Ôn tập gk WWW

1. Các cài đặt
2. Setup Intelligi

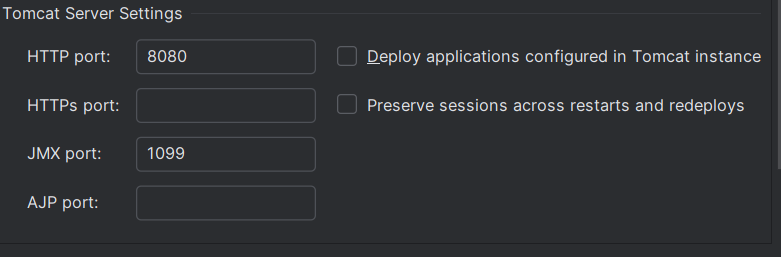
* Tk: [22669281.sang@student.iuh.edu.vn](mailto:22669281.sang@student.iuh.edu.vn)
* Mk: Huusang3704.

1. Plugin

* Database tools and SQL for WebStorm: test connect data, text lệnh sql

1. Add dependences
2. “Build Project” nếu chưa add được tomcat
3. Run 🡪 edit config 🡪 tomcat serve local

* Deployement -> artifact 🡪 explore



1. Kết nối database

<!-- https://mvnrepository.com/artifact/jakarta.servlet.jsp.jstl/jakarta.servlet.jsp.jstl-api -->

<dependency>

<groupId>jakarta.servlet.jsp.jstl</groupId>

<artifactId>jakarta.servlet.jsp.jstl-api</artifactId>

<version>3.0.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.glassfish.web/jakarta.servlet.jsp.jstl -->

<dependency>

<groupId>org.glassfish.web</groupId>

<artifactId>jakarta.servlet.jsp.jstl</artifactId>

<version>3.0.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->

<dependency>

<groupId>org.mariadb.jdbc</groupId>

<artifactId>mariadb-java-client</artifactId>

<version>3.5.5</version>

</dependency>

1. Các bước thực hiện
2. Tạo file content.xml

* Format tên data; user; passwork cho đúng

1. Tạo cây thư mục

* Model
* Dao
* Servlet
* Util
* Webapp

+ META-INF 🡪 context.xml

+ images –> lưu file ảnh

1. Tạo csdl
2. Viết các class theo thứ tự

* Model
* Dao
* Servlet
* Jsp: Chú ý nhớ add namespace phù hợp vào. VD: <c:if> cần namespace taglib

Đề có thể cho

* gửi jsp, nhận, hiển thị

+ Radio:

+ Button

+ Chọn nhiều

+ Select ( danh mục ) chọn 1

+ Hình ảnh

Nội dung viết

1. CRUD
2. Modal: thuộc tính, get, set
3. Until

public class DBUtil {  
 private DataSource dataSource;  
  
 public DBUtil(DataSource dataSource) {  
 this.dataSource = dataSource;  
 }  
  
 public Connection getConnection() {  
 Connection connection;  
 try{  
 connection = dataSource.getConnection();  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 return connection;  
 }  
}

1. Dao

* Khai báo private DBUtil
* Contructor dao(DataSource dataSouce)

private DBUtil dbUtil;  
private DepartmentDAO departmentDAO;   
  
public EmployeeDAO(DataSource dataSource) {  
 dbUtil = new DBUtil(dataSource);  
 departmentDAO = new DepartmentDAO(dataSource);  
}

* 1. CRUD
     1. R ( getAll)

public List<Department> getAllDepartment() {  
 List<Department> departments = new ArrayList<>();  
 String sql = "SELECT \* FROM departments";  
 try (Connection connection = dbUtil.getConnection();  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(sql);  
 ) {  
 while (resultSet.next()) {  
 String id = resultSet.getString("id");  
 String name = resultSet.getString("name");  
  
 Department department = new Department(id, name);  
  
 departments.add(department);  
 }  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 return departments;  
}

