第9章　宠物商城购物模块

本章要点：

- 能理解基于 MVC 开发模式的功能模块开发方法。
- 会熟练编写模型代码，并进行单元测试。
- 会熟练编写控制器代码和视图代码。
- 会熟练配置项目的部署信息和 Web 服务器信息。
- 会熟练进行视图拆分设计。

9.1 首页展示

项目 9-1

■ 9.1.1　功能简介

首页展示的主要功能是展示最新上架的 12 只宠物，并在页面顶部展示站点导航信息以及页面底部的版权信息等内容。结合功能需求和 MVC 开发模式，确定实现首页展示功能需要编写的文件，如 图 9-1 所示。

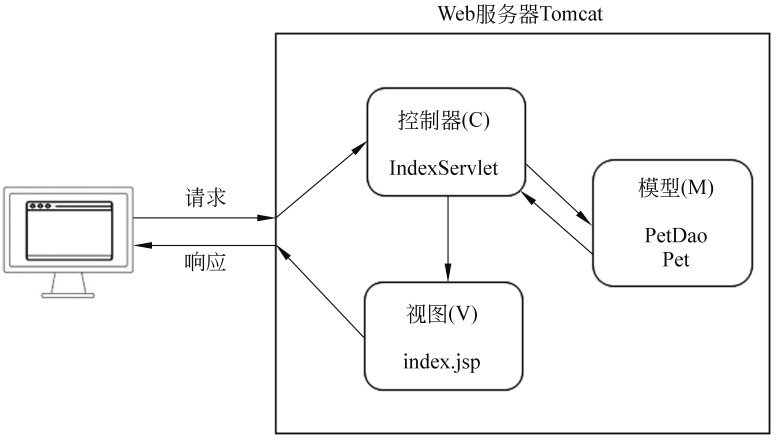

图 9-1　首页展示的 MVC 模式图

■ 9.1.2 模型代码的编写

在 PetStore 项目的 com.example.domain 包上右击，在弹出的快捷菜单中选择 New→Java Class 命令，在弹出的窗口中输入 Pet，按 Enter 键创建文件并输入源代码。

源程序：**Pet.java** 文件

```
package com.example.domain;
public class Pet {
    private int id;
    private int category_id;
    private String title;
    private String tag;
    private String photo;
    private double price;
    private int stock;
    private Date ondate;
    private String descs;
    public Pet() { }     //无参构造方法
    public int getId() { return id;}
    public void setId(int id) {this.id = id;}
    public int getCategory_id() {return category_id;}
    public void setCategory_id(int category_id)
       {this.category_id = category_id;}
    public String getTitle() {return title;}
    public void setTitle(String title) {this.title = title;}
    public String getTag() {return tag;}
    public void setTag(String tag) {this.tag = tag;}
    public String getPhoto() {return photo;}
    public void setPhoto(String photo) {this.photo = photo;}
    public double getPrice() {return price;}
    public void setPrice(double price) {this.price = price;}
    public int getStock() {return stock;}
    public void setStock(int stock) {this.stock = stock;}
    public Date getOndate() {return ondate;}
    public void setOndate(Date ondate) {this.ondate = ondate;}
    public String getDescs() {return descs;}
    public void setDescs(String descs) {this.descs = descs;}
}
```

Pet.java 文件的代码中设计了一个简单的宠物类，只有一些属性及其 getter 和 setter 方法，没有业务逻辑方法。在模型（M）中满足这些规则的类可以理解为简单的实体类，是为了方便开发人员表示数据库表中的数据。所以实体类的设计简单方便，依据数据库表字段的属性名和数据类型，即可快速编写实体类代码。

在本项目中，每个实体类添加了无参构造方法（构造器），这是为后续 JDBCTemplate 对象的查询方法获取数据库表中数据后自动封装为实体对象做准备。

在 PetStore 项目的 com.example.dao 包上右击，在弹出的快捷菜单中选择 New→Java Class 命令，在弹出的窗口中输入 PetDao，按 Enter 键创建文件并输入源代码。

源程序：PetDao.java 文件

```
package com.example.dao;
import com.example.domain.Pet;
import com.example.utils.JDBCUtils;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import java.util.List;
public class PetDao {
    private JdbcTemplate template =
                                new JdbcTemplate(JDBCUtils.getDataSource());
    // 获取最新上架的 12 只宠物对象列表
    public List<Pet> getNewList(){
        List<Pet> petList = null;
        try {
            String sql = "select * from pets order by ondate desc limit 12";
            petList =
              template.query(sql, new BeanPropertyRowMapper<>(Pet.class));
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            return petList;
        }
    }
}
```

PetDao.java 文件的代码中设计了一个宠物数据存取类，主要包含从数据库中获取宠物数据对象的方法以及修改数据库中宠物数据的方法。目前只有 getNewList 一个方法，第 10～12 章中将会逐步增加其他方法。

程序说明：

- 本项目中的数据存取类使用了 JDBCTemplate 工具类，该工具代码简洁并且能自动封装查询数据结果为相应的实体对象，有利于提高开发效率。
- getNewList 方法可获取最新上架的 12 只宠物的对象列表，业务功能通过 SQL 语句完成。SQL 语句中的“order by ondate desc”表示根据 ondate 字段降序排列，即最新上架的宠物排在前面；“limit 12”表示仅获取前 12 条数据。

■ 9.1.3　模型代码的测试

在 IDEA 创建 Java Enterprises 项目时，默认包含了 JUnit 单元测试框架，开发人员可以轻松地使用该框架进行单元测试。

JUnit 用于编写和运行可重复的自动化测试开源框架，通过单元测试可以保证项目的代码按预期工作。JUnit 可以用于测试整个对象，以及对象的一部分或者几个对象之间的交互。JUnit 提供了注解以确定测试方法，提供断言测试的预期结果。JUnit 的主要优点有如下几个。

- JUnit 优雅简洁。不复杂且不需要花费太多的时间。
- JUnit 测试可以自动运行，检查预期的结果，并提供即时反馈。
- JUnit 测试可以组织成测试套件包含测试实例，甚至其他测试套件。
- JUnit 可显示测试进度，如果测试没有问题，进度条是绿色的，如果测试失败则会变成红色。

