Project Title :Shopaholics(Angular Store)

Abstract :The project shopaholics is gives a clear idea of how a shopping cart is maintained in an e-commerce website.How a user reaches his destination i.e,from the welcome screen of the application to the checkout,payment and confirmation of his purchase.This project includes some javascript screens which makes a flow from home page,product page,cart,wishlist,checkout,end screen.

The javascript screens are collaborated by the Spring MVC architecture.Here we use angular JS for the User Interface i.e,view part for the rich look of the application.The MVc architecture helps a lot in the flow of the application the webconatiner takes care of the MVC architecture.

Introduction: Any shopping application needs to serve all the requirements of the user in a particular context.

The Shopaholics application allows the user to view its homepag where he can view the categories of things he need to purchase the offers etc.When the user selects a product he is redirected to the product page where he can find the description of th eparticular product ,its price,offers,review,ratings etc.The user now have two options either to select the product,go back.If the user selects the product he can find two ore options either to go to the cart and checkout or add the product to the wishlist so that he can checkout the product later.

Once the user select the product and goes to cart,there he can find the quantity change option ,the price variations based on quantity etc.When the user checksout the producct from the cart,the page is redirected to the payment page where he has different options like Cash on delivery,credit/debitetc.Once the user is done with the payment he has his confirmation and then he can go either back to continue shopping or end the application.

Detailed System Description : The whole system has three tasks.

1.Creating a flow between all the view pages

2.Populating the data in the database which I have maintained for the appplication to the view javascript screens.

3.Checkout and payment.

The project comes to nice finishing if we maintain this 3 tasks properly.So the below mentioned classes gives a complete description of how they used in the project.

Store and Product class:This is the homepage of the application.Store class is javascript file.It has a getproduct() method to retrieve the products from the database.To set the current product when Url provides a productsku,the store controller uses this method.

The Product class has three key variables which are used by the cart,sku(unique id),name and price.All the other variables are used elsewhere in th e application but not in the cart.

ShoppingCart:
The shoppingCart class is a js file and implements different methods like addItems(),clearItems(),gettotalCount(),gettotalPrice(),checkout(). After all this java classes we have a default.html class which provides the master view to the application .This class contains all the angular js attributes like ng-app,ng-view etc .

Requirements: Spring MVC
JAVA
JavaScript
HTML
Angular JS
DBMS
SQL

Literature Survey: When I started developing the shopping cart, I had the following requirements in mind:

Must be 100% pure JavaScript (so it is easy to integrate into any site)
Must follow the MVVM architecture (so it is easy to customize its look and feel)
Must be safe (we don't want to be responsible for storing people's credit card numbers etc.)
Must be fast and reliable (we don't want users to give up before they checkout!)
Must be flexible (it should allow payments to be processed using different services)
Must be extensible (adding new payment methods should be easy)
Must be easy to use (because there's no reason for it to be complicated)

I believe the "shoppingCart" class described above addresses all these requirements. It uses jQuery and integrates well with AngularJS applications. The "shoppingCart" class contains all the logic and provides the object model needed to create flexible and attractive views.

Shopping Cart Sample :

To know how the application works we need to have a look at its views.There are three main views for this application:

Store :This is the main view of the application in which it has the view list of all the products of the online angular store.This is how the store looks like

Product Details:

This page shows the entire view of a product which is selected by the customer.It provides the entire information about the product.It gives a quick summary to the customer and in this page the customer can see the availability of the product and he can add it to the cart immediately.

Cart :This is the final view of the application where the user can see the checkout process and the items which are added into the cart and the total value of the cart.
The application starts with the definition of Angular JS which is in the app.js

```
// App Module: the name AngularStore matches the ng-app attribute
// in the main &lt;html> tag. The route provides parses the URL and
// injects the appropriate partial page
var storeApp = angular.module('AngularStore', []).
 config(['$routeProvider', function($routeProvider) {
 $routeProvider.
  when('/store', {
   templateUrl: 'partials/store.htm',
   controller: storeController }).
  when('/products/:productSku', {
   templateUrl: 'partials/product.htm',
   controller: storeController }).
  when('/cart', {
   templateUrl: 'partials/shoppingCart.htm',
   controller: storeController }).
  otherwise({
   redirectTo: '/store' });
}]);
```

It contains a routeProvider that specifies which view should be displayed based on the URL.

Then comes the definition of "Data Service" which provides the data shared throughout the application.

```
// create a data service that provides a store and a shopping
// cart that will be shared by all views
// (instead of creating fresh ones for each view).
storeApp.factory("DataService", function() {
 var myStore = new store();
 var myCart = new shoppingCart("AngularStore");
 myCart.addCheckoutParameters("PayPal", "your PayPal merchant
account id");
 myCart.addCheckoutParameters("Google", "your Google merchant
account id ", {
  ship_method_name_1: "UPS Next Day Air",
  ship_method_price_1: "20.00",
  ship_method_currency_1: "USD",
  ship_method_name_2: "UPS Ground",
  ship_method_price_2: "15.00",
  ship_method_currency_2: "USD"
 });
```

```
  return {
    store: myStore,
    cart: myCart
  };
});
```

The service creates an object called store which contains the list of products available.When the “shoppingCart” object is created, it automatically loads its contents from local storage, so users can add items to the cart, close the application, and continue shopping later on.