* + - 1. R (getById)

public Department getDepartmentByID(String departmentID) {  
 String sql = "SELECT \* FROM departments WHERE id=?";  
 try (Connection connection = dbUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(sql)) {  
 preparedStatement.setString(1, departmentID);  
  
 try (ResultSet resultSet = preparedStatement.executeQuery()) {  
 if (resultSet.next()) {  
 String id = resultSet.getString("ID");  
 String name = resultSet.getString("NAME");  
  
 Department department = new Department(id, name);  
 return department;  
 }  
 }  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 return null;  
}

* + - 1. R (byName ): y chang ở trên khác mỗi câu lệnh sql

public List<Department> searchDepartmentByName(String keyword) {  
 List<Department> departments = new ArrayList<>();  
 String sql = "SELECT \* FROM departments WHERE name LIKE ?";….

preparedStatement.setString(1, "%" + name + "%");

* + 1. C ( save)

public void saveEmployee(Employee employee) {  
 String sql = "INSERT INTO employees(name, department\_id, salary) VALUES (?,?,?)";  
  
 try(Connection connection = dbUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(sql)){  
 preparedStatement.setString(1, employee.getName());  
 preparedStatement.setString(2, employee.getDepartments().getId());  
 preparedStatement.setDouble(3, employee.getSalary());  
  
 preparedStatement.executeUpdate();  
  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
}

* + 1. U (update )

public void updateScore(Score score, String idStu) {  
 //valid điểm  
 if (score.getScore() < 0 || score.getScore() > 10) {  
 throw new IllegalArgumentException("ScorePoint 0 - 10");  
 }  
  
 // valid sinh viên tồn tại  
 Student student = studentDAO.getStudentByID(idStu);  
 if (student == null) {  
 throw new IllegalArgumentException("Student not tồn tại");  
 }  
  
 String sql = "UPDATE scores SET subject\_name = ?, score = ? WHERE student\_id = ? AND score\_id = ?";  
 try (Connection connection = dbUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(sql)) {  
 preparedStatement.setString(1, score.getSubjectName());  
 preparedStatement.setDouble(2, score.getScore());  
 preparedStatement.setString(7, idStu);  
 preparedStatement.setString(8, score.getScoreId());  
  
 preparedStatement.executeUpdate();  
  
  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
}

* + 1. Delete

//Remove  
public void removeDepartment(String departmentId) {  
 String sql = "DELETE FROM departments WHERE id=?";  
 try(Connection connection = dbUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(sql)){  
 preparedStatement.setString(1, departmentId);  
 preparedStatement.executeUpdate();  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
}

1. Servlet

* @WebServlet(“/departments”)
* Public class DepartmentServlet extend HttpServlet{}
* Khai báo DAO
* Khai báo dataSource: @Resource(“jdbc/employeedb”)

@WebServlet("/departments")  
public class DepartmentServlet extends HttpServlet {  
 private EmployeeDAO employeeDAO;  
 private DepartmentDAO departmentDAO;  
  
 @Resource(name = "jdbc/employeedb")  
 private DataSource dataSource;

* 1. Init

@Override  
public void init(ServletConfig config) throws ServletException {  
 try {  
 departmentDAO = new DepartmentDAO(dataSource);  
 employeeDAO = new EmployeeDAO(dataSource);  
 } catch (Exception e){  
 throw new RuntimeException(e);  
 }  
}

* 1. doGet
     1. R (getAll)

@Override  
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String action = req.getParameter("action");  
 if(action==null) action= "list";  
  
 switch (action) {  
 case "list":// getAll  
 List<Department> departments = departmentDAO.getAllDepartment();  
 req.setAttribute("departments", departments);  
 req.getRequestDispatcher("department-list.jsp").forward(req, resp);  
 break;

* + - 1. R(search)

case "search":  
 String keyword = req.getParameter("keyword");  
 List<Department> departmentsList;  
 if(keyword != null && !keyword.trim().isEmpty()) {  
 departmentsList = departmentDAO.searchDepartmentByName(keyword);  
 } else {  
 departmentsList = departmentDAO.getAllDepartment();  
 }  
 req.setAttribute("departments", departmentsList);  
 req.getRequestDispatcher("department-list.jsp").forward(req, resp);  
 break;

