M30COM Report

# Introduction

This document is to support and document my journey while creating the electronics product store application.

In it I hope to demonstrate my understanding of Web Development using Java EE.

This has been an interesting experience as I have a back ground with Asp.Net and c# and I like having the opportunity to compare the two similar but different attempts at solving the worlds interconnected problems

# Overview

The task was to create a web site where users could purchase electronics products. The application was required to have a list of products from 5 different categories that customers could add or remove from there shopping cart.

The technologies I was limited to use were and attempted to incorporate in the project were:

* Java servlets
* JavaBeans
* JFS Pages (Java Faces)
* HTML/CSS
* Jarvascript

# Design Process

## User

OverView

You must submit a report on the system describing the features that you have incorporated and reflection on development process. The report should include:

~~• Use case scenarios and their UML representation of the application~~

• Application design (Design patterns, ER diagram, wireframes and sketches etc.). A brief evaluation report not more than 1 page reflecting development process and technologies used

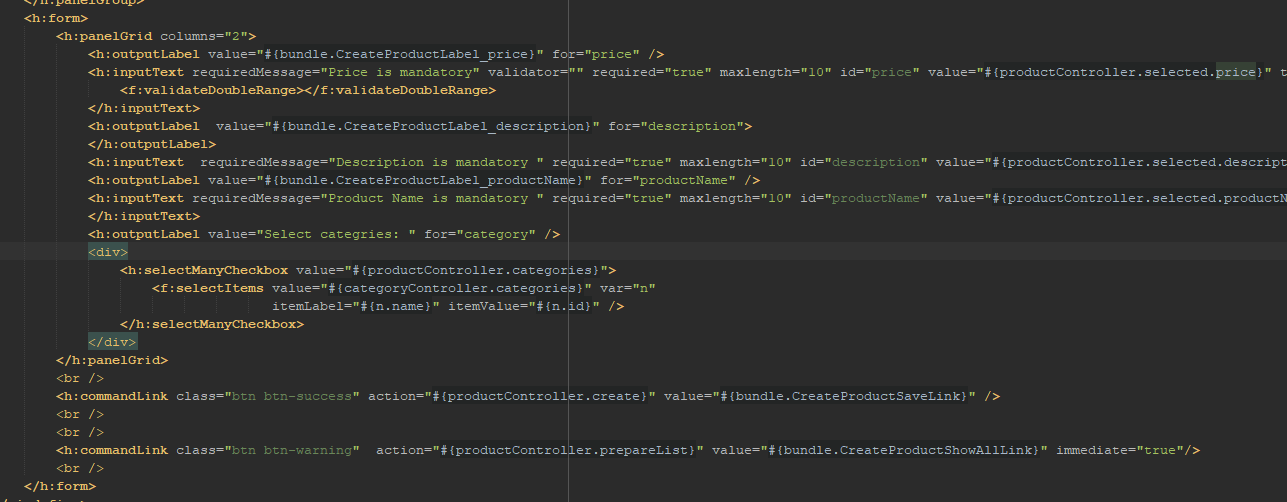
• Correct use of appropriate technologies (JSP, Servlet, JavaServer Faces, Java Beans, JDBC etc.)

• Evidence of implementation (e.g. screen shots, code, database, test plan and test results) of the application

# Validation

## Client side

I have used html attributes for the client side attributes.

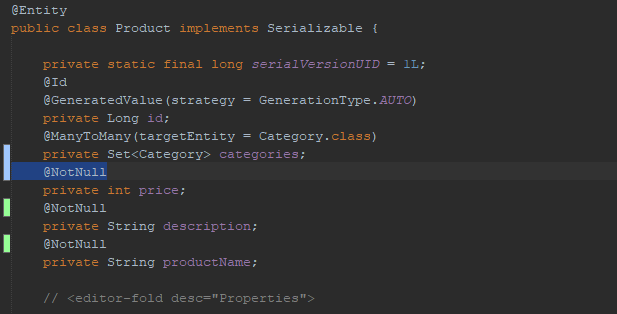


Here you can see for both the name and description inputs I have used the required attribute to make sure that when you create a category you must give it name and a description.

I have also used the max-length attribute to apply some further constraints to ensure for sensible inputs.

For the price I apply the validateDoubleRange validatior from the core jsf namespace.

# Server side validation



I have opted for using the java beans annotations for the majority of the server side validation.

The @NotNUll attribute will instruct the SQL DBMS to create the column for that property with the constraint NOTNULL.

