Xây dựng ứng dụng web căn bản với ZK framework

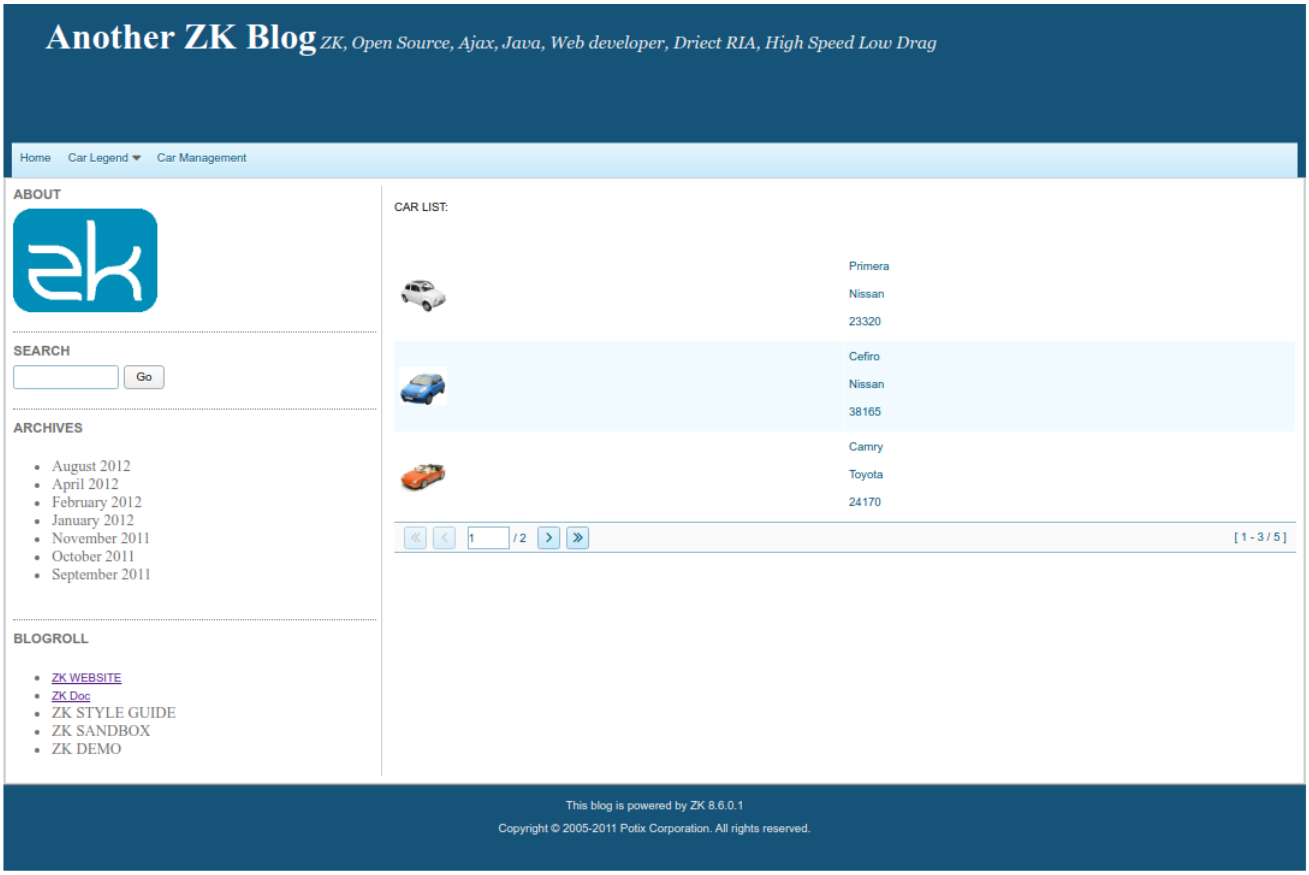
***Lưu ý**: Cần đọc slide giới thiệu tổng quan các thành phần trong ZK và thực hành bài hướng dẫn tạo ứng dụng đăng ký user trước khi thực hành tạo ứng dụng này.*

Nội dung hướng dẫn

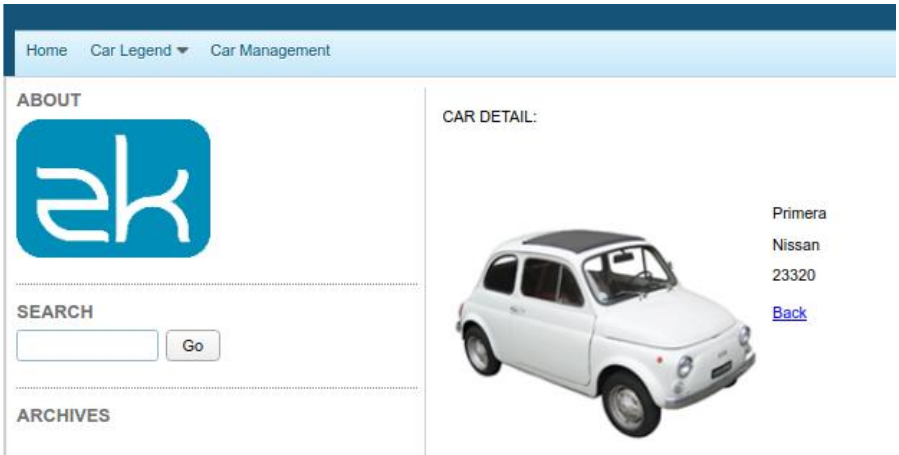
1. Bài toán ứng dụng

1.1 Đặc tả yêu cầu

Xây dựng ứng dụng web căn bản với các chức năng hiển thị sản phẩm, phân trang, tìm kiếm, thêm, sửa, xóa. Yêu cầu được minh họa qua các màn hình giao diện sau:

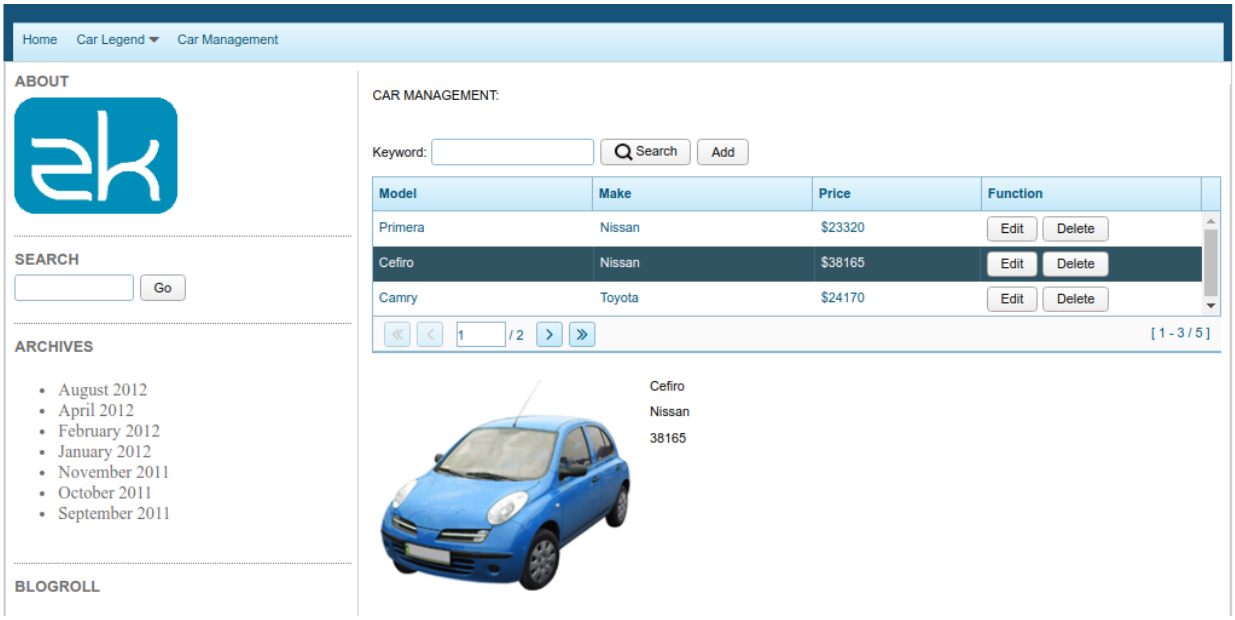


Trang chủ, khi click vào một ô tô hiển thị nội dung chi tiết. Nếu bấm "Back" quay về trang trước.



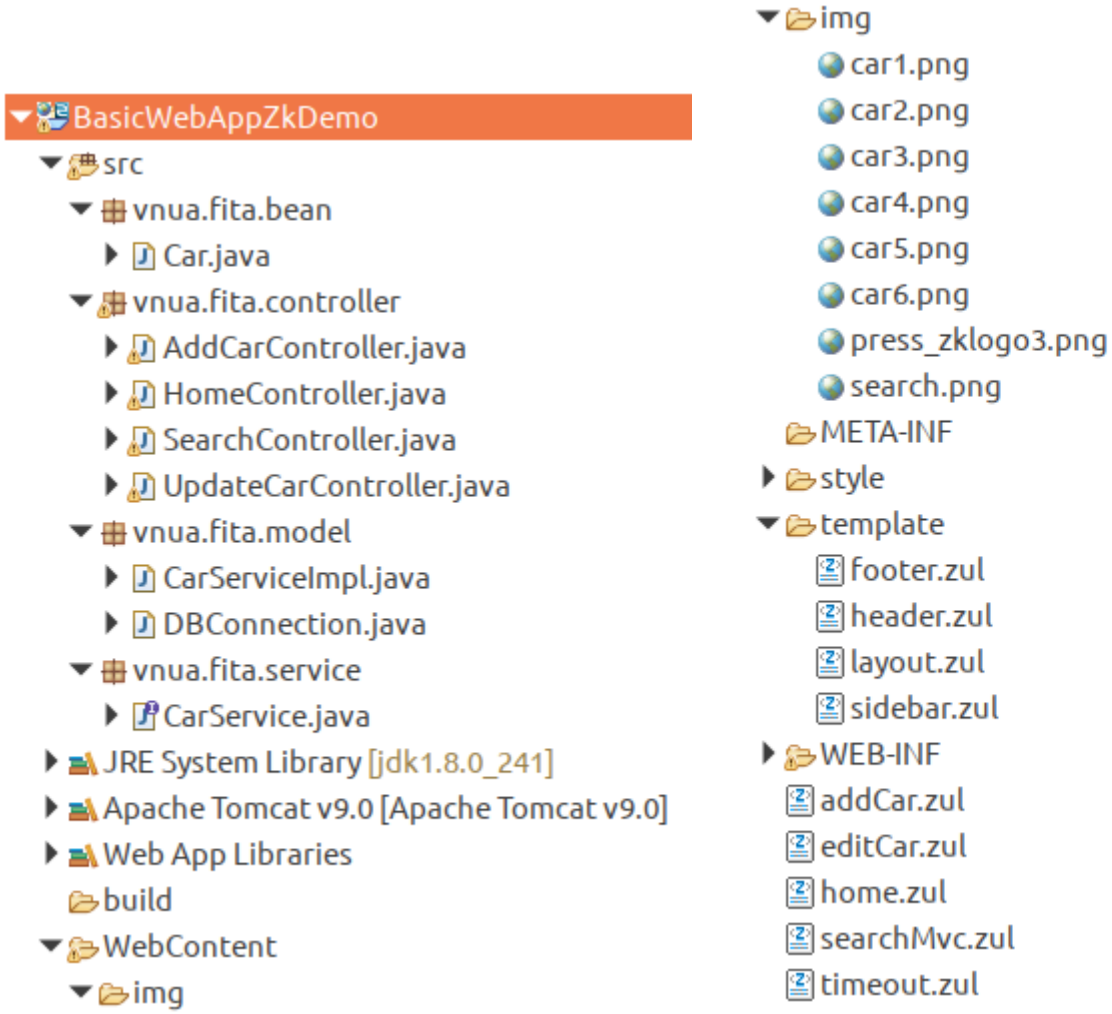
Menu Car Legend cho chọn các loại xe cụ thể, cách hiển thị tương tác tương tự trang chủ.

