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
23	String actionRequestParameter = request.getParameter(ACTION_IDENTIFIER);
24	if (null == actionRequestParameter) {
25	actionRequestParameter = "goToHome";
26	}
27	ControllerAction command = (ControllerAction) actions.get(actionRequestParameter);
28	if (null == command) {
29	throw new IllegalArgumentException("No command for form action: " + actionRequestParameter);
30	}
31	command.execute(request, response);
32	}
33	public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
34	processAction(request, response);
35	}
36	public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
37	processAction(request, response);
38	}
39	}

	@Entity
	@Table(name = "customers")
1	public class Customer implements Serializable {
2	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue
3	private Long id;
4	private String name;
5	public void setName(String name) {
6	this.name = name;
7	}
8	private String address;
9	private String email;
	@OneToMany(targetEntity = Order.class, mappedBy = "customer", fetch = FetchType.LAZY)
10	private List<Order> orders;
11	public Customer() {
	// JPA
13	}
14	public Long getId() {
15	return id;
16	}
17	public String getName() {
18	return name;
19	}
20	public void setAddress(String address) {
21	this.address = address;
22	}
23	public String getAddress() {
24	return address;
25	}
26	public void setOrders(List<Order> orders) {
27	this.orders = orders;
28	}

29	<code>public List<Order> getOrders() {</code>
30	<code> return orders;</code>
31	<code>}</code>
32	<code>public void setEmail(String email) {</code>
33	<code> this.email = email;</code>
34	<code>}</code>
35	<code>public String getEmail() {</code>
36	<code> return email;</code>
37	<code>}</code>
38	<code>}</code>

5

1	<code>public class CustomerRepository implements ServletContextListener {</code>
	<code> @PersistenceUnit(unitName = "store-pu")</code>
2	<code> private EntityManagerFactory emf;</code>
3	<code> private EntityManager em;</code>
4	<code> public CustomerRepository() {</code>
5	<code> emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");</code>
6	<code> System.out.println("[SERVLET-TEST-INFO]: CustomerRepository Constructor");</code>
7	<code> }</code>
	<code> @Override</code>
8	<code> public void contextDestroyed(ServletContextEvent sce) {</code>
9	<code> if (emf.isOpen()) {</code>
10	<code> emf.close();</code>
11	<code> }</code>
12	<code> }</code>
	<code> @Override</code>
13	<code> public void contextInitialized(ServletContextEvent sce) {</code>
14	<code> ServletContext context = sce.getServletContext();</code>
15	<code> context.setAttribute("customerRepository", this);</code>
16	<code> System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado!</code>
17	<code> CustomerRepository");</code>
	<code> }</code>
18	<code> public final EntityManager entityManager() {</code>
19	<code> if (em == null !em.isOpen()) {</code>
20	<code> em = emf.createEntityManager();</code>
21	<code> }</code>
22	<code> return em;</code>
23	<code> }</code>

24	public void persistOrMerge(Serializable entity, Serializable id) {
25	em = entityManager();
26	if (entity == null) {
27	throw new IllegalArgumentException("entity");
28	}
29	try {
30	em.getTransaction().begin();
31	if (id == null) {
32	em.persist(entity);
33	} else {
34	em.merge(entity);
35	}
36	em.getTransaction().commit();
37	} finally {
38	em.close();
39	}
40	}
41	public Customer findCustomerById(Long id) {
42	em = entityManager();
43	return em.find(Customer.class, id);
44	}
45	public List findAllCustomers() {
46	em = entityManager();
47	Query q = em.createQuery("select c from Customer c");
48	return q.getResultList();
49	}
50	public Customer newCustomer() {
51	Customer c = new Customer();
52	persistOrMerge(c, c.getId());
53	return c;
54	}
55	}

6

1	public class GoToAction implements ControllerAction {
2	private String jspFileName;
3	public GoToAction(String jspFileName) {
4	this. setJspFileName(jspFileName);
5	}
	@Override