* + - 1. R ( lọc )

case "listByDanhMuc":  
 String idDanhMuc = req.getParameter("idDM");  
 List<TinTuc> tinTucList = new ArrayList<>();  
 if(idDanhMuc!=null && !idDanhMuc.isEmpty()) {  
 tinTucList = danhSachTinTucQuanLyDAO.getTinTucByDanhMuc(idDanhMuc);  
 } else {  
 tinTucList = danhSachTinTucQuanLyDAO.getAllTinTuc();  
 }  
 List<DanhMuc> danhMucss = danhMucDAO.getAllDanhMuc();  
 req.setAttribute("danhmucs", danhMucss);;  
 req.setAttribute("tintucs", tinTucList);  
 req.getRequestDispatcher("tintuc-list.jsp").forward(req, resp);  
 break;

* Kết hợp lọc ở jsp

<form method="get" action="${pageContext.request.contextPath}/tintucs">  
 <input type="hidden" name="action" value="listByDanhMuc"/>  
 <label for="idDM">Lọc theo danh mục:</label>  
 <select name="idDM" id="idDM" onchange="this.form.submit()">  
 <option value="">Tất cả</option>  
 <**c:forEach** items="${danhmucs}" var="dm">  
 <option value="${dm.maDM}" <**c:if** test="${param.idDM == dm.maDM}">selected</**c:if**>>${dm.tenDM}</option>  
 </**c:forEach**>  
 </select>  
</form>

* + 1. C ( save thêm mới )

case "new":  
// req.setAttribute("departments", departmentDAO.getAllDepartment()); // load danh sách department để chọn  
// req.getRequestDispatcher("employee-form.jsp").forward(req, resp);  
// break;  
  
 String deptId = req.getParameter("deptId");  
 if(deptId!=null) {  
 Department department = departmentDAO.getDepartmentByID(deptId);  
 req.setAttribute("selectedDept", department); // truyền dept hiện tại sang jsp để hiển thị trên selected  
 req.setAttribute("fixDept", true); // không cho đổi department  
 } else {  
 req.setAttribute("departments", departmentDAO.getAllDepartment());  
 req.setAttribute("fixDept", false);;  
 }  
  
 req.getRequestDispatcher("employee-form.jsp").forward(req, resp);  
 break;

* + 1. Update

case "edit": // U  
 String deptId = req.getParameter("id");  
 Department department = departmentDAO.getDepartmentByID(deptId);  
  
 req.setAttribute("department", department);  
 req.getRequestDispatcher("department-form.jsp").forward(req,resp);  
 break;

* + 1. Delete

case "delete": // D  
 departmentDAO.removeDepartment(req.getParameter("id"));  
 resp.sendRedirect("departments");  
 break;

* 1. doPost

@Override  
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String id = req.getParameter("maTT");  
 String tieuDe = req.getParameter("tieuDe");  
 String noiDung = req.getParameter("noiDung");  
 String lienKet = req.getParameter("lienKet");  
 String maDM = req.getParameter("maDM");  
  
 DanhMuc danhMuc = danhMucDAO.getDanhMucById(maDM);  
  
 TinTuc tinTuc = new TinTuc(id, tieuDe, noiDung, lienKet, danhMuc);  
 if (id != null && !id.trim().isEmpty()) {  
 danhSachTinTucQuanLyDAO.updateTinTuc(tinTuc);  
 } else {  
 danhSachTinTucQuanLyDAO.saveTinTuc(tinTuc);  
 }  
 resp.sendRedirect(req.getContextPath() + "/tintucs");

// nếu muốn gửi tới trang có thêm action, id thì

resp.sendRedirect(req.getContextPath() + "/scores?action=list&studentId=" + idStu);  
  
}

1. jsp
   1. List hiển thị

* Danh sách: dùng table, forEach ( nhớ add namspace )

<table class="table table-primary">  
 <tr>  
 <th>ID</th>  
 <th>Name Employee</th>  
 <th>Salary</th>  
 <th>Dept</th>  
 <th>Action</th>  
 </tr>  
  
 <**c:forEach** var="emp" items="${employees}">  
 <tr>  
 <td>${emp.id}</td>  
 <td>${emp.name}</td>  
 <td>${emp.salary}</td>  
 <td>${emp.departments.name}</td>  
 <td>

