Merged Opened 2 weeks ago by Angelo Flores

Resolve BROW-2785

All threads resolved

Collapse all threads

Compare and

src/main/java/com/nordstrom/dynamo/tables/BrandTable.java

+24 -0 Viewed

```
+ package com.nordstrom.dynamo.tables;
+
+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
+ import com.nordstrom.indexer.config.AwsConfig;
+ import com.nordstrom.indexer.model.Brand;
+ import com.nordstrom.utils.DateTimeUtils;
+ import org.springframework.beans.factory.annotation.Autowired;
+ import org.springframework.stereotype.Component;
+ import org.springframework.web.context.annotation.RequestScope;
+
+ /**
+  * Brand table inheritance of HashKeyTable.
+  */
+ @Component
+ public class BrandTable extends HashKeyTable<Integer, Brand> {
+
+   /**
+    * C'tor.
+    */
+   @Autowired
+     public BrandTable(AwsConfig awsConfig, AmazonDynamoDB dynamoDb, DateTimeUtils dateTimeUtils) {
+     super(awsConfig, dynamoDb, dateTimeUtils, "brandname-brandid", Brand.class);
+   }
+ }
```

src/main/java/com/nordstrom/dynamo/tables/HashKeyTable.java

+241 -0 Viewed

```
+ package com.nordstrom.dynamo.tables;
+
+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.Builder;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.TableNameOverride;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBScanExpression;
+ import com.amazonaws.services.dynamodbv2.datamodeling.ScanResultPage;
+ import com.nordstrom.indexer.config.AwsConfig;
+ import com.nordstrom.utils.DateTimeUtils;
+ import com.nordstrom.utils.interfaces.HasEndDateTime;
+ import com.nordstrom.utils.interfaces.HasExpirationTime;
+ import com.nordstrom.utils.interfaces.HasModified;
+ import java.util.List;
+ import org.eclipse.jetty.util.StringUtil;
+
+ /**
+  * Abstract class for hash key dynamo tables.
+  *
+  * @param <HashKeyT> Hash key type of the typle (primary key).
+  * @param <ItemT> object type of DB.
+  */
+ public abstract class HashKeyTable<HashKeyT, ItemT> {
+   protected final AmazonDynamoDB dynamoDb;
+   protected final DateTimeUtils dateTimeUtils;
+   protected DynamoDBMapper mapper;
+   protected DynamoDBMapperConfig mapperConfig;
+   protected final Class<ItemT> itemType;
+   private final boolean hasModified;
+   private final boolean hasEndDateTime;
+   private final boolean hasExpirationTime;
+   final String hashKeyError  = "Hash key is null";
+   final String rangeKeyError  = "Range key is null";
+
+   /**
+    * Abstract class for hash key dynamo tables.
+    *
+    * @param dynamoDb specifies dynamo db client.
+    * @param itemType specifies item class type.
+    */
+   public HashKeyTable(
+             AmazonDynamoDB dynamoDb,
+             DateTimeUtils dateTimeUtils,
+             Class<ItemT> itemType
+   ) {
+     this.dynamoDb = dynamoDb;
+     this.dateTimeUtils = dateTimeUtils;
+     this.itemType = itemType;
+     this.hasModified = HasModified.class.isAssignableFrom(itemType);
+     this.hasEndDateTime = HasEndDateTime.class.isAssignableFrom(itemType);
+     this.hasExpirationTime = HasExpirationTime.class.isAssignableFrom(itemType);
+   }
+
+   /**
+    * C'tor for hash key table using standard {app}-{env}-{name} convention.
+    *
+    * @param dynamoDb specifies dynamo db client.
```

```
+     */
+    public HashKeyTable(
+            AwsConfig awsConfig,
+            AmazonDynamoDB dynamoDb,
+            DateTimeUtils dateTimeUtils,
+            String tableIdentifier,
+            Class<ItemT> itemType
+    ) {
+      this(dynamoDb, dateTimeUtils, itemType);
+
+      String tableName = String.format(
+            "%s-%s-%s",
+            "product",
+            awsConfig.getEnvironment(),
+            tableIdentifier
+      );
+
+      this.initialize(tableName);
+    }
+
+    /**
+     * Method gets an item from table by hash key.
+     *
+     * @param hashKey specifies hash key of the item object.
+     * @param rangeKey specifies range key of the item object.
+     * @return Item object.
+     * @throws IllegalArgumentException in case of illegal arguments.
+     */
+    public ItemT get(HashKeyT hashKey, String rangeKey) throws IllegalArgumentException {
+      return mapper.load(itemType, hashKey, rangeKey, this.mapperConfig);
+    }
+
+    /**
+     * Method gets an item from table by hash key.
+     *
+     * @param hashKey specifies hash key of the item object.
+     * @param rangeKey specifies range key of the item object.
+     * @return Item object.
+     * @throws IllegalArgumentException in case of illegal arguments.
+     */
+    public ItemT get(HashKeyT hashKey, Integer rangeKey) throws IllegalArgumentException {
+      return mapper.load(itemType, hashKey, rangeKey, this.mapperConfig);
