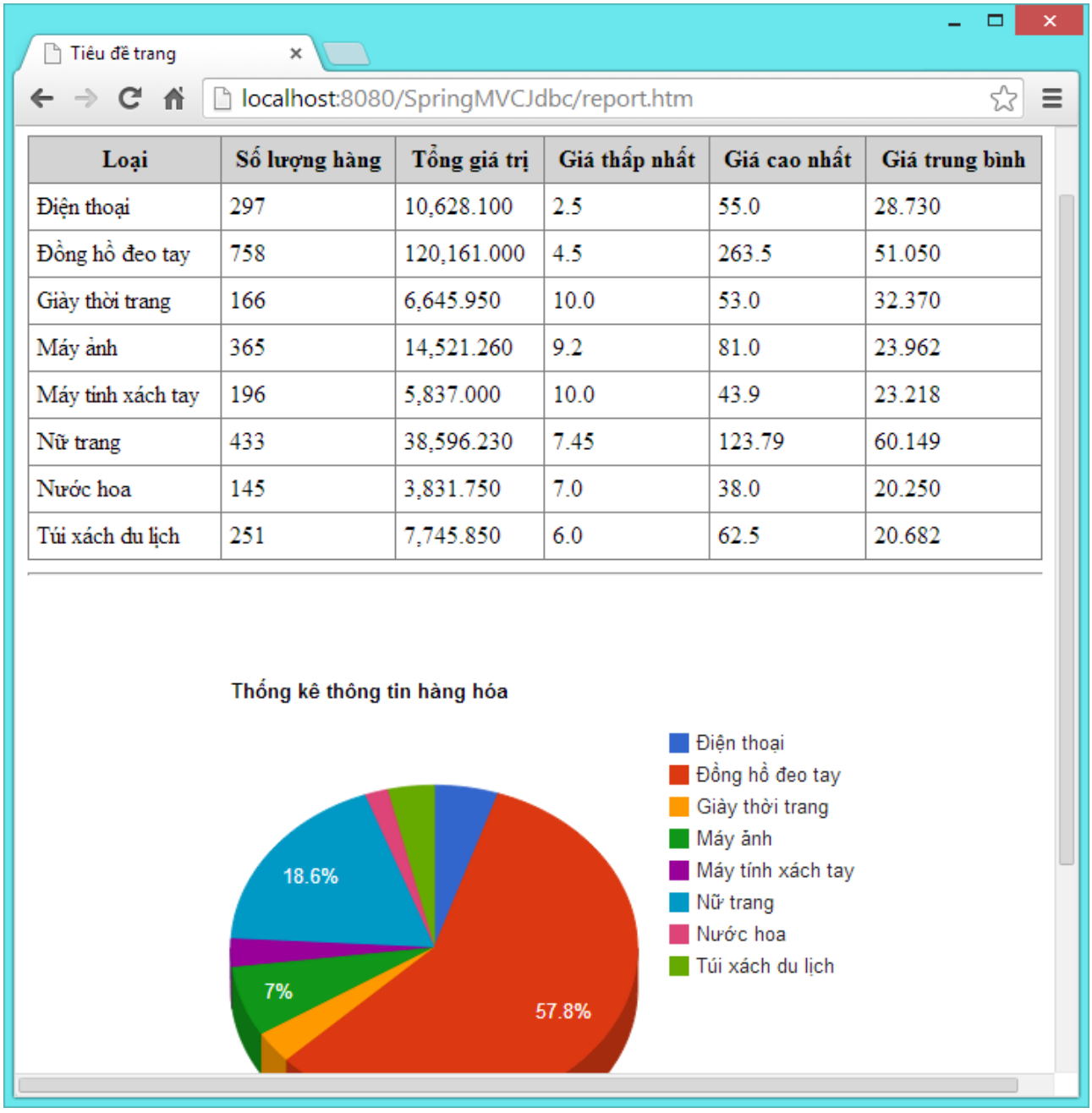
## MỤC TIÊU

Kết thúc bài thực hành này, bạn có khả năng

- ✓ Thống kê số liệu
- ✓ Kết hợp vơi Google chart để vẽ biểu đồ

## MÔ TẢ

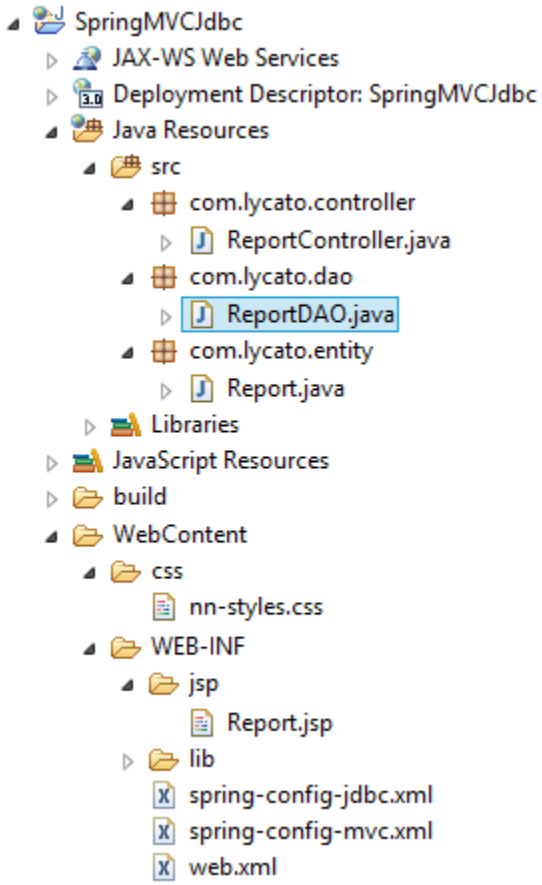



| Loại | Số lượng hàng | Tổng giá trị | Giá thấp nhất | Giá cao nhất | Giá trung bình |
|---|---|---|---|---|---|
| Điện thoại | 297 | 10,628.100 | 2.5 | 55.0 | 28.730 |
| Đồng hồ đeo tay | 758 | 120,161.000 | 4.5 | 263.5 | 51.050 |
| Giày thời trang | 166 | 6,645.950 | 10.0 | 53.0 | 32.370 |