6	public void execute(HttpServletRequest request, HttpServletResponse response) {
7	ServletContext context = request.getSession().getServletContext();
8	try {
9	context.getRequestDispatcher("/") + jspFileName).forward(request, response);
10	} catch (Exception e) {
11	e.printStackTrace();
12	}
13	}
14	public void setJspFileName(String jspFileName) {
15	this.jspFileName = jspFileName;
16	}
17	public String getJspFileName() {
18	return jspFileName;
19	}
20	}

7

1	public abstract class ModelAction {
2	public Object execute(HttpServletRequest request, HttpServletResponse response) {
3	setPersistenceRepository(request);
4	getRequestParameters(request);
5	return newEntity();
6	}
7	Object newEntity() {
8	Object o = createEntity();
9	setEntityAssociations(o);
10	return o;
11	}
12	abstract Object createEntity();
13	abstract void getRequestParameters(HttpServletRequest request);
14	abstract void setPersistenceRepository(HttpServletRequest request);
15	abstract void setEntityAssociations(Object o);
16	}

8

1	public class NewCategoryAction extends ModelAction {
2	Object params [];
3	private CategoryRepository categoryRepository ;
4	@Override
5	public void setPersistenceRepository(HttpServletRequest request) {
6	ServletContext context = request.getSession().getServletContext();
7	categoryRepository = (CategoryRepository)
8	context.getAttribute("categoryRepository");
9	}
10	@Override
11	public Object createEntity() {
12	Category c = categoryRepository .newCategory();
13	setEntityAssociations(c);
14	categoryRepository .persistOrMerge(c, c.getId());
15	return c;
16	}
17	@Override
18	public void setEntityAssociations(Object o) {
19	((Category) o).setCategoryName((String) params [0]);
20	}
21	@Override
22	public void getRequestParameters(HttpServletRequest request) {
23	params = new Object[1];
24	params [0] = request.getParameter("CategoryName");
25	}
26	}

9

1	public class NewCustomerAction extends ModelAction {
2	Object params [];
3	private CustomerRepository customerRepository ;
4	@Override
5	public void setPersistenceRepository(HttpServletRequest request) {
6	ServletContext context = request.getSession().getServletContext();
7	customerRepository = (CustomerRepository)
8	context.getAttribute("customerRepository");
9	}
10	@Override
11	public Object createEntity() {
12	Customer c = customerRepository .newCustomer();
13	setEntityAssociations(c);

11	return c;
12	}
	@Override
13	public void setEntityAssociations(Object o) {
14	((Customer) o).setName((String) params[0]);
15	((Customer) o).setAddress((String) params[1]);
16	((Customer) o).setEmail((String) params[2]);
17	}
	@Override
18	public void getRequestParameters(HttpServletRequest request) {
19	params = new Object[3];
20	params[0] = request.getParameter("CustomerName");
21	params[1] = request.getParameter("CustomerAddress");
22	params[2] = request.getParameter("CustomerMail");
23	}
24	}

10

1	public class NewOrderAction extends ModelAction {
2	Object params[];
3	private OrderRepository orderRepository;
	@Override
4	public void setPersistenceRepository(HttpServletRequest request) {
5	ServletContext context = request.getSession().getServletContext();
6	orderRepository = (OrderRepository) context.getAttribute("orderRepository");
7	}
	@Override
8	public Object createEntity() {
9	Order o = orderRepository.newOrder();
10	setEntityAssociations(o);
11	orderRepository.persistOrMerge(o, o.getId());
12	return o;
13	}
	@Override
14	public void setEntityAssociations(Object o) {
15	((Order) o).setCustomer((Customer) params[0]);
16	((Order) o).setProducts((List<Product>) params[1]);
17	}
	@Override
18	public void getRequestParameters(HttpServletRequest request) {

19	<code>params = new Object[2];</code>
20	<code>params[0] = request.getAttribute("customer");</code>
21	<code>params[1] = request.getAttribute("products");</code>
22	<code>}</code>
23	<code>}</code>