+    }
+
+    /**
+     * Gets all items from table.
+     *
+     * @return List of objects.
+     */
+    public List<ItemT> getAll() {
+      return paginatedScan(new DynamoDBScanExpression());
+    }
+
+    /**
+     * Method saves an item.
+     *
+     * @param item specifies the item object to save.
+     * @throws IllegalArgumentException in case of illegal arguments.
+     */
+    public void save(ItemT item, String modified) throws IllegalArgumentException {
+      if (item == null) {
+        throw new IllegalArgumentException("Item object is null");
+      }
+      if (this.hasModified) {
+        ((HasModified) item).setModified(modified);
+      }
+      if (this.hasExpirationTime && this.hasEndDateTime) {
+        HasEndDateTime itemWithEndDateTime = (HasEndDateTime) item;
+        HasExpirationTime itemWithExpirationTime = (HasExpirationTime) item;
+
+        String endDateTime = itemWithEndDateTime.getEndDateTime();
+        long expirationTime = DateTimeUtils.calculateExpirationTime(endDateTime);
+
+        itemWithExpirationTime.setExpirationTime(expirationTime);
+      }
+
+      this.mapper.save(item, this.mapperConfig);
+    }
+
+    public void save(ItemT item) throws IllegalArgumentException {
+      save(item, dateTimeUtils.getDateTimeNowUtcAsString());
+    }
+
+    /**
+     * Method deletes an item by hash key.
+     *
+     * @param hashKey specifies hash key of the item object.
+     * @param rangeKey specifies range key of the item object.
+     * @throws IllegalArgumentException in case of illegal arguments.
+     */
+    public void delete(HashKeyT hashKey, String rangeKey) throws IllegalArgumentException {
+      if (hashKey == null) {
+        throw new IllegalArgumentException(hashKeyError);
+      }
+      if (StringUtil.isBlank(rangeKey)) {
+        throw new IllegalArgumentException(rangeKeyError);
+      }
+
+      ItemT record = get(hashKey, rangeKey);
```

```
158 +
159 +     this.delete(record);
160 +   }
161 +
162 +   /**
163 +    * Method deletes an item by hash key.
164 +    *
165 +    * @param hashKey specifies hash key of the item object.
166 +    * @param rangeKey specifies range key of the item object.
167 +    * @throws IllegalArgumentException in case of illegal arguments.
168 +    */
169 +   public void delete(HashKeyT hashKey, Integer rangeKey) throws IllegalArgumentException {
170 +     if (hashKey == null) {
171 +       throw new IllegalArgumentException(hashKeyError);
172 +     }
173 +     if (rangeKey == null) {
174 +       throw new IllegalArgumentException(rangeKeyError);
175 +     }
176 +
177 +     ItemT record = get(hashKey, rangeKey);
178 +
179 +     this.delete(record);
180 +   }
181 +
182 +   /**
183 +    * Method deletes an item by object.
184 +    *
185 +    * @param item specifies the item to delete.
186 +    * @throws IllegalArgumentException in case of illegal arguments.
187 +    */
188 +   protected void delete(ItemT item) throws IllegalArgumentException {
189 +     if (item == null) {
190 +       throw new IllegalArgumentException("Item object is null");
191 +     }
192 +
193 +     this.mapper.delete(item, this.mapperConfig);
194 +   }
195 +
196 +   /**
197 +    * Used for testing.
198 +    *
199 +    * @param mapper mock set.
200 +    */
201 +   public void setMapper(DynamoDBMapper mapper) {
202 +     this.mapper = mapper;
203 +   }
204 +
205 +   /**
206 +    * Scans an entire table with a scan expression.
207 +    * To be used on large tables with multiple pages.
208 +    *
209 +    * @param scanExpression Specifies a scan expression.
210 +    * @return List of objects.
211 +    */
212 +   protected List<ItemT> paginatedScan(DynamoDBScanExpression scanExpression) {
213 +     ScanResultPage<ItemT> value = this.mapper.scanPage(
214 +         this.itemType, scanExpression, this.mapperConfig
215 +     );
216 +     List<ItemT> values;
217 +
218 +     if (value.getLastEvaluatedKey() != null) {
219 +       value = this.mapper.scanPage(
220 +           this.itemType,
221 +           scanExpression.withExclusiveStartKey(value.getLastEvaluatedKey()),
222 +           this.mapperConfig
223 +       );
224 +       values = value.getResults();
225 +     } else {
226 +       values = value.getResults();
227 +     }
228 +
229 +     return values;
230 +   }
231 +
232 +   /**
233 +    * Initializes the table object.
234 +    */
235 +   protected void initialize(String tableName) {
236 +     Builder builder = new DynamoDBMapperConfig.Builder();
237 +     builder.setTableNameOverride(new TableNameOverride(tableName));
238 +     this.mapperConfig = builder.build();
239 +     this.mapper = new DynamoDBMapper(dynamoDb);
240 +   }
241 + }
```

