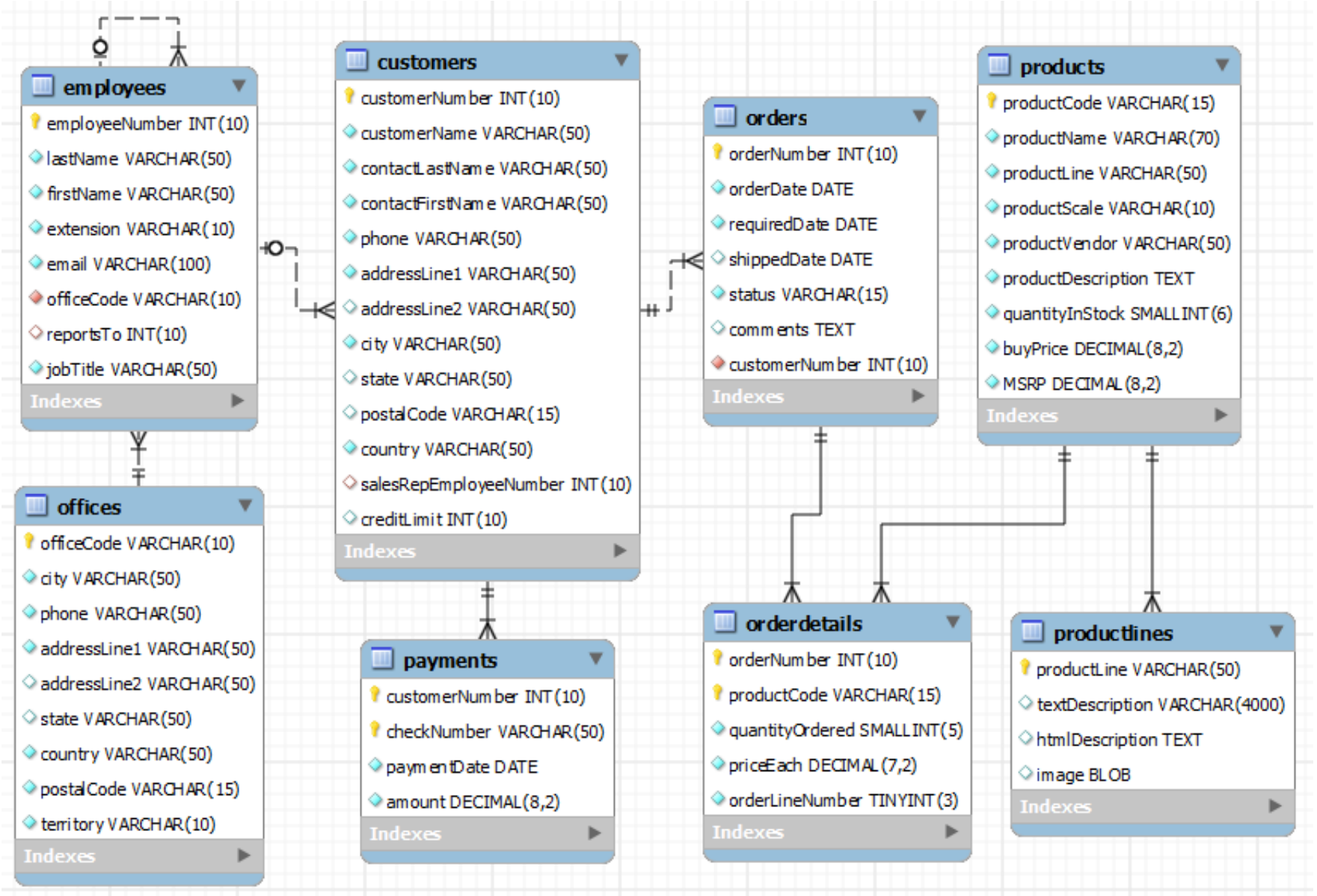
MVC Using JPA

classicmodels



Classic Models Online Shop Web App

- Create JavaEE project
  - Name: classicmodels
  - Template: Web Application
  - Group: sit.int202
  - Application server: Tomcat 10.0.xxx

  - Version: Jakarta EE 9
  - Dependencies:
    - Specifications:
      - Servlet
      - Persistence (JPA)
    - Implementations:
      - Hibernate
      - Hibernate Validator