| Máy ảnh | 365 | 14,521.260 | 9.2 | 81.0 | 23.962 |
| Máy tính xách tay | 196 | 5,837.000 | 10.0 | 43.9 | 23.218 |
| Nữ trang | 433 | 38,596.230 | 7.45 | 123.79 | 60.149 |
| Nước hoa | 145 | 3,831.750 | 7.0 | 38.0 | 20.250 |
| Túi xách du lịch | 251 | 7,745.850 | 6.0 | 62.5 | 20.682 |

Hoạt động: Chạy report.htm hiển thị trang web gồm số liệu thống kê và biểu đồ được vẽ dựa vào số liệu thống kê ở bảng trên.

## THỰC HIỆN

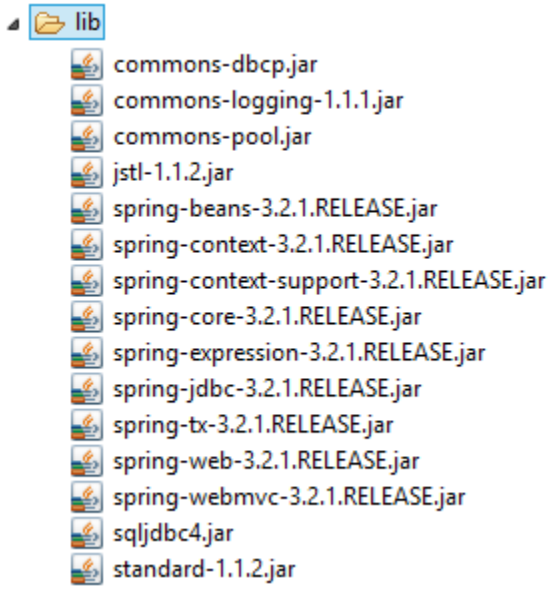
Trong bài này, bạn sẽ phải tạo một project có tổ chức như sau:




- ✓ Bước 1: Thư viện và cấu hình project
- ✓ Bước 2: CSDL
- ✓ Bước 4: Tạo lớp DAO và Entity
- ✓ Bước 5: Tạo giao diện
- ✓ Bước 6: Tạo Controller
- ✓ Bước 7: Chạy

## Bước 1: Thư viện và cấu hình project

### Thư viện




Bên cạnh các thư viện của Thư viện cần thiết cho ứng dụng

- ✓ SQLServerDriver
  - sqljdbc4.jar
- ✓ JdbcTemplate
  - commons-dbcp.jar
  - spring-jdbc-3.2.1.RELEASE.jar
  - spring-tx-3.2.1.RELEASE.jar

### Cấu hình

❖ **Web.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app                                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
	<display-name>SpringMVCEmail</display-name>
	<welcome-file-list>
		<welcome-file>index.jsp</welcome-file>
	</welcome-file-list>
	<servlet>
		<servlet-name>dispatcher</servlet-name>
		<servlet-class>
			org.springframework.web.servlet.DispatcherServlet
		</servlet-class>
		<init-param>
			<param-name>contextConfigLocation</param-name>
			<param-value>/WEB-INF/spring-config-*.xml</param-value>
		</init-param>
		<load-on-startup>1</load-on-startup>
```

```
	</servlet>
	<servlet-mapping>
		<servlet-name>dispatcher</servlet-name>
		<url-pattern>*.htm</url-pattern>
	</servlet-mapping>
</web-app>
```

Cấu hình để Spring MVC nạp nhiều file cấu hình: spring-config-*.xml. Dấu * sẽ đại diện cho nhóm ký tự bất kỳ. Cụ thể ở bài này là mvc, gmail và upload

- **spring-config-mvc.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:context="http://www.springframework.org/schema/context"
	xmlns:p="http://www.springframework.org/schema/p"
	xmlns:tx="http://www.springframework.org/schema/tx"
	xmlns:mvc="http://www.springframework.org/schema/mvc"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
			http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
			http://www.springframework.org/schema/context
			http://www.springframework.org/schema/context/spring-context-3.2.xsd
			http://www.springframework.org/schema/tx
			http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
			http://www.springframework.org/schema/mvc
			http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

	<!-- Declare a view resolver -->
	<bean id="viewResolver" p:prefix="/WEB-INF/jsp/" p:suffix=".jsp"
	class="org.springframework.web.servlet.view.InternalResourceViewResolver"/>

	<!-- Spring MVC Annotation -->
	<mvc:annotation-driven />
	<context:annotation-config />

	<!-- Where to find component -->
	<context:component-scan base-package="com.lycato" />
</beans>
```

- ✓ Khai báo bean InternalResourceViewResolver để xử lý view
- ✓ Chỉ rõ package tìm kiếm các component là com.lycato
- ✓ Chỉ rõ ứng dụng Spring này được phép sử dụng annotation