11

1	<code>public class NewPaymentAction extends ModelAction {</code>
2	<code>Object params[];</code>
3	<code>private PaymentRepository paymentRepository;</code>
	<code>@Override</code>
4	<code>public void setPersistenceRepository(HttpServletRequest request) {</code>
5	<code>ServletContext context = request.getSession().getServletContext();</code>
6	<code>paymentRepository = (PaymentRepository)</code> <code>context.getAttribute("paymentRepository");</code>
7	<code>}</code>
	<code>@Override</code>
8	<code>public Object createEntity() {</code>
9	<code>Payment object = paymentRepository.newPayment();</code>
10	<code>setEntityAssociations(object);</code>
11	<code>paymentRepository.persistOrMerge(object, object.getId());</code>
12	<code>return object;</code>
13	<code>}</code>
	<code>@Override</code>
14	<code>public void setEntityAssociations(Object payment) {</code>
15	<code>((Payment) payment).setPaymentStatus("NOT YET PAID");</code>
16	<code>((Payment) payment).setPaymentOrder((Order) params[0]);</code>
17	<code>((Payment) payment).setPaymentType((String) params[1]);</code>
18	<code>}</code>
	<code>@Override</code>
19	<code>public void getRequestParameters(HttpServletRequest request) {</code>
20	<code>params = new Object[2];</code>
21	<code>params[0] = request.getAttribute("order");</code>
22	<code>params[1] = request.getParameter("paymentType");</code>
23	<code>}</code>
24	<code>}</code>

12

1	<code>public class NewProductAction extends ModelAction {</code>
2	<code>private ProductRepository productRepository;</code>
3	<code>Object params[];</code>

	@Override
4	public void setPersistenceRepository(HttpServletRequest request) {
5	ServletContext context = request.getSession().getServletContext();
6	productRepository = (ProductRepository)
7	context.getAttribute("productRepository");
	}
	@Override
8	public Object createEntity() {
9	Product p = productRepository.newProduct();
10	setEntityAssociations(p);
11	productRepository.persistOrMerge(p, p.getId());
12	return p;
13	}
	@Override
14	public void setEntityAssociations(Object product) {
15	((Product) product).setProductName((String) params[0]);
16	((Product) product).setProductDescription((String) params[1]);
17	Double price;
18	if (params[2] == null) {
19	price = Double.valueOf("0");
20	} else {
21	price = Double.valueOf((String) params[2]);
22	}
23	((Product) product).setProductPrice(price);
	//#if defined(DisplayWhatIsNew)
24	((Product) product).setProductInsertDate(new Date());
	//#endif
25	}
	@Override
26	public void getRequestParameters(HttpServletRequest request) {
27	params = new Object[3];
28	params[0] = request.getParameter("ProductName");
29	params[1] = request.getParameter("ProductDescription");
30	params[2] = request.getParameter("ProductPrice");
31	}
32	}

13

1	public class NewSellerAction extends ModelAction {
2	SellerRepository sellerRepository;
3	Object params[];

	@Override
4	public Object createEntity() {
5	Seller s = sellerRepository .newSeller();
6	setEntityAssociations(s);
7	sellerRepository .persistOrMerge(s, s.getId());
8	return s;
9	}
	@Override
10	public void setEntityAssociations(Object seller) {
11	((Seller) seller).setName((String) params [0]);
12	((Seller) seller).setAddress((String) params [1]);
13	((Seller) seller).setEmail((String) params [2]);
14	}
	@Override
15	public void setPersistenceRepository(HttpServletRequest request) {
16	ServletContext context = request.getSession().getServletContext();
17	sellerRepository = (SellerRepository) context.getAttribute("sellerRepository");
18	}
	@Override
19	public void getRequestParameters(HttpServletRequest request) {
20	params = new Object[3];
21	params [0] = request.getParameter("SellerName");
22	params [1] = request.getParameter("SellerAddress");
23	params [2] = request.getParameter("SellerMail");
24	}
25	}

14

	@Entity
	@Table(name = "orders")
1	public class Order implements Serializable {
2	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue
3	private Long id ;
	@ManyToOne(targetEntity = Customer.class, fetch = FetchType.LAZY)
4	private Customer customer ;
	@OneToMany(targetEntity = Product.class, mappedBy = "productOrder", fetch = FetchType.LAZY)
	@Column(name = "orderProducts")
5	private List<Product> products ;