// link thường  
 <a href="${pageContext.request.contextPath}/employees?action=edit&id=${emp.id}">Edit</a> |

// link 2 tham số

<a href="${pageContext.request.contextPath}/scores?action=edit&scoreId=${item.scoreId}&studentId=${item.student.studentID}">Sửa điểm</a>

</td>  
 </tr>  
 </**c:forEach**>  
</table>

* 1. Form add or chỉnh sửa – lấy danh mục từ servlet để hiển thị

<form name = "tintucForm" action="${pageContext.request.contextPath}/tintucform" method="post" onsubmit="return validateTinTinForm();">  
 <input type="hidden" name="maTT" value="${tinTuc.maTT}"> <br>  
 Tiêu đề: <input type="text" name="tieuDe" value="${tinTuc.tieuDe}"> <br>  
 Nội dung: <input type="text" name="noiDung" value="${tinTuc.noiDungTT}"><br>  
 liên kết: <input type="text" name="lienKet" value="${tinTuc.lienKet}"><br>  
 Danh mục:  
 <select name="maDM">  
 <**c:forEach** items="${danhMucs}" var="danhMuc">  
 <option value="${danhMuc.maDM}">  
 ${danhMuc.tenDM}  
 </option  
 </**c:forEach**>  
 </select>

<input type="submit" value="save">  
</form>

* + 1. Form có dropdowm select – chọn 1

Ngành:  
 <select name="major" required>  
 <option value="Information Techonology">Information techonology</option>  
 <option value="Software Engineering">Software Engineering</option>  
 <option value="Computer Science">Computer Science</option>  
 <option value="Data Science">Data Science</option>  
 <option value="Artificial Intelligence">Artificial Intelligence</option>  
 </select><br>

* 1. Chức năng thêm ở jsp
     1. Tìm kiếm

<form method="get" action="${pageContext.request.contextPath}/departments">  
 <input type="text" name="keyword" placeholder="Nhập tên phòng ban">  
 <input type="submit" value="Tìm kiếm">  
 <input type="hidden" name="action" value="search">  
</form>

* 1. Valid ở jsp

<script>  
 function validateTinTinForm() {  
  
 var tieuDe = *document*.forms["tintucForm"]["tieuDe"].value.trim();  
  
 var errors = [];  
  
 // kiểm tra bắt buộc nhập hoặc dùng required ở input text ở form  
 if(!tieuDe) errors.push("Tiêu đề là bắt buộc");

// kiểm tra chuôix  
 if(lienKet && !/^http:\/\//.test(lienKet)) {  
 errors.push("Liên kết phỉa bắt đầu bằng http://");  
 }

// kiểm tra số

if(scorePoint) {  
 var scoreVal = parseFloat(scorePoint);  
 if (isNaN(scoreVal) || scoreVal<0 || scoreVal>10) erross.push("Điểm từ 0 -10");  
}

// kiểm tra ngày

if(examDate) {  
 var today = new *Date*();  
 today.setHours(0,0,0,0); // loại bỏ giờ phút giây  
 var exam = new *Date*(examDate);  
 if(exam > today) erross.push("Ngày không được trước ngày hiện tại");  
}

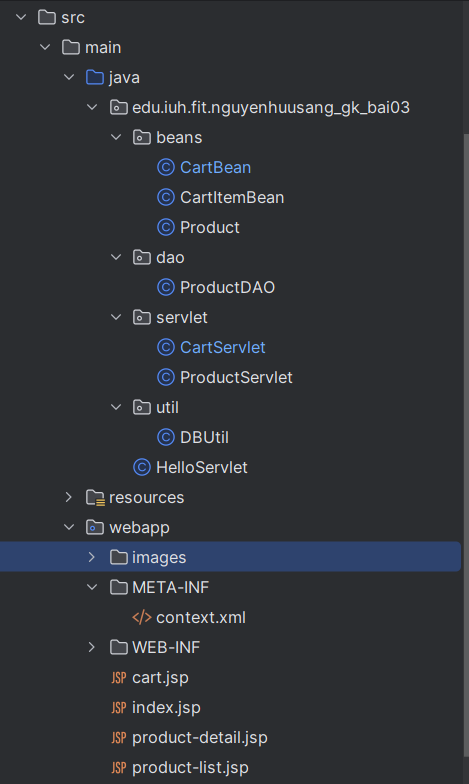
// Nội dung không quà 255 ký tự  
 if(noiDung.length > 255){  
 errors.push("Nội dung không được vượt quá 255 ký tự");  
 }  
  
 if(errors.length > 0) {  
 alert(errors.join("\n"));  
 return false;  
 }  
 return true;  
  
