

trigger CheckboxSelectionTrigger on Product\_Item\_\_c (before insert, before update) {

if(Trigger.isBefore) {

for(Product\_Item\_\_c product : Trigger.New) {

if(product.Amount\_\_c > 0) {

product.InStock\_\_c = true;

} else {

product.Amount\_\_c = 0;

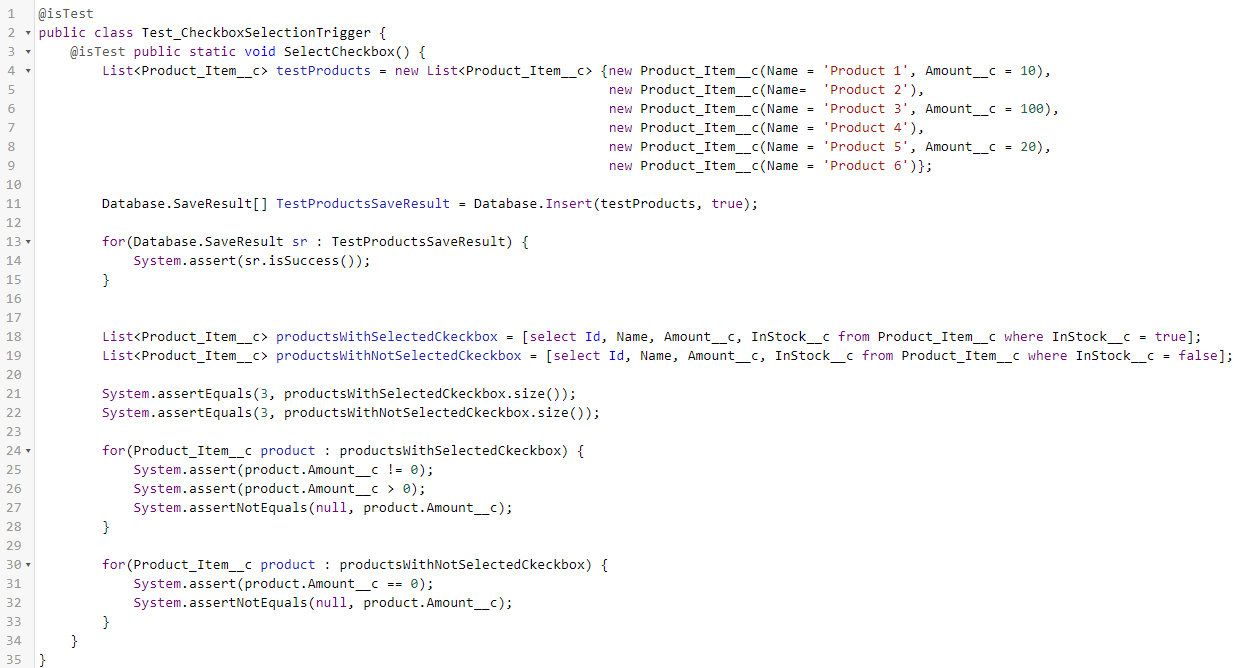
}

}

}

}





@isTest

public class Test\_CheckboxSelectionTrigger {

@isTest public static void SelectCheckbox() {

List<Product\_Item\_\_c> testProducts = new List<Product\_Item\_\_c> {new Product\_Item\_\_c(Name = 'Product 1', Amount\_\_c = 10),

new Product\_Item\_\_c(Name= 'Product 2'),

new Product\_Item\_\_c(Name = 'Product 3', Amount\_\_c = 100),

new Product\_Item\_\_c(Name = 'Product 4'),

new Product\_Item\_\_c(Name = 'Product 5', Amount\_\_c = 20),

new Product\_Item\_\_c(Name = 'Product 6')};

Database.SaveResult[] TestProductsSaveResult = Database.Insert(testProducts, true);

for(Database.SaveResult sr : TestProductsSaveResult) {

System.assert(sr.isSuccess());

}

List<Product\_Item\_\_c> productsWithSelectedCkeckbox = [select Id, Name, Amount\_\_c, InStock\_\_c from Product\_Item\_\_c where InStock\_\_c = true];

List<Product\_Item\_\_c> productsWithNotSelectedCkeckbox = [select Id, Name, Amount\_\_c, InStock\_\_c from Product\_Item\_\_c where InStock\_\_c = false];

System.assertEquals(3, productsWithSelectedCkeckbox.size());

System.assertEquals(3, productsWithNotSelectedCkeckbox.size());

for(Product\_Item\_\_c product : productsWithSelectedCkeckbox) {

System.assert(product.Amount\_\_c != 0);

System.assert(product.Amount\_\_c > 0);

System.assertNotEquals(null, product.Amount\_\_c);

}

for(Product\_Item\_\_c product : productsWithNotSelectedCkeckbox) {

System.assert(product.Amount\_\_c == 0);

System.assertNotEquals(null, product.Amount\_\_c);

}

}

}

Change pagination 🡪 in footer









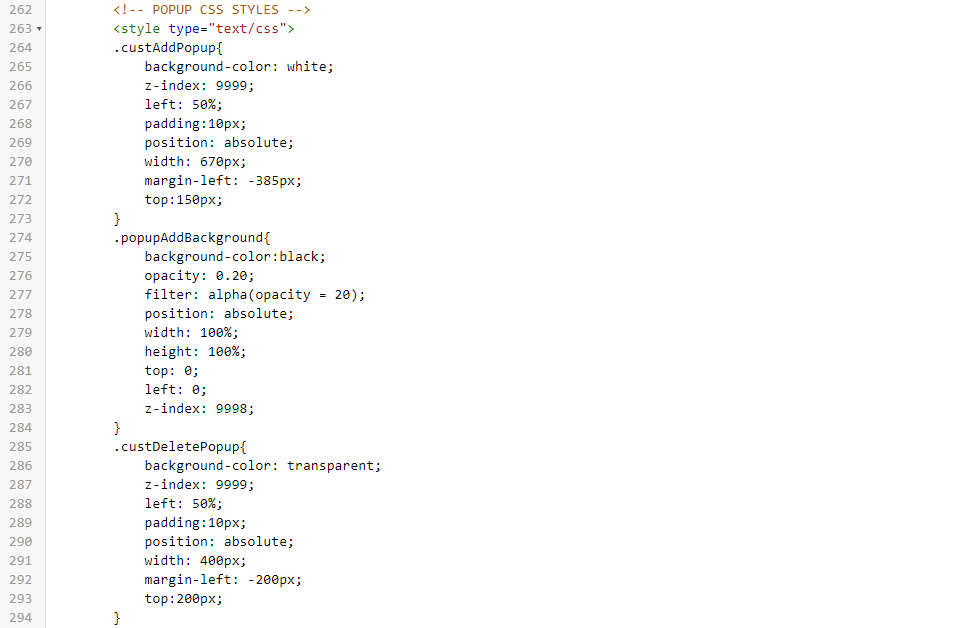


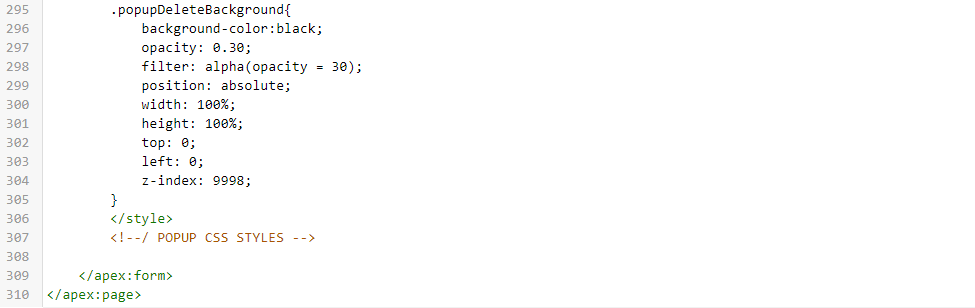












<apex:page lightningStylesheets="true" standardController="Product\_Item\_\_c" extensions="Controller\_ProductListVisualforcePage" docType="html-5.0">

