Commerce Search Module:

- The hybris Commerce Search provides you with easy-to-use administrative tools for search and navigation using Solr server.
- Solr (pronounced "solar") is an open source enterprise search platform, written in Java, from the Apache Lucene project.
- Its major features include full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features.
- Providing distributed search and index replication, Solr is designed for scalability and Fault tolerance.
- Solr is the second-most popular enterprise search engine after Elasticsearch.(kibana).
- Solr runs as a standalone full-text search server.
- - It uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it usable from most popular programming languages.
- Apache Lucene and Apache Solr are both produced by the same Apache Software Foundation development team since the two projects were merged in 2010. It is common to refer to the technology or products as Lucene/Solr or Solr/Lucene.
- Solr is bundled as the built-in search in many applications such as CMS/ECM systems.
- The major Hadoop distributions from Cloudera,Hortonworks and MapR all bundle Solr as the search engine for their Big Data platforms.
- Solr exposes industry standard HTTP REST-like APIs with both XML and JSON support, and will integrate with any system or programming language supporting these standards.

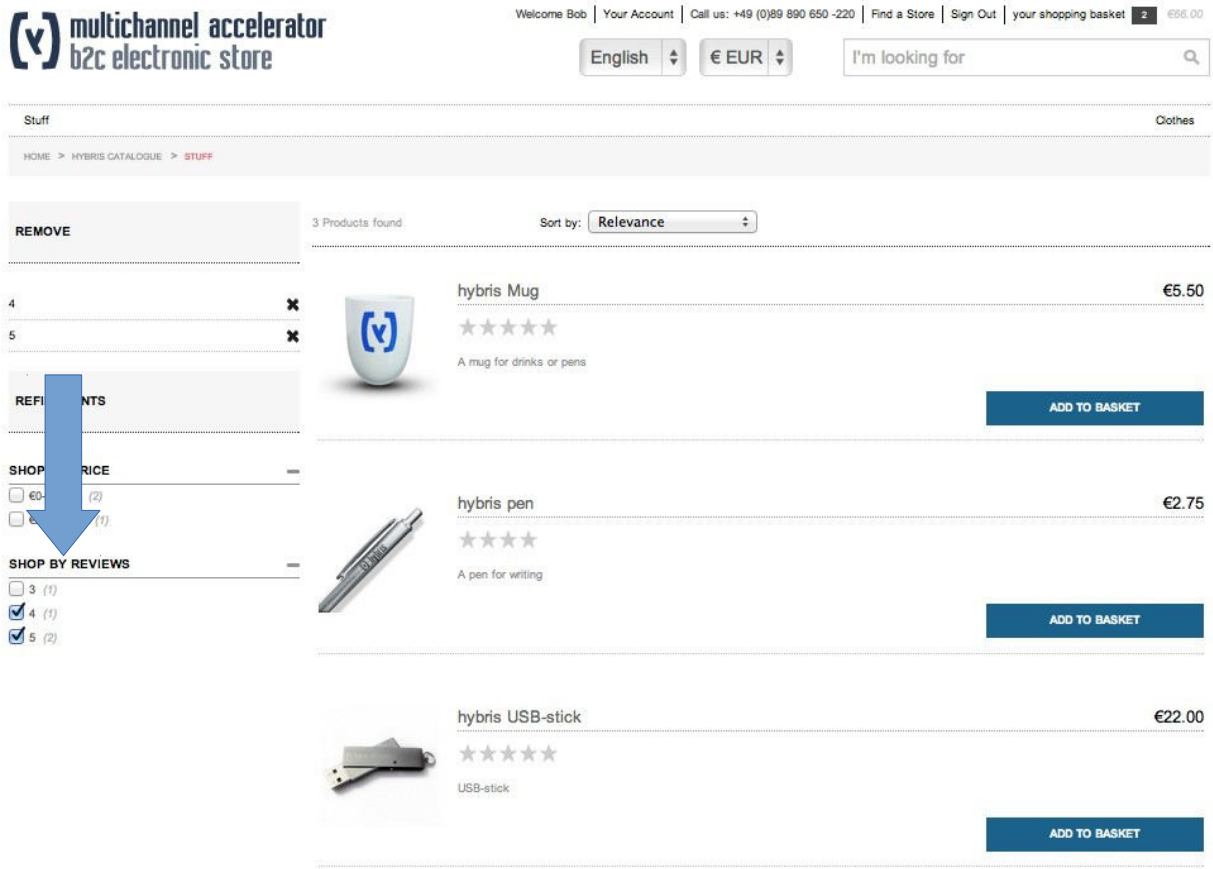
SolR Attributes and Facets

Task Description

- We want to add a ranged **review rating** property and display it as a facet with possible values from the set (1-5).

- This would allow the customer to refine search results based on the average of the ratings given products by other customers. We will use a ranged property here to limit the number of facets by rounding the floating point value of the rating average (AvgRating) to the nearest integer value.

Create ranges for your New Ranged Index Property:



The customer may add a review by adding number from 1-5. Their average will be a floating point number, and we will define reasonable ranges to present as facets. Instead of displaying the range, we will show the nearest integer value.

Let's add our **SolrValueRange**:

1	0.5	1.49
2	1.5	2.49
3	2.5	3.49
4	3.5	4.49
5	4.5	5