 }  
  
</script>

<form name = "tintucForm" action="${pageContext.request.contextPath}/tintucform" method="post" onsubmit="return validateTinTinForm();">  
 <input type="hidden" name="maTT" value="${tinTuc.maTT}"> <br>

<form>

Giỏ hàng

1. Cấu trúc thư mục





1. Model

* Product
* CartItemBean ( sản phẩm trong giỏ hàng )
* CartBean ( giỏ hàng )
  1. CartitemBean

public class CartItemBean {  
 private Product product;  
 private int quantity;  
  
public CartItemBean() {  
}  
  
public CartItemBean(Product product, int quantity) {  
 this.product = product;  
 this.quantity = quantity;  
}

// Thêm hàm tính tổng tiền của 1 item

public double getSubtotal() {  
 return product.getPrice()\*quantity;  
}

* 1. CartBean

public class CartBean {  
 private List<CartItemBean> items;  
  
 public CartBean() {  
 items = new ArrayList<>();  
 }  
  
 public CartBean(List<CartItemBean> items) {  
 this.items = items;  
 }

// thêm sản phẩm  
public void addProduct(Product p) {  
 for (CartItemBean item : items) {// chạy hết item trong giỏ hàng  
 if (item.getProduct().getId() == p.getId()) { // kiểm tra xem item đó đã tồn tại trong bean chưa  
 item.setQuantity(item.getQuantity() + 1); // nếu tồn tại rồi thì tăng quality + 1  
 return;  
 }  
 }  
 items.add(new CartItemBean(p, 1));// nếu item chưa có trong giỏ thì thêm item đó vào với quality là 1  
}

// xóa sản phẩm  
public void removeProduct(int productID) {  
 items.removeIf(item -> item.getProduct().getId() == productID);  
}

// cập nhật số lượng  
public void updateProduct(int productID, int quality) {  
 for (CartItemBean item : items) {  
 if (item.getProduct().getId() == productID) {  
 if (quality > 0) {  
 item.setQuantity(quality);  
 } else {  
 removeProduct(productID);  
 }  
 return;  
 }  
 }  
}

// tính tổng tiền  
public double getTotal() {  
 double total = 0;  
 for (CartItemBean itemBean : items) {  
 total += itemBean.getSubtotal();  
 }  
 return total;  
}  
  
// xóa hết giỏ hàng  
public void clear() {  
 items.clear();  
}

1. Dao ( như bth CRUD không có gì khác )
2. Servlet
   1. ProductServlet

@Override  
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String idstr = req.getParameter("id");//Lấy tham soo "id" từ URL request  
  
 if(idstr!=null) { // nếu có id thì  
 int id = Integer.*parseInt*(idstr); // chuyển về int để tìm sản phẩm trong DB  
 Product product = productDAO.getProductById(id);// gọi DAO tìm  
 if(product!=null) {// nếu tìm thấy  
 req.setAttribute("product", product);// đưa đối tượng vào resquest để JSP hiển thị  
 RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/product-detail.jsp");// lấy dispacher trỏ đến trang (/..)  
 dispatcher.forward(req, resp);// hướng hướng nội bộ (forward ) đến trang (/...) ( product-detail)  
 } else {// nếu không tìm thấy  
 resp.sendError(HttpServletResponse.*SC\_NOT\_FOUND*, "Product not found");  
 return;  
 }  
 }  
 List<Product> products = productDAO.getAllProducts();// nếu không có id --> tức là người dùng chỉ vào /products  
 req.setAttribute("products", products);  
 RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/product-list.jsp");  
 dispatcher.forward(req, resp);  
}