<apex:slds />

<apex:pageMessages />

<apex:form >

<apex:pageBlock title="Meizu Global Store" >

<apex:pageBlockButtons location="top" >

<apex:input type="date" value="{! searchAddedDate }" style="width: 186px; height: 32px; margin-right:5px;" />

<apex:inputText value="{! searchTitle }" style="width: 186px; height: 32.2px; margin-right:30px; margin-bottom:3.75px;" />

<button class="slds-button slds-button\_neutral" style="margin-right:300px; margin-bottom:3.5px;">

<apex:commandLink value="Search" action="{! searchProduct }" reRender="Products\_List"/>

</button>

<button class="slds-button slds-button\_brand" style="font-weight:bold;">

<apex:commandLink action="{! showAddPopup}" value="New Product" reRender="AddNewProductPopup"/>

</button>

</apex:pageBlockButtons>

<apex:pageBlockTable value="{! product }" var="pt" id="Products\_List">

<!-- Name -->

<apex:column width="20%">

<apex:facet name="header">

<apex:commandLink action="{! sortByName}" reRender="Products\_List">

<apex:outputText value="Name"/>

</apex:commandLink>

</apex:facet>

<apex:outputText value="{! pt.Name }"/>

</apex:column>

<!-- Type -->

<apex:column width="15%">

<apex:facet name="header">

<apex:commandLink action="{! sortByType }" reRender="Products\_List">

<apex:outputText value="Type"/>

</apex:commandLink>

</apex:facet>

<apex:outputText value="{! pt.Type\_\_c }"/>

</apex:column>

<!-- Amount -->

<apex:column width="15%">

<apex:facet name="header">

<apex:commandLink action="{! sortByAmount}" reRender="Products\_List">

<apex:outputText value="Amount"/>

</apex:commandLink>

</apex:facet>

<apex:outputText value="{! pt.Amount\_\_c }" style="margin-left: 15px;" />

</apex:column>

<!-- Price -->

<apex:column width="15%">

<apex:facet name="header">

<apex:commandLink action="{! sortByPrice}" reRender="Products\_List">

<apex:outputText value="Price"/>

</apex:commandLink>

</apex:facet>

<apex:outputText value="{! pt.Price\_\_c }"/>&nbsp;$

</apex:column>

<!-- Added Date -->

<apex:column width="15%">

<apex:facet name="header">

<apex:commandLink action="{! sortByAddedDate}" reRender="Products\_List">

<apex:outputText value="Added Date"/>

</apex:commandLink>

</apex:facet>

<apex:outputText value="{0,date,dd/MM/yy}">

<apex:param value="{! pt.Added\_Date\_\_c}"/>

</apex:outputText>

</apex:column>

<!-- Release Date -->

<apex:column width="15%">

<apex:facet name="header">

<apex:commandLink action="{! sortByReleaseDate}" reRender="Products\_List">

<apex:outputText value="Release Date"/>

</apex:commandLink>

</apex:facet>

<apex:outputText value="{0,date,dd/MM/yy}">

<apex:param value="{! pt.Release\_Date\_\_c}"/>

</apex:outputText>

</apex:column>

<!-- Delete button -->

<apex:column width="5%">

<button class="slds-button slds-button\_destructive slds-m-right\_x-large">

<!--<apex:commandLink action="{! deleteProduct }" value="Delete" onclick="return confirm('Are you sure?');">

<apex:param assignTo="{! productId }" value="{! pt.Id }" name="pruductId"/>

</apex:commandLink>-->

<apex:commandLink action="{! showDeletePopup}" value="Delete" reRender="DeleteProductPopup" style="margin-right:2px; font-weight:bold;">

<apex:param assignTo="{! productId }" value="{! pt.Id }" name="pruductId"/>

</apex:commandLink>

</button>

</apex:column>

</apex:pageBlockTable>

</apex:pageBlock>

<!-- Pagination -->

<div style="height: 30px;" class="slds-align\_absolute-center">

<span class="slds-badge">

<!-- First page -->

<apex:commandLink action="{! First }" value="First"/>

</span>

<span class="slds-badge">

<!-- Previous page -->

<!-- active -->

<apex:commandLink action="{! Previous }" value="<< Previous" rendered="{! hasPrevious }" />

<!-- inactive (no earlier pages) -->

<apex:outputText style="color: #ccc;" value="<< Previous" rendered="{! NOT(hasPrevious) }" />

</span>

<span class="slds-badge">

<!-- Next page -->

<!-- active -->

<apex:commandLink action="{! Next }" value="Next >>" rendered="{! hasNext }" />

<!-- inactive (no more pages) -->

<apex:outputText style="color: #ccc;" value="Next >>" rendered="{! NOT(hasNext) }" />

</span>

<span class="slds-badge">

<!-- Last page -->

<apex:commandLink action="{! Last }" value="Last"/>

</span>

</div>

<!--/ Pagination -->

<!-- ADD NEW PRODUCT POPUP -->

<apex:outputPanel id="AddNewProductPopup">

<apex:outputPanel layout="block" rendered="{! displayAddPopup }" styleClass="popupAddBackground"/>

<apex:outputPanel layout="block" rendered="{! displayAddPopup }" styleClass="custAddPopup">

<fieldset class="slds-box slds-theme--default slds-container--medium" style="width: 650px;">

<legend class="slds-text-heading--small slds-p-vertical--medium" style="font-weight:bold">Add New Product</legend>

<!-- FIELD SECTION -->

<div class="demo-only demo-only--sizing slds-grid" style="width: 600px;" >

<!-- Left Half -->

<div class="slds-size\_1-of-2 ">

<div class="slds-text-align\_center ">

<div class="slds-grid slds-grid\_vertical ">

<div class="slds-grid">

<div class="slds-size\_1-of-2" style="width: 110px;">

<div class="slds-text-align\_right slds-m-right\_x-small slds-m-top\_xx-small">

<apex:outputText value="Product Name"/><abbr class="slds-required">\*</abbr>

</div>

</div>

<div class="slds-size\_1-of-2" style="width: 190px;">

<div class="slds-text-align\_left ">

<apex:inputField value="{! newProduct.Name }" onkeydown="if(event.keyCode==13){this.blur();actionFunction();}"/>

</div>

</div>

</div>

<div class="slds-grid slds-m-top\_small">

<div class="slds-size\_1-of-2" style="width: 110px;">

<div class="slds-text-align\_right slds-m-right\_x-small slds-m-top\_xx-small">

<apex:outputText value="Release Date"/>

</div>

</div>

<div class="slds-size\_1-of-2" style="width: 190px; height: 32px;">

<div class="slds-text-align\_left">

<apex:input type="date" value="{! release\_date }" onkeydown="if(event.keyCode==13){this.blur();actionFunction();}" id="relDate"/>

</div>

</div>

</div>

<div class="slds-grid slds-m-top\_small">

<div class="slds-size\_1-of-2" style="width: 110px;">

<div class="slds-text-align\_right slds-m-right\_x-small slds-m-top\_xx-small">

<apex:outputText value="Added Date"/>

</div>

</div>

<div class="slds-size\_1-of-2" style="width: 190px; height: 32px;">

<div class="slds-text-align\_left">

<apex:input type="date" value="{! added\_date }" onkeydown="if(event.keyCode==13){this.blur();actionFunction();}" id="addDate"/>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Right Half -->

<div class="slds-size\_1-of-2 ">

<div class="slds-text-align\_center ">

<div class="slds-grid slds-grid\_vertical ">

<div class="slds-grid">

<div class="slds-size\_1-of-2" style="width: 110px;">

<div class="slds-text-align\_right slds-m-right\_small slds-m-top\_xx-small">

<apex:outputText value="Amount"/>

</div>

</div>

<div class="slds-size\_1-of-2">

<div class="slds-text-align\_left" style="width: 190px;">

<apex:inputField value="{! newProduct.Amount\_\_c }" onkeydown="if(event.keyCode==13){this.blur();actionFunction();}"/>

</div>

</div>

</div>

<div class="slds-grid slds-m-top\_small">

<div class="slds-size\_1-of-2" style="width: 110px;">

<div class="slds-text-align\_right slds-m-right\_small slds-m-top\_xx-small">

<apex:outputText value="Price"/>

</div>

</div>

<div class="slds-size\_1-of-2">

<div class="slds-text-align\_left" style="width: 190px;">

<apex:inputField value="{! newProduct.Price\_\_c }" onkeydown="if(event.keyCode==13){this.blur();actionFunction();}"/>