Other Dependencies:
- Project Lombok 1.18.2x
- JSTL 3.0.x
- MySQL Connector 8.0.3x
- Eclipse Expressly 5.0.x

Create Office Entity

```
@Entity
@Table(name = "Offices")
public class Office {
    @Id
    private String officeCode;
    private String
addressLine1;
    private String
addressLine2;
    private String city;
    private String state;
    private String country;
    private String postalCode;
    private String phone;
    private String territory;
}
```

```
@Entity
@Table(name = "Offices")
public class Office {
    @Id
    private String officeCode;
        :
    private String phone;
    private String territory;

    @OneTomany(mappedBy= ”officeCode")
    private List<Employee> employees;
}
```

Modify persistence.xml

```
<persistence-unit name=“classic-models">
  <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
  <class>sit.int202.classicmodelweb.entities.Office</class>
  <properties>
     <property name="jakarta.persistence.jdbc.driver"
                          value="com.mysql.cj.jdbc.Driver"/>
     <property name="jakarta.persistence.jdbc.url"
                          value="jdbc:mysql://localhost:3306/classicmodels"/>
     <property name="jakarta.persistence.jdbc.user" value="root"/>
     <property name="jakarta.persistence.jdbc.password" value=“mysql@sit"/>
     <property name="hibernate.dialect"
                          value="org.hibernate.dialect.MySQL8Dialect"/>
   </properties>
</persistence-unit>
```

Testing Entity Manager Services

M E N U

==========

1) Add New Office
2) Edit Office
3) Delete Office
4) Search Office
5) List All Office

---

0) List All Office

Select your choice:

Entity Manager Builder

```
import jakarta.persistence.EntityManager;
import jakarta.persistence.EntityManagerFactory;
import jakarta.persistence.Persistence;


public class EntityManagerBuilder {
    private final static EntityManagerFactory emf  =
       Persistence.createEntityManagerFactory("classic-models");
    public static EntityManager getEntityManager() {
        return emf.createEntityManager();
    }
}
```

Office Repository

```
@AllArgsConstructor
@NoArgsConstructor
public class OfficeRepository {
    private EntityManager entityManager;


    private EntityManager getEntityManager() {
        if (entityManager == null || !entityManager.isOpen()) {
            entityManager = EntityManagerBuilder.getEntityManager();
        }
        return entityManager;
    }
    public List<Office> findAll() {
        return getEntityManager().createQuery(
       "select o from Office o").getResultList();
    }
    public Office find(String officeCode) {
        return getEntityManager().find(Office.class, officeCode);
    }
```

TestJpa

```
public class TestJpa {
    private static final Scanner scanner =
          new Scanner(System.in);
    private static final OfficeRepository
repository =
           new OfficeRepository();

    public static void main(String[] args) {
        int choice = 0;
        do {
            choice = menu();
            switch (choice) {
                case 4:
                    searchOffice();
                    break;
                case 5:
                    listAllOffice();
                    break;
            }
        } while (choice > 0);
    }
```

```
   private static int menu() {
        System.out.println("\n\n");
        System.out.println("M E N U");

System.out.println("====================");
        System.out.println("1) Add New Office");
        System.out.println("2) Edit Office");
        System.out.println("3) Delete Office");
        System.out.println("4) Search Office");
        System.out.println("5) List All
Office");
        System.out.println("--------------------
");
        System.out.println("0) List All
Office");
        System.out.print("\nSelect your choice:
");
        return scanner.nextInt();
    }
```

TestJpa (2)

```
private static void searchOffice() {
    System.out.print("Enter country or city to find: ");
    String cityOrCoutry = scanner.next();
    List<Office> offices = repository.findByCityOrCountry(cityOrCoutry);
    if (!offices.isEmpty()) {
        System.out.printf("Office search by %s*\n", cityOrCoutry);
        System.out.println("-------------------------");
        offices.forEach(office -> printOffice(office));
    } else {
        System.out.printf("Office search by %s* does not exist !!!\n",
cityOrCoutry);
    }
}

private static void listAllOffice() {
    repository.findAll().forEach(o -> printOffice(o));
}
private static void printOffice(Office o) {
    System.out.println("Office Code: " + o.getOfficeCode());
    System.out.println("City: " + o.getCity());
    System.out.println("Country: " + o.getCountry());
    System.out.println("-------------");
}
```