//DooPosost insert Overide là đủ

@Override  
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 super.doPost(req, resp);  
}

* 1. CartServlet
     1. doGet

@Override  
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 req.getRequestDispatcher("cart.jsp").forward(req, resp);  
}

* + 1. doPost

@Override  
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 HttpSession session = req.getSession();// lấy sess hiện tại của user ( chưa có thì tạo mới) dùng để lưu cart  
 CartBean cart = (CartBean) session.getAttribute("cart"); // lấy cart từ sess nếu user đã có giỏ từ trc  
 if(cart == null) { // nếu ch có, tạo CartBean mới và lưu vào sess để dùng cho các request tiếp theo  
 cart = new CartBean();  
 session.setAttribute("cart", cart);  
 }  
  
 String action = req.getParameter("action");// lấy tham số action từ form/ request để thao tác add, update  
 try {  
 if("add".equals(action)) {  
 int id = Integer.*parseInt*(req.getParameter("id"));// lấy id từ form  
 Product p = productDAO.getProductById(id);// tìm sản phẩm đó từ id trong dao  
 cart.addProduct(p);// add p đó vào cart  
 } else if ("update".equals(action)) {  
 int id = Integer.*parseInt*(req.getParameter("productId"));// lấy id từ form  
 int quality = Integer.*parseInt*(req.getParameter("quantity")); // lấy quantity  
 cart.updateProduct(id, quality); // update  
 } else if ("remove".equals(action)) {  
 int id = Integer.*parseInt*(req.getParameter("productId"));// lấy id  
 cart.removeProduct(id);  
 } else if ("clear".equals(action)) {  
 cart.clear();  
 }  
 } catch (Exception e) {  
 throw new ServletException(e);  
 }  
 resp.sendRedirect("cart");// sau khi xử lý giỏ hàng, chuyển hướng trình duyệt về /cart để hiển thị giỏ hàng mới, đồng thời tránh việc POST bị gửi lại khi refresh;  
}

1. Jsp
   1. Product-list

<**c:forEach** items="${products}" var="p">  
 <div class="s">  
 <b>${p.model}</b><br>  
  
 <img src="images/${p.imgurl}" class=""> <br>  
 Price: ${p.price} <br>  
 <form action="${pageContext.request.contextPath}/cart" method="post">  
 <input type="text" size="2" value="1" name="quantity"> <br>  
 <input type="hidden" name="id" value="${p.id}">  
 <input type="hidden" name="price" value="${p.price}">  
 <input type="hidden" name="model" value="${p.model}">  
 <input type="hidden" name="action" value="add"> <br>  
 <input type="submit" name="addToCart" value="Add To Cart"> <br>  
 </form>  
 <a href="${pageContext.request.contextPath}/product?id=${p.id}">Product Detail</a><br>  
 </div>  
  
</**c:forEach**>

* 1. Product-detail

<**c:if** test="${not empty book}">  
 <ul>  
 <li>Title: ${book.title}</li>  
 <li>Author: ${book.author}</li>  
 <li>Quantity: ${book.quantity}</li>  
 <li>Price: ${book.price}</li>  
 <li>Description: ${book.description}</li>  
 <img src="${pageContext.request.contextPath}/images/${book.imgurl}">  
 </ul>  
  
 </**c:if**>  
  
<p>  
 <a href="${pageContext.request.contextPath}/books">Back to Book List</a>

* 1. Cart.jsp

<h2>Cart</h2>  
<**c:if** test="${empty cart.books}">  
 <p>Cart is empty</p>  
</**c:if**>  
  
<**c:if** test="${not empty cart.books}">  
 <table>  
 <tr>  
 <th>BookID</th>  
 <th>Title book</th>  
 <th>Price</th>  
 <th>Quantity</th>  
 <th>Total</th>  
 <th>Remove</th>  
 </tr>  
  
