

1	@Entity
2	@Table(name = "categories")
3	public class Category implements java.io.Serializable {
4	
5	private static final long serialVersionUID = 1L;
6	
7	@Id
8	@GeneratedValue
9	private Long id;
10	
11	private String categoryName;
12	
13	@OneToOne(mappedBy = "productCategory")
14	private Product categoryProduct;
15	
16	public Category() {
17	// Used by JPA.
18	}
19	
20	public void setId(Long id) {
21	this.id = id;
22	}
23	
24	public Long getId() {
25	return id;
26	}
27	
28	public String getCategoryName() {
29	return this.categoryName;
30	}
31	
32	public void setCategoryName(String categoryName) {
33	this.categoryName = categoryName;
34	}
35	
36	public Product getCategoryProduct() {
37	return this.categoryProduct;
38	}
39	
40	public void setCategoryProduct(Product categoryProduct) {
41	this.categoryProduct = categoryProduct;
42	}
43	}

44	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	}

181	@Entity
182	@Table(name = "customers")
183	public class Customer implements Serializable {
184	
185	private static final long serialVersionUID = 1L;
186	
187	@Id
188	@GeneratedValue
189	private Long id;
190	
191	private String name;
192	
193	public void setName(String name) {
194	this.name = name;
195	}
196	
197	private String address;
198	
199	private String email;
200	
201	@OneToMany(targetEntity = Order.class, mappedBy = "customer", fetch = FetchType.LAZY)
202	private List<Order> orders;
203	
204	public Customer() {
205	// JPA
206	}
207	
208	public Long getId() {
209	return id;
210	}
211	
212	public String getName() {
213	return name;
214	}
215	
216	public void setAddress(String address) {
217	this.address = address;
218	}
219	
220	public String getAddress() {
221	return address;
222	}
223	
224	public void setOrders(List<Order> orders) {
225	this.orders = orders;
226	}
227	

228	public List<Order> getOrders() {
229	return orders;
230	}
231	
232	public void setEmail(String email) {
233	this .email = email;
234	}
235	
236	public String getEmail() {
237	return email;
238	}
239	}

5

240	public class CustomerRepository implements ServletContextListener {
241	
242	@PersistenceUnit(unitName = "store-pu")
243	private EntityManagerFactory emf;
244	
245	private EntityManager em;
246	
247	public CustomerRepository() {
248	emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");
249	System.out.println("[SERVLET-TEST-INFO]: CustomerRepository Constructor");
250	}
251	
252	@Override
253	public void contextDestroyed(ServletContextEvent sce) {
254	if (emf.isOpen()) {
255	emf.close();
256	}
257	}
258	
259	@Override
260	public void contextInitialized(ServletContextEvent sce) {
261	ServletContext context = sce.getServletContext();
262	context.setAttribute("customerRepository", this);
263	System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! CustomerRepository");
264	}
265	
266	public final EntityManager entityManager() {
267	if (em == null !em.isOpen()) {
268	em = emf.createEntityManager();
269	}
270	return em;
271	}
272	

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)
399	context.getAttribute("customerRepository");
400	}
401	
402	@Override
403	public Object createEntity() {
404	Customer c = customerRepository .newCustomer();
405	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	<code>params = new Object[2];</code>
451	<code>params[0] = request.getAttribute("customer");</code>
452	<code>params[1] = request.getAttribute("products");</code>
453	<code>}</code>
454	<code>}</code>

11

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

12

488	<code>public class NewProductAction extends ModelAction {</code>
489	
490	<code>private ProductRepository productRepository;</code>
491	<code>Object params[];</code>

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	

537	@Override
538	public Object createEntity() {
539	Seller s = sellerRepository .newSeller();
540	setEntityAssociations(s);
541	sellerRepository .persistOrMerge(s, s.getId());
542	return s;
543	}
544	
545	@Override
546	public void setEntityAssociations(Object seller) {
547	((Seller) seller).setName((String) params [0]);
548	((Seller) seller).setAddress((String) params [1]);
549	((Seller) seller).setEmail((String) params [2]);
550	}
551	
552	@Override
553	public void setPersistenceRepository(HttpServletRequest request) {
554	ServletContext context = request.getSession().getServletContext();
555	sellerRepository = (SellerRepository) context.getAttribute("sellerRepository");
556	}
557	
558	@Override
559	public void getRequestParameters(HttpServletRequest request) {
560	params = new Object[3];
561	params [0] = request.getParameter("SellerName");
562	params [1] = request.getParameter("SellerAddress");
563	params [2] = request.getParameter("SellerMail");
564	}
565	}