Add insert(), delete(), getTransaction methods to Office Repository

```
public boolean insert(Office office) {
    try {
        EntityManager entityManager =
getEntityManager();
        entityManager.getTransaction().begin();
        entityManager.persist(office);
        entityManager.getTransaction().commit();
    } catch (Exception e) {
        return false;
    }
    return true;
}


public boolean delete(String officeCode) {
    EntityManager entityManager =
getEntityManager();
    Office office = find(officeCode);
    if (office != null) {
        entityManager.getTransaction().begin();
        entityManager.remove(office);
        entityManager.getTransaction().commit();
        return true;
    }
    return false;
}
```

```
public boolean delete(Office office) {
    if (office != null) {
        EntityManager entityManager =
getEntityManager();
        if (entityManager.contains(office)) {

entityManager.getTransaction().begin();
            entityManager.remove(office);

entityManager.getTransaction().commit();
            return true;
        } else {
            return
delete(office.getOfficeCode());
        }
    }
    return false;
}


//use for update
public EntityTransaction getTransaction() {
    return getEntityManager().getTransaction();
}
```

Add method for add new Office to TestJpa

```
private static void addNewOffice() {
    int x = (int) (Math.random() * 99 + 7);
    Office newOffice = new Office();
    newOffice.setOfficeCode(String.valueOf(x));
    newOffice.setAddressLine1("126 Pracha-Utit, Bangmod");
    newOffice.setAddressLine2("Thungkru");
    newOffice.setCity("Bangkok");                    newOffice.setCountry("Thailand");
    newOffice.setPostalCode("10140");                newOffice.setPhone("+66 2 470 9872");
    newOffice.setState("");                          newOffice.setTerritory("SE-A");
    if (repository.insert(newOffice)) {
        System.out.println("Inserted New Office ::");
    } else {
        System.out.println("Can't insert Office ::");
    }
    printOffice(newOffice);
}
```

Add method for update Office to TestJpa

```
private static void updateOffice() {
    System.out.print("Enter office code to UPDATE: ");
    String officeCode = scanner.next();
    Office office = repository.find(officeCode);
    if (office != null) {
        System.out.println(":: Updating Offfice ::");
        printOffice(office);
        repository.getTransaction().begin();
        System.out.print("Enter new city: ");
        office.setCity(scanner.next());
        System.out.print("Enter new country: ");
        office.setCountry(scanner.next());
        repository.getTransaction().commit();
        System.out.printf("Office %s has been updated already !!", officeCode);
    }
}
```

Add method for delete Office to TestJpa

```
private static void deleteOffice() {
    System.out.print("Enter office code to DELETE (by code): ");
    String officeCode = scanner.next();
    System.out.println(officeCode + (repository.delete(officeCode) ?
            " was deleted" : " does not exist !!!"));
    System.out.println("------------");
    System.out.print("Enter office code to DELETE (by Object): ");
    officeCode = scanner.next();
    Office office = repository.find(officeCode);
    if (repository.delete(office)) {
        System.out.printf("Office %s was deleted\n", officeCode);
    } else {
        System.out.printf("Office NOT FOUND or Error occurred while delete Office
%s \n", officeCode);
    }
}
```

Add Name Query to Office Entity

```
@NamedQueries({
        @NamedQuery(name = "Office.FIND_BY_CITY_OR_COUNTRY",
                query = "select o from Office o where lower(o.city) like :city or
lower(o.country) like :country")
})
```

```
public class Office {
    @Id
    private String officeCode;
    private String addressLine1;
    private String addressLine2;
```

Add method findByCityOrCountry to Office Repository

```
public List<Office> findByCityOrCountry(String cityOrCountry) {
    cityOrCountry = cityOrCountry.toLowerCase()+'%';
    Query query =
getEntityManager().createNamedQuery("Office.FIND_BY_CITY_OR_COUNTRY");
    query.setParameter("city", cityOrCountry);
    query.setParameter("country", cityOrCountry);
    return query.getResultList();
}
```

Add method searchOffice to TestJpa

```
private static void searchOffice() {
    System.out.print("Enter country or city to find: ");
    String cityOrCoutry = scanner.next();
    List<Office> offices = repository.findByCityOrCountry(cityOrCoutry);
    if (!offices.isEmpty()) {
        System.out.printf("Office search by %s*\n", cityOrCoutry);
        System.out.println("--------------------------");
        offices.forEach(office -> printOffice(office));
    } else {
        System.out.printf("Office search by %s* does not exist !!!\n",
cityOrCoutry);
    }
}
```