merchandiseinitialdata/resources/merchandiseinitialdata/import/coredata/stores/hybris/solr.impex

```
...
# Define price ranges
INSERT_UPDATE SolrValueRange;&rangeValueRefID;solrValueRangeSet(name)
[unique=true];name[unique=true];from;to

;avgRatingRef1;hybrisAvgRating;1;0.5;1.49
;avgRatingRef2;hybrisAvgRating;2;1.5;2.49
;avgRatingRef3;hybrisAvgRating;3;2.5;3.49
;avgRatingRef4;hybrisAvgRating;4;3.5;4.49
;avgRatingRef5;hybrisAvgRating;5;4.5;5
...
```

We now create a value range set to collect our range values and give it a name to be referenced by our indexed property:

merchandiseinitialdata/resources/merchandiseinitialdata/import/coredata/stores/hybris/solr.impex

```
...
# Define price range set
INSERT_UPDATE
SolrValueRangeSet;name[unique=true];qualifier;type;solrValueRanges(&ra
ngeValueRefID);
;
hybrisAvgRating;;double;avgRatingRef1,avgRatingRef2,avgRatingRef3,avgR
atingRef4,avgRatingRef5;;
...
```

Create our Index Property

Create a new indexed property (**reviewAvgRatingRange**) for the merchandise's shop product type (**hybrisProductType**).

merchandiseinitialdata/resources/merchandiseinitialdata/import/coredata/stores/hybris/solr.impex

```
...
# Other facet properties
INSERT_UPDATE SolrIndexedProperty;solrIndexedType(identifier)
[unique=true];name[unique=true];type(code);sortableType(code);curr
ency[default=false];localized[default=false];multiValue[default=fa
lse];facet[default=true];facetType(code);facetSort(code);priority;
visible;useForSpellchecking[default=false];useForAutocomplete[defa
ult=false];fieldValueProvider;facetDisplayNameProvider;customFacet
SortProvider;topValuesProvider;rangeSets(name)

;$solrIndexedType; reviewAvgRatingRange ;double ;; ; ; ;true
;MultiSelectOr ;Alpha ;2000 ;true; ;
;rangedAvgCustomerRatingProvider;;;;hybrisAvgRating
```

Note:-Set 'facet' to true - it will be display as facet on the store front

- Set 'facetType' to MultiSelectOr - Customer can do refinements like (please show me products with 4 *OR* 5 stars)
- Define **fieldValueProvider** - which is a bean id of the java class that provides values for the property while creating solr index (we still need to implement and register **rangedAvgCustomerRatingProvider**)
- Set a rangeSet to use with this index property