</div>

</div>

</div>

<div class="slds-grid slds-m-top\_small">

<div class="slds-size\_1-of-2" style="width: 110px;">

<div class="slds-text-align\_right slds-m-right\_small slds-m-top\_xx-small">

<apex:outputText value="Type"/>

</div>

</div>

<div class="slds-size\_1-of-2" style="width: 186px;">

<div class="slds-text-align\_left slds-m-left\_xxx-small slds-m-top\_xx-small">

<apex:inputField value="{! newProduct.Type\_\_c }" onkeydown="if(event.keyCode==13){this.blur();actionFunction();}" style="width: 186px; height: 26px;"/>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!--/ FIELD SECTION -->

<div class="slds-align\_absolute-center slds-m-top\_xx-large">

<button class="slds-button slds-button\_neutral"><apex:commandLink action="{! closeAddPopup}" value="Cancel" reRender="AddNewProductPopup"/></button>

<button class="slds-button slds-button\_brand"><apex:commandLink action="{! addNewProduct }" value="Save" /></button>

</div>

</fieldset>

</apex:outputPanel>

</apex:outputPanel>

<!--/ ADD NEW PRODUCT POPUP -->

<!-- DELETE PRODUCT POPUP -->

<apex:outputPanel id="DeleteProductPopup">

<apex:outputPanel layout="block" rendered="{! displayDeletePopup }" styleClass="popupDeleteBackground"/>

<apex:outputPanel layout="block" rendered="{! displayDeletePopup }" styleClass="custDeletePopup">

<fieldset class="slds-box slds-theme--default slds-container--small">

<div class="slds-align\_absolute-center slds-text-heading\_medium" style="font-weight:bold"><h1> Are you sure ? </h1></div>

<div class="slds-align\_absolute-center slds-m-top\_large">

<button class="slds-button slds-button\_destructive">

<apex:commandLink action="{! closeDeletePopup}" value="No" reRender="DeleteProductPopup"/>

</button>

<button class="slds-button slds-button\_success" style="text-color:white" >

<apex:commandLink action="{! deleteProduct }" value="Yes">

<apex:param value="{! productId }" name="pruductId"/>

</apex:commandLink>

</button>

</div>

</fieldset>

</apex:outputPanel>

</apex:outputPanel>

<!--/ DELETE PRODUCT POPUP -->

<!-- POPUP CSS STYLES -->

<style type="text/css">

.custAddPopup{

background-color: white;

z-index: 9999;

left: 50%;

padding:10px;

position: absolute;

width: 670px;

margin-left: -385px;

top:150px;

}

.popupAddBackground{

background-color:black;

opacity: 0.20;

filter: alpha(opacity = 20);

position: absolute;

width: 100%;

height: 100%;

top: 0;

left: 0;

z-index: 9998;

}

.custDeletePopup{

background-color: transparent;

z-index: 9999;

left: 50%;

padding:10px;

position: absolute;

width: 400px;

margin-left: -200px;

top:200px;

}

.popupDeleteBackground{

background-color:black;

opacity: 0.30;

filter: alpha(opacity = 30);

position: absolute;

width: 100%;

height: 100%;

top: 0;

left: 0;

z-index: 9998;

}

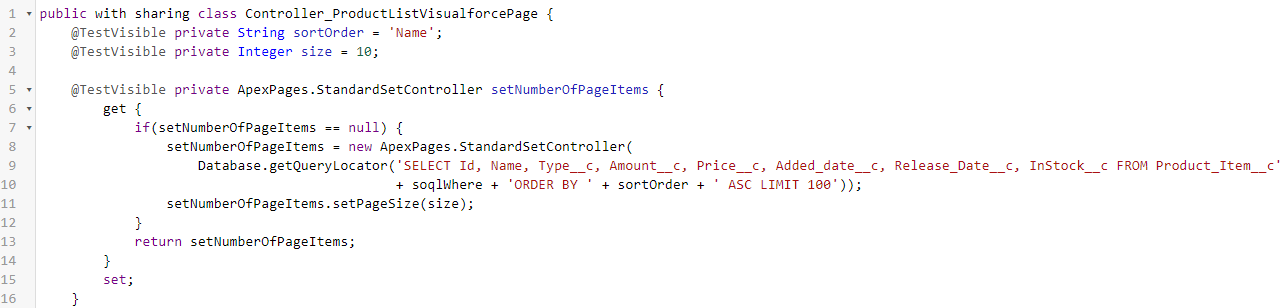
</style>

<!--/ POPUP CSS STYLES -->

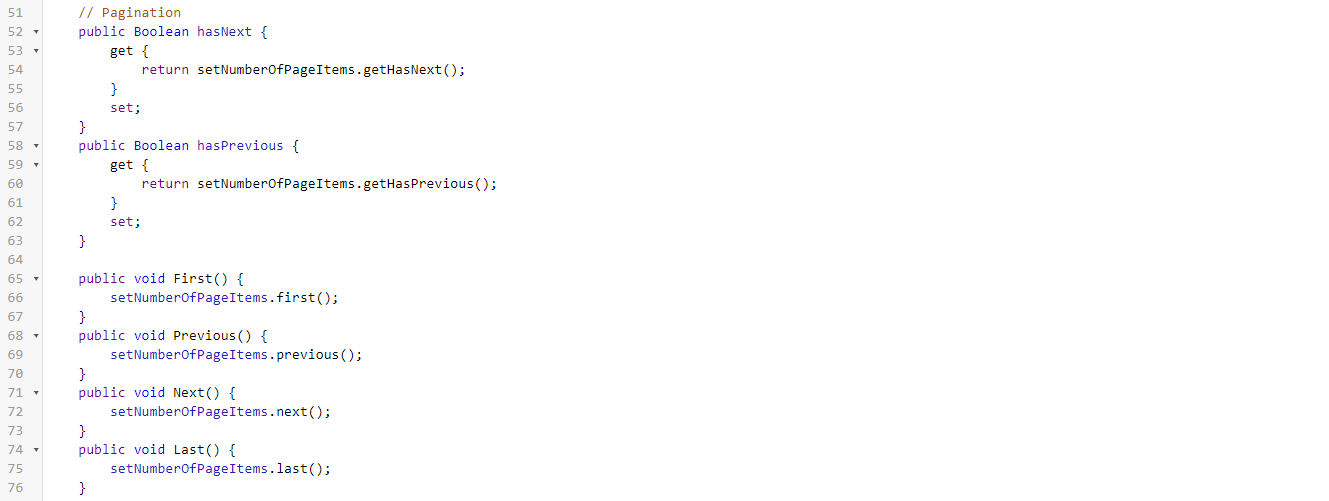
</apex:form>

</apex:page>



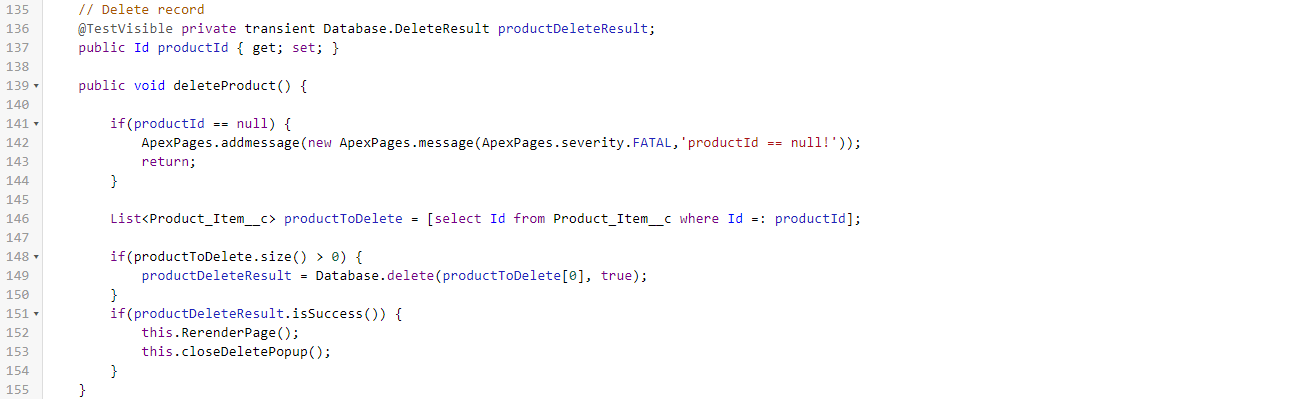














public with sharing class Controller\_ProductListVisualforcePage {

@TestVisible private String sortOrder = 'Name';