src/main/java/com/nordstrom/dynamo/tables/StyleTable.java +24 -0 Viewed

```
+ package com.nordstrom.dynamo.tables;
+
+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
+ import com.nordstrom.indexer.config.AwsConfig;
+ import com.nordstrom.indexer.model.Style;
+ import com.nordstrom.utils.DateTimeUtils;
+ import org.springframework.beans.factory.annotation.Autowired;
+ import org.springframework.stereotype.Component;
+ import org.springframework.web.context.annotation.RequestScope;
+
+ /**
+  * Brand table inheritance of HashKeyTable.
+  */
+ @Component
+ public class StyleTable extends HashKeyTable<String, Style> {
+
+   /**
+    * C'tor.
+    */
+   @Autowired
+   public StyleTable(AwsConfig awsConfig, AmazonDynamoDB dynamoDb, DateTimeUtils dateTimeUtils) {
+     super(awsConfig, dynamoDb, dateTimeUtils, "stylegroup-styleid", Style.class);
+   }
+ }
```

src/main/java/com/nordstrom/indexer/config/ApplicationConfig.java +4 -33 Viewed

```
  package com.nordstrom.indexer.config;

- import com.nordstrom.indexer.model.Market;
- import org.springframework.beans.factory.annotation.Autowired;
  import org.springframework.beans.factory.annotation.Value;
  import org.springframework.context.annotation.Configuration;

+ /**
+  * Configuration for application data.
+  */
  @Configuration
  public class ApplicationConfig {
-
-   @Value("${ENVIRONMENT}")
+   @Value("${nordstrom.aws.env}")
    private String environment;

-   @Value("${scope}")
-   private String scope;
-
-   private final KafkaConfig kafkaConfig;
-
    @Value("${logging.loggingApplicationId}")
    private String loggingApplicationId;
```