Menu Car Management hiển thị trang quản lý ô tô với các chức năng tìm kiếm, phân trang, xem, thêm, sửa , xóa:



Mặc định hiển thị danh sách tất cả ô tô. Nếu ô keyword để trống, kết quả tìm kiếm trả về danh sách tất cả ô tô. Khi click chọn một ô tô trong danh sách kết quả tìm kiếm, hình ảnh, thông tin được hiện bên dưới.

1.2 Cấu trúc ứng dụng



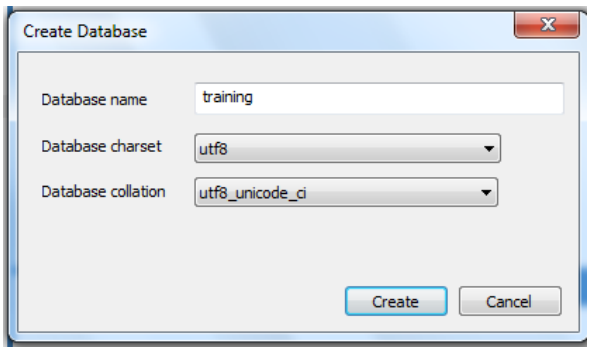
1.3 Các file cần download

Các file cần download nằm trong thư mục đính kèm:

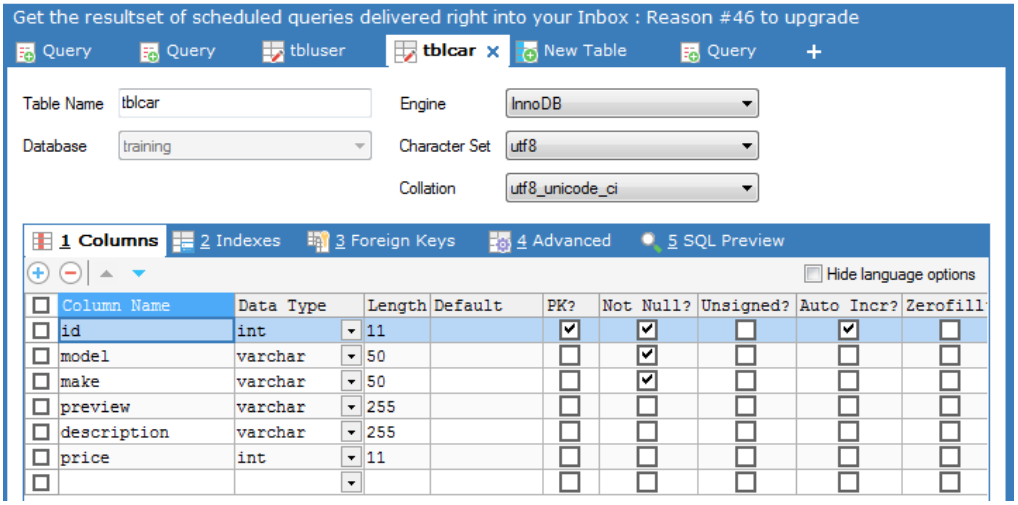
- Car.java, CarService.java, CarServiceImpl.java, DBConnection.java, AddCarController.java
- img: thư mục chứa ảnh
- default.zul, sidebar.zul, footer.zul
- tblcar_data.sql: chứa các câu lệnh sql để tạo dữ liệu cho bảng tblcar
- web.xml, zk.xml: paste vào thư mục web-inf