@TestVisible private Integer size = 10;

@TestVisible private ApexPages.StandardSetController setNumberOfPageItems {

get {

if(setNumberOfPageItems == null) {

setNumberOfPageItems = new ApexPages.StandardSetController(

Database.getQueryLocator('SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c, Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c FROM Product\_Item\_\_c' + soqlWhere + 'ORDER BY ' + sortOrder + ' ASC LIMIT 100'));

setNumberOfPageItems.setPageSize(size);

}

return setNumberOfPageItems;

}

set;

}

// Product records to visualforce page

public List<Product\_Item\_\_c> getProduct() {

return (List<Product\_Item\_\_c>) setNumberOfPageItems.getRecords();

}

private void RerenderPage() {

this.setNumberOfPageItems = null;

}

// Sort by all Fields

public void sort(string modifier) {

this.sortOrder = modifier;

this.RerenderPage();

}

public void sortByName() {

this.sort('Name');

}

public void sortByType() {

this.sort('Type\_\_c');

}

public void sortByPrice() {

this.sort('Price\_\_c');

}

public void sortByAmount() {

this.sort('Amount\_\_c');

}

public void sortByAddedDate() {

this.sort('Added\_Date\_\_c');

}

public void sortByReleaseDate() {

this.sort('Release\_Date\_\_c');

}

// Pagination

public Boolean hasNext {

get {

return setNumberOfPageItems.getHasNext();

}

set;

}

public Boolean hasPrevious {

get {

return setNumberOfPageItems.getHasPrevious();

}

set;

}

public void First() {

setNumberOfPageItems.first();

}

public void Previous() {

setNumberOfPageItems.previous();

}

public void Next() {

setNumberOfPageItems.next();

}

public void Last() {

setNumberOfPageItems.last();

}

// Add New Record Popup

public Boolean displayAddPopup { get; set; }

public void closeAddPopup() {

displayAddPopup = false;

}

public void showAddPopup() {

displayAddPopup = true;

}

// Delete Record Popup

public Boolean displayDeletePopup { get; set; }

public void closeDeletePopup() {

displayDeletePopup = false;

}

public void showDeletePopup() {

displayDeletePopup = true;

}

// Add New record

public Product\_Item\_\_c newProduct { get; set;}

public Date added\_date { get; set;}

public Date release\_date { get; set;}

public Controller\_ProductListVisualforcePage(ApexPages.StandardController controller) {

newProduct = (Product\_Item\_\_c)controller.getRecord();

}

@TestVisible private transient Database.SaveResult productSaveResult;

public void addNewProduct() {

if(newProduct.Name == null) {

ApexPages.addmessage(new ApexPages.message(ApexPages.severity.FATAL,' Please fill in at least a Name field!'));

return;

}

newProduct.Release\_Date\_\_c = release\_date;

newProduct.Added\_Date\_\_c = added\_date;

try{

productSaveResult = Database.insert(newProduct, true);

}

catch(DMLException e) { // catch validation rule

Apexpages.addMessage(new ApexPages.Message(ApexPages.SEVERITY.FATAL, e.getDmlMessage(0)));

}

catch(Exception ex) { // catch other possible exceptions

Apexpages.addMessage(new ApexPages.Message(ApexPages.SEVERITY.FATAL, ex.getMessage()));

}

this.RerenderPage();

this.closeAddPopup();

newProduct = new Product\_Item\_\_c();

}

// Delete record

@TestVisible private transient Database.DeleteResult productDeleteResult;

public Id productId { get; set; }

public void deleteProduct() {

if(productId == null) {

ApexPages.addmessage(new ApexPages.message(ApexPages.severity.FATAL,'productId == null!'));

return;

}

List<Product\_Item\_\_c> productToDelete = [select Id from Product\_Item\_\_c where Id =: productId];

if(productToDelete.size() > 0) {

productDeleteResult = Database.delete(productToDelete[0], true);

}

if(productDeleteResult.isSuccess()) {

this.RerenderPage();

this.closeDeletePopup();

}

}

// Search

public Date searchAddedDate { get; set; }

public String searchTitle { get; set; }

private String soqlWhere = ' ';

public void searchProduct() {

if(searchAddedDate != null && searchTitle != null) {

// If user fill up both of search fields

searchTitle = String.escapeSingleQuotes(searchTitle.trim());

soqlWhere = ' WHERE Added\_Date\_\_c =: searchAddedDate AND Name LIKE ' + '\'%' + searchTitle + '%\'' + ' ';

this.RerenderPage();

} else if (searchAddedDate != null && searchTitle == null) {

// If user fill up only Added Date search field

soqlWhere = ' WHERE Added\_Date\_\_c =: searchAddedDate ';

this.RerenderPage();

} else if (searchAddedDate == null && searchTitle != null) {

// If user fill up only Product Name search field

searchTitle = String.escapeSingleQuotes(searchTitle.trim());

soqlWhere = ' WHERE Name LIKE ' + '\'%' + searchTitle + '%\'' + ' ';

this.RerenderPage();

} else if(searchAddedDate == null && searchTitle == null) {

// If search fields are empty - "soqlWhere" variable dont change and = ' '

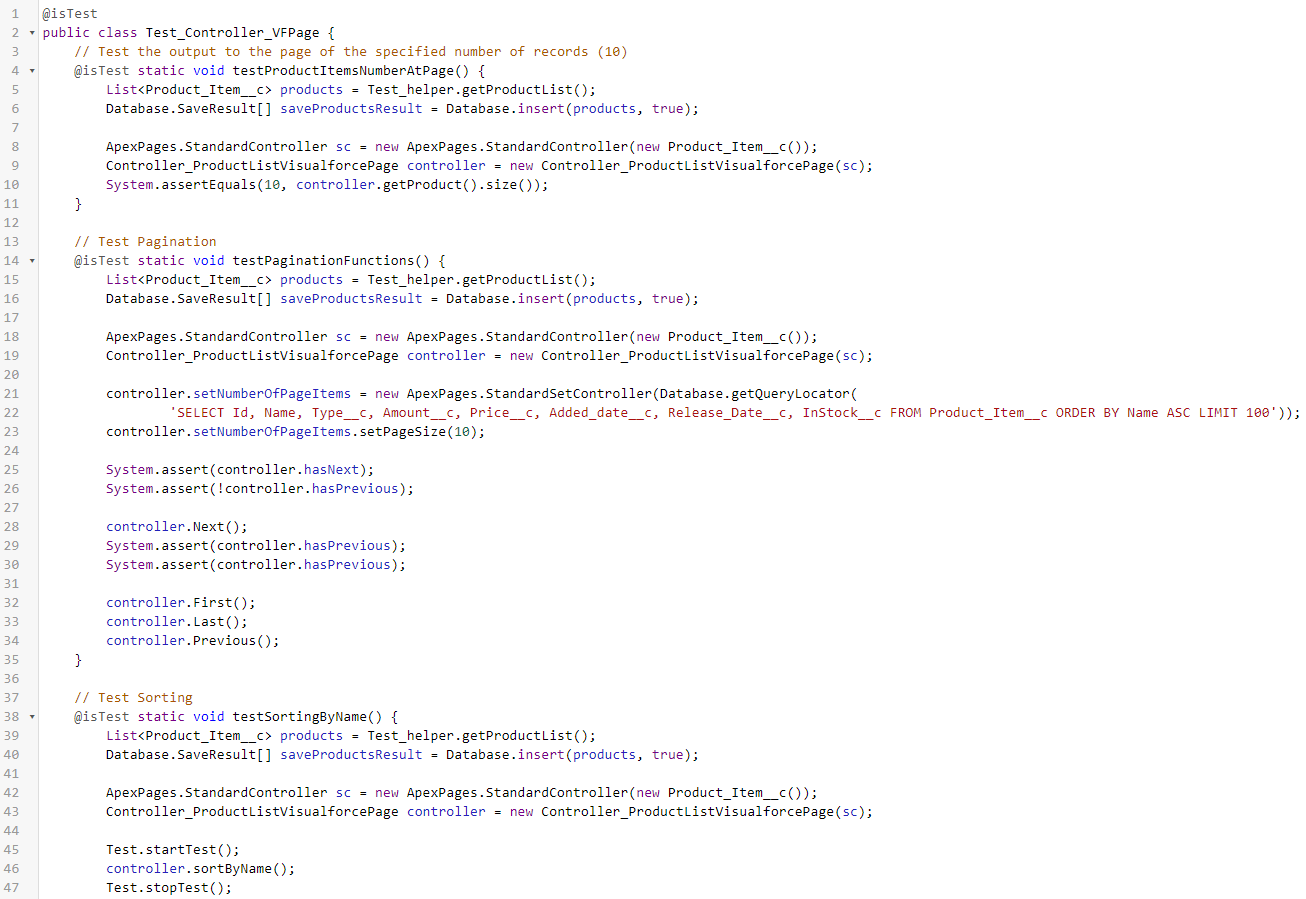
this.RerenderPage();