在 MVC 开发模式下，模型、控制器、视图需要协同工作，具体的功能模块需要三个部分的代码都正常运行才能顺利完成。为了简化系统开发过程的复杂度，模型、控制器、视图三个部分若能进行独立的单元测试，则可以将复杂问题简单化，提高开发效率。

模型部分的 Java 代码不涉及 Web 运行环境，适合使用 JUnit 框架进行测试。此外，IDEA 集成了快捷编写测试代码的方式，减少了测试代码编写的工作量。下面对 PetDao 中的 getNewList 方法编写单元测试代码。

打开 PetStore 项目的 PetDao.Java 文件，在方法 getNewList 内右击，在弹出的菜单中选择 Generate→Test 命令，弹出 Create Test 对话框，如图 9-2 所示。

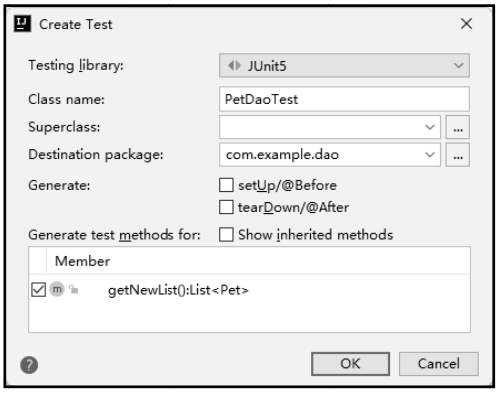

图 9-2　Create Test 对话框

测试类的名称采用被测试类名加上后缀 Test 的方式，这种命名方式简单直观，便于查找和维护。选择需要测试的方法 getNewList，最后单击 OK 按钮完成。按照这个步骤创建的测试类文件 PetDaoTest.java，保存在 PetStore 项目的文件夹 src\test\java 中，如图 9-3 所示。

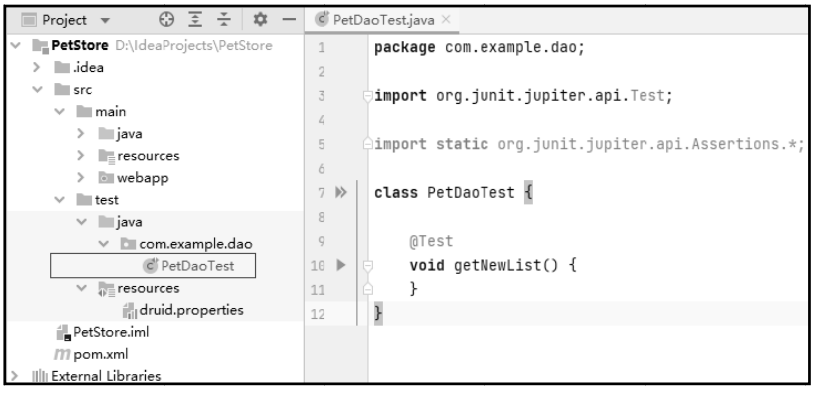

图 9-3　PetDaoTest 类文件

源程序：**PetDaoTest.java 文件**

```
package com.example.dao;
import com.example.domain.Pet;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.Before Each;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;
class PetDaoTest {
    private PetDao petDao;
    //每个测试方法执行前，初始化 petDao 对象
    @BeforeEach
    public void init(){
        petDao = new PetDao();
    }
    @Test
    void getNewList() {
        List<Pet> newPetList = petDao.getNewList();//执行被测试方法
        assertEquals(12,newPetList.size());//断言：测试结果列表大小是否为 12
        for (Pet pet : newPetList) {        //输出宠物信息，非测试必须代码
            System.out.println(pet.getTitle());
        }
    }
}
```

在测试类文件 PetDaoTest.java 中输入源代码，其中@Test 是注解声明，表示 PetDaoTest 类中的 getNewList 方法是一个测试方法，可以在 JUnit 框架中运行测试。

assertEquals 方法主要用于比较传递进去的两个参数，当两个参数相等时则测试通过，否则测试失败。该方法中的第一个参数是期望值，第二个参数是测试方法运行的结果值，本实例中期望值为 12，实际值为 newPetList 列表包含的数据个数，当两个值一致时表示 getNewList 方法是正确的，测试通过。assertEquals 方法有多个重载方法，可以支持多种数据类型。

测试代码编写完成后，可以直接在代码界面运行测试。在图 9-4 中，单击行号 13 右侧的三角箭头，在弹出的快捷菜单中选择“Run 'getNewList()'”命令，执行测试方法。