2. Thiết kế cơ sở dữ liệu

- Tạo database

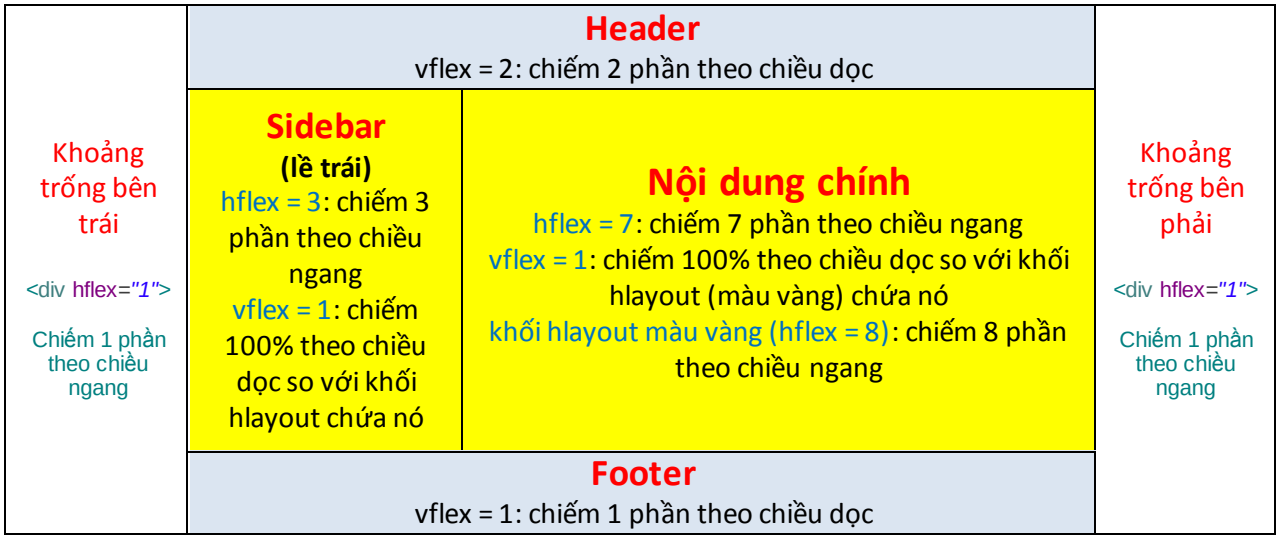


- Tạo bảng tblcar



- Chèn dữ liệu vào bảng tblcar
  Sử dụng các lệnh insert trong file tblcar_data.sql để chạy và chèn dữ liệu vào bảng

3. Thiết kế layout tổng thể



- **layout.zul**

```
<zk>
    <style><![CDATA[
      .fixed-center .header {
        padding-left: 8px;
                padding-right: 8px;
      }
      .center-left,
      .center-right {
        border: none;
        padding-right: 8px;
      }
      .center {
        border: 1px solid #CFCFCF;

      }

    ]]></style>
    <include id="includeStyle" src="style/default.zul" />
    <hlayout width="100%" height="900px">
            <div hflex="1"></div>
            <div hflex="8" vflex="1" class="fixed-center">
                    <include vflex="2" class="header" src="template/header.zul" />
                    <hlayout vflex="7" class="center">
                            <div vflex="1" hflex="3" class="center-left">
                                    <include src="template/sidebar.zul" class="center-left-inner" />
                            </div>
                            <div vflex="1" hflex="7" class="center-right" self="@insert(content)">
                                    <!-- include home.zul or nissan.zul after -->
                            </div>
                    </hlayout>
                    <include vflex="1" class="footer" src="template/footer.zul" />
            </div>
            <div hflex="1"></div>
    </hlayout>

</zk>
```

Thêm style trong khối thẻ và bằng include

Sử dụng các lớp style được khai báo

Căn theo chiều ngang 3 khối chính (khối khoảng trống trái, khối trang web, khối khoảng trống phải)

Cách để chèn động trang nội dung về sau

- **header.zul**

```
<vlayout vflex = "1" hflex = "1">
        <div vflex="4">
                <separator class="header-bar" />
                <label class="header-title1" value="Another ZK Blog" />
                <label class="header-title2" value="ZK, Open Source, Ajax, Java, Web developer, Driect
RIA, High Speed Low Drag" />
        </div>
        <div vflex="1" style="background-color:white;border-bottom:0px">
                <menubar id="menubar" vflex = "1"       width="100%" autodrop="true" style="border-
bottom:0px">
                        <menuitem label="Home" href="/home.zul" />

                        <menu label="Car Legend">
                                <menupopup>
                                        <menuitem label="Nissan" href="/home.zul?type=nissan"/>
                                        <menuitem label="Toyota" href="/home.zul?type=toyota"/>
                                </menupopup>
                        </menu>
                        <menuitem label="Car Management" href="/searchMvc.zul" />
                </menubar>
        </div>
</vlayout>
```

Đặt vflex, hflex = 1 để khối chiếm 100% theo chiều dọc và ngang

Các mục nhỏ sử dụng menuitem, nếu có menu con, cần sử dụng thẻ menu chứa menupopup

Header chia hai khối căn theo chiều dọc, khối menu chiếm một phần (vflex = 1), khối text chiếm 4 phần (vflex = 4). Cần cấu hình style để bỏ đường viền của menu và khối div chứa nó

- **zk.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<zk>
        <library-property>
                <name>org.zkoss.theme.preferred</name>
                <value>sapphire</value>  <!-- or silvertail, atlantic, deepsea, gardensalad  etc. -->
        </library-property>
</zk>
```

Thẻ cấu hình chọn một theme có sẵn trong danh sách của zk. Nếu ko cấu hình, tùy từng phiên bản zk có theme mặc định khác nhau. Xem ZK Official Themes để biết chi tiết

4. Xây dựng trang nội dung chính home.zul

4.1 Thiết kế layout

```
<?init class="org.zkoss.zk.ui.util.Composition"
arg0="/template/layout.zul"?>
<zk>
        <vlayout self="@define(content)" apply="vnua.fita.controller.HomeController">

                <div height="50px" style = "margin-top:20px;">
                        <label id = "titleLabel"/>
                </div>
                <listbox id="carListbox" mold="paging" pageSize="3"
                        emptyMessage="No car found in the result" style="border: 0px;">

                </listbox>

                <hlayout id="carDetail" style="margin-top:20px" width="100%"
                        visible="false">
                        <image id="previewImage" width="250px" />
                        <vlayout hflex="1">