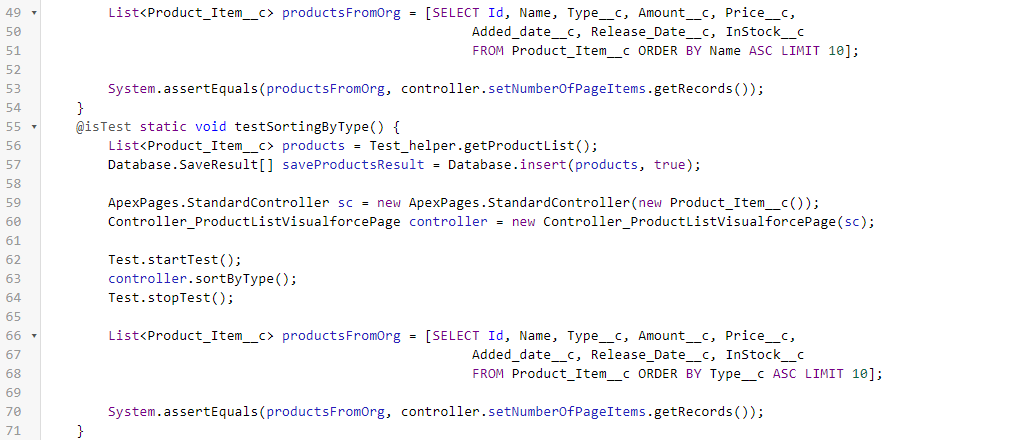
}

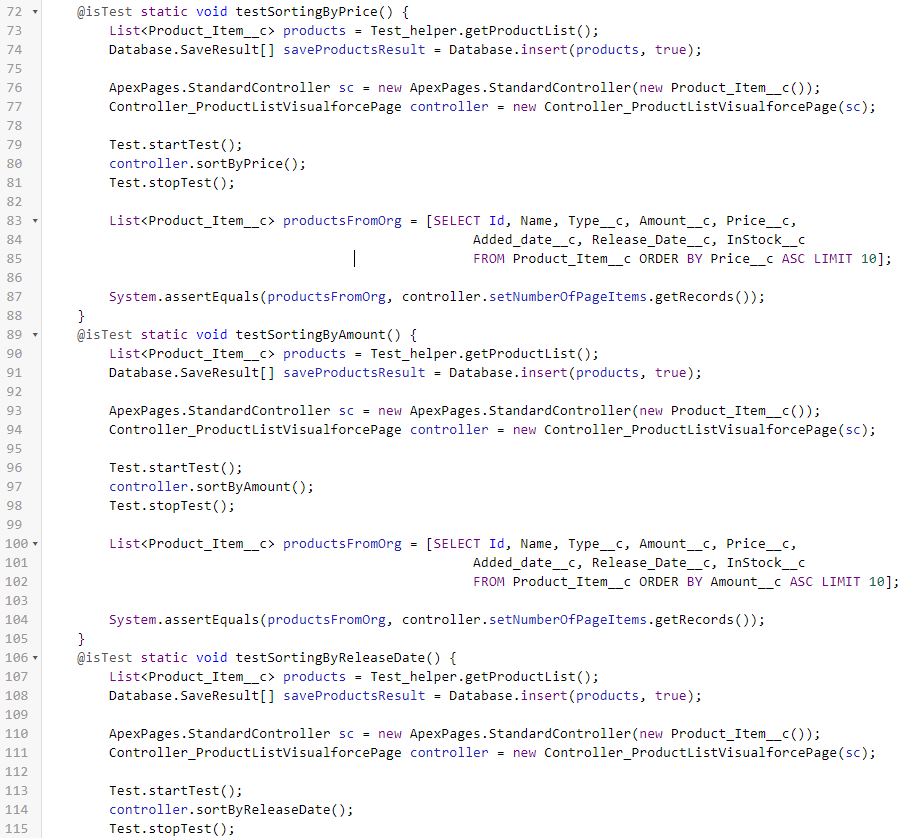
}

}

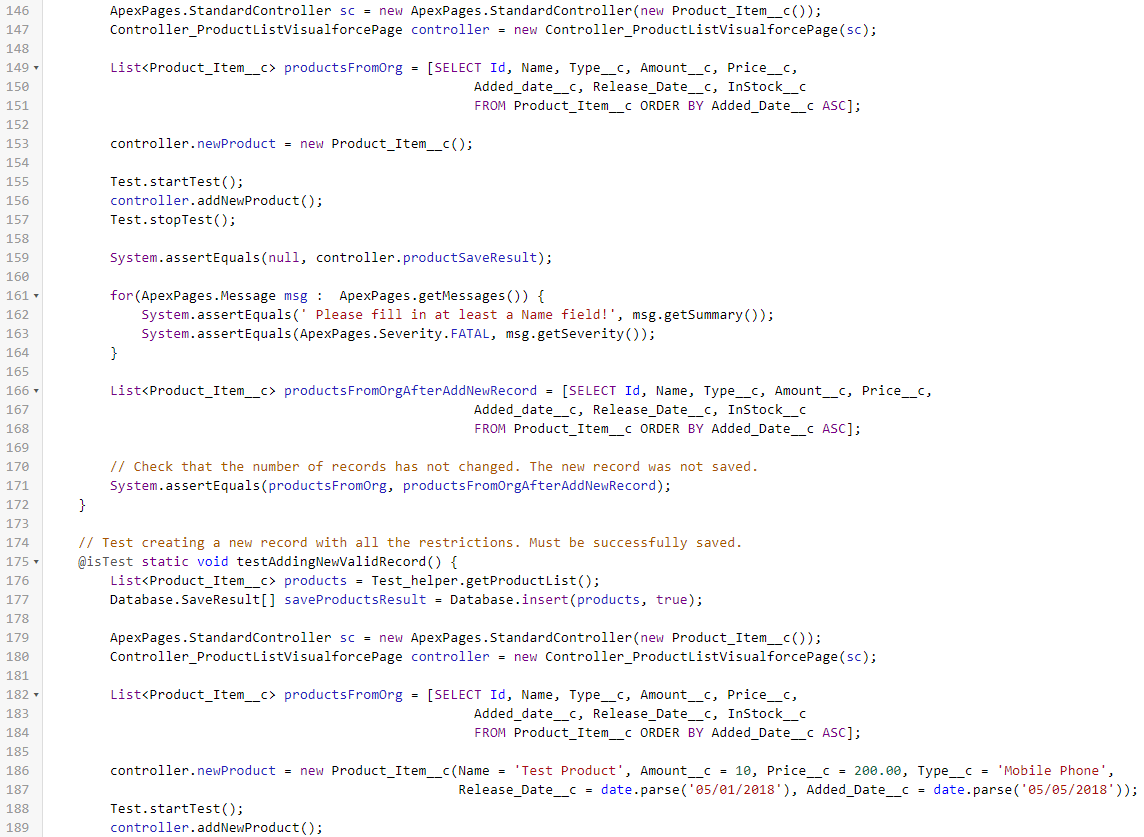


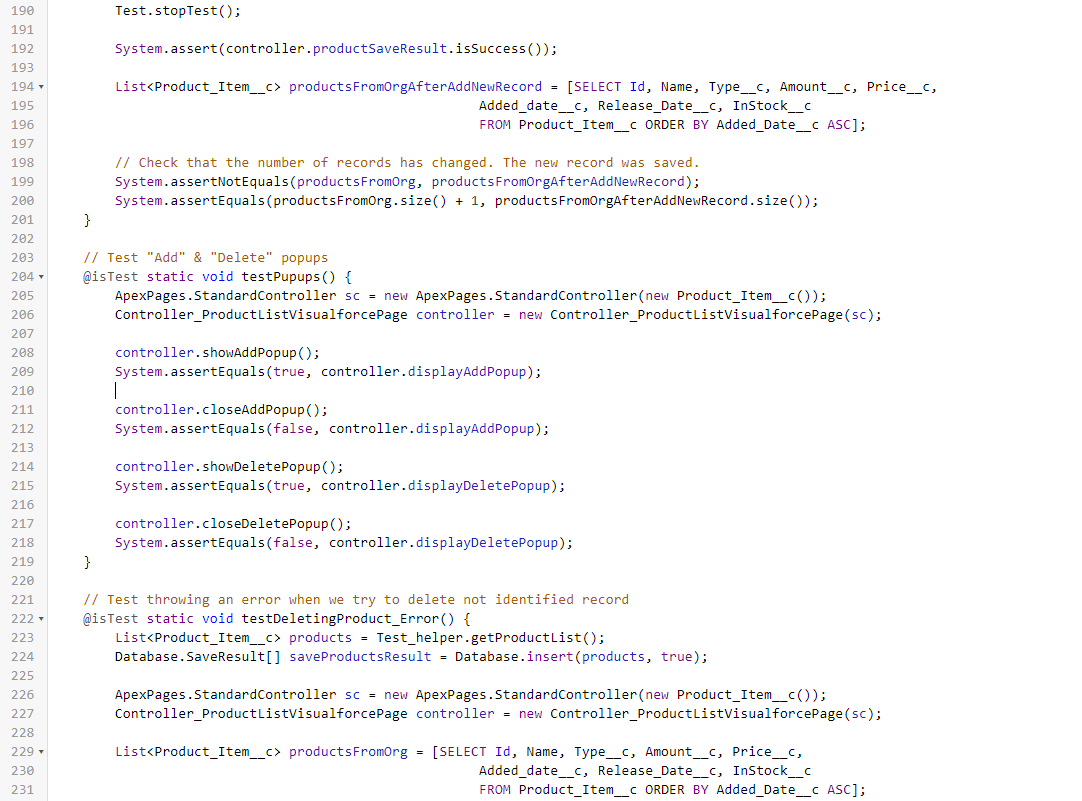




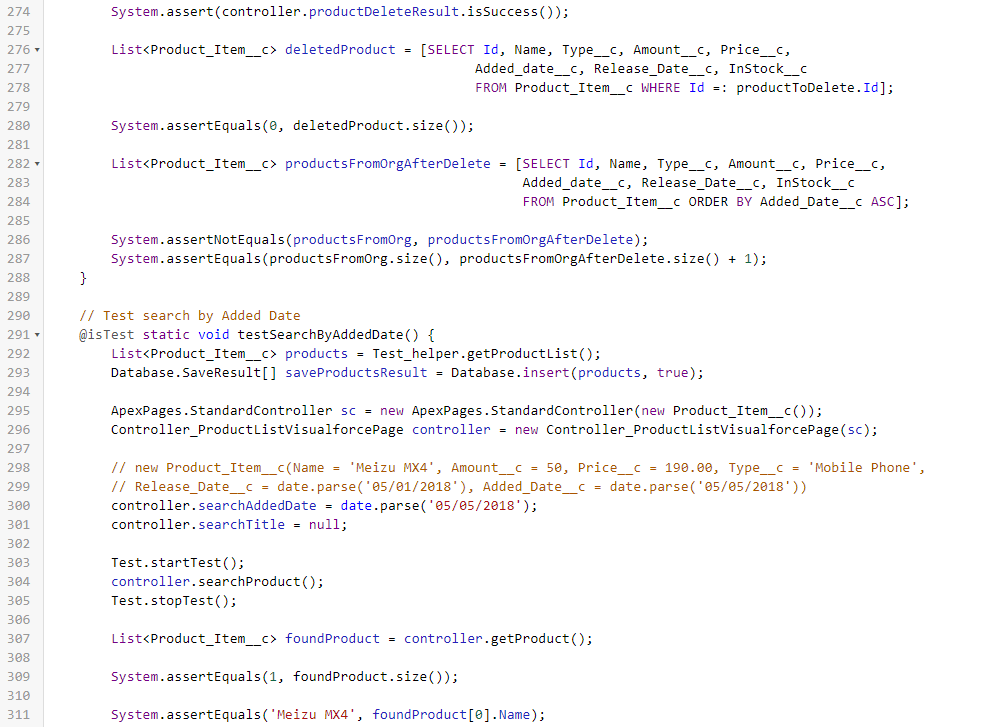






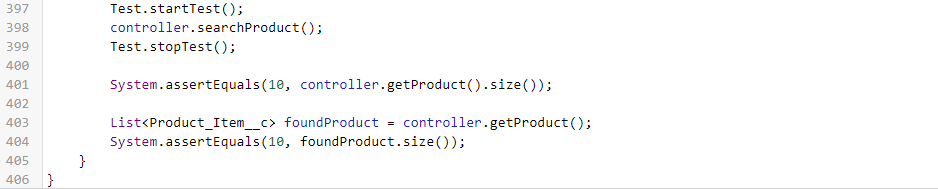












@isTest

public class Test\_Controller\_VFPage {

// Test the output to the page of the specified number of records (10)

@isTest static void testProductItemsNumberAtPage() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

System.assertEquals(10, controller.getProduct().size());

}

// Test Pagination

@isTest static void testPaginationFunctions() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

controller.setNumberOfPageItems = new ApexPages.StandardSetController(Database.getQueryLocator(

'SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c, Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c FROM Product\_Item\_\_c ORDER BY Name ASC LIMIT 100'));

controller.setNumberOfPageItems.setPageSize(10);

System.assert(controller.hasNext);

System.assert(!controller.hasPrevious);

controller.Next();

System.assert(controller.hasPrevious);

System.assert(controller.hasPrevious);

controller.First();

controller.Last();

controller.Previous();

}

// Test Sorting

@isTest static void testSortingByName() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

Test.startTest();

controller.sortByName();

Test.stopTest();

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Name ASC LIMIT 10];

System.assertEquals(productsFromOrg, controller.setNumberOfPageItems.getRecords());

}

@isTest static void testSortingByType() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