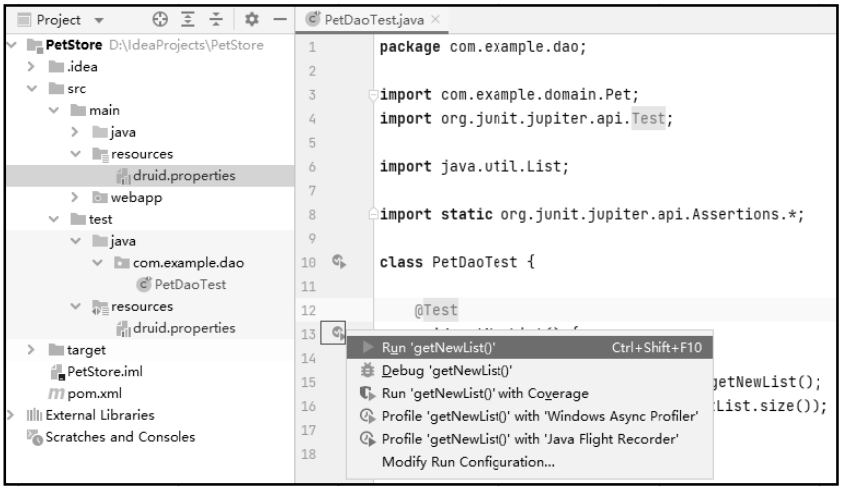

图 9-4　运行测试方法

测试运行结果如图 9-5 所示，方法名称前出现的“√”符号，表示测试通过，说明 getNewList 方法成功获取了数据库中的 12 条宠物数据。

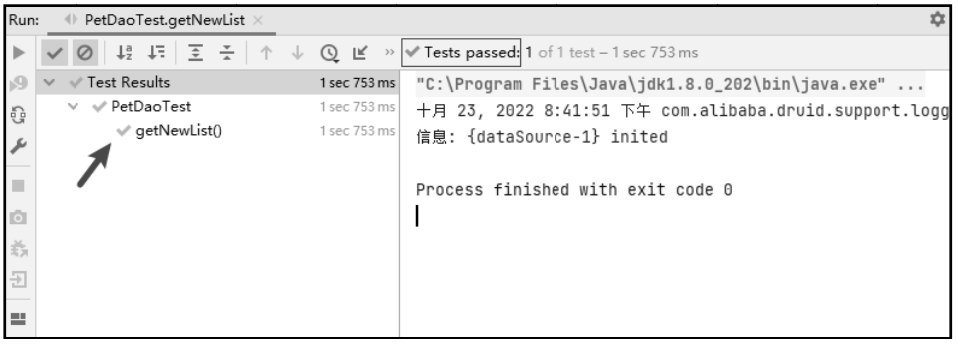

图 9-5　方法测试通过

测试运行结果如图 9-6 所示时，方法名称前出现叹号，表示测试失败，说明 getNewList 方法未能正确执行，开发人员需根据错误提示信息解决问题后再次进行测试，直到测试通过。

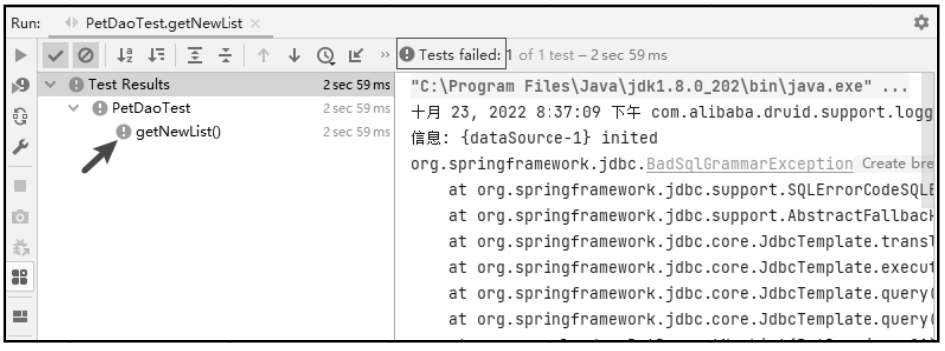

图 9-6　方法测试失败

测试说明：

- 执行测试程序时，使用的数据库连接配置文件 druid.properties 是在 src\test\resources 文件夹中，该文件与 src\main\resources 文件夹中的 druid.properties 文件内容保持一致。
- 当项目业务需求发生变化，数据库表结构进行调整时，所有 com.example.dao 包中的类都应该重新测试，确保模型（M）中的程序是能正确运行的。
- 模型（M）中的方法在编写完成后，都应当进行单元测试，为后续整体功能模块测试奠定基础。整体功能模块测试时需要涉及 MVC 三者之间的调用关系和数据流转，有一定的复杂性。此时若模型（M）中还有 Bug，会大幅增加整体功能模块的测试难度。

■ 9.1.4　控制器代码

在 PetStore 项目的 com.example.servlet 包上右击，在弹出的快捷菜单中选择 New→Servlet 命令，在弹出的窗口中输入 IndexServlet，按 Enter 键创建文件并输入源代码。

源程序：**IndexServlet.java** 文件

```
package com.example.servlet;

import com.example.dao.PetDao;
import com.example.domain.Pet;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.IOException;
import java.util.List;

@WebServlet("/IndexServlet")
public class IndexServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException{
        //1.获取请求参数（本次请求处理中无参数）

        //2.使用模型（M）对象执行业务方法，获取业务数据
        PetDao petDao = new PetDao();
        List<Pet> petList = petDao.getNewList();
        //3.将数据传递给视图（V）并展示（请求转发，浏览器的 URL 无变化）
        request.setAttribute("petList",petList);
        request.getRequestDispatcher("/index.jsp")
                .forward(request,response);
        //4.将数据传递给视图（V）并展示（重定向，浏览器的 URL 变化）
        //request.getSession().setAttribute("petList",petList);
        //response.sendRedirect(request.getContextPath()+"/index.jsp");
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        this.doGet(request, response);
    }
}
```

本项目中的控制器（C），即 Servlet 的代码文件，包含如下三个步骤。

- 获取请求参数。从 request 对象中读取请求参数值，该步骤根据业务功能的具体情况是可选的。
- 使用模型（M）对象执行业务方法，获取业务数据。该步骤的结果通常为获取一组对象或者单个对象。
- 将数据传递给视图。该步骤通常使用 request 对象或者 Session 对象向视图传递数据。request 对象用 setAttribute 方法写入数据后，使用请求转发方式向客户端展示视图，session 对象用 setAttribute 方法写入数据后，使用重定向方式向客户展示视图，开发人员可以根据实际需求选择使用这两种方式中的一种。