                        </vlayout>
                </hlayout>
        </vlayout>
</zk>
```

Chèn toàn bộ khối khai báo @define(content) vào vùng @insert(content) trong layout.zul

Khai báo dùng layout.zul cho trang

Khai báo controller xử lý

Khối tiêu đề

Khối hiển thị danh sách các ô tô

Khối thông tin chi tiết ô tô được chọn

ảnh hiển thị bên trái, thông tin chi tiết được xếp theo chiều dọc bên phải. Visible = false mặc định khối carDetail ẩn, thuộc tính này bằng true khi ô tô được chọn

Code chi tiết:

```
<?init class="org.zkoss.zk.ui.util.Composition"
arg0="/template/layout.zul"?>
<zk>
        <vlayout self="@define(content)" apply="vnua.fita.controller.HomeController">

                <div height="50px" style = "margin-top:20px;">
                        <label id = "titleLabel"/>
                </div>
                <listbox id="carListbox" mold="paging" pageSize="3"
                        emptyMessage="No car found in the result" style="border: 0px;">
                        <template name="model">
                                <listitem>
                                        <listcell>
                                                <image src="${each.preview}" width="50px" />
                                        </listcell>
                                        <listcell>
                                                <vlayout hflex="1">
                                                        <label value="${each.model}" />
                                                        <label value="${each.make}" />
                                                        <label value="${each.price}" />
                                                        <label value="${each.description}" />
                                                </vlayout>

                                        </listcell>

                                </listitem>
                        </template>
                </listbox>

                <hlayout id="carDetail" style="margin-top:20px" width="100%"
                        visible="false">
                        <image id="previewImage" width="250px" />
                        <vlayout hflex="1">
```

Phân trang: 3 bản ghi/trang

Template: Cách lặp qua một List các đối tượng dữ liệu Car được cung cấp bởi HomeController

Đường dẫn ảnh trong thư mục img

Listitem: ứng với dòng

Listcell ứng với cột

${each.<tên thuộc tính của đối tượng Car chứa dữ liệu >

```
                        <label id="modelLabel" />
                        <label id="makeLabel" />
                        <label id="priceLabel" />
                        <label id="descriptionLabel" />
                        <a id="back" label="Back" />
                    </vlayout>
                </hlayout>
            </vlayout>

</zk>
```

4.2 Viết code cho lớp tầng Controller

HomeController.java

```
package vnua.fita.controller;

import java.util.List;
import java.util.Set;

import org.zkoss.zk.ui.Component;
import org.zkoss.zk.ui.Executions;
import org.zkoss.zk.ui.select.SelectorComposer;
import org.zkoss.zk.ui.select.annotation.Listen;
import org.zkoss.zk.ui.select.annotation.Wire;
import org.zkoss.zul.A;
import org.zkoss.zul.Hlayout;
import org.zkoss.zul.Image;
import org.zkoss.zul.Label;
import org.zkoss.zul.ListModelList;
import org.zkoss.zul.Listbox;
import org.zkoss.zul.ext.Selectable;

import vnua.fita.bean.Car;
import vnua.fita.model.CarServiceImpl;
import vnua.fita.service.CarService;


public class HomeController extends SelectorComposer<Component> {

    private static final long serialVersionUID = 1L;
    @Wire
    private Label titleLabel;
    @Wire
    private Listbox carListbox;
    @Wire
    private Hlayout carDetail;
    @Wire
    private Label modelLabel;
    @Wire
    private Label makeLabel;
    @Wire
    private Label priceLabel;
    @Wire
    private Label descriptionLabel;
    @Wire
    private Image previewImage;
    @Wire
    private A back;
```

Các thành phần cần quản lý trên trang home.zul

```
	// đối tượng của tầng model dùng để truy vấn database
	private CarService carService = new
CarServiceImpl("jdbc:mysql://localhost:3306/training??useUnicode=true&&characterEncoding=UTF-8",
"root", "123456");

	@Override
	public void doAfterCompose(Component comp) throws Exception {
		// Gọi lại phương thức lớp cha để khởi tạo các thành phần mặc định
		super.doAfterCompose(comp);

		// keyword chỉ loại ô tô, được truyền đến bởi chức năng xem ô tô theo loại
		// trên menu, hoặc chức năng tìm kiếm ô tô ở trang lề phải sidebar
		String keyword = (String) Executions.getCurrent().getParameter("type");

		if(keyword != null) {
			titleLabel.setValue(keyword.toUpperCase() + " CAR LIST:");
		}else { // keyword = null khi được gọi bởi trang chủ (home) trên menu
			titleLabel.setValue("CAR LIST:");
		}

		// Lấy về danh sách ô tô tìm được, nếu keword = null trả về ds tất cả ô tô
		// tìm kiếm theo trường model hoặc make của ô tô có chứa từ khóa keyword
		List<Car> result = carService.search(keyword);

		// set model cho listbox có id = carListbox, model sẽ được duyệt bởi template
		// trên trang giao diện home.zul
		// cần chuyển kiểu List > ListModelList của ZK
		carListbox.setModel(new ListModelList<Car>(result));
	}

	// Xử lý sự kiện select một ô tô trong danh sách (listbox) trên trang home.zul
	@Listen("onSelect = #carListbox")
	public void showDetail(){

		// Lấy về các dòng được chọn trên listbox
		Set<Car> selection = ((Selectable<Car>)carListbox.getModel()).getSelection();

		if (selection!=null && !selection.isEmpty()){
			// Lấy thông tin ô tô đầu tiên trong ds chọn để hiển thị chi tiết
			Car selected = selection.iterator().next();