Change the Update Query

An update query fetches the products that need to be updated in the solr index. As the solr index now contains information about customer reviews, a separate item, we need to ensure that the query will not only fetch the products that were modified since the time of the last index, but also those for which customer reviews have changed:

merchandiseinitialdata/resources/merchandiseinitialdata/import/coredata/stores/hybris/solr.impe

```
...
# Create the queries that will be used to extract data for Solr
INSERT_UPDATE SolrIndexerQuery;solrIndexedType(identifier)
[unique=true];identifier[unique=true];type(code);injectCurrentDate[default=true];injectCurrentTime[defa
ult=true];injectLastIndexTime[default=true];query;user(uid)
;$solrIndexedType;$solrIndexedType-fullQuery;full;;;false;"SELECT {PK} FROM
{Product}";anonymous
;$solrIndexedType;$solrIndexedType-updateQuery;update;;;;"SELECT DISTINCT tbl.pk, tbl.code
FROM (
   {{
      SELECT DISTINCT {p:PK} AS pk, {p:code} AS code
      FROM {Product AS p LEFT JOIN CustomerReview AS cr ON {cr:product}={p:PK} }
      WHERE {p:varianttype} IS NULL AND ({p:modifiedtime} >= ?lastIndexTime OR
{cr:modifiedtime} >= ?lastIndexTime)
   }}
   UNION
   {{
      SELECT DISTINCT {p:PK} AS pk, {p:code} AS code
      FROM {VariantProduct AS p JOIN Product AS bp1 ON {p:baseProduct}={bp1:PK} LEFT JOIN
CustomerReview AS cr ON {cr:product}={bp1:PK} }
      WHERE {p:varianttype} IS NULL AND ({bp1:modifiedtime} >= ?lastIndexTime OR
{cr:modifiedtime} >= ?lastIndexTime)
   }}
   UNION
   {{
      SELECT DISTINCT {p:PK} AS pk, {p:code} AS code
      FROM {VariantProduct AS p JOIN VariantProduct AS bp1 ON {p:baseProduct}={bp1:PK} JOIN
Product AS bp2 ON {bp1:baseProduct}={bp2:PK} LEFT JOIN CustomerReview AS cr ON
{cr:product}={bp2:PK} }
      WHERE {p:varianttype} IS NULL AND ({bp2:modifiedtime} >= ?lastIndexTime OR
{cr:modifiedtime} >= ?lastIndexTime)
   }}
) tbl ORDER BY tbl.code";anonymous
...
```

Here is a complete listing of `solr.impex`:

To the **`solr.impex`** file:

`solr.impex`

```
#
# Import the Solr configuration for the B2CTelco store
#
$productCatalog=b2ctelcoProductCatalog
$catalogVersions=catalogVersions(catalog(id),version);
$serverConfigName=b2ctelcoSolrServerConfig
$indexConfigName=b2ctelcoSolrIndexConfig
$searchConfigName=b2ctelcoPageSize
$facetSearchConfigName=b2ctelcoIndex
$facetSearchConfigDescription=b2ctelco Solr Index
$searchIndexNamePrefix=b2ctelco
$solrIndexedType=b2ctelcoProductType
$indexBaseSite=b2ctelco
$indexLanguages=en
$indexCurrencies=USD

#
# Setup the Solr server, indexer, and search configs
#



#
# Setup the Solr server, indexer, and search configs
#

# Create the solr server configuration
INSERT_UPDATE SolrServerConfig;name[unique=true];mode(code);embeddedMaster
;$serverConfigName;standalone;false;

INSERT_UPDATE SolrEndpointUrl;solrServerConfig(name)
[unique=true];url[unique=true];master[unique=true,default=false]
;$serverConfigName;http://localhost:8983/solr;true

# Create the solr indexer configuration
INSERT_UPDATE
SolrIndexConfig;name[unique=true];batchSize;numberOfThreads;indexMode(code);
;$indexConfigName;100;1;TWO_PHASE;

# Create the faceted search configuration
INSERT_UPDATE SolrSearchConfig;description[unique=true];pageSize
;$searchConfigName;20

#
# Setup the indexed types, their properties, and the update queries
#

# Declare the indexed type Product
```

```
INSERT_UPDATE SolrIndexedType;identifier[unique=true];type(code);variant;sorts(&sortRefID)
;$solrIndexedType;Product;false;sortRef1,sortRef2,sortRef3,sortRef4,sortRef5,sortRef6

INSERT_UPDATE
SolrFacetSearchConfig;name[unique=true];description;indexNamePrefix;languages(isocode);curre
ncies(isocode);solrServerConfig(name);solrSearchConfig(description);solrIndexConfig(name);solr
IndexedTypes(identifier);enabledLanguageFallbackMechanism;$catalogVersions
;$facetSearchConfigName;$facetSearchConfigDescription;$searchIndexNamePrefix;
$indexLanguages;$indexCurrencies;$serverConfigName;$searchConfigName;
$indexConfigName;$solrIndexedType;true;$productCatalog:Online,$productCatalog:Staged

UPDATE BaseSite;uid[unique=true];solrFacetSearchConfiguration(name)
;$indexBaseSite;$facetSearchConfigName




# Define price ranges
INSERT_UPDATE SolrValueRange;&rangeValueRefID;solrValueRangeSet(name)
[unique=true];name[unique=true];from;to

 ;avgRatingRef1    ;hybrisAvgRating;1;0.5;1.49
;avgRatingRef2;hybrisAvgRating;2;1.5;2.49
;avgRatingRef3;hybrisAvgRating;3;2.5;3.49
;avgRatingRef4;hybrisAvgRating;4;3.5;4.49
;avgRatingRef5;hybrisAvgRating;5;4.5;5



# Define price range set
INSERT_UPDATE
SolrValueRangeSet;name[unique=true];qualifier;type;solrValueRanges(&rangeValueRefID);

;
hybrisAvgRating;;double;avgRatingRef1,avgRatingRef2,avgRatingRef3,avgRatingRef4,avgRating
Ref5;;


# Other facet properties
INSERT_UPDATE SolrIndexedProperty;solrIndexedType(identifier)
[unique=true];name[unique=true]
;type(code);sortableType(code);currency[default=false];localized[default=false];multiValue[defaul
t=false];facet[default=true];facetType(code);facetSort(code);priority;visible;useForSpellchecking[
default=false];useForAutocomplete[default=false];fieldValueProvider;facetDisplayNameProvider;
customFacetSortProvider;topValuesProvider;rangeSets(name)

                    ;$solrIndexedType;          reviewAvgRatingRange;double ;   ;
;          ;          ;true          ;MultiSelectOr    ;Alpha    ;2000     ;true;
;  ;rangedAvgCustomerRatingProvider;;;;hybrisAvgRating
```

```
# Create the queries that will be used to extract data for Solr
INSERT_UPDATE SolrIndexerQuery;solrIndexedType(identifier)
[unique=true];identifier[unique=true];type(code);injectCurrentDate[default=true];injectCurrentTim
e[default=true];injectLastIndexTime[default=true];query;user(uid)
;$solrIndexedType;$solrIndexedType-fullQuery;full;;;false;"SELECT {PK} FROM
{Product}";anonymous
;$solrIndexedType;$solrIndexedType-updateQuery;update;;;;"SELECT DISTINCT tbl.pk, tbl.code
FROM (
   {{
      SELECT DISTINCT {p:PK} AS pk, {p:code} AS code
      FROM {Product AS p LEFT JOIN CustomerReview AS cr ON {cr:product}={p:PK} }
      WHERE {p:varianttype} IS NULL AND ({p:modifiedtime} >= ?lastIndexTime OR
{cr:modifiedtime} >= ?lastIndexTime)
   }}
   UNION
   {{
      SELECT DISTINCT {p:PK} AS pk, {p:code} AS code
      FROM {VariantProduct AS p JOIN Product AS bp1 ON {p:baseProduct}={bp1:PK} LEFT
JOIN CustomerReview AS cr ON {cr:product}={bp1:PK} }
      WHERE {p:varianttype} IS NULL AND ({bp1:modifiedtime} >= ?lastIndexTime OR
{cr:modifiedtime} >= ?lastIndexTime)
   }}
   UNION
   {{
      SELECT DISTINCT {p:PK} AS pk, {p:code} AS code
      FROM {VariantProduct AS p JOIN VariantProduct AS bp1 ON {p:baseProduct}={bp1:PK}
JOIN Product AS bp2 ON {bp1:baseProduct}={bp2:PK} LEFT JOIN CustomerReview AS cr ON
{cr:product}={bp2:PK} }
      WHERE {p:varianttype} IS NULL AND ({bp2:modifiedtime} >= ?lastIndexTime OR
{cr:modifiedtime} >= ?lastIndexTime)
   }}
) tbl ORDER BY tbl.code";anonymous

# Define the available sorts
INSERT_UPDATE SolrSort;&sortRefID;indexedType(identifier)
[unique=true];code[unique=true];useBoost
;sortRef0;$solrIndexedType;stockAvailability;false
;sortRef1;$solrIndexedType;relevance;true
;sortRef2;$solrIndexedType;topRated;false
;sortRef3;$solrIndexedType;name-asc;false
;sortRef4;$solrIndexedType;name-desc;false
;sortRef5;$solrIndexedType;price-asc;false
;sortRef6;$solrIndexedType;price-desc;false

# Define the sort fields
INSERT_UPDATE SolrSortField;sort(indexedType(identifier),code)
[unique=true];fieldName[unique=true];ascending[unique=true]
;$solrIndexedType:relevance;inStockFlag;false
;$solrIndexedType:relevance;score;false
;$solrIndexedType:topRated;inStockFlag;false
;$solrIndexedType:topRated;reviewAvgRating;false
;$solrIndexedType:name-asc;name;true
```

```
;$solrIndexedType:name-desc;name;false
;$solrIndexedType:price-asc;lowestBundlePriceValue;true
;$solrIndexedType:price-desc;lowestBundlePriceValue;false
```