■ 9.1.5 视图代码

打开 PetStore 项目 src\main\webapp 文件夹中 index.jsp 文件，并输入源代码。

源程序：**index.jsp 文件**

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="ctx" value="${pageContext.request.contextPath}" />
<!DOCTYPE html>
<html>
<head>
    <title>宠物商城</title>
    <style>
        .petbox{
           width: 300px;
           padding: 0 20px;
           float: left;
        }
    </style>
</head>
<body>
    <h1>首页展示</h1>
    <c:forEach items="${petList}" var="pet">
     <div class="petbox">
        <img src="${ctx}/petimg/${pet.photo}"
               width="200" height="250" />
        <p>${pet.title}</p>
        <p>${pet.tag}</p>
        <p>${pet.price}</p>
        <p><a href="${ctx}/DetailServlet?id=${pet.id}">查看详情</a></p>
     </div>
    </c:forEach>
</body>
</html>
```

程序说明：

- JSP 页面使用 HTML 标记结合 JSTL 标签。
- “<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>”表示在页面中导入 JSTL 标签库。
- “<c:set var="ctx" value="${pageContext.request.contextPath}" />”表示在页面中定义 ctx 变量，在当前页面内有效，为了简化代码编写。
- 使用 JSTL 的 forEach 标签遍历 Servlet 传递的数据对象列表 petList，在页面上展示宠物的各项数据信息。
- “查看详情”超链接中 DetailServlet 的详细代码暂未编写，是为后续宠物详情功能做准备。

■ 9.1.6　项目部署配置

在项目功能模块测试运行前，需要对项目进行部署配置。在 IDEA 中，选择快捷菜单中的 Edit Configurations 命令，如图 9-7 所示。

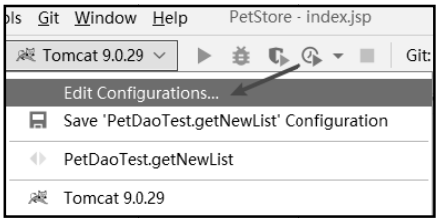

图 9-7　选择项目配置菜单

在弹出 Run/Debug Configurations 对话框中，选择 Web 服务器 Tomcat 9.0.29，再选择 Deployment 选项卡部署配置选项，选中或者添加 PetStore:war exploded 选项，配置 Application context 网站的虚拟路径为“/PetStore”，配置界面如图 9-8 所示。

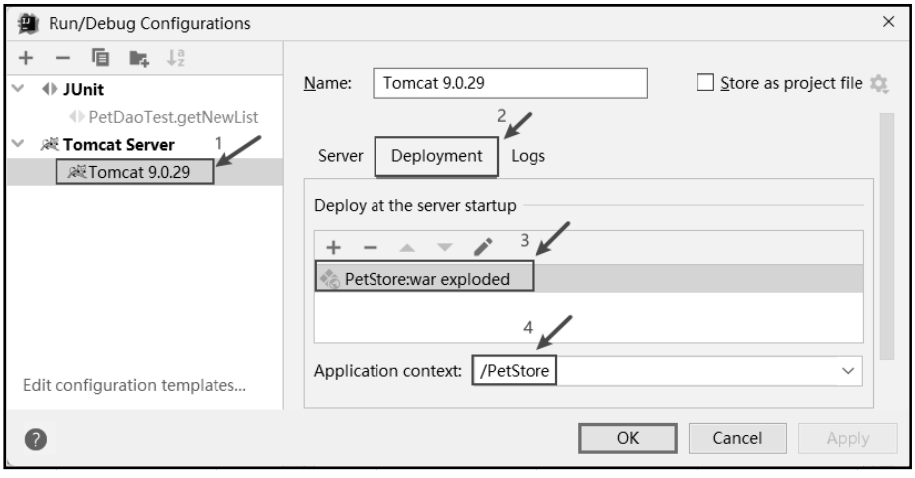

图 9-8　项目部署配置

在弹出的界面中，继续选择 Server 选项卡，设置项目启动时浏览器的默认访问网址为 http://localhost:8080/PetStore/IndexServlet。另外设置 On 'Update' action 和 On frame deactivation 的值为 Update classes and resources，当源代码发生改变并保存时，项目会自动重新编译并部署到 Tomcat 服务器，方便开发人员预览最新代码的运行效果，配置界面如图 9-9 所示。

■ 9.1.7　功能测试

单击运行工具栏中的“运行”按钮 ▶，启动 Tomcat。首页展示的效果图如图 9-10 所示。

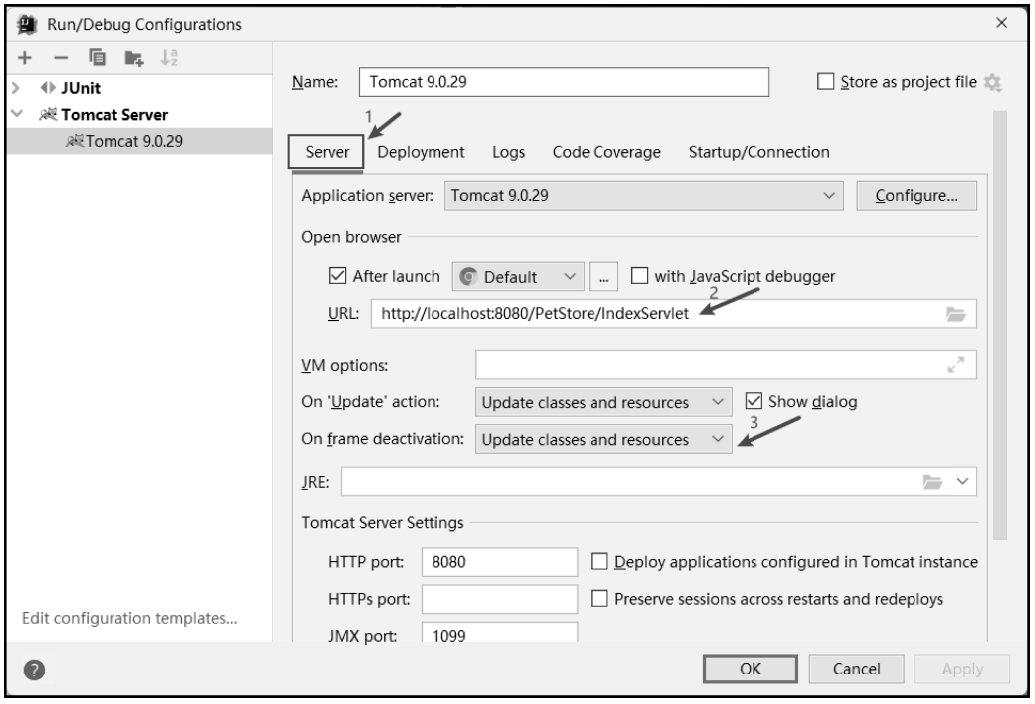

图 9-9　Web 服务器配置

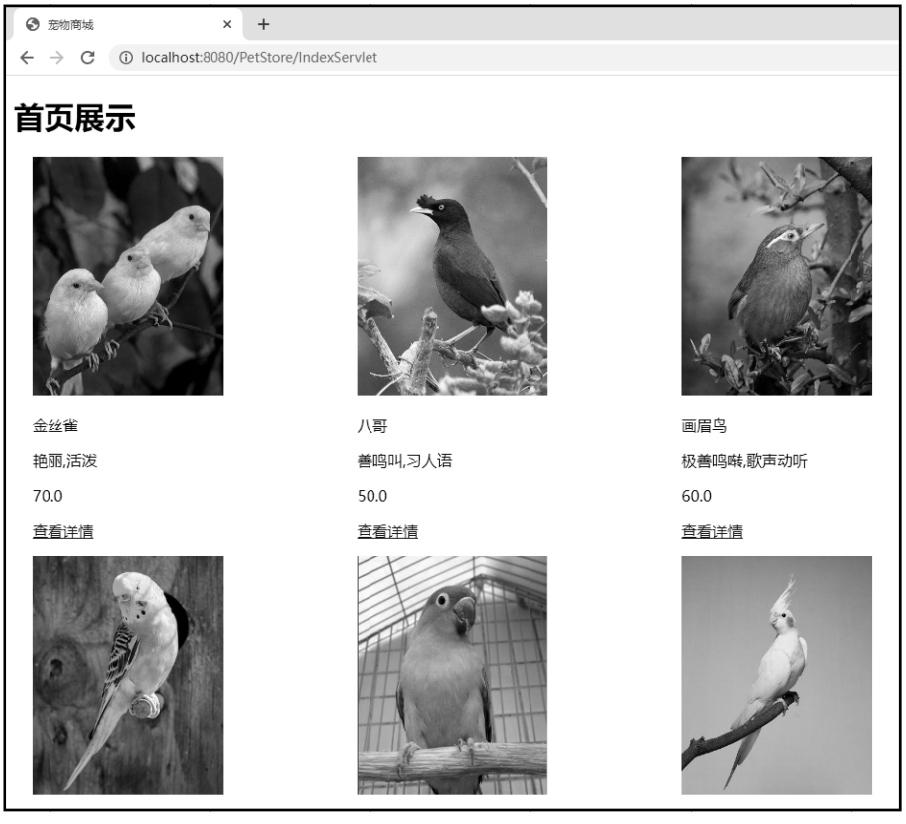

图 9-10　首页展示的效果图

在测试首页展示效果时，浏览器的网址应为 http://localhost:8080/PetStore/IndexServlet。本项目采用 MVC 开发模式，客户端请求由控制器（C），即 Servlet 进行处理，Servlet 调用模型（M）对象的方法获取业务数据，再将数据传递给视图（V）并响应给客户端浏览器，所以浏览器地址栏显示为访问 IndexServlet。

■ 9.1.8　视图优化

在视图 index.jsp 页面代码中，重点在于宠物数据信息的展示，未对页面进行美化修饰，页面样式比较简单。下面通过引入 Bootstrap 前端样式框架，对 index.jsp 页面进行优化。

在 PetStroe 项目的 webapp\img 文件夹添加 favicon.ico 和 logo.png 图片文件，webapp\css 文件夹添加 bootstrap-4.6.1.min.css、font-awesome-3.2.1.min.css 和 site.css 样式文件，webapp\js 文件夹添加 jquery-3.6.0.min.js 脚本文件，webapp\font 文件夹添加 fontawesome-webfont.eot、fontawesome-webfont.svg、fontawesome-webfont.ttf、fontawesome-webfont.woff 和 FontAwesome.otf。这些文件的用途如表 9-1 所示。读者可以下载本书配套资源中的项目源代码，其中包含了上述文件。

表 9-1　图片与样式文件清单

文　件　夹	文　　件	说　　明
img	favicon.ico	网页个性化小图标
img	logo.png	站点 LOGO 图片
css	site.css	网站样式
css	bootstrap-4.6.1.min.css	Bootstrap 前端样式库文件
css	font-awesome-3.2.1.min.css	Font Awesome 图标字体库样式文件
js	jquery-3.6.0.min.js	jQuery 提供的 JavaScript 库文件
font	fontawesome-webfont.eot	Font Awesome 字体文件
font	fontawesome-webfont.svg	Font Awesome 字体文件
font	fontawesome-webfont.ttf	Font Awesome 字体文件
font	fontawesome-webfont.woff	Font Awesome 字体文件
font	FontAwesome.otf	Font Awesome 字体文件