			// gán giá trị cho các thành phần trong vùng car detail
			previewImage.setSrc(selected.getPreview());
			modelLabel.setValue(selected.getModel());
			makeLabel.setValue(selected.getMake());
			priceLabel.setValue(selected.getPrice().toString());
			descriptionLabel.setValue(selected.getDescription());
		}
		titleLabel.setValue("CAR DETAIL:");
		// ẩn khối danh sách ô tô
		carListbox.setVisible(false);
		// hiện khối thông tin chi tiết về ô tô được chọn
		carDetail.setVisible(true);
	}

	// Xử lý sự kiện click vào đường link “back” tại vùng thông tin chi tiết ô tô
	@Listen("onClick = #back")
	public void backCarList(){
		// Hiển thị khối danh sách ô tô
		carListbox.setVisible(true);
```

```
                // ẩn khối thông tin chi tiết ô tô
                carDetail.setVisible(false);
        }
}
```

4.3 Viết code cho lớp tầng Model

```
package vnua.fita.model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedList;
import java.util.List;

import vnua.fita.bean.Car;
import vnua.fita.service.CarService;


public class CarServiceImpl implements CarService{

        private String jdbcURL;
        private String jdbcUsername;
        private String jdbcPassword;
        private Connection conn;

        // Constructor với các tham số để kết nối
        public CarServiceImpl(String jdbcURL, String jdbcUsername, String jdbcPassword) {
                this.jdbcURL = jdbcURL;
                this.jdbcUsername = jdbcUsername;
                this.jdbcPassword = jdbcPassword;
        }

        // Trả về danh sách ô tô tìm được, nếu keword = null hoặc "" trả về ds tất cả ô tô
        // tìm kiếm theo trường model hoặc make của ô tô có chứa từ khóa keyword
        public List<Car> search(String keyword){
                List<Car> result = new LinkedList<Car>();
                PreparedStatement pst = null;
                try {
                        conn = DBConnection.create(jdbcURL, jdbcUsername, jdbcPassword);
                        String sqlCommand ="Select * from tblcar ";

                        // Mệnh đề Where chỉ xuất hiện trong câu sql khi keyword khác null, khác rỗng
                        if ((keyword!=null && !"".equals(keyword))){
                                sqlCommand += "WHERE "
                                                + "model LIKE ? "
                                                + " OR "
                                                + "make LIKE ?";
                        }

                        pst = conn.prepareStatement(sqlCommand);
                        if ((keyword!=null && !"".equals(keyword))){
                                pst.setString(1, "%" + keyword.toLowerCase() + "%");
                                pst.setString(2, "%" + keyword.toLowerCase() + "%");
                        }

                        ResultSet rs = pst.executeQuery();
```

```
                while (rs.next()) {
                        Integer id = rs.getInt(1);
                        String model = rs.getString(2);
                        String make = rs.getString(3);
                        String preview = rs.getString(4);
                        String description = rs.getString(5);
                        Integer price = rs.getInt(6);
                        Car car = new Car(id, model, make, description,  preview, price);
                        result.add(car);
                }
        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                DBConnection.closeConnect(conn);
                DBConnection.closeStatement(pst);
        }
        return result;
}

// Việc cài đặt dành cho sinh viên và bạn đọc
// Xem bài hướng dẫn ứng dụng đăng ký user
public boolean insertCar(Car  car) {
        return false;
}

// Việc cài đặt dành cho sinh viên và bạn đọc
// Tham khảo: https://mkyong.com/jdbc/jdbc-preparestatement-example-update-a-record/
public boolean updateCar(Car  car) {
        return false;
}

// Việc cài đặt dành cho sinh viên và bạn đọc
// Tham khảo: https://mkyong.com/jdbc/jdbc-preparestatement-example-delete-a-record/
public boolean deleteCar(int carId) {
        return false;
}

// Việc cài đặt dành cho sinh viên và bạn đọc
public Car selectCar(int carId) {
        return null;
}

}
```

5. Xây dựng trang cho các loại ô tô cụ thể

Trên menubar, gọi trang home.zul, truyền tham số loại ô tô để tìm kiếm:

```
<menuitem label="Nissan" href="/home.zul?type=nissan"/>
<menuitem label="Toyota" href="/home.zul?type=toyota"/>
```

6. Xây dựng chức năng tìm kiếm từ trang lề trái

```
<label class="sidebar-title" value="SEARCH" />
<hlayout>
        <textbox width="110px" id="keyword"/>
        <button id="submitBtn" label="Go">
                <attribute name="onClick"><![CDATA[
                        Executions.sendRedirect("/home.zul?type="+keyword.getValue());
                        ]]>
                </attribute>
        </button>
</hlayout>
```

Xử lý sự kiện onClick ngay trên trang sidebar.zul, gọi trang home.zul với tham số type là từ khóa tìm kiếm

7. Xây dựng trang quản lý ô tô (searchMvc.zul)

7.1 Thiết kế layout

Layout trang **searchMvc.zul** gần tương tự như trang home.zul, nó có thêm các thành phần cho phép thực hiện tìm kiếm, thêm, sửa, xóa.