14

566	@Entity
567	@Table(name = "orders")
568	public class Order implements Serializable {
569	
570	private static final long <i>serialVersionUID</i> = 1L;
571	
572	@Id
573	@GeneratedValue
574	private Long id ;
575	
576	@ManyToOne(targetEntity = Customer. class , fetch = FetchType. <i>LAZY</i>)
577	private Customer customer ;
578	
579	@OneToMany(targetEntity = Product. class , mappedBy = "productOrder", fetch = FetchType. <i>LAZY</i>)
580	@Column(name = "orderProducts")
581	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	}

676	}
677	
678	public Order findById(Long id) {
679	em = entityManager();
680	return em.find(Order.class, id);
681	}
682	
683	public List<Order> findAllOrders() {
684	em = entityManager();
685	Query q = em.createQuery("select o from Order o");
686	return q.getResultList();
687	}
688	
689	public Order newOrder() {
690	Order o = new Order();
691	persistOrMerge(o, o.getId());
692	
693	return o;
694	}
695	}

16

696	@Entity
697	@Table(name = "payments")
698	public class Payment implements java.io.Serializable {
699	
700	private static final long serialVersionUID = 1L;
701	
702	@Id
703	@GeneratedValue
704	private Long id;
705	
706	private String paymentType;
707	
708	@OneToOne
709	private Order paymentOrder;
710	
711	private String paymentStatus;
712	
713	public Payment() {
714	// Used by JPA.
715	}
716	
717	public void setId(Long id) {
718	this.id = id;
719	}
720	

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", " 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)
901	context.getAttribute("productRepository");
902	productRepository.persistOrMerge(p, p.getId());
903	}
904	
904	private void setProductAssociations(HttpServletRequest request, HttpServletResponse
	response, Product p) {

905	setProductSeller(request, response, p);
906	setProductCategory(request, response, p);
907	}
908	
909	private void setProductCategory(HttpServletRequest request, HttpServletResponse response, Product p) {
910	NewCategoryAction newCategoryAction = new NewCategoryAction();
911	Category c = (Category) newCategoryAction.execute(request, response);
912	p.setProductCategory(c);
913	}
914	
915	private void setProductSeller(HttpServletRequest request, HttpServletResponse response, Product p) {
916	NewSellerAction newSellerAction = new NewSellerAction();
917	Seller s = (Seller) newSellerAction.execute(request, response);
918	p.setProductSeller(s);
919	}
920	}

20

921	@Entity
922	@Table(name = "products")
923	public class Product implements Serializable {
924	
925	private static final long serialVersionUID = 1L;
926	
927	@Id
928	@GeneratedValue
929	private Long id;
930	
931	private String productName;
932	
933	private String productDescription;
934	
935	private Double productPrice;
936	
937	@ManyToOne(targetEntity = Order.class, fetch = FetchType.LAZY)
938	private Order productOrder;
939	
940	@OneToOne
941	private Seller productSeller;
942	
943	@OneToOne
944	private Category productCategory;
945	// #if defined(DisplayWhatIsNew)
946	private Date productInsertDate;
947	
948	// #endif
949	

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

999	<code>return productSeller;</code>
1000	<code>}</code>
1000	
1000	<code>//#if defined(DisplayWhatIsNew)</code>
1000	<code>public void setProductInsertDate(Date productInsertDate) {</code>
1000	<code> this.productInsertDate = productInsertDate;</code>
1000	<code>}</code>
1000	
1000	<code>public Date getProductInsertDate() {</code>
1000	<code> return productInsertDate;</code>
1000	<code>}</code>
1000	
1000	<code>//#endif</code>
1000	<code>public void setProductCategory(Category productCategory) {</code>
1000	<code> this.productCategory = productCategory;</code>
1000	<code>}</code>
1000	
1000	<code>public Category getProductCategory() {</code>
1000	<code> return productCategory;</code>
1000	<code>}</code>
1000	<code>}</code>