	@OneToOne(mappedBy = "paymentOrder")
6	private Payment orderPayment;
7	public Order() {
	// JPA
8	}
9	public Order(Customer c) {
	customer = c;
10	}
11	public void setId(Long id) {
12	this.id = id;
13	}
14	public Long getId() {
15	return id;
16	}
17	public void setCustomer(Customer client) {
18	this.customer = client;
19	}
20	public Customer getClient() {
21	return customer;
22	}
23	public void setProducts(List<Product> products) {
24	this.products = products;
25	}
26	public List<Product> getProducts() {
27	return products;
28	}
29	public Payment getOrderPayment() {
30	return this.orderPayment;
31	}
32	public void setOrderPayment(Payment orderPayment) {
33	this.orderPayment = orderPayment;
34	}
35	}

15

1	public class OrderRepository implements ServletContextListener {
---	--

	@PersistenceUnit(unitName = "store-pu")
2	private EntityManagerFactory emf;
3	private EntityManager em;
4	public OrderRepository() {
5	emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
6	System.out.println("[SERVLET-TEST-INFO]: OrderRepository Constructor");
7	}
	@Override
8	public void contextDestroyed(ServletContextEvent sce) {
9	if (emf.isOpen()) {
10	emf.close();
11	}
12	}
	@Override
13	public void contextInitialized(ServletContextEvent sce) {
14	ServletContext context = sce.getServletContext();
15	context.setAttribute("orderRepository", this);
16	System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! OrderRepository");
17	}
18	public final EntityManager entityManager() {
19	if (em == null !em.isOpen()) {
20	em = emf.createEntityManager();
21	}
22	return em;
23	}
24	public void persistOrMerge(Serializable entity, Serializable id) {
25	em = entityManager();
26	if (entity == null) {
27	throw new IllegalArgumentException("entity");
28	}
29	try {
30	em.getTransaction().begin();
31	if (id == null) {
32	em.persist(entity);
33	} else {
34	em.merge(entity);
35	}
36	em.getTransaction().commit();
37	} finally {
38	em.close();

39	}
40	}
41	public Order findOrderById(Long id) {
42	em = entityManager();
43	return em.find(Order.class, id);
44	}
45	public List<Order> findAllOrders() {
46	em = entityManager();
47	Query q = em.createQuery("select o from Order o");
48	return q.getResultList();
49	}
50	public Order newOrder() {
51	Order o = new Order();
52	persistOrMerge(o, o.getId());
53	return o;
54	}
55	}

16

	@Entity
	@Table(name = "payments")
1	public class Payment implements java.io.Serializable {
2	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue
3	private Long id;
4	private String paymentType;
	@OneToOne
5	private Order paymentOrder;
6	private String paymentStatus;
7	public Payment() {
	// Used by JPA.
8	}
9	public void setId(Long id) {
10	this .id = id;
11	}

12	public Long getId() {
13	return id;
14	}
15	public String getPaymentStatus() {
16	return this.paymentStatus;
17	}
18	public void setPaymentStatus(String paymentStatus) {
19	this.paymentStatus = paymentStatus;
20	}
21	public Order getPaymentOrder() {
22	return this.paymentOrder;
23	}
24	public void setPaymentOrder(Order paymentOrder) {
25	this.paymentOrder = paymentOrder;
26	}
27	public void setPaymentType(String paymentType) {
28	this.paymentType = paymentType;
29	}
30	public String getPaymentType() {
31	return paymentType;
32	}
33	}

17

1	public class PaymentRepository implements ServletContextListener {
	@PersistenceUnit(unitName = "store-pu")
2	private EntityManagerFactory emf;
3	private EntityManager em;
4	public PaymentRepository() {
5	emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
6	System.out.println("[SERVLET-TEST-INFO]: PaymentRepository Constructor");
7	}
	@Override
8	public void contextDestroyed(ServletContextEvent sce) {
9	if (emf.isOpen()) {
10	emf.close();