Test.startTest();

controller.sortByType();

Test.stopTest();

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Type\_\_c ASC LIMIT 10];

System.assertEquals(productsFromOrg, controller.setNumberOfPageItems.getRecords());

}

@isTest static void testSortingByPrice() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

Test.startTest();

controller.sortByPrice();

Test.stopTest();

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Price\_\_c ASC LIMIT 10];

System.assertEquals(productsFromOrg, controller.setNumberOfPageItems.getRecords());

}

@isTest static void testSortingByAmount() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

Test.startTest();

controller.sortByAmount();

Test.stopTest();

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Amount\_\_c ASC LIMIT 10];

System.assertEquals(productsFromOrg, controller.setNumberOfPageItems.getRecords());

}

@isTest static void testSortingByReleaseDate() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

Test.startTest();

controller.sortByReleaseDate();

Test.stopTest();

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Release\_Date\_\_c ASC LIMIT 10];

System.assertEquals(productsFromOrg, controller.setNumberOfPageItems.getRecords());

}

@isTest static void testSortingByAddedDate() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

Test.startTest();

controller.sortByAddedDate();

Test.stopTest();

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC LIMIT 10];

System.assertEquals(productsFromOrg, controller.setNumberOfPageItems.getRecords());

}

// Test adding new empty record (trigger an error message: 'Please fill in at least a Name field!')