↧ Show 20 lines ↕ Show all unchanged lines ↥ Show 20 lines

```
    @Value("${logging.loggingApplicationType}")
    private String loggingApplicationType;

-   @Value("${market}")
-   private Market market;
-
-   @Autowired
-   public ApplicationConfig(final KafkaConfig kafkaConfig) {
-     this.kafkaConfig = kafkaConfig;
-   }
-
    public String getEnvironment() {
      return environment;
    }

-   public String getScope() {
-     return scope;
-   }
-
-   public String getKafkaSourceTopic() {
-     return this.kafkaConfig.getSourceTopic();
-   }
-
-   public String getKafkaGroupId() {
-     return kafkaConfig.getGroupId();
-   }
-
    public String getLoggingApplicationId() {
      return loggingApplicationId;
    }
```

↧ Show 20 lines ↕ Show all unchanged lines ↥ Show 20 lines

```
    public String getLoggingApplicationType() {
      return loggingApplicationType;
    }

-   public Market getMarket() {
-     return market;
-   }
  }
```

src/main/java/com/nordstrom/indexer/config/AwsConfig.java +78 -0 Viewed

```
+ package com.nordstrom.indexer.config;
+
+ import com.amazonaws.auth.AWSCredentialsProvider;
+ import com.amazonaws.auth.AWSCredentialsProviderChain;
+ import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
+ import com.amazonaws.auth.ProcessCredentialsProvider;
+ import com.amazonaws.auth.profile.ProfileCredentialsProvider;
+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
+ import org.springframework.boot.context.properties.ConfigurationProperties;
+ import org.springframework.stereotype.Component;
+
+ /**
+  * Configuration for AWS account data.
+  */
+ @Component
+ @ConfigurationProperties(prefix = "nordstrom.aws")
+ public class AwsConfig {
+   private AmazonDynamoDB amazonDynamoDb;
+   private String env;
+   private String region;
+   private String profileCredentialsProviderCommand;
+   private String profileCredentialsProvider;
+
+   public AmazonDynamoDB getAmazonDynamoDb() {
+     return amazonDynamoDb;
+   }
+
+   /**
+    * Configuring Amazon connection after prop files are filled.
+    */
+   public void setAmazonDynamoDb() {
+     AWSCredentialsProvider awsCredentials = new AWSCredentialsProviderChain(
+         ProcessCredentialsProvider.builder()
+         .withCommand(getProfileCredentialsProviderCommand()).build(),
+         new ProfileCredentialsProvider(getProfileCredentialsProvider()),
+         new DefaultAWSCredentialsProviderChain()
+     );
+
+     this.amazonDynamoDb = AmazonDynamoDBClientBuilder.standard()
+         .withCredentials(awsCredentials)
+         .withRegion(getRegion())
+         .build();
+   }
+
+   public String getEnvironment() {
+     return env;
+   }
+
+   public void setEnv(String env) {
+     this.env = env;
+   }
+
+   public String getRegion() {
+     return region;
+   }
+
+   public void setRegion(String region) {
+     this.region = region;
+   }
+
+   public String getProfileCredentialsProviderCommand() {
+     return profileCredentialsProviderCommand;
+   }
+
+   public void setProfileCredentialsProviderCommand(String profileCredentialsProviderCommand) {
+     this.profileCredentialsProviderCommand = profileCredentialsProviderCommand;
+   }
+
+   public String getProfileCredentialsProvider() {
+     return profileCredentialsProvider;
+   }
+
+   public void setProfileCredentialsProvider(String profileCredentialsProvider) {
+     this.profileCredentialsProvider = profileCredentialsProvider;
+   }
+ }
+
```