打开 index.jsp 文件，输入优化后的源代码，其中宠物数据展示的核心代码未变，主要增加了优化后的样式代码。

源程序：优化后的 index.jsp 文件

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="ctx" value="${pageContext.request.contextPath}" />
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>宠物商城</title>
    <link rel="shortcut icon" href="img/favicon.ico" />
```

```
    <link rel="stylesheet" href="css/bootstrap-4.6.1.min.css" />
    <link rel="stylesheet" href="css/font-awesome-3.2.1.min.css">
    <link rel="stylesheet" href="css/site.css" />
    <script src="js/jquery-3.6.0.min.js"></script>
</head>
<body>
<div class="d-flex flex-column flex-md-row align-items-center p-3
                        px-md-4 mb-3 bg-white border-bottom shadow-sm">
    <img src="img/logo.png" width="64" height="64" class="mb-2">
    <h5 class="my-0 mr-md-auto font-weight-normal">宠物商城</h5>
    <nav class="my-2 my-md-0 mr-md-3">
        <a class="p-2 text-dark" href="#">首页</a>
        <a class="p-2 text-dark" href="#">购物车</a>
        <a class="p-2 text-dark" href="#">联系客服</a>
    </nav>
</div>

<div class="container">
    <div class="card-deck mb-3 text-center">
        <c:forEach items="${petList}" var="pet">
            <div class="card mb-4 shadow-sm">
                <div class="card-header">
                    <a href="${ctx}/DetailServlet?id=${pet.id}">
                        <img src="${ctx}/petimg/${pet.photo}" class="pet-pic">
                    </a>
                </div>
                <div class="card-body">
                    <h1 class="card-title pricing-card-title">
     <small class="text-muted">${pet.title}</small>
                    </h1>
                    <p class="pet-desc">${pet.descs}</p>
                    <p><span class="pet-tag">${pet.tag}</span></p>
                    <p class="pet-price">￥${pet.price}</p>
                    <a class="btn btn-lg btn-block btn-outline-primary"
                        href="${ctx}/DetailServlet?id=${pet.id}">查看详情</a>
                </div>
            </div>
        </c:forEach>
    </div>
</div>
<footer class="footer mt-1 py-3">
    <div class="container">
        <div class="row">
            <div class="col-12 col-md">
                <img src="img/logo.png" width="24" height="24" class="mb-2">
                <small class="d-block mb-3 text-muted">© 2023</small>
            </div>
            <div class="col-6 col-md">
                <h5>备案信息</h5>
                <ul class="list-unstyled text-small">
                    <li><a class="text-muted" href="#">备案号</a></li>
```

```
                </ul>
            </div>
            <div class="col-6 col-md">
                <h5>工商信息</h5>
                <ul class="list-unstyled text-small">
                    <li><a class="text-muted" href="#">商业资质</a></li>
                </ul>
            </div>
            <div class="col-6 col-md">
                <h5>关于我们</h5>
                <ul class="list-unstyled text-small">
                    <li><a class="text-muted" href="#">宠物商城有限公司</a> </li>
                </ul>
            </div>
        </div>
    </div>
</footer>
</body>
</html>
```