solr_en.impex

```
#
# Import the Solr configuration for the Merchandise store
#

# Index Type
$solrIndexedType=b2ctelcoProductType

# Language
$lang=en

# Solr Indexed Property
UPDATE SolrIndexedProperty;solrIndexedType(identifier)
[unique=true];name[unique=true];displayName[lang=$lang]


 ;$solrIndexedType;reviewAvgRatingRange;"Reviews"

# Define the available sorts
UPDATE SolrSort;indexedType(identifier)[unique=true];code[unique=true];name[lang=$lang]

            ;$solrIndexedType;topRated;"Top Rated"
```

Implement Value Provider for the new Index Property:

The **commerceservices** extension has a value provider for the product's average customer review score:

Com.lycamobile.ProductReviewAverageRatingValueProvider

It is used to index a property called reviewAvgRating.
Our goal is to index reviewAvgRatingRange - meaning we want to index not the exact review score, which is a floating point value, but the name of the range that the review score falls within.

We can extend from the existing ProductReviewAverageRatingValueProvider.

To do so, we need to create our own extension, which extends **commerceservices**:

Create own Extension

1.) Please call **ant extgen** to create **lycamobilecommerceservices**.

```
extgen:
[input]
    [input] Please choose a template for generation.
    [input] Press [Enter] to use the default value ([yempty],
springmvcstore, flexstore, storetemplate, ycockpit,
fulfilmentprocess, soapspring)
press Enter
    [input]
    [input] Please choose the name of your extension. It has to
start with a letter followed by letters and/or numbers.
    [input] Press [Enter] to use the default value [training]
lycamobilecommerceservices.

    [input]
    [input] Please choose the package name of your extension. It
has to fulfill java package name convention.
    [input] Press [Enter] to use the default value [org.training]
Com.lycamobile
```