src/main/java/com/nordstrom/indexer/config/KafkaConfig.java deleted +0 -31 Viewed

```java
package com.nordstrom.indexer.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

@Configuration
@ConfigurationProperties(prefix = "kafka")
public class KafkaConfig {

  private String sourceTopic;

  @Value("${spring.kafka.consumer.group-id}")
  private String groupId;

  public String getSourceTopic() {
    return sourceTopic;
  }

  public void setSourceTopic(final String sourceTopic) {
    this.sourceTopic = sourceTopic;
  }

  public String getGroupId() {
    return groupId;
  }

  public void setGroupId(String groupId) {
    this.groupId = groupId;
  }
}
```

src/main/java/com/nordstrom/indexer/config/MetricsConfig.java +3 -0 Viewed

```java
package com.nordstrom.indexer.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

/**
 * Configuration for Metrics.
 */
@Configuration
@ConfigurationProperties("statsd")
public class MetricsConfig {

  @Value("${STATSD_HOST:localhost}")
  private String host;

  @Value("${STATSD_PORT:8128}")
  private int port;

  @Value("${POD_NAME:oid-indexer}")
  private String podName;

  private String prefix;

  public String getHost() {
    return host;
  }

  public void setHost(String host) {
    this.host = host;
  }

  public int getPort() {
    return port;
  }

  public void setPort(int port) {
    this.port = port;
  }

  public String getPodName() {
    return podName;
  }

  public void setPodName(String podName) {
    this.podName = podName;
  }

  public String getPrefix() {
    return prefix;
  }

  public void setPrefix(String prefix) {
    this.prefix = prefix;
  }

}
```

src/main/java/com/nordstrom/indexer/consumer/EventProcessingException.java +8 -0 Viewed

```
+ package com.nordstrom.indexer.consumer;
+
+ class EventProcessingException extends RuntimeException {
+
+   public EventProcessingException(final Throwable cause) {
+     super(cause);
+   }
+ }
```

src/main/java/com/nordstrom/indexer/consumer/OisStreamConsumer.java +48 -14 Viewed

```
  package com.nordstrom.indexer.consumer;

+ import com.nordstrom.indexer.processor.Processor;
+ import com.nordstrom.indexer.processor.ProcessorFactory;
+ import com.nordstrom.offer.StyleOffer;
  import com.nordstrom.sharedlib.logging.ApplicationLogger;
-
+ import com.nordstrom.utils.ValidationHelper;
  import java.util.List;
-
- import org.apache.avro.generic.GenericData;
+ import org.apache.commons.lang3.time.StopWatch;
  import org.apache.kafka.clients.consumer.ConsumerRecord;
  import org.springframework.beans.factory.annotation.Autowired;
  import org.springframework.kafka.annotation.KafkaListener;
  import org.springframework.stereotype.Service;

+ /**
+  * Kafka Consumer for Ois Topic.
+  */
  @Service
  public class OisStreamConsumer {

    private final ApplicationLogger logger;
+   private final ProcessorFactory processorFactory;
+
+   private final String topicName = "ois-rolling-us-avro";

    /**
-    * Creates a consumer to read messages from the OIS stream.
-    * @param logger logger to log messages
+    * C'tor.
     */
    @Autowired
    public OisStreamConsumer(
-       final ApplicationLogger logger
-   ) {
+       final ApplicationLogger logger,
+       ProcessorFactory processorFactory) {
      this.logger = logger;
+     this.processorFactory = processorFactory;
    }

-
    /**
-    * Processes a batch of messages from the OIS stream.
-    * @param consumerRecords generic representation of the kafka messages
+    * Processes stream records.
     */
-   @KafkaListener(topics = {"${kafka.sourceTopic}"}, containerFactory = "batchConsumerFactory", groupId =
"${spring.kafka.consumer.group-id}")
-   public void process(List<ConsumerRecord<String, GenericData.Record>> consumerRecords) {
-     // Do nothing
+   @KafkaListener(
+       topics = {"${spring.kafka.sourceTopic}"}, containerFactory = "batchConsumerFactory",
+       groupId = "${spring.kafka.consumer.group-id}"
+   )
+   public void process(List<ConsumerRecord<String, StyleOffer>> consumerRecords) {
+     StopWatch stopWatch = new StopWatch();
+     stopWatch.start();
+     if (consumerRecords == null) {
+       logger.warn(String.format("%s has served a null record, and will not be processed.",
+           topicName));
+     } else {
+       stopWatch.split();
+       logger.warn(String.format("Processing OIS EventRecords completed in %s ns. Batch size=%s."
+           + "Filtered batch size=%s",
+           stopWatch.getSplitTime(), consumerRecords.size(), consumerRecords.size()));
+
+       try {
+         consumerRecords.parallelStream().forEach(this::processRecord);
+       } catch (Exception exception) {
+         logger.error(exception, String.format("Failed to process records %s.", consumerRecords));
+         throw new EventProcessingException(exception);
+       } finally {
+         stopWatch.stop();
+       }
+     }
    }

+   protected void processRecord(ConsumerRecord<String, StyleOffer> consumerRecord) {
+     ValidationHelper.validateArgumentForNull(consumerRecord);
+
+     Processor processor = processorFactory.getProcessor(consumerRecord);
+     processor.execute(consumerRecord);
+   }
  }
```