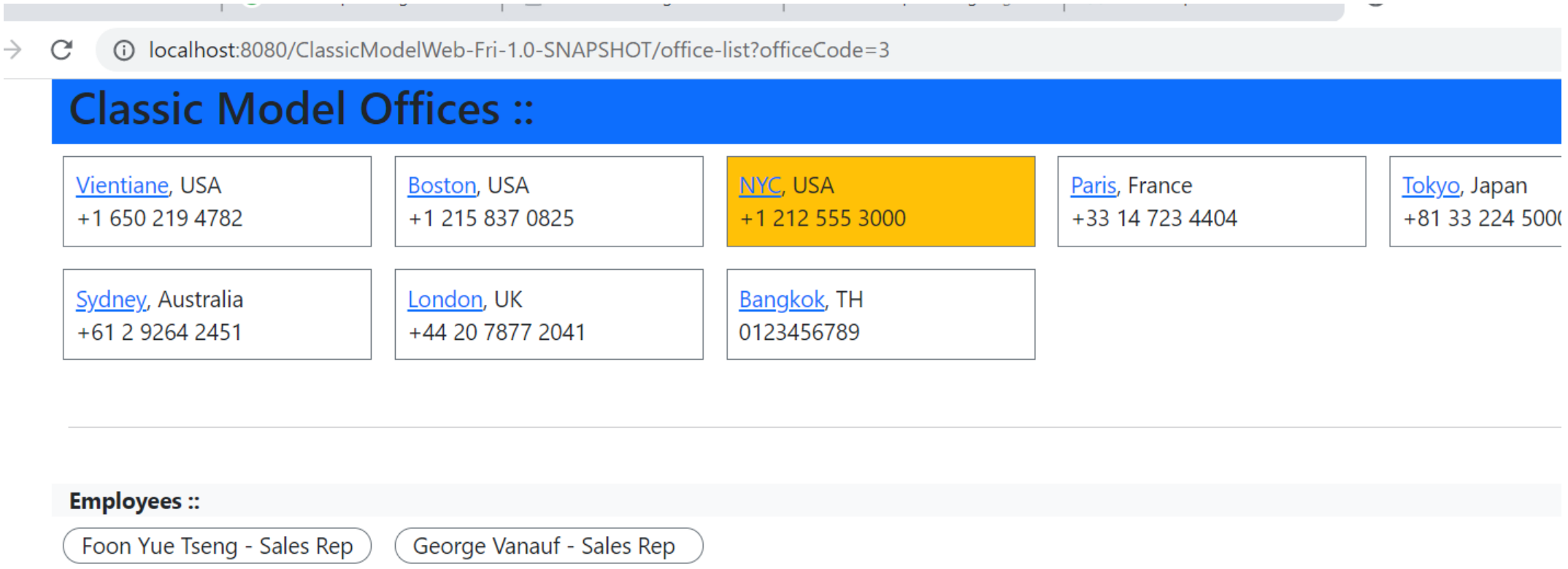
Create Employee Entity & Modify Office Entity

```
@Getter
@Setter
@Entity
@Table(name = "employees")
public class Employee {
    @Id
    private Integer employeeNumber;
    private String firstName;
    private String lastName;
    private String extension;
    private String email;
    private String officeCode;
    private Integer reportsTo;
    private String jobTitle;

}
```

Add Employee Entity then add it to persistence.xml

```
<persistence-unit name="classic-models">
  <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
  <class>sit.int202.classicmodelweb.entities.Employee</class>
  <class>sit.int202.classicmodelweb.entities.Office</class>

  <properties>
        :
        :
  </properties>
</persistence-unit>
```

Practice: Office-Employee Listing



OfficeListServlet

```
@WebServlet(name = "OfficeListServlet", value = "/office-list")
public class OfficeListServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, …. {
        OfficeRepository officeRepository = new OfficeRepository();
        request.setAttribute("offices", officeRepository.findAll());
        String officeCode = request.getParameter("officeCode");
        if (officeCode != null) {
            request.setAttribute("selectedOffice", officeRepository.find(officeCode));
        }
        getServletContext().getRequestDispatcher("/OfficeList.jsp").forward(request, response);
    }
```