```
<?init class="org.zkoss.zk.ui.util.Composition"
arg0="/template/layout.zul"?>

<vlayout width="100%" apply="vnua.fita.controller.SearchController"  self="@define(content)">
        <div height="50px" style = "margin-top:20px;">CAR  MANAGEMENT:</div>
        <hbox align="center" style = "margin-bottom:  10px;">
                Keyword:
                <textbox id="keywordBox" />
                <button id="searchButton" label="Search" image="/img/search.png" />
                <button id="addButton" label="Add"/>
        </hbox>
        <listbox id="carListbox" rows="3" emptyMessage="No  car found in the result" mold="paging"
pageSize="3">
                <listhead>
                        <listheader label="Model" />
                        <listheader label="Make" />
                        <listheader label="Price" width="20%"/>
                        <listheader label="Function" />
                </listhead>
                <template name="model">
                        <listitem>
                                <listcell label="${each.model}"></listcell>
                                <listcell label="${each.make}"></listcell>
                                <listcell>$<label value="${each.price}" /></listcell>
                                <listcell>
                                        <hlayout>
                                                <button label="Edit"
forward="onClick=carListbox.onEdit"   id = "Edit${each.id}"/>
                                                <button label="Delete"
forward="onClick=carListbox.onDelete"   id = "Delete${each.id}"/>
                                        </hlayout>
                                </listcell>
                        </listitem>
                </template>
        </listbox>
```

Khối Tìm kiếm và nút Add để thêm mới

Rows = 3: cho biết số dòng hiển thị ứng với vị trí thanh cuộn. pageSize cho biết số dòng/trang

Sự kiện click trên nút Edit của mỗi dòng được gắn với carListbox

Gắn id của ô tô tương ứng vào id của nút Edit để tách ra xử lý về sau

Khối thông tin chi tiết ô tô luôn hiển thị

```
<hlayout style="margin-top:20px" width="100%">
        <image id="previewImage" width="250px" />
        <vlayout hflex="1">
                <label id="modelLabel" />
                <label id="makeLabel" />
                <label id="priceLabel" />
                <label id="descriptionLabel" />
        </vlayout>
</hlayout>
</vlayout>
```

7.2 Viết code cho lớp tầng Controller

```
package vnua.fita.controller;


import java.util.HashMap;
import java.util.List;
import java.util.Set;

import org.zkoss.zk.ui.Component;
import org.zkoss.zk.ui.Executions;
import org.zkoss.zk.ui.event.Event;
import org.zkoss.zk.ui.event.EventListener;
import org.zkoss.zk.ui.event.ForwardEvent;
import org.zkoss.zk.ui.select.SelectorComposer;
import org.zkoss.zk.ui.select.annotation.*;
import org.zkoss.zul.*;
import org.zkoss.zul.ext.Selectable;

import vnua.fita.bean.Car;
import vnua.fita.model.CarServiceImpl;
import vnua.fita.service.CarService;


public class SearchController extends SelectorComposer<Component> {

        private static final long serialVersionUID = 1L;

        @Wire
        private Textbox keywordBox;
        @Wire
        private Listbox carListbox;
        @Wire
        private Label modelLabel;
        @Wire
        private Label makeLabel;
        @Wire
        private Label priceLabel;
        @Wire
        private Label descriptionLabel;
        @Wire
        private Image previewImage;


        private CarService carService = new
CarServiceImpl("jdbc:mysql://localhost:3306/training??useUnicode=true&&characterEncoding=UTF-8",
"root", "123456");
```

```
	@Override
	public void doAfterCompose(Component comp) throws Exception {
		super.doAfterCompose(comp);

		// Mặc định load danh sách tất cả ô tô
		List<Car> result = carService.search(null);
		carListbox.setModel(new ListModelList<Car>(result));
	}

	//Xử lý sự kiện click nút Add để thêm mới ô tô > mở ra một hộp thoại
	@Listen("onClick = #addButton")
	public void showAddModal() {
		Window window = (Window)Executions.createComponents(
			"addCar.zul", null, null);
		window.doModal();
	  }

	//Xử lý sự kiện click nút Edit, truyền carId cho hộp thoại mở ra
	@Listen("onEdit=#carListbox")
	public void showUpdateModalDialog(ForwardEvent evt) {
		// Lấy về đối tượng Edit Button từ ForwardEvent
		Button editBtn = (Button)evt.getOrigin().getTarget();

		// Lấy về id của nút Edit
		String editBtnId = editBtn.getId();

		// Tách ra carId từ id của nút Edit
		Integer carId = Integer.valueOf(editBtnId.substring(4));

		// đưa dữ liệu vào HashMap để truyền vào hộp thoại editCar.zul được mở ra
		final HashMap<String, Object> map = new HashMap<String, Object>();
		map.put("carId", carId);
		Window window = (Window)Executions.createComponents(
				"editCar.zul", null, map);
		window.doModal();
	}

	// Xử lý sự kiện click nút Delete, hiện hộp thoại confirm và xử lý
	@Listen("onDelete=#carListbox")
	public void doDelete(ForwardEvent evt) {
		Messagebox.show("Are you sure to delete?", "Delete?", Messagebox.YES |
Messagebox.NO,
			Messagebox.QUESTION, new EventListener<Event>() {
			  @Override
			  public void onEvent(final Event confirmEvt) throws InterruptedException {
			   // Nếu đồng ý xóa
			   if (Messagebox.ON_YES.equals(confirmEvt.getName())) {
				// Tách ra carId từ id của nút Delete
				Button deleteBtn = (Button)evt.getOrigin().getTarget();
				String deleteBtnId = deleteBtn.getId();
				Integer carId = Integer.valueOf(deleteBtnId.substring(6));

				// Dòng code minh họa lấy được id
				Messagebox.show("Car Id: " + carId);

				// Code: gọi phương thức tầng model, xóa ô tô trong database
				// ???

				// Code: Gọi lại phương thức search() bên dưới để cập nhật kết quả
```

```
                                        // ???