11	}
12	}
	@Override
13	public void contextInitialized(ServletContextEvent sce) {
14	ServletContext context = sce.getServletContext();
15	context.setAttribute("paymentRepository", this);
16	System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! PaymentRepository");
17	}
18	public final EntityManager entityManager() {
19	if (em == null !em.isOpen()) {
20	em = emf.createEntityManager();
21	}
22	return em;
23	}
24	public void persistOrMerge(Serializable entity, Serializable id) {
25	em = entityManager();
26	if (entity == null) {
27	throw new IllegalArgumentException("entity");
28	}
29	try {
30	em.getTransaction().begin();
31	if (id == null) {
32	em.persist(entity);
33	} else {
34	em.merge(entity);
35	}
36	em.getTransaction().commit();
37	} finally {
38	em.close();
39	}
40	}
41	public Payment findPaymentById(Long id) {
42	em = entityManager();
43	return em.find(Payment.class, id);
44	}
45	public List findAllPayments() {
46	em = entityManager();
47	Query q = em.createQuery("select obj from Payment obj");
48	return q.getResultList();
49	}
50	public Payment newPayment() {

51	Payment object = new Payment();
52	persistOrMerge(object, object.getId());
53	return object;
54	}
55	}

18

1	public class ProcessCheckoutFormAction implements ControllerAction {
	@Override
2	public void execute(HttpServletRequest request, HttpServletResponse response) {
3	ControllerAction paymentAction = null ;
4	createEntities(request, response);
5	String paymentType = request.getParameter("PaymentType");
6	paymentAction = selectPaymentMethod(paymentAction, paymentType);
7	paymentAction.execute(request, response);
8	}
9	private ControllerAction selectPaymentMethod(ControllerAction paymentAction, String paymentType) {
10	if (paymentType.equals("Default")) {
11	paymentAction = new GoToAction("payment.jsp");
12	}
	//#if defined(Bankslip)
13	if (paymentType.equals("Bankslip")) {
14	paymentAction = new GoToAction("bankslip.jsp");
15	}
	//#endif
	//#if defined(Paypal)
16	if (paymentType.equals("Paypal")) {
17	paymentAction = new GoToAction("paypal.jsp");
18	}
	//#endif
19	return paymentAction;
20	}
21	private void createEntities(HttpServletRequest request, HttpServletResponse response)
22	{
23	NewCustomerAction newCustomerAction = new NewCustomerAction();
24	Customer c = (Customer) newCustomerAction.execute(request, response);
24	request.setAttribute("customer", c);
25	request.setAttribute("products", request.getSession().getAttribute("products"));
26	NewOrderAction newOrderAction = new NewOrderAction();

27	Order o = (Order) newOrderAction.execute(request, response);
28	request.setAttribute("order", o);
29	NewPaymentAction newPaymentAction = new NewPaymentAction();
30	newPaymentAction.execute(request, response);
31	}
32	}

19

1	public class ProcessSellerFormAction implements ControllerAction {
	@Override
2	public void execute(HttpServletRequest request, HttpServletResponse response) {
3	try {
4	Product p = createProduct(request, response);
5	setProductAssociations(request, response, p);
6	updateEntity(request, p);
7	forward(request, response);
8	} catch (Exception e) {
9	e.printStackTrace();
10	}
11	}
12	private Product createProduct(HttpServletRequest request, HttpServletResponse response)
	{
13	NewProductAction newProductAction = new NewProductAction();
14	Product p = (Product) newProductAction.execute(request, response);
15	return p;
16	}
17	private void forward(HttpServletRequest request, HttpServletResponse response) {
18	try {
19	request.setAttribute("message", " Product inserted with success");
20	
21	GoToAction responseAction = new GoToAction("response.jsp");
22	responseAction.execute(request, response);
23	} catch (Exception e) {
24	e.printStackTrace();
25	}
26	}
27	private void updateEntity(HttpServletRequest request, Product p) {
28	ServletContext context = request.getSession().getServletContext();
29	ProductRepository productRepository = (ProductRepository)
30	context.getAttribute("productRepository");
30	productRepository.persistOrMerge(p, p.getId());
31	}