- **spring-config-jdbc.xml**

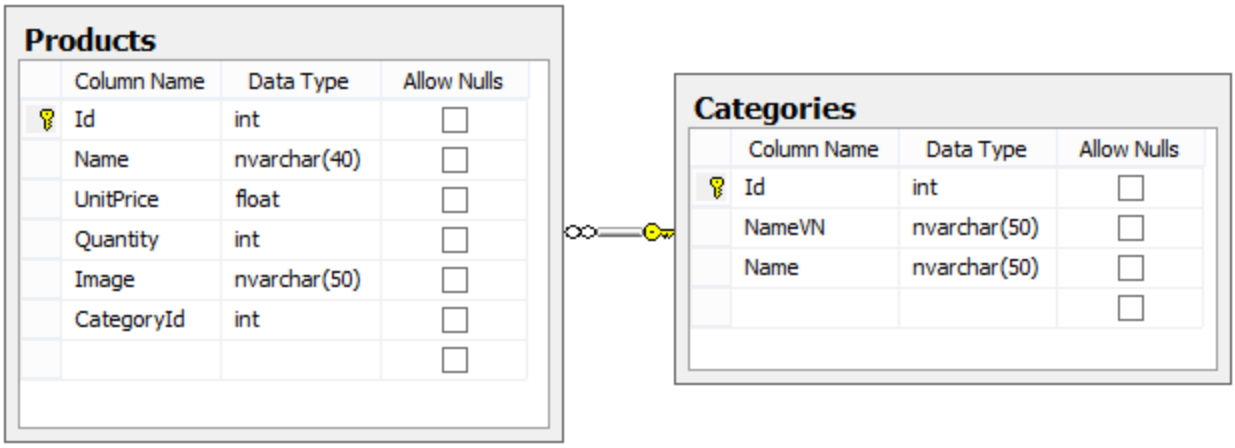
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:context="http://www.springframework.org/schema/context"
	xmlns:p="http://www.springframework.org/schema/p"
	xmlns:tx="http://www.springframework.org/schema/tx"
	xmlns:mvc="http://www.springframework.org/schema/mvc"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
			http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
			http://www.springframework.org/schema/context
			http://www.springframework.org/schema/context/spring-context-3.2.xsd
			http://www.springframework.org/schema/tx
			http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
```

```
                    http://www.springframework.org/schema/mvc
                    http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

        <bean id="dataSource" destroy-method="close"
                class="org.apache.commons.dbcp.BasicDataSource"
                p:driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
                p:url="jdbc:sqlserver://localhost:1433;DatabaseName=Seminar"
                p:username="sa"
                p:password="songlong"/>

        <bean id="jdbcTemplate"
                class="org.springframework.jdbc.core.JdbcTemplate">
                <constructor-arg ref="dataSource"/>
        </bean>

</beans>
```

Trong file cấu hình này bạn phải khai báo 2 bean.

- ✓ BasicDataSource: bean này cấu hình các thông số kết nối CSDL
- ✓ JdbcTemplate: bean này được khai báo đến làm việc với CSDL được tiêm vào và sử dụng sau này trong ứng dụng

## Bước 2: CSDL




Hình: CSDL Seminar chứa 2 bảng Products và Categories

## Bước 4: Tạo lớp mô tả và truy xuất dữ liệu

### Lớp mô tả dữ liệu (Entity)

#### *Report.java*

```
package com.lycato.entity;

public class Report {
        String group;
        Double sum, avg, min, max;
        Integer count;

        public String getGroup() {
                return group;
```

```
	}
	public void setGroup(String group) {
		this.group = group;
	}
	public Double getSum() {
		return sum;
	}
	public void setSum(Double sum) {
		this.sum = sum;
	}
	public Double getAvg() {
		return avg;
	}
	public void setAvg(Double avg) {
		this.avg = avg;
	}
	public Double getMin() {
		return min;
	}
	public void setMin(Double min) {
		this.min = min;
	}
	public Double getMax() {
		return max;
	}
	public void setMax(Double max) {
		this.max = max;
	}
	public Integer getCount() {
		return count;
	}
	public void setCount(Integer count) {
		this.count = count;
	}
}
```

### Lớp truy xuất dữ liệu (DAO)

Lớp này chứa các phương thức getProductReport() để lấy thông tin báo cáo hàng hóa. Lớp này được chú thích bởi @Repository để có thể tiêm vào ReportController trong ứng dụng bởi @Autowire để sử dụng sau này.

#### *ReportDAO.java*

```
package com.lycato.dao;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;

import com.lycato.entity.Report;

@Repository
public class ReportDAO{
	/**
	 * Inject từ <bean class="...JdbcTemplate>
	 */
	@Autowired
```

```
	protected JdbcTemplate jdbc;

	/**
	 * Truy vấn thực thể
	 * @return danh sách thực thể truy vấn được
	 */
	public List<Report> getProductReport() {
		String sql = "SELECT c.NameVN as [Group], SUM(p.UnitPrice * p.Quantity) AS
Sum, SUM(p.Quantity) AS Count, MIN(p.UnitPrice) AS Min, MAX(p.UnitPrice) AS Max,
AVG(p.UnitPrice) AS Avg FROM Products AS p INNER JOIN Categories AS c ON p.CategoryId =
c.Id GROUP BY c.NameVN";
		return jdbc.query(sql, getRowMapper());
	}