src/main/java/com/nordstrom/indexer/dependency/KafkaBootstrapper.java +11 -6 Viewed

```
1  1   package com.nordstrom.indexer.dependency;
2  2
3      - import org.apache.avro.generic.GenericData;
   3   + import com.nordstrom.offer.StyleOffer;
4  4   import org.apache.kafka.common.serialization.StringDeserializer;
5  5   import org.springframework.boot.autoconfigure.kafka.KafkaProperties;
6  6   import org.springframework.context.annotation.Bean;
7  7   import org.springframework.context.annotation.Configuration;
8  8   import org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;
9  9   import org.springframework.kafka.core.ConsumerFactory;
10 10  import org.springframework.kafka.core.DefaultKafkaConsumerFactory;
11 11
12     - 
   12  + /**
   13  +  * Bootstrap to connect Kafka Consumer.
   14  +  */
13 15  @Configuration
14 16  public class KafkaBootstrapper {
15 17
16 18    private final KafkaProperties kafkaProperties;
17 19
18 20    public KafkaBootstrapper(final KafkaProperties kafkaProperties) {
19 21      this.kafkaProperties = kafkaProperties;
20 22    }
21 23
22 24    @Bean(name = "batchConsumerFactory")
23     -   ConcurrentKafkaListenerContainerFactory<String, GenericData.Record> kafkaListenerContainerFactory(
24     -       ConsumerFactory<String, GenericData.Record> consumerFactory) {
25     -     ConcurrentKafkaListenerContainerFactory<String, GenericData.Record> factory =
   25  +   ConcurrentKafkaListenerContainerFactory<String, StyleOffer> kafkaListenerContainerFactory(
   26  +       ConsumerFactory<String, StyleOffer> consumerFactory) {
   27  +     ConcurrentKafkaListenerContainerFactory<String, StyleOffer> factory =
26 28          new ConcurrentKafkaListenerContainerFactory<>();
27 29      factory.setConsumerFactory(consumerFactory);
28 30      factory.setBatchListener(true);
29 31
30 32      return factory;
31 33    }
32 34
33 35    @Bean(name = "retryConsumerFactory")
34 36    ConcurrentKafkaListenerContainerFactory<String, String> retryKafkaListenerContainerFactory() {
35 37      ConcurrentKafkaListenerContainerFactory<String, String> factory =
36 38          new ConcurrentKafkaListenerContainerFactory<>();
37 39
38 40      factory.setConsumerFactory(new DefaultKafkaConsumerFactory<>(
39     -         kafkaProperties.buildConsumerProperties(), new StringDeserializer(), new StringDeserializer()));
   41  +         kafkaProperties.buildConsumerProperties(),
   42  +         new StringDeserializer(),
   43  +         new StringDeserializer()
   44  +     ));
40 45      factory.setBatchListener(false);
41 46
42 47      return factory;
43 48    }
44 49  }
```

src/main/java/com/nordstrom/indexer/dependency/SharedLibraryBeanBootstrapper.java +3 -1 Viewed