源程序：**site.css** 文件

```
/*****通用样式*****/
body{
  font-family: "Microsoft YaHei","微软雅黑",Arial,sans-serif;
}
.container {
  max-width: 960px;
  margin: 0 auto;
}
.footer {
    border-top: 1px solid #f5f5f5;
  }
  .footer .container {
    width: auto;
    max-width: 680px;
  }

/***** index.jsp 页面样式 ******/
.card-deck .card {
  min-width: 220px;
}
.pet-pic{
  border-radius: 50%;
  transition: .5s transform;
  width: 100%;
  border: solid 3px #eee;
  box-shadow: #eee 1px 0px 15px 2px;
}
.pet-desc{
  text-overflow: ellipsis;
  white-space: nowrap;
```

```
  overflow: hidden;
}
.pet-tag{
  background: #2d8cf0;
  padding:3px 6px;
  margin-left: 3px;
  border-radius: 6px;
  color: #fff;
}
.pet-price{
  font-weight: bold;
  margin-right: 30px;
  color:#c30;
}
/***** detail.jsp 页面样式 *****/
input[name=quantity]{
  width: 120px;
}
```

程序说明：

- 增加了图片和样式文件，使用 Bootstrap 前端样式框架，整体页面效果优化。
- 测试本页面时，如果出现样式文件无法正常加载的情况，可以尝试重新启动 IDEA 软件。

优化后的首页展示的效果图如图 9-11 所示。

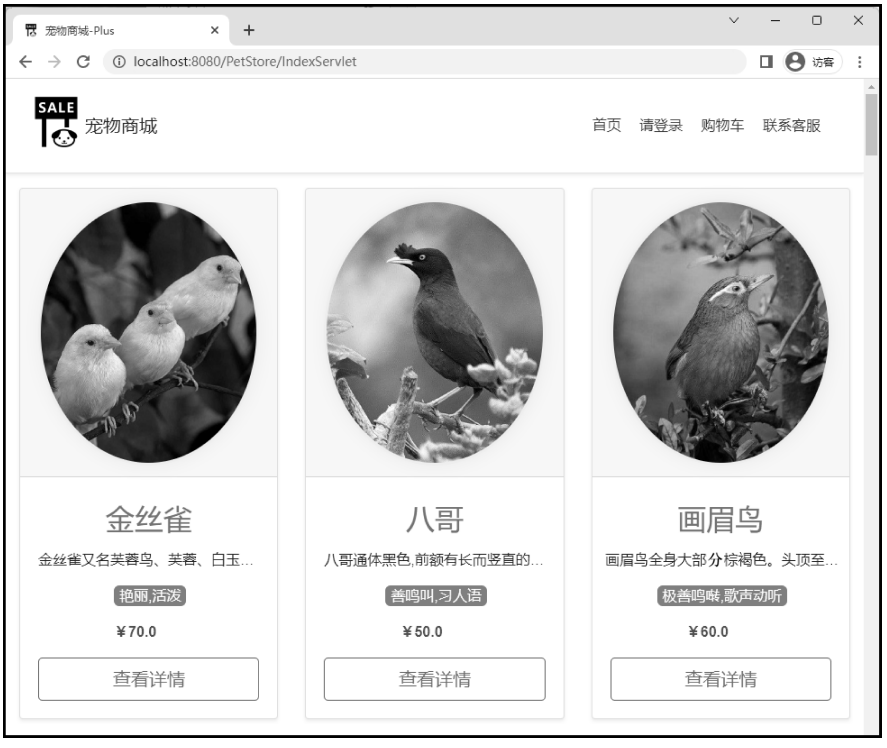

图 9-11　优化后的首页展示的效果图

9.2 宠物详情

项目 9-2

■ 9.2.1　功能简介

宠物详情的主要功能是展示单只宠物的详细信息，用户在该页面上可以输入购买数量并将宠物加入购物车，其中购物车添加功能在 9.2.2 节实现。结合功能需求和 MVC 开发模式，确定实现宠物详情功能需要编写的文件，如图 9-12 所示。

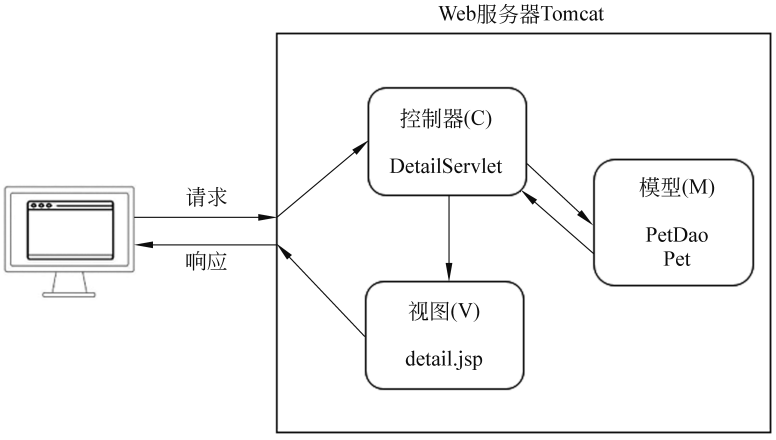

图 9-12　宠物详情的 MVC 模式图

■ 9.2.2　模型代码的编写

打开 PetDao.java 文件，在方法 getNewList 后面添加新方法 getById。

源程序：PetDao.java 文件中的 getById 方法

```
// 获取单个宠物对象
public Pet getById(int id){
 Pet pet = null;
 try {
   String sql = "select * from pets where id = ?";
   pet = template.queryForObject(sql,
           new BeanPropertyRowMapper<>(Pet.class),id);
 } catch (Exception e) {
    e.printStackTrace();
 } finally {
    return pet;
 }
}
```