32	private void setProductAssociations(HttpServletRequest request, HttpServletResponse response, Product p) {
33	setProductSeller(request, response, p);
34	setProductCategory(request, response, p);
35	}
36	private void setProductCategory(HttpServletRequest request, HttpServletResponse response, Product p) {
37	NewCategoryAction newCategoryAction = new NewCategoryAction();
38	Category c = (Category) newCategoryAction.execute(request, response);
39	p.setProductCategory(c);
40	}
41	private void setProductSeller(HttpServletRequest request, HttpServletResponse response, Product p) {
42	NewSellerAction newSellerAction = new NewSellerAction();
43	Seller s = (Seller) newSellerAction.execute(request, response);
44	p.setProductSeller(s);
45	}
46	}

20

	@Entity
	@Table(name = "products")
1	public class Product implements Serializable {
2	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue
3	private Long id;
4	private String productName;
5	private String productDescription;
6	private Double productPrice;
	@ManyToOne(targetEntity = Order. class , fetch = FetchType.LAZY)
7	private Order productOrder;
	@OneToOne
8	private Seller productSeller;
	@OneToOne
9	private Category productCategory;
	//#if defined(DisplayWhatIsNew)
10	private Date productInsertDate;

	//#endif
11	public Product() {
	// JPA
12	}
13	public Long getId() {
14	return id;
15	}
16	public void setId(Long id) {
17	this.id = id;
18	}
19	public void setProductName(String productName) {
20	this.productName = productName;
21	}
22	public String getProductName() {
23	return productName;
24	}
25	public void setProductDescription(String productDescription) {
26	this.productDescription = productDescription;
27	}
28	public String getProductDescription() {
29	return productDescription;
30	}
31	public void setProductPrice(Double productPrice) {
32	this.productPrice = productPrice;
33	}
34	public Double getProductPrice() {
35	return productPrice;
36	}
37	public void setProductOrder(Order productOrder) {
38	this.productOrder = productOrder;
39	}
40	public Order getProductOrder() {
41	return productOrder;
42	}
43	public void setProductSeller(Seller productSeller) {
44	this.productSeller = productSeller;

45	}
46	public Seller getProductSeller() {
47	return productSeller;
48	}
	//#if defined(DisplayWhatIsNew)
49	public void setProductInsertDate(Date productInsertDate) {
50	this .productInsertDate = productInsertDate;
51	}
52	public Date getProductInsertDate() {
53	return productInsertDate;
54	}
	//#endif
55	public void setProductCategory(Category productCategory) {
56	this .productCategory = productCategory;
57	}
58	public Category getProductCategory() {
59	return productCategory;
60	}
61	}

21

1	public class ProductRepository implements ServletContextListener {
	@PersistenceUnit(unitName = "store-pu")
2	private EntityManagerFactory emf;
3	private EntityManager em;
4	public ProductRepository() {
5	emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
6	System.out.println("[SERVLET-TEST-INFO]: ProductRepository Constructor");
7	}
	@Override
8	public void contextDestroyed(ServletContextEvent sce) {
9	if (emf.isOpen()) {
10	emf.close();
11	}
12	}
	@Override
13	public void contextInitialized(ServletContextEvent sce) {
14	ServletContext context = sce.getServletContext();
15	context.setAttribute("productRepository", this);

16	System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! ProductRepository");
17	}
18	public final EntityManager entityManager() {
19	if (em == null !em.isOpen()) {
20	em = emf.createEntityManager();
21	}
22	return em;
23	}
24	public void persistOrMerge(Serializable entity, Serializable id) {
25	em = entityManager();
26	if (entity == null) {
27	throw new IllegalArgumentException("entity");
28	}
29	try {
30	em.getTransaction().begin();
31	if (id == null) {
32	em.persist(entity);
33	} else {
34	em.merge(entity);
35	}
36	em.getTransaction().commit();
37	} finally {
38	em.close();
39	}
40	}
41	public Product newProduct() {
42	Product p = new Product();
43	persistOrMerge(p, p.getId());
44	return p;
45	}
46	public Product findProductById(Long id) {
47	em = entityManager();
48	return em.find(Product.class, id);
49	}
50	public List findAllProducts() {
51	em = entityManager();
52	Query q = em.createQuery("select p from Product p");
53	return q.getResultList();
54	}
55	public List findAllProductsAvailable() {
56	em = entityManager();