                                    } else { // Nếu ko đồng ý xóa
                                        return;
                                    }
                                }
                            }
                        );
                }

                // Xử lý sự kiện bấm nút search
                @Listen("onClick = #searchButton")
                public void search(){
                        String keyword = keywordBox.getValue();
                        List<Car> result = carService.search(keyword);
                        carListbox.setModel(new ListModelList<Car>(result));
                }

                // Xử lý sự kiện click chọn một ô tô trong danh sách, hiển thị thông tin chi tiết
                @Listen("onSelect = #carListbox")
                public void showDetail(){
                        Set<Car> selection = ((Selectable<Car>)carListbox.getModel()).getSelection();
                        if (selection!=null && !selection.isEmpty()){
                                Car selected = selection.iterator().next();
                                previewImage.setSrc(selected.getPreview());
                                modelLabel.setValue(selected.getModel());
                                makeLabel.setValue(selected.getMake());
                                priceLabel.setValue(selected.getPrice().toString());
                                descriptionLabel.setValue(selected.getDescription());
                        }
                }
        }
```

7.3 Viết code cho lớp tầng Model

Code lớp CarServiceImpl.java đã được trình bày ở trên. Các chức năng dùng chung phương thức search.

8. Gợi ý các chức năng thêm, cập nhật ô tô

8.1 Chức năng thêm – khi bấm nút Add

Trang **addCar.zul**:

```
        <window id="modalDialog" closable="true" title="Add New Popup" border="normal"
apply="vnua.fita.controller.AddCarController">
        <!-- Them cac truong thuoc tinh Car -->
        <image id="previewImage" />
        Upload your hot shot:
        <fileupload label="Upload" onUpload="previewImage.setContent(event.media)" />

        <button id="addCarBtn" hflex="1" label="Add Car"/>
        </window>
```

Lớp **AddCarController.java**: Xem thêm code lớp này trong file download gắn kèm, xem bài hướng dẫn ứng dụng đăng ký user để làm tương tự. Tên các thuộc tính của ô tô nên đặt giống trong lớp Car.java, kèm theo tên viết tắt của thành phần UI, ví dụ: modelTb, makeTb, priceTb, descriptionTb. (Tb: Textbox)

```
public class AddCarController extends SelectorComposer<Component> {
    private static final long serialVersionUID = 1L;

    @Wire
    Window modalDialog;
    @Wire
    Image previewImage; // Hình ảnh ô tô
    // Khai báo thêm các thuộc tính của ô tô



    // Xử lý sự kiện khi bấm nút Add trên hộp thoại thêm ô tô
    @Listen("onClick = #addCarBtn")
    public void addCar() {
        // Thêm ảnh vào thư mục img
        saveImage();

        // Gọi phương thức của lớp tầng model để insert các trường dữ liệu vào database

        Messagebox.show("Add new car successfull!");
        // Đóng hộp thoại
        modalDialog.detach();
        // Quay lại trang quản lý ô tô
        Executions.sendRedirect("searchMvc.zul");
        }

    }
```

8.2 Chức năng cập nhật – khi bấm nút Edit

Trang **editCar.zul**:

```
<window id="modalDialog" closable="true" title="Edit Popup" border="normal"
apply="vnua.fita.controller.UpdateCarController">
        <!-- Them cac truong thuoc tinh Car -->
        <vlayout>
            <label id="msg"></label>

            <button id="updateCarBtn" hflex="1" label="Update Car"/>
        </vlayout>

</window>
```

Lớp **UpdateCarController.java**:

```
package vnua.fita.controller;


import org.zkoss.zk.ui.Component;
import org.zkoss.zk.ui.Executions;
import org.zkoss.zk.ui.select.SelectorComposer;
import org.zkoss.zk.ui.select.annotation.Listen;
import org.zkoss.zk.ui.select.annotation.Wire;
```

```
import org.zkoss.zul.Label;
import org.zkoss.zul.Messagebox;
import org.zkoss.zul.Window;

public class UpdateCarController  extends SelectorComposer<Component>  {
   private static final long serialVersionUID = 1L;

   @Wire
   Window modalDialog;
   @Wire
   Label msg; // Thuộc tính có tính chất minh hoạ

   // Khai báo các thuộc tính của ô tô
   // ??


   @Override
   public void doAfterCompose(Component  comp) throws Exception {
         super.doAfterCompose(comp);


         // Lấy carId được truyền tới bởi trang SearchController  qua HashMap
         Integer carId = (Integer)Executions.getCurrent().getArg().get("carId");

         // Hiển thi message lên hộp thoai có tính chất minh
         msg.setValue(carId.toString());

         // Gọi phương thức tầng model load dữ liêu từ database căn cứ vào carId có được
         // set vào các trường thuộc tính để hiển thi lên giao diên trang editCar.zul cho phép sửa
         // ??
   }

   // Xử lý sư kiên khi bấm nút Update trên hộp thoai chỉnh sửa thông tin ô
   @Listen("onClick = #updateCarBtn")
   public void updateCar()  {
         // Gọi phương thức tầng model cập nhật dữ liêu vào database
         // Gọi phương thức getX() trên các thuộc tính để lấy dữ liệu đưa vào đối tượng Car
         // truyền vào phương thức tầng model
         // ??

         Messagebox.show("Update  successfull!");
         // Đóng hộp thoai
         modalDialog.detach();
         // Quay lai trang quản lý ô tô
      Executions.sendRedirect("searchMvc.zul");
   }
}
```

9. Gắn thêm chức năng đăng nhập

Đang cập nhật ...