 <**c:forEach** var="item" items="${cart.books}">  
 <tr>  
 <td>${item.book.id}</td>  
 <td>${item.book.title}</td>  
 <td>${item.book.price}</td>  
 <td>  
 <form action="${pageContext.request.contextPath}/cart" method="post" style="display:inline;">  
 <input type="hidden" name="action" value="update">  
 <input type="hidden" name="id" value="${item.book.id}">  
 <input type="number" name="quantity" value="${item.quantity}" min="1">  
 <input type="submit" value="Update">  
 </form>  
 </td>  
 <td>${item.book.price \* item.quantity}</td>  
 <td>  
 <form action="${pageContext.request.contextPath}/cart" method="post" style="display: inline;">  
 <input type="hidden" name="action" value="remove">  
 <input type="hidden" name="id" value="${item.book.id}">  
 <input type="submit" value="Remove">  
 </form>  
 </td>  
 </tr>  
 </**c:forEach**>  
 </table>  
  
 <p><strong>Total: </strong>${cart.total}</p>  
  
</**c:if**>

Thêm

1. DAO
   1. Save có tự động sinh ra id kiểu string

if(score.getScoreId()==null || score.getScoreId().isEmpty()) {  
 score.setScoreId(java.util.UUID.*randomUUID*().toString());  
}

* 1. Save tự động sinh id theo quy luật

// Sinh student\_id  
if(student.getStudentID() == null || student.getStudentID().isEmpty()) {  
 int year = student.getEnrollment\_year();  
 String prefix = String.*valueOf*(year);  
  
 // lấy số thứ tự lớn nhất của năm đó  
 String sqlMax = "SELECT MAX(student\_id) FROM students WHERE student\_id LIKE ?";  
 String maxID = null;  
 try(Connection connection = dbUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(sqlMax)) {  
 preparedStatement.setString(1, prefix + "%");  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if(resultSet.next()) {  
 maxID = resultSet.getString(1);  
 }  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
  
 int nextNum = 1;  
 if(maxID != null) {  
 // lấy 4 số cuối rồi +1  
 String numStr = maxID.substring(prefix.length());  
 nextNum = Integer.*parseInt*(numStr) +1 ;  
 }  
 String newId = String.*format*("%s%04d", prefix, nextNum);  
 student.setStudentID(newId);  
  
}  
  
// valid năm nhập học  
if(student.getEnrollment\_year() < 2020) {  
 throw new IllegalArgumentException("Enrollment Year >= 2020");  
}  
  
//valid email tồn tại  
String sqlEmailCheck = "SELECT \* FROM students WHERE email = ?";  
try (Connection connection = dbUtil.getConnection();  
PreparedStatement preparedStatement = connection.prepareStatement(sqlEmailCheck)){  
 preparedStatement.setString(1, student.getEmail());  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if(resultSet.next() && resultSet.getInt(1) > 0) { // getInt() để lấy giá trị ở cột đầu tiên của dòng đó  
 throw new IllegalArgumentException("Email đã tồn tại");  
 }  
} catch (SQLException e) {  
 throw new RuntimeException(e);  
}

1. Servlet
2. JSP

* Chọn link giữa 2 điều kiện

<**c:choose**>  
 <**c:when** test="${not empty param.deptId}">  
 <a href="${pageContext.request.contextPath}/employees?action=new&deptId=${param.deptId}">Add Employee</a> <br>  
 </**c:when**>  
 <**c:otherwise**>  
 <a href="${pageContext.request.contextPath}/employees?action=new">Add Employee</a> <br>  
 </**c:otherwise**>  
</**c:choose**>

* Điều kiện không cho thay đổi trong select

<**c:if** test="${fixDept}">  
 <input type="hidden" name="departmentId" value="${selectedDept.id}">  
 Department:  
 <select name="departmentId" disabled>  
 <option value="${selectedDept.id}" selected> ${selectedDept.name}</option>  
 </select>  
</**c:if**>  
  
<**c:if** test="${!fixDept}">  
 Department:  
 <select name="departmentId">  
 <**c:forEach** var="dept" items="${departments}">  
 <option value="${dept.id}"  
 <**c:if** test="${employee != null && dept.id == employee.departments.id}">selected</**c:if**>  
 >${dept.name}</option>  
  
 </**c:forEach**>  
 </select>  
</**c:if**>