# Source Control

# Notes

# Java Platform, Enterprise Edition (Java EE)

Java Platform, Enterprise Edition (Java EE) is the standard in community-driven enterprise software.

Java EE is developed using the [Java Community Process](http://www.jcp.org/), with contributions from industry experts, commercial and open source organizations, Java User Groups, and countless individuals. Each release integrates new features that align with industry needs, improves application portability, and increases developer productivity.

# Jave persistence API (JPA)

The **Java Persistence API** (**JPA**) is one possible approach to ORM (Object Relational Mapping). Via **JPA** the developer can map, store, update and retrieve data from relational databases to **Java** objects and vice versa. **JPA** can be used in **Java**-EE and **Java**-SE applications. **JPA** is a specification and several implementations are available.

# JDBC Resource

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API.

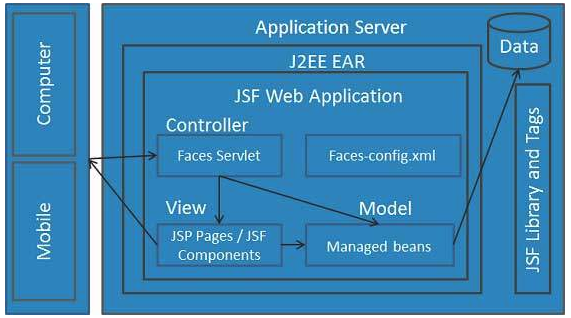
A JDBC resource (data source) provides applications with a means of connecting to a database. Typically, the administrator creates a JDBC resource for each database accessed by the applications deployed in a domain. (However, more than one JDBC resource can be created for a database.)

# JavaServer Faces

**JavaServer Faces** (JSF) is a MVC web framework that simplifies the construction of User Interfaces (UI) for server-based applications using reusable UI components in a page.

JSF provides a facility to connect UI widgets with data sources and to server-side event handlers.

The JSF specification defines a set of standard UI components and provides an Application Programming Interface (API) for developing components. JSF enables the reuse and extension of the existing standard UI components.



**Components:**

## Class FacesServlet

**FacesServlet** is a servlet that manages the request processing lifecycle for web applications that are utilizing JavaServer Faces to construct the user interface.

# Java Bean

Managed beans

L

A Java Bean is a java class that should follow following conventions:

* It should have a no-arg constructor.
* It should be Serializable.
* It should provide methods to set and get the values of the properties, known as getter and setter methods.

//Employee.java

Example Java Bean

**package** mypack;

**public** **class** Employee **implements** java.io.Serializable{

**private** **int** id;

**private** String name;

**public** Employee(){}

**public** **void** setId(**int** id){**this**.id=id;}

**public** **int** getId(){**return** id;}

**public** **void** setName(String name){**this**.name=name;}

**public** String getName(){**return** name;}

}

# Steps/Process:

Creating EntityModelsPackage

Created ModelPackage

Creating Entity Class with option “ create Persistence Unit” selected

PersistenceUnit Name is ElectronicProductStorePU with EclipseLink (JPA 2.1)(default) as Persistence Provider

Created 4 entities Customer, Order,Product,Purchase. Added there cardinatilites, onetomany many to one on the properies

Tutorial: <https://www.tutorialspoint.com/jpa/jpa_entity_relationships.htm>

* @ManyToOne Relation
  + Child:
    - @ManyToOne
    - private Department department;
  + Parent: nothing
* @OneToMany Relation
  + Child: Private Object objectname;
    - @OneToMany( targetEntity=Employee.class )
    - private List employeelist;
  + Parent: nothing
* @OneToOne Relation
* @ManyToMany Relation

Had Issue with Creating Order Entity Due to ORDER being a keyword in sql, the same with desc

Creat JSF pages from Entity classes

Create JDBC Resource for JPA Api to Connect to the database.

Setting:

JNDI name: org.apache.derby.jdbc.ClientDataSource

Creat a new JDBC Connectio Pool

DataSource class anemorg.apache.derby.jdbc.ClientDataSource

Database connection is extracted for excisting data base created earlier (Electronic product store)

URL: jdbc:derby://localhost:1527/ElectronicProduct

serverName: localhost

portNumber: 1527

DatabaseName:L ElectronicProduct

User: chris

Password: 1234

Configuerd User

Name:admin

Password:1234

Groups: CanEditProductList,CanViewOrders

I managed to get the authentication working, main issue was either not restarting the Glass server or not clean building