程序说明：

- getById 方法根据宠物编号获取宠物对象，业务功能通过 SQL 语句完成。SQL 语句中的“?”为参数值占位符，其具体值由该方法的参数 id 传入。
- JDBCTemplate 对象的 queryForObject 方法用于获取单个对象，可自动封装数据库查询结果为 Pet 对象。

■ 9.2.3 模型代码的测试

打开 PetStore 项目的 PetDaoTest.java 文件，在测试方法 getNewList 后添加新的测试方法 getById。

源程序：**PetDaoTest.java 文件中的 getById 测试方法**

```
@Test
void getById() {
    Pet pet = petDao.getById(1);
    assertEquals(1,pet.getId());
    System.out.println(pet.getTitle());//输出宠物名称，非必须代码
}
```

单击测试方法 getById 的三角箭头，在弹出的快捷菜单中选择“Run 'getById()'”命令，执行测试方法。测试运行结果如图 9-13 所示，方法名称前出现“√”符号，表示测试通过，说明 getById 方法获取了编号为 1 的宠物对象。

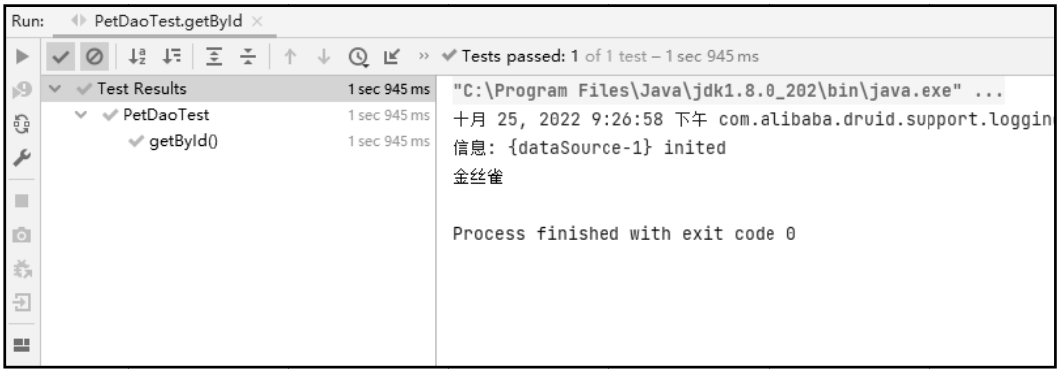

图 9-13 方法 getById 测试通过

■ 9.2.4 控制器代码

在 PetStore 项目的 com.example.servlet 包上右击，在弹出的快捷菜单中选择 New→Servlet 命令，在弹出的窗口中输入 DetailServlet，按 Enter 键创建文件并输入源代码。

源程序：**DetailServlet.java 文件**

```
package com.example.servlet;
import com.example.dao.PetDao;
import com.example.domain.Pet;
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.IOException;

@WebServlet("/DetailServlet ")
public class DetailServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
       HttpServletResponse response) throws ServletException, IOException {
        //1.获取请求参数 id
        String id = request.getParameter("id");
        //2.使用模型（M）对象执行业务方法，获取业务数据
        PetDao petDao = new PetDao();
        Pet pet = petDao.getById(Integer.parseInt(id));
        //3.将数据传递给视图（V）并展示（请求转发，浏览器的 URL 无变化）
        request.setAttribute("pet",pet);
        request.getRequestDispatcher("/detail.jsp")
                .forward(request,response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        this.doGet(request, response);
    }
}
```

本项目中的控制器（C），即 Servlet 的代码文件，包含如下三个步骤。

- 获取请求参数。从 request 对象中读取请求参数宠物编号的值。
- 使用模型（M）对象执行业务方法，获取业务数据。根据编号获取宠物对象。
- 将数据传递给视图。request 对象的 setAttribute 方法写入宠物对象数据后，使用请求转发方式向客户端展示视图 detail.jsp，视图 detail.jsp 中使用 JSTL 显示宠物对象数据。

■ 9.2.5　视图代码

在 PetStore 项目中的 src\main\webapp 文件夹上右击，在弹出的快捷菜单中选择 New→JSP/JSPX 命令，在弹出的窗口中输入 detail.jsp，按 Enter 键创建文件并输入源代码。

源程序：detail.jsp 文件

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="ctx" value="${pageContext.request.contextPath}" />
<!DOCTYPE html>
<html>
```