@isTest static void testAddingNewEmptyRecord() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

controller.newProduct = new Product\_Item\_\_c();

Test.startTest();

controller.addNewProduct();

Test.stopTest();

System.assertEquals(null, controller.productSaveResult);

for(ApexPages.Message msg : ApexPages.getMessages()) {

System.assertEquals(' Please fill in at least a Name field!', msg.getSummary());

System.assertEquals(ApexPages.Severity.FATAL, msg.getSeverity());

}

List<Product\_Item\_\_c> productsFromOrgAfterAddNewRecord = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

// Check that the number of records has not changed. The new record was not saved.

System.assertEquals(productsFromOrg, productsFromOrgAfterAddNewRecord);

}

// Test сreating a new record with all the restrictions. Must be successfully saved.

@isTest static void testAddingNewValidRecord() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

controller.newProduct = new Product\_Item\_\_c(Name = 'Test Product', Amount\_\_c = 10, Price\_\_c = 200.00, Type\_\_c = 'Mobile Phone',

Release\_Date\_\_c = date.parse('05/01/2018'), Added\_Date\_\_c = date.parse('05/05/2018'));

Test.startTest();

controller.addNewProduct();

Test.stopTest();

System.assert(controller.productSaveResult.isSuccess());

List<Product\_Item\_\_c> productsFromOrgAfterAddNewRecord = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

// Check that the number of records has changed. The new record was saved.

System.assertNotEquals(productsFromOrg, productsFromOrgAfterAddNewRecord);

System.assertEquals(productsFromOrg.size() + 1, productsFromOrgAfterAddNewRecord.size());

}

// Test "Add" & "Delete" popups

@isTest static void testPupups() {

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

controller.showAddPopup();

System.assertEquals(true, controller.displayAddPopup);

controller.closeAddPopup();

System.assertEquals(false, controller.displayAddPopup);

controller.showDeletePopup();

System.assertEquals(true, controller.displayDeletePopup);

controller.closeDeletePopup();

System.assertEquals(false, controller.displayDeletePopup);

}

// Test throwing an error when we try to delete not identified record

@isTest static void testDeletingProduct\_Error() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

controller.productId = null;

Test.startTest();

controller.deleteProduct();

Test.stopTest();

System.assertEquals(null, controller.productDeleteResult);

for(ApexPages.Message msg : ApexPages.getMessages()) {

System.assertEquals('productId == null!', msg.getSummary());

System.assertEquals(ApexPages.Severity.FATAL, msg.getSeverity());

}

List<Product\_Item\_\_c> productsFromOrgAfterAddNewRecord = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

// Check that the number of records has not changed. The record wasn't deleted.

System.assertEquals(productsFromOrg, productsFromOrgAfterAddNewRecord);

}

// Test successfull deleting record.

@isTest static void testDeletingProduct\_Success() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

List<Product\_Item\_\_c> productsFromOrg = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

Product\_Item\_\_c productToDelete = productsFromOrg[0];

controller.productId = productToDelete.Id;

Test.startTest();

controller.deleteProduct();

Test.stopTest();

System.assertNotEquals(null, controller.productDeleteResult);

System.assert(controller.productDeleteResult.isSuccess());

List<Product\_Item\_\_c> deletedProduct = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c WHERE Id =: productToDelete.Id];

System.assertEquals(0, deletedProduct.size());

List<Product\_Item\_\_c> productsFromOrgAfterDelete = [SELECT Id, Name, Type\_\_c, Amount\_\_c, Price\_\_c,

Added\_date\_\_c, Release\_Date\_\_c, InStock\_\_c

FROM Product\_Item\_\_c ORDER BY Added\_Date\_\_c ASC];

System.assertNotEquals(productsFromOrg, productsFromOrgAfterDelete);

System.assertEquals(productsFromOrg.size(), productsFromOrgAfterDelete.size() + 1);

}

// Test search by Added Date

@isTest static void testSearchByAddedDate() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

// new Product\_Item\_\_c(Name = 'Meizu MX4', Amount\_\_c = 50, Price\_\_c = 190.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/01/2018'), Added\_Date\_\_c = date.parse('05/05/2018'))

controller.searchAddedDate = date.parse('05/05/2018');

controller.searchTitle = null;

Test.startTest();

controller.searchProduct();

Test.stopTest();

List<Product\_Item\_\_c> foundProduct = controller.getProduct();

System.assertEquals(1, foundProduct.size());

System.assertEquals('Meizu MX4', foundProduct[0].Name);

System.assertEquals(50, foundProduct[0].Amount\_\_c);

System.assertEquals(190.00, foundProduct[0].Price\_\_c);

System.assertEquals('Mobile Phone', foundProduct[0].Type\_\_c);

System.assertEquals(Date.parse('05/01/2018'), foundProduct[0].Release\_Date\_\_c);

System.assertEquals(true, foundProduct[0].InStock\_\_c);

}

// Test search by product name of part of product name (title)

@isTest static void testSearchByProductTitleOrItsPart() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

//new Product\_Item\_\_c(Name = 'Thunderobot 911GT-Y6', Amount\_\_c = 10, Price\_\_c = 3193.33, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/19/2018'), Added\_Date\_\_c = date.parse('05/22/2018'))

controller.searchAddedDate = null;

controller.searchTitle = 'robot';

Test.startTest();

controller.searchProduct();

Test.stopTest();

List<Product\_Item\_\_c> foundProduct = controller.getProduct();

System.assertEquals(1, foundProduct.size());

System.assertEquals('Thunderobot 911GT-Y6', foundProduct[0].Name);

System.assertEquals(10, foundProduct[0].Amount\_\_c);

System.assertEquals(3193.33, foundProduct[0].Price\_\_c);

System.assertEquals('Laptop', foundProduct[0].Type\_\_c);

System.assertEquals(Date.parse('05/19/2018'), foundProduct[0].Release\_Date\_\_c);

System.assertEquals(true, foundProduct[0].InStock\_\_c);

}

// Test search by both product name(or its part) and added date

@isTest static void testSearchProductByNameAndAddedDate() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

//new Product\_Item\_\_c(Name = 'Thunderobot 911GT-Y6', Amount\_\_c = 10, Price\_\_c = 3193.33, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/19/2018'), Added\_Date\_\_c = date.parse('05/22/2018'))

//new Product\_Item\_\_c(Name = 'Meizu Pro 6', Amount\_\_c = 600, Price\_\_c = 299.99, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/04/2018'), Added\_Date\_\_c = date.parse('05/22/2018'))

controller.searchAddedDate = date.parse('05/22/2018');

controller.searchTitle = 'ro';

Test.startTest();

controller.searchProduct();

Test.stopTest();

List<Product\_Item\_\_c> foundProduct = controller.getProduct();

System.assertEquals(2, foundProduct.size());

System.assertEquals('Meizu Pro 6', foundProduct[0].Name);

System.assertEquals(600, foundProduct[0].Amount\_\_c);

System.assertEquals(299.99, foundProduct[0].Price\_\_c);

System.assertEquals('Mobile Phone', foundProduct[0].Type\_\_c);

System.assertEquals(Date.parse('05/04/2018'), foundProduct[0].Release\_Date\_\_c);

System.assertEquals(true, foundProduct[0].InStock\_\_c);

System.assertEquals('Thunderobot 911GT-Y6', foundProduct[1].Name);

System.assertEquals(10, foundProduct[1].Amount\_\_c);

System.assertEquals(3193.33, foundProduct[1].Price\_\_c);

System.assertEquals('Laptop', foundProduct[1].Type\_\_c);

System.assertEquals(Date.parse('05/19/2018'), foundProduct[1].Release\_Date\_\_c);

System.assertEquals(true, foundProduct[1].InStock\_\_c);

}

// Test empty search request

@isTest static void testEmptySearchRequest() {

List<Product\_Item\_\_c> products = Test\_helper.getProductList();