57	Query q = em.createQuery("select p from Product p where p.productOrder = NULL");
58	return q.getResultList();
59	}
	//#if defined(DisplayWhatIsNew)
60	public List findAllProductsAvailableByNearestDate() {
61	em = entityManager();
62	Query q = em.createQuery("select p from Product p where p.productOrder = NULL ORDER BY p.productInsertDate DESC");
63	return q.getResultList();
64	}
	//#endif
	//#if defined(DisplayByCategory)
65	public List findAllProductsByCategory(String category) {
66	em = entityManager();
67	Query q = em.createQuery("select p from Product p where p.productOrder = NULL AND p.productCategory.categoryName=:cat");
68	q.setParameter("cat", category);
69	return q.getResultList();
70	}
	//#endif
71	}

22

	@Entity
	@Table(name = "sellers")
1	public class Seller implements Serializable {
2	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue
3	private Long id;
4	private String name;
5	private String address;
6	private String email;
	@OneToOne(mappedBy = "productSeller")
7	private Product product;
8	public Seller() {
	// JPA
9	}
10	public Long getId() {

11	<code>return id;</code>
12	<code>}</code>
13	<code>public void setId(Long id) {</code>
14	<code> this.id = id;</code>
15	<code>}</code>
16	<code>public String getName() {</code>
17	<code> return name;</code>
18	<code>}</code>
19	<code>public void setName(String name) {</code>
20	<code> this.name = name;</code>
21	<code>}</code>
22	<code>public void setEmail(String email) {</code>
23	<code> this.email = email;</code>
24	<code>}</code>
25	<code>public String getEmail() {</code>
26	<code> return email;</code>
27	<code>}</code>
28	<code>public void setAddress(String address) {</code>
29	<code> this.address = address;</code>
30	<code>}</code>
31	<code>public String getAddress() {</code>
32	<code> return address;</code>
33	<code>}</code>
34	<code>public void setProduct(Product product) {</code>
35	<code> this.product = product;</code>
36	<code>}</code>
37	<code>public Product getProduct() {</code>
38	<code> return product;</code>
39	<code>}</code>
40	<code>}</code>

23

1	<code>public class SellerRepository implements ServletContextListener {</code>
	<code> @PersistenceUnit(unitName = "store-pu")</code>
2	<code> private EntityManagerFactory emf;</code>
3	<code> private EntityManager em;</code>

4	public SellerRepository() {
5	emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
6	System.out.println("[SERVLET-TEST-INFO]: SellerRepository Constructor");
7	}
	@Override
8	public void contextDestroyed(ServletContextEvent sce) {
9	if (emf.isOpen()) {
10	emf.close();
11	}
12	}
	@Override
14	public void contextInitialized(ServletContextEvent sce) {
15	ServletContext context = sce.getServletContext();
16	context.setAttribute("sellerRepository", this);
17	System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! SellerRepository");
18	}
19	public final EntityManager entityManager() {
20	if (em == null !em.isOpen()) {
21	em = emf.createEntityManager();
22	}
23	return em;
24	}
25	public void persistOrMerge(Serializable entity, Serializable id) {
26	em = entityManager();
27	
28	if (entity == null) {
29	throw new IllegalArgumentException("entity");
30	}
31	try {
32	em.getTransaction().begin();
33	if (id == null) {
34	em.persist(entity);
35	} else {
36	em.merge(entity);
37	}
38	em.getTransaction().commit();
39	} finally {
40	em.close();
41	}
42	}
43	public Seller findSellerById(Long id) {
44	em = entityManager();

45	<code>return em.find(Seller.class, id);</code>
46	<code>}</code>
47	<code>public Seller findSellerByEmail(String sellerEmail) {</code>
48	<code>em = entityManager();</code>
49	<code>return (Seller) em.createQuery("select s from Seller s where s.email = '" + sellerEmail + "'").getSingleResult();</code>
50	<code>}</code>
51	<code>public List<Seller> findAllSellers() {</code>
52	<code>em = entityManager();</code>
53	<code>Query q = em.createQuery("select s from Seller s");</code>
54	<code>return q.getResultList();</code>
55	<code>}</code>
56	<code>public Seller newSeller() {</code>
57	<code>Seller s = new Seller();</code>
58	<code>persistOrMerge(s, s.getId());</code>
59	<code>return s;</code>
60	<code>}</code>
61	<code>}</code>