OfficeEmployeeList.jsp (1)

```
<body>
 <div class="container">
  <div class="row bg-primary">
     <h2>Classic Model Offices ::</h2>
  </div>
  <div class="row">
  <c:forEach items="${offices}" var="office">
    <div class="col-2 border border-secondary p-2 m-2
     ${office.id == selectedOffice.id ? 'bg-warning' : ''}">
     <div>
       <a href="office-list?officeCode=${office.id}">
            ${office.city}</a>, ${office.country}
     </div>
     <div> ${office.phone}</div>
    </div>
  </c:forEach>
 </div>
```

OfficeEmployeeList.jsp (2)

```
        <br>
        <div class="row bg-light">
          <b>Employees ::</b>
        </div>
        <div class="row">
          <c:forEach items="${selectedOffice.employeeList}" var="employee">
            <div class="col-2 pl-2 m-2 border border-secondary rounded-pill">
              <div> ${employee.firstName}

                          ${employee.lastName} - ${employee.jobTitle}

                    </div>
            </div>
          </c:forEach>
        </div>
    </div>
</body>
```

Practice: Product Listing

Product Entity

```
@NamedQueries({
      @NamedQuery(name = "Product.FindAll", query = "SELECT p FROM Product p"),
      @NamedQuery(name = "Product.count", query = "SELECT count(p) as count FROM Product p")
})
@Table(name = "products", indexes = {
      @Index(name = "productLine", columnList = "productLine")
})
@Entity
public class Product {
   @Id
   @Column(name = "productCode", nullable = false, length = 15)
   private String id;

   @Column(name = "productName", null
```

Product Repository

```
public class ProductRepository {
    private static int PAGE_SIZE = 10;
    public int getDefaultPageSize() {
        return PAGE_SIZE;
    }
    public List<Product> findAll(int page, int pageSize) {
        int startPosition = (page-1) * pageSize;
        EntityManager entityManager = getEntityManager();
        Query query = entityManager.createNamedQuery("Product.FindAll");
        query.setFirstResult(startPosition);
        query.setMaxResults(pageSize);
        List<Product> productList = query.getResultList();
         entityManager.close();
         return productList;
    }

   public int countAll() {
        EntityManager entityManager = getEntityManager();
        int number = ((Number) entityManager.createNamedQuery("Product.count").getSingleResult()).intValue();
        entityManager.close();
        return number;
    }
```

ProductListServlet

```
protected void doGet(HttpServletRequest request, … {
    ProductRepository productRepository = new ProductRepository();
    String pageParam = request.getParameter("page");
    String pageSizeParam = request.getParameter("pageSize");
    int page = pageParam==null ? 1 : Integer.valueOf(pageParam);
    int pageSize = pageSizeParam==null ?
          productRepository.getDefaultPageSize() : Integer.valueOf(pageSizeParam);

    List<Product> productList = productRepository.findAll(page, pageSize);
    request.setAttribute("products", productList);
    request.setAttribute("page", page);
    request.setAttribute("pageSize", pageSize);
    request.setAttribute("itemCount", productRepository.countAll());
    getServletContext().getRequestDispatcher("/ProductList.jsp").forward(request, response);
}
```

ProductList.jsp (1)

```
<body>
<div class="container ml-2 p-2">
   <div class="d-flex flex-row">Product List ::</div>
   <hr>
   <c:forEach items="${products}" var="p" varStatus="vs">
      <div class="row">
         <div class="col-1">${vs.count + (page-1)*pageSize})</div>
         <div class="col-4"> ${p.id}: ${p.productName}</div>
         <div class="col-1"> ${p.productScale}</div>
         <div class="col-1" style="text-align: right">${p.msrp}</div>
      </div>
   </c:forEach>
   <hr>
```

ProductList.jsp (2)

```
  <div class="d-flex flex-row">
    <div>page: </div>
    <c:forEach begin="1" end="${itemCount/pageSize + (itemCount%pageSize>0?1 :0)}" varStatus="vs">
      <c:choose>
        <c:when test="${vs.count==page}">
          <div class="p-1 text-danger">[${vs.count}]</div>
        </c:when>
        <c:otherwise>
          <div class="p-1"><a href="product-list?pageSize=${pageSize}&page=${vs.count}">[${vs.count}]</a></div>
        </c:otherwise>
      </c:choose>
    </c:forEach>
  </div>
</div>
</body>
```