	/**
	 * Ánh xạ cấu trúc bản ghi theo thuộc tính của bean
	 * @return ánh xạ bản ghi theo thuộc tính bean
	 */
	private RowMapper<Report> getRowMapper() {
		return new BeanPropertyRowMapper<Report>(Report.class);
	}
}
```

Câu lệnh SQL thống kê số liệu như sau

```
SELECT
	c.NameVN as [Group],
	SUM(p.UnitPrice * p.Quantity) AS Sum,
	SUM(p.Quantity) AS Count,
	MIN(p.UnitPrice) AS Min,
	MAX(p.UnitPrice) AS Max,
	AVG(p.UnitPrice) AS Avg
FROM Products AS p
	INNER JOIN Categories AS c
		ON p.CategoryId = c.Id
GROUP BY c.NameVN
```

## Bước 5: Tạo giao diện

### Report.jsp

```
<%@page pageEncoding="utf-8" contentType="text/html; charset=utf-8" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jstl/fmt_rt" prefix="fmt" %>

<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Tiêu đề trang</title>
	<link href="css/nn-styles.css" rel="stylesheet">
</head>
<body>
	<h2>THỐNG KÊ THÔNG TIN HÀNG HÓA</h2>
	<table class="grid-view">
	<tr>
		<th>Loại</th>
		<th>Số lượng hàng</th>
		<th>Tổng giá trị</th>
		<th>Giá thấp nhất</th>
		<th>Giá cao nhất</th>
```

```
            <th>Giá trung bình</th>
        </tr>
        <c:forEach var="i" items="${items}">
        <tr>
            <td>${i.group}</td>
            <td>${i.count}</td>
            <td><fmt:formatNumber value="${i.sum}"
                minFractionDigits="3" maxFractionDigits="3"/></td>
            <td>${i.min}</td>
            <td>${i.max}</td>
            <td><fmt:formatNumber value="${i.avg}"
                minFractionDigits="3" maxFractionDigits="3"/></td>
        </tr>
        </c:forEach>
        </table>
        <hr>
        <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
      google.load("visualization", "1", {packages:["corechart"]});
      google.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Loại', 'Tổng giá trị'],
          <c:forEach var="i" items="${items}">
          ['${i.group}', ${i.sum}],
          </c:forEach>
        ]);

        var options = {
          title: 'Thống kê thông tin hàng hóa',
          is3D: true,
        };

        var chart = new
google.visualization.PieChart(document.getElementById('piechart_3d'));
        chart.draw(data, options);
      }
    </script>
        <div id="piechart_3d" style="width: 700px; height: 500px;"></div>
</body>
</html>
```

### nn-styles.css

Định nghĩa style trình bày bảng số liệu thống kê.

```
table.grid-view{
        width:100%;
        border-collapse: collapse;
}
table.grid-view th, table.grid-view td{
        padding: 5px;
        border: 1px solid gray;
}
table.grid-view th{
        background-color: lightgray;

}
```

## Bước 6: Tạo Controller

```
package com.lycato.controller;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;

import com.lycato.dao.ReportDAO;

@Controller
public class ReportController {
	@Autowired
	ReportDAO rdao;

	@RequestMapping("report")
	public String report(ModelMap model) {
		model.addAttribute("items", rdao.getProductReport());
		return "Report";
	}
}
```

Khi bạn tương tác vào chương trình tương:

| STT | Hành động | Phương thức | Mô tả |
|---|---|---|---|
| 1 | Chạy products.htm | showProducts(), GET | Hiển thị menu loại và danh sách hàng hóa |
| 2 | Nhấp [loại] | showProducts(categoryId), POST | Hiển thị menu loại và danh sách hàng hóa của loại được chọn |

## Bước 7: Chạy

http://localhost:8080/SpringMVCJdbc/products.htm