In the controller.js file the data service which has been created is used by the "storeController" objects that will drive all the viewsof the application.

```
function storeController($scope, $routeParams, DataService) {

  // get store and cart from service
  $scope.store = DataService.store;
  $scope.cart = DataService.cart;

  // use routing to pick the selected product
  if ($routeParams.productSku != null) {
    $scope.product =
$scope.store.getProduct($routeParams.productSku);
  }
}
```

Store class:

The store classis defined in the store.js file.It has a getProduct method which retrieves the product by SKU.This method is used by the “storeController” to set the current product when the URL routing specifies a productSku

```
// store (contains the products)
function store() {
  this.products = [
    new product("APL", "Apple", "Eat one every...", 12, 90, 0, 2, 0, 1, 2),
    new product("AVC", "Avocado", "Guacamole...", 16, 90, 0, 1, 1, 1,
2),
    new product("BAN", "Banana", "These are...", 4, 120, 0, 2, 1, 2, 2),
    // more products...
    new product("WML", "Watermelon", "Nothing...", 4, 90, 4, 4, 0, 1,
1)
  ];
  this.dvaCaption = ["Negligible", "Low", "Average", "Good", "Great" ];
  this.dvaRange = ["below 5%", "between 5 and 10%",... "above 40%"];
}
store.prototype.getProduct = function (sku) {
```

```
 for (var i = 0; i &lt; this.products.length; i++) {
  if (this.products[i].sku == sku)
   return this.products[i];
 }
 return null;
}
```

Product class :The product class is defined in the product.js

```
// product class
function product(sku, name, description, price,
           cal, carot, vitc, folate, potassium, fiber) {
 this.sku = sku; // product code (SKU = stock keeping unit)
 this.name = name;
 this.description = description;
 this.price = price;
 this.cal = cal;
 this.nutrients = {
  "Carotenoid": carot,
  "Vitamin C": vitc,
  "Folates": folate,
  "Potassium": potassium,
  "Fiber": fiber
 };
}
```

The product class has three properties that will be used by the shopping cart: sku (unique ID), name, and price.

AngularJS views:

The default.htm file contains the master view. It is implemented as follows:

```
&lt;!doctype html>
&lt;html ng-app="AngularStore">
 &lt;head>
  &lt;!-- includes for jQuery, Angular, and Bootstrap -->
  &lt;!-- ... -->
  &lt;!-- includes for the Angular Store app -->
  &lt;script src="js/product.js" type="text/javascript">&lt;/script>
  &lt;script src="js/store.js" type="text/javascript">&lt;/script>
  &lt;script src="js/shoppingCart.js"
type="text/javascript">&lt;/script>
  &lt;script src="js/app.js" type="text/javascript">&lt;/script>
  &lt;script src="js/controller.js" type="text/javascript">&lt;/script>
  &lt;link href="css/style.css" rel="stylesheet" type="text/css"/>
 &lt;/head>
 &lt;body>
  &lt;div class="container-fluid">
   &lt;div class="row-fluid">
    &lt;div class="span10 offset1">
```

```
            &lt;h1 class="well" >
               &lt;a href="default.htm">
                  &lt;img src="img/logo.png" height="60" width="60"
alt="logo"/>
               &lt;/a>
               Angular Store
            &lt;/h1>
            &lt;div ng-view>&lt;/div>
         &lt;/div>
      &lt;/div>
   &lt;/div>
  &lt;/body>
&lt;/html>
```

The definitions of the attributes of angular js used in the application:
The "**ng-app**" attribute associates the page with the AngularStore module defined in the app.js file. This attribute takes care of the URL routing, view injection, and providing each view with the appropriate controllers.
The "**ng-view**" div marks the place where AngularJS will inject the partial pages that correspond to the routed views. Recall that our application has three partial pages: store.htm, product.htm, and shoppingCart.htm.
The parts of the page around the "ng-view" div remain in place as you switch views, acting as a master page. In this sample, this area shows the app logo and a title.

User Manual:

This system is very easy to use.It helps the user to go to every other page by jus one click.If the user clicks on a product in the homepage he goes to the product page.If he clicks cart or wishlist he goes to the related pages.If he clicks back he goes back to the homepage.When the user clicks checkout in the cart page he goes to the payment page and after that exits the application

Conclusion:

The Shopaholics application fulfill the requirments of a shopping application.It is 100% javascript and doesnot need any servers to run it.It is very easy to go through it for the flow and use.I have achieved the rquirements of the project by the above mentioned technologies and styling scripts.

References:

AngularJS by Google. The AngularJS home page, with links to samples and documentation.
A Look Into AngularJS – The "Super-heroic JavaScript MVW Framework". A nice, brief summary of the AngularJS framework.
Building Huuuuuge Apps with AngularJS. The best documentation I found on how to structure large (or not so large) AngularJS

applications.
Egghead.io. John Lindquist's series of how-to videos on AngularJS.
nopCommerce - An open-source shopping cart framework. A CodeProject article describing a full (client/server) shopping cart framework.
Use of the PayPal payment system in ASP.NET. A CodeProject article about the PayPal payment system, including a lot of interesting and useful details.