2.) Import the extension to your eclipse workspace
3.) Change the extensioninfo.xml file of the new extension to require the commerceservices extension:

merchandisecommerceservices/extensioninfo.xml

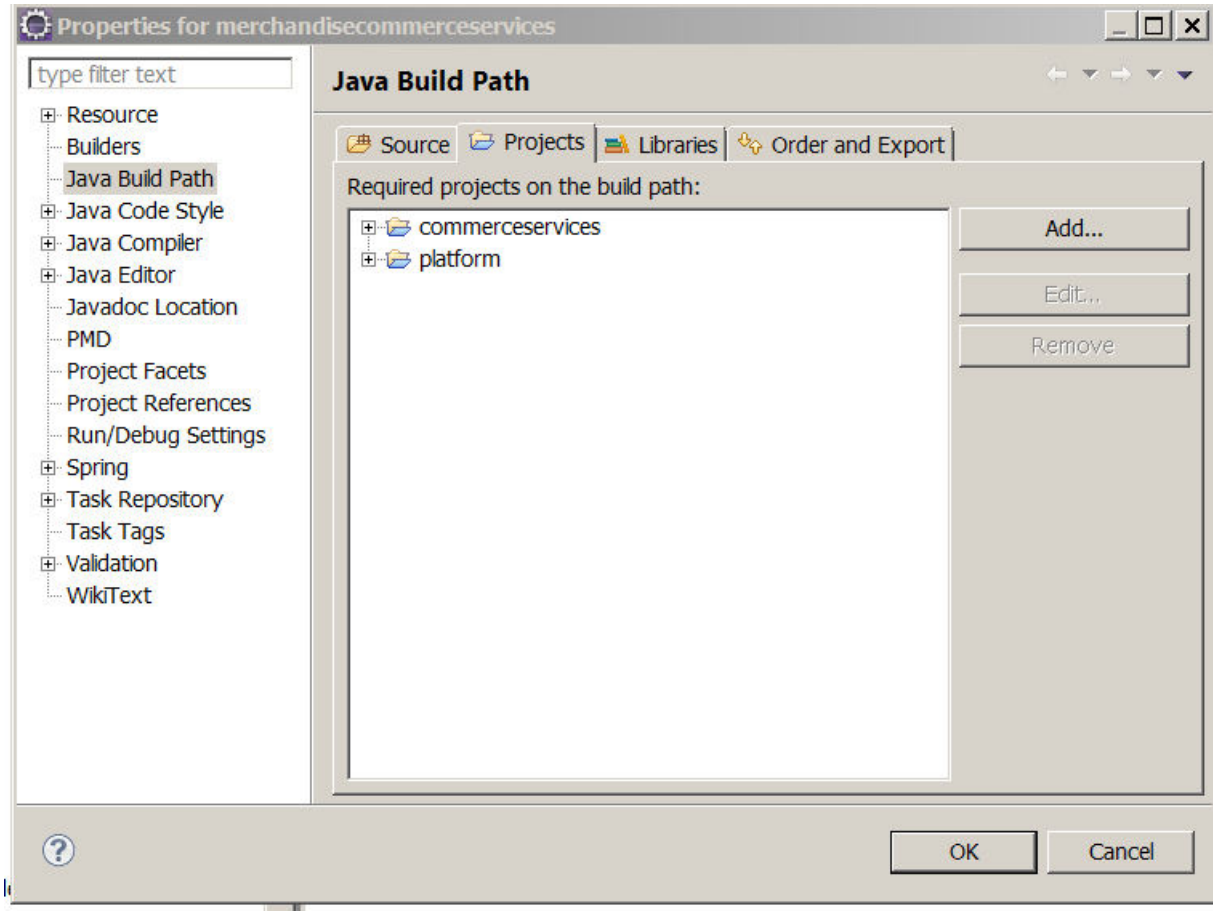
```
<extensioninfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="extensioninfo.xsd">

<extension name="merchandisecommerceservices"
abstractclassprefix="Generated"
classprefix="Merchandisecommerceservices"
managername="MerchandisecommerceservicesManager"
managersuperclass="de.hybris.platform.jalo.extension.Extension"
>

<requires-extension name="commerceservices"/>


<coremodule packageroot="de.hybris.merchandise"
manager="de.hybris.merchandise.jalo.MerchandisecommerceservicesMan
ager"
generated="true" />

</extension>

</extensioninfo>
```

4.) Add the new extension to **localextensions.xml** so it is included in the build and deployment.

config/localextensions.xml

```
...
<extensions>
...
<extension name="lycamobilecommerceservices"/>
...
</extensions>
```

5.) Add commerceservices to the buildpath of our new merchandisecommerceservices extension.