Database.SaveResult[] saveProductsResult = Database.insert(products, true);

ApexPages.StandardController sc = new ApexPages.StandardController(new Product\_Item\_\_c());

Controller\_ProductListVisualforcePage controller = new Controller\_ProductListVisualforcePage(sc);

controller.searchAddedDate = null;

controller.searchTitle = null;

Test.startTest();

controller.searchProduct();

Test.stopTest();

System.assertEquals(10, controller.getProduct().size());

List<Product\_Item\_\_c> foundProduct = controller.getProduct();

System.assertEquals(10, foundProduct.size());

}

}





@isTest

public class Test\_Helper {

@isTest public static List<Product\_Item\_\_c> getProductList() {

return new List<Product\_Item\_\_c> { new Product\_Item\_\_c(Name = 'Meizu MX4', Amount\_\_c = 50, Price\_\_c = 190.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/01/2018'), Added\_Date\_\_c = date.parse('05/05/2018')),

new Product\_Item\_\_c(Name = 'Meizu MX4 Pro', Amount\_\_c = 100, Price\_\_c = 220.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/07/2018'), Added\_Date\_\_c = date.parse('05/10/2018')),

new Product\_Item\_\_c(Name = 'Meizu MX6', Amount\_\_c = 90, Price\_\_c = 235.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/11/2018'), Added\_Date\_\_c = date.parse('05/12/2018')),

new Product\_Item\_\_c(Name = 'Meizu M2', Amount\_\_c = 135, Price\_\_c = 112.99, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/03/2018'), Added\_Date\_\_c = date.parse('02/10/2018')),

new Product\_Item\_\_c(Name = 'Meizu M3', Amount\_\_c = 77, Price\_\_c = 148.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('04/01/2018'), Added\_Date\_\_c = date.parse('04/07/2018')),

new Product\_Item\_\_c(Name = 'Meizu M5s', Amount\_\_c = 233, Price\_\_c = 151.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('04/11/2018'), Added\_Date\_\_c = date.parse('02/18/2018')),

new Product\_Item\_\_c(Name = 'Meizu M6s', Amount\_\_c = 111, Price\_\_c = 160.99, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('04/17/2018'), Added\_Date\_\_c = date.parse('04/22/2018')),

new Product\_Item\_\_c(Name = 'Meizu E2', Amount\_\_c = 250, Price\_\_c = 240.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('03/12/2018'), Added\_Date\_\_c = date.parse('03/15/2018')),

new Product\_Item\_\_c(Name = 'Meizu Pro 6 Plus', Amount\_\_c = 500, Price\_\_c = 440.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('04/19/2018'), Added\_Date\_\_c = date.parse('04/29/2018')),

new Product\_Item\_\_c(Name = 'Meizu Pro 6', Amount\_\_c = 600, Price\_\_c = 299.99, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/04/2018'), Added\_Date\_\_c = date.parse('05/22/2018')),

new Product\_Item\_\_c(Name = 'Meizu Pro 7', Amount\_\_c = 450, Price\_\_c = 433.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('04/28/2018'), Added\_Date\_\_c = date.parse('05/01/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Redmi 3', Amount\_\_c = 118, Price\_\_c = 116.50, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('04/13/2018'), Added\_Date\_\_c = date.parse('04/19/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Redmi Note 4X', Amount\_\_c = 97, Price\_\_c = 135.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('03/16/2018'), Added\_Date\_\_c = date.parse('03/23/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Redmi 5', Amount\_\_c = 27, Price\_\_c = 88.99, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('02/01/2018'), Added\_Date\_\_c = date.parse('02/12/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Mi A1', Amount\_\_c = 144, Price\_\_c = 195.34, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('01/11/2018'), Added\_Date\_\_c = date.parse('01/15/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Mi Max2', Amount\_\_c = 720, Price\_\_c = 531.90, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('01/14/2018'), Added\_Date\_\_c = date.parse('01/24/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Mi 5s', Amount\_\_c = 589, Price\_\_c = 286.99, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('02/09/2018'), Added\_Date\_\_c = date.parse('02/13/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Mi 6X', Amount\_\_c = 890, Price\_\_c = 689.00, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('04/08/2018'), Added\_Date\_\_c = date.parse('04/21/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Mi Mix', Amount\_\_c = 541, Price\_\_c = 489.90, Type\_\_c = 'Mobile Phone', Release\_Date\_\_c = date.parse('05/11/2018'), Added\_Date\_\_c = date.parse('05/25/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Mi Pad 3', Amount\_\_c = 174, Price\_\_c = 329.00, Type\_\_c = 'Tablet', Release\_Date\_\_c = date.parse('01/14/2018'), Added\_Date\_\_c = date.parse('01/21/2018')),

new Product\_Item\_\_c(Name = 'Lenovo P8', Amount\_\_c = 616, Price\_\_c = 228.98, Type\_\_c = 'Tablet', Release\_Date\_\_c = date.parse('03/03/2018'), Added\_Date\_\_c = date.parse('03/08/2018')),

new Product\_Item\_\_c(Name = 'Lenovo Tab4', Amount\_\_c = 311, Price\_\_c = 241.99, Type\_\_c = 'Tablet', Release\_Date\_\_c = date.parse('03/27/2018'), Added\_Date\_\_c = date.parse('03/28/2018')),

new Product\_Item\_\_c(Name = 'Lenovo Tab3', Amount\_\_c = 519, Price\_\_c = 136.99, Type\_\_c = 'Tablet', Release\_Date\_\_c = date.parse('02/22/2018'), Added\_Date\_\_c = date.parse('02/24/2018')),

new Product\_Item\_\_c(Name = 'Acer A315-31-C9ZN', Amount\_\_c = 498, Price\_\_c = 495.66, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('01/16/2018'), Added\_Date\_\_c = date.parse('01/27/2018')),

new Product\_Item\_\_c(Name = 'Acer SF315-51G-513S', Amount\_\_c = 911, Price\_\_c = 969.36, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/15/2018'), Added\_Date\_\_c = date.parse('05/25/2018')),

new Product\_Item\_\_c(Name = 'Acer ES1-531-C7TF', Amount\_\_c = 1200, Price\_\_c = 380.79, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('04/23/2018'), Added\_Date\_\_c = date.parse('04/27/2018')),

new Product\_Item\_\_c(Name = 'Acer VX5-591G-58AX', Amount\_\_c = 881, Price\_\_c = 917.69, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('02/19/2018'), Added\_Date\_\_c = date.parse('02/24/2018')),

new Product\_Item\_\_c(Name = 'ASUS F540UP7200', Amount\_\_c = 290, Price\_\_c = 1133.33, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('03/18/2018'), Added\_Date\_\_c = date.parse('03/29/2018')),

new Product\_Item\_\_c(Name = 'ASUS F541UJ7200', Amount\_\_c = 30, Price\_\_c = 776.58, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/02/2018'), Added\_Date\_\_c = date.parse('05/13/2018')),

new Product\_Item\_\_c(Name = 'Xiaomi Mi Notebook Pro', Amount\_\_c = 110, Price\_\_c = 1099.00, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/01/2018'), Added\_Date\_\_c = date.parse('05/11/2018')),

new Product\_Item\_\_c(Name = 'Apple MacBook Air 15', Amount\_\_c = 339, Price\_\_c = 909.99, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/06/2018'), Added\_Date\_\_c = date.parse('05/18/2018')),

new Product\_Item\_\_c(Name = 'Apple Macbook Pro 15 ', Amount\_\_c = 12, Price\_\_c = 2999.99, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/07/2018'), Added\_Date\_\_c = date.parse('05/17/2018')),

new Product\_Item\_\_c(Name = 'MSI GL62M 7REX-1252CN', Amount\_\_c = 55, Price\_\_c = 1151.49, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/17/2018'), Added\_Date\_\_c = date.parse('05/21/2018')),

new Product\_Item\_\_c(Name = 'Thunderobot 911GT-Y6', Amount\_\_c = 10, Price\_\_c = 3193.33, Type\_\_c = 'Laptop', Release\_Date\_\_c = date.parse('05/19/2018'), Added\_Date\_\_c = date.parse('05/22/2018'))};

}

}