21

102	<code>public class ProductRepository implements ServletContextListener {</code>
102	
102	<code> @PersistenceUnit(unitName = "store-pu")</code>
102	<code> private EntityManagerFactory emf;</code>
102	
102	<code> private EntityManager em;</code>
102	
102	<code> public ProductRepository() {</code>
102	<code> emf = (EntityManagerFactory) Persistence.createEntityManagerFactory("store-pu");</code>
102	<code> System.out.println("[SERVLET-TEST-INFO]: ProductRepository Constructor");</code>
103	<code> }</code>
103	
103	<code> @Override</code>
103	<code> public void contextDestroyed(ServletContextEvent sce) {</code>
103	<code> if (emf.isOpen()) {</code>
103	<code> emf.close();</code>
103	<code> }</code>
103	<code> }</code>
103	
103	<code> @Override</code>
104	<code> public void contextInitialized(ServletContextEvent sce) {</code>
104	<code> ServletContext context = sce.getServletContext();</code>
104	<code> context.setAttribute("productRepository", this);</code>
104	<code> System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado!</code>
104	<code>ProductRepository");</code>
104	<code> }</code>
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
109	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
110	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	<code> this.id = id;</code>
114	<code> }</code>
114	
114	<code> public String getName() {</code>
114	<code> return name;</code>
114	<code> }</code>
114	
114	<code> public void setName(String name) {</code>
114	<code> this.name = name;</code>
114	<code> }</code>
115	
115	<code> public void setEmail(String email) {</code>
115	<code> this.email = email;</code>
115	<code> }</code>
115	
115	<code> public String getEmail() {</code>
115	<code> return email;</code>
115	<code> }</code>
115	
115	<code> public void setAddress(String address) {</code>
116	<code> this.address = address;</code>
116	<code> }</code>
116	
116	<code> public String getAddress() {</code>
116	<code> return address;</code>
116	<code> }</code>
116	
116	<code> public void setProduct(Product product) {</code>
116	<code> this.product = product;</code>
116	<code> }</code>
117	
117	<code> public Product getProduct() {</code>
117	<code> return product;</code>
117	<code> }</code>
117	<code>}</code>

23

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

118	}
118	
118	@Override
118	public void contextDestroyed(ServletContextEvent sce) {
118	if (emf.isOpen()) {
119	emf.close();
119	}
119	}
119	
119	@Override
119	public void contextInitialized(ServletContextEvent sce) {
119	ServletContext context = sce.getServletContext();
119	context.setAttribute("sellerRepository", this);
119	System.out.println("[SERVLET-TEST-INFO]: Contexto inicializado! SellerRepository");
119	}
120	
120	public final EntityManager entityManager() {
120	if (em == null !em.isOpen()) {
120	em = emf.createEntityManager();
120	}
120	return em;
120	}
120	
120	public void persistOrMerge(Serializable entity, Serializable id) {
120	em = entityManager();
121	
121	if (entity == null) {
121	throw new IllegalArgumentException("entity");
121	}
121	try {
121	em.getTransaction().begin();
121	if (id == null) {
121	em.persist(entity);
121	} else {
121	em.merge(entity);
122	}
122	em.getTransaction().commit();
122	} finally {
122	em.close();
122	}
122	}
122	
122	public Seller findSellerById(Long id) {
122	em = entityManager();
122	return em.find(Seller.class, id);
123	}
123	
123	public Seller findSellerByEmail(String sellerEmail) {
123	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", " 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
129	form first!");
129	GoToAction responseAction = new GoToAction("response.jsp");
130	responseAction.execute(request, response);
130	} else {
130	ProcessCheckoutFormAction checkoutFormAction = new
130	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

132	public class VerifySellerFormAction implements ControllerAction {
132	
132	@Override
132	public void execute(HttpServletRequest request, HttpServletResponse response) {
132	List params = getRequestParameters(request);
132	Boolean emptyFields = false ;
132	
132	for (Object param : params) {
132	if (param == null param.equals("")) {
132	emptyFields = true ;
133	}
133	}
133	
133	if (emptyFields) {
133	request.setAttribute("message", "Form with blank fields! Complete your form first!");
133	GoToAction responseAction = new GoToAction("response.jsp");
133	responseAction.execute(request, response);
133	} else {
133	ProcessSellerFormAction sellerFormAction = new ProcessSellerFormAction();
133	sellerFormAction.execute(request, response);
134	}
134	}
134	
134	private List getRequestParameters(HttpServletRequest request) {
134	List params = new ArrayList();
134	addParameters(request, params);
134	return params;
134	}
134	
134	private void addParameters(HttpServletRequest request, List params) {
135	params.add(request.getParameter("CategoryName"));
135	params.add(request.getParameter("SellerName"));
135	params.add(request.getParameter("SellerAddress"));
135	params.add(request.getParameter("SellerMail"));
135	params.add(request.getParameter("ProductName"));
135	params.add(request.getParameter("ProductDescription"));
135	params.add(request.getParameter("ProductPrice"));
135	}
135	}