24

1	<code>public class VerifyCatalogFormAction implements ControllerAction {</code>
	<code>@Override</code>
2	<code>public void execute(HttpServletRequest request, HttpServletResponse response) {</code>
3	<code>String[] productsIds = request.getParameterValues("productsIds");</code>
4	<code>if (productsIds == null) {</code>
5	<code>request.setAttribute("message", "
 ERROR: You must select at least a product!");</code>
6	<code>GoToAction responseAction = new GoToAction("response.jsp");</code>
7	<code>responseAction.execute(request, response);</code>
8	<code>}</code>
9	<code>else {</code>
10	<code>getProductList(request, productsIds);</code>
11	<code>GoToAction checkoutAction = new GoToAction("checkout.jsp");</code>
12	<code>checkoutAction.execute(request, response);</code>
13	<code>}</code>
14	<code>}</code>
15	<code>private void getProductList(HttpServletRequest request, String[] productsIds) {</code>
16	<code>ServletContext context = request.getSession().getServletContext();</code>
17	<code>ProductRepository productRepository = (ProductRepository) context.getAttribute("productRepository");</code>

18	List<Product> products = new ArrayList();
19	for (String productId : productsIds) {
20	products.add(productRepository.findProductById(Long.valueOf(productId)));
21	}
22	HttpSession session = request.getSession();
23	session.setAttribute("products", products);
24	}
25	}

25

1	public class VerifyCheckoutFormAction implements ControllerAction {
	@Override
2	public void execute(HttpServletRequest request, HttpServletResponse response) {
3	List params = getRequestParameters(request);
4	Boolean emptyFields = false;
5	for (Object param : params) {
6	if (param == null param.equals("")) {
7	emptyFields = true;
8	}
9	}
10	if (emptyFields) {
11	request.setAttribute("message", "Form with blank fields! Complete your form first!");
12	GoToAction responseAction = new GoToAction("response.jsp");
13	responseAction.execute(request, response);
14	} else {
15	ProcessCheckoutFormAction checkoutFormAction = new ProcessCheckoutFormAction();
16	checkoutFormAction.execute(request, response);
17	}
18	}
19	private List getRequestParameters(HttpServletRequest request) {
20	List params = new ArrayList();
21	addParameters(request, params);
22	return params;
23	}
24	private void addParameters(HttpServletRequest request, List params) {
25	params.add(request.getParameter("CustomerName"));
26	params.add(request.getParameter("CustomerAddress"));
27	params.add(request.getParameter("CustomerMail"));

28	params.add(request.getParameter("paymentType"));
29	}
30	}

26

1	public class VerifySellerFormAction implements ControllerAction {
	@Override
2	public void execute(HttpServletRequest request, HttpServletResponse response) {
3	List params = getRequestParameters(request);
4	Boolean emptyFields = false ;
5	for (Object param : params) {
6	if (param == null param.equals("")) {
7	emptyFields = true ;
8	}
9	}
10	if (emptyFields) {
11	request.setAttribute("message", "Form with blank fields! Complete your form first!");
12	GoToAction responseAction = new GoToAction("response.jsp");
13	responseAction.execute(request, response);
14	} else {
15	ProcessSellerFormAction sellerFormAction = new ProcessSellerFormAction();
16	sellerFormAction.execute(request, response);
17	}
18	}
19	private List getRequestParameters(HttpServletRequest request) {
20	List params = new ArrayList();
21	addParameters(request, params);
22	return params;
23	}
24	private void addParameters(HttpServletRequest request, List params) {
25	params.add(request.getParameter("CategoryName"));
26	params.add(request.getParameter("SellerName"));
27	params.add(request.getParameter("SellerAddress"));
28	params.add(request.getParameter("SellerMail"));
29	params.add(request.getParameter("ProductName"));
30	params.add(request.getParameter("ProductDescription"));
31	params.add(request.getParameter("ProductPrice"));
32	}
33	}

27