```
<head>
    <title>宠物商城</title>
    <style>
        .petbox{
            width: 600px;
            margin: 0 auto;
        }
    </style>
</head>
<body>
   <h1>"宠物详情"</h1>
   <div class="petbox">
       <img src="${ctx}/petimg/${pet.photo}" width="200" height="250" />
       <p>${pet.title}</p>
       <p>${pet.descs}</p>
       <p>${pet.tag}</p>
       <p>价格：${pet.price}</p>
       <p>库存：${pet.stock}</p>
       <form action="${ctx}/AddToCartServlet" method="post">
           <p>
               数量：<input type="text" name="quantity" value="1">
               <input type="hidden" name="id" value="${pet.id}">
           </p>
           <p>
               <input type="submit" value="加入购物车">
               <a href="${ctx}/IndexServlet">返回首页</a>
           </p>
       </form>
   </div>
</body>
</html>
```

程序说明：

- JSP 页面使用 HTML 标记结合 JSTL 标签。
- 使用 EL 表达式输出控制器（C），即 Servlet，通过 Request 传递的宠物对象 pet 的属性，在页面上展示宠物的详细信息。
- 在页面中设计了一个表单，包含隐藏表单元素宠物 id 和文本表单元素购买宠物的数量，表单提交到 AddToCartServlet，为后续购物车添加功能做准备。

■ 9.2.6 功能测试

单击运行工具栏中的“运行”按钮 ▶，启动 Tomcat。9.1.6 节中配置了项目启动后，默认功能是首页展示，在首页展示页面单击宠物的“查看详情”超链接，测试宠物详情功能，效果如图 9-14 所示。

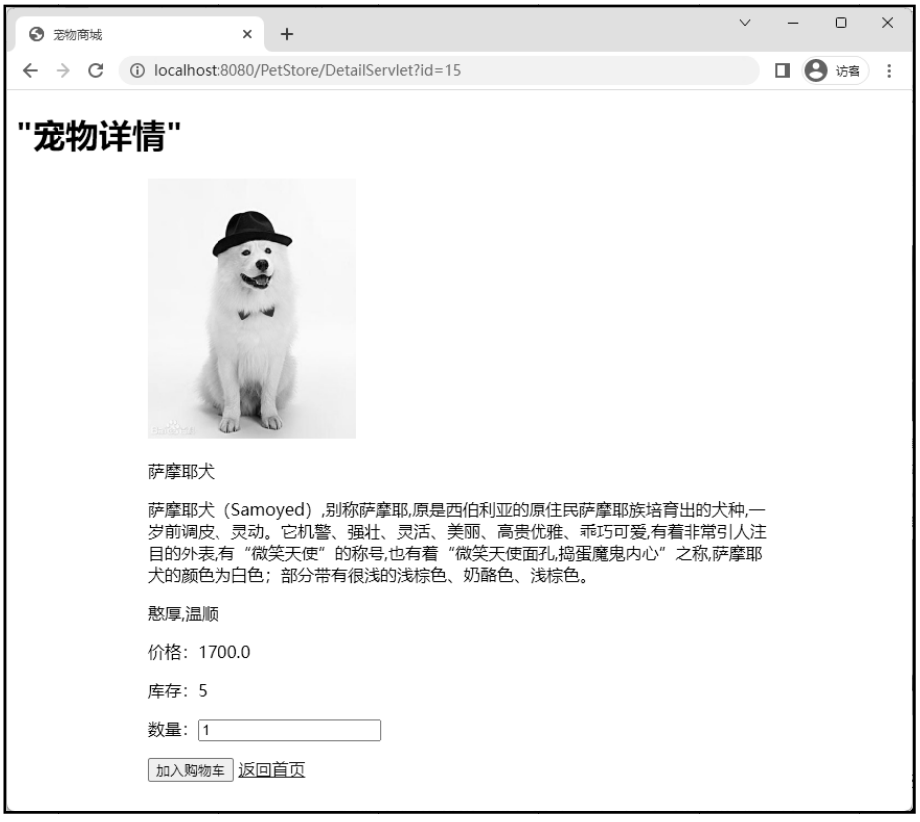

图 9-14　宠物详情的效果图

本项目采用 MVC 开发模式，客户端请求由控制器（C），即 Servlet 进行处理，再通过请求转发方式响应到客户端浏览器，所以地址栏显示为访问具体 Servlet。在宠物详情功能中，用户在首页单击“宠物详情”超链接，如 http://localhost:8080/PetStore/DetailServlet?id=15，发送请求到 Tomcat 服务器，由 DetailServlet 具体处理该请求，并通过服务器内部请求转发将视图响应给客户端，所以浏览器地址栏展示为“宠物详情”超链接网址。

■ 9.2.7　视图优化

当前 PetStroe 项目的 detail.jsp 文件内容重点在于展示单个宠物详细信息，页面没有使用样式代码进行效果优化，接下来通过样式代码的调整，对 detail.jsp 页面进行优化。

源程序：detail.jsp 新文件

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="ctx" value="${pageContext.request.contextPath}" />
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>宠物商城</title>
```

```
    <link rel="shortcut icon" href="img/favicon.ico" />
    <link rel="stylesheet" href="css/bootstrap-4.6.1.min.css" />
    <link rel="stylesheet" href="css/font-awesome-3.2.1.min.css">
    <link rel="stylesheet" href="css/site.css" />
    <script src="js/jquery-3.6.0.min.js"></script></head>
<body>
<div class="d-flex flex-column flex-md-row align-items-center p-3
                      px-md-4 mb-3 bg-white border-bottom shadow-sm">
    <img src="img/logo.png" width="64" height="64" class="mb-2">
    <h5 class="my-0 mr-md-auto font-weight-normal">宠物商城</h5>
    <nav class="my-2 my-md-0 mr-md-3">
        <a class="p-2 text-dark" href="#">首页</a>
        <a class="p-2 text-dark" href="#">购物车</a>
        <a class="p-2 text-dark" href="#">联系客服</a>
    </nav>
</div>

<div class="container">
    <div class="row no-gutters border rounded flex-md-row mb-4
                shadow-sm h-md-250">
        <div class="col-auto d-none d-lg-block">
            <img src="${ctx}/petimg/${pet.photo}"
                      width="300" heght="400" class="mb-2">
        </div>
        <div class="col p-4 d-flex flex-column">
            <h3 class="d-inline-block mb-2 text-dark">${pet.title}</h3>
            <div class="mb-2 text-muted">
                <span class="pet-tag">${pet.tag}</span>
            </div>
            <p class="card-text">${pet.descs}</p>
            <p>价格：￥<span id="pet-price">${pet.price}</span></p>
            <p>库存：<span id="pet-stock">${pet.stock}</span></p>
            <form action="${ctx}/AddToCartServlet" method="post">
                <p>
                    <label for="pet-quantity">数量：</label>
                    <input type="text" id="pet-quantity"
                           name="quantity" value="1">
                    <input type="hidden" name="id" value="${pet.id}">
                </p>
                <nav>
                    <input class="btn btn-warning" type="submit"
                           value="加入购物车">
                    <a class="btn btn-warning" href="${ctx}/IndexServlet">
                        返回首页
                    </a>
                </nav>
            </form>
```