Value Provider Implementation

In this new extension, implement the value provider class for the new index property:

`lycamobilecommerceservices`/src/com/lycamobile/`lycamobile`ProductReviewAverageRatingValueProvider.java

`lycamobile`ProductReviewAverageRatingValueProvider.java

```
/**
 *
 */
package com.lycamobile;

import
de.hybris.platform.commerceservices.search.solrfacetsearch.provider.impl.ProductReviewAv
erageRatingValueProvider;
import de.hybris.platform.core.model.c2l.LanguageModel;
import de.hybris.platform.solrfacetsearch.config.IndexedProperty;
import de.hybris.platform.solrfacetsearch.config.exceptions.FieldValueProviderException;
import de.hybris.platform.solrfacetsearch.provider.FieldValue;

import java.util.Collection;
import java.util.List;

import org.apache.commons.collections.CollectionUtils;
import org.apache.log4j.Logger;




/**
 * @author jagadish
 *
 */
public class LycamobileProductReviewAverageRatingValueProvider extends
ProductReviewAverageRatingValueProvider
{
	private static final Logger LOG =
Logger.getLogger(LycamobileProductReviewAverageRatingValueProvider.class);

	@Override
	protected void addFieldValues(final List<FieldValue> fieldValues, final
IndexedProperty indexedProperty,
			final LanguageModel language, final Object value)
	{
		List<String> rangeNameList = null;
		try
		{
			rangeNameList = getRangeNameList(indexedProperty, value);
		}
		catch (final FieldValueProviderException e)
		{
			LOG.error("Could not get Range value", e);
		}
		String rangeName = null;
		if (CollectionUtils.isNotEmpty(rangeNameList))
		{
			rangeName = rangeNameList.get(0);
		}
		final Collection<String> fieldNames =
```

```
getFieldNameProvider().getFieldNames(indexedProperty,
                                     language == null ? null : language.getIsocode());
                final Object valueToPass = (rangeName == null ? value : rangeName);
                for (final String fieldName : fieldNames)
                {
                        fieldValues.add(new FieldValue(fieldName, valueToPass));
                }
        }

}
```

Now register the new solr value provider **`LycamobileProductReviewAverageRatingValueProvider`** in your new extension's spring configuration.

lycamobilecommerceservices/resources/lycamobilecommerceservices-spring.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:aop="http://www.springframework.org/schema/aop"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">

   <bean id="rangedAvgCustomerRatingProvider"
class="com.lycamobile.LycamobileProductReviewAverageRatingValueProvider"
parent="productReviewAverageRatingValueProvider" />


</beans>
```

Rebuild and Test

1. Rebuild the codebase with **ant all**
2. Run the system update with project data for **b2ctelcostore**. Select Yes for 'Import Core Data' and 'Activate Solr Cron Jobs'.
3. Check these changes in the hmc (System/Facet search/SOLR Item types).
4. Add a few customer reviews to products in the catalog. Select a product and click on 'Write a review' to add product reviews.
5. Go to hmc - Marketing - Product Reviews. Searching for products with Approval Status 'Pending' will display the reviews just added.
6. Open a pending product review and change its approval status to 'Approved'. (Don't forget to save.)

If you complete all these steps, you should be able to Shop by Reviews in the merchandise shop:

It was hard to find where can I add reviews. Catalog -> Products -> Select Product -> Select tab community -> Add Review
we should write reviews in the store home page. go to store homepage–> click on product–>write a review. Then check for the reviews in the hmc.