↕ Show all unchanged lines ↥ Show 20 lines

```
4  4   import com.nordstrom.sharedlib.config.LoggingOptions;
5  5   import com.nordstrom.sharedlib.logging.ApplicationLogger;
6  6   import com.nordstrom.sharedlib.logging.JsonLogger;
7      - 
8  7   import org.springframework.context.annotation.Bean;
9  8   import org.springframework.context.annotation.Configuration;
10 9
   10  + /**
   11  +  * Bootstrap connection for logging.
   12  +  */
11 13  @Configuration
12 14  public class SharedLibraryBeanBootstrapper {
13 15
14 16    @Bean
15 17    public ApplicationLogger configureLogger(ApplicationConfig applicationConfig) {
16 18      LoggingOptions options = prepareLoggingOptions(applicationConfig);
17 19      return new JsonLogger(options);
18 20    }
19 21
20 22    private LoggingOptions prepareLoggingOptions(ApplicationConfig applicationConfig) {
21 23      return new LoggingOptions()
22 24          .withEnvironment(applicationConfig.getEnvironment())
23 25          .withApplicationType(applicationConfig.getLoggingApplicationType())
24 26          .withLoggingApplicationId(applicationConfig.getLoggingApplicationId())
25 27          .withLoggingNameSpace(applicationConfig.getLoggingNamespace());
26 28    }
27 29  }
```

src/main/java/com/nordstrom/indexer/model/Brand.java +31 -0 Viewed

```java
+ package com.nordstrom.indexer.model;
+
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
+
+ /**
+  * Brand POJO for item in dynamo.
+  */
+ public class Brand {
+   private Integer brandId;
+   private String brandName;
+
+   @DynamoDBHashKey(attributeName = "brandId")
+   public Integer getBrandId() {
+     return brandId;
+
+   }
+
+   public void setBrandId(Integer brandId) {
+     this.brandId = brandId;
+   }
+
+   @DynamoDBRangeKey(attributeName = "brandName")
+   public String getBrandName() {
+     return brandName;
+   }
+
+   public void setBrandName(String brandName) {
+     this.brandName = brandName;
+   }
+ }
```

src/main/java/com/nordstrom/indexer/model/Market.java deleted +0 -7 Viewed

```java
- package com.nordstrom.indexer.model;
-
- public enum Market {
-     US,
-     CA,
-     NONE
- }
```

src/main/java/com/nordstrom/indexer/model/Style.java +32 -0 Viewed

```java
+ package com.nordstrom.indexer.model;
+
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;
+
+ /**
+  * Style POJO for item in dynamo.
+  */
+ @DynamoDBTable(tableName = "product-dev-stylegroup-styleid")
+ public class Style {
+   private String styleGroupId;
+   private Integer styleId;
+
+   @DynamoDBHashKey(attributeName = "styleGroupId")
+   public String getStyleGroupId() {
+     return styleGroupId;
+   }
+
+   public void setStyleGroupId(String styleGroupId) {
+     this.styleGroupId = styleGroupId;
+   }
+
+   @DynamoDBRangeKey(attributeName = "styleId")
+   public Integer getStyleId() {
+     return styleId;
+   }
+
+   public void setStyleId(Integer styleId) {
+     this.styleId = styleId;
+   }
+ }
```

src/main/java/com/nordstrom/indexer/processor/DeleteProcessor.java +56 -0 Viewed

```java
+ package com.nordstrom.indexer.processor;
+
+ import com.nordstrom.dynamo.tables.BrandTable;
+ import com.nordstrom.dynamo.tables.StyleTable;
+ import com.nordstrom.offer.StyleOffer;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+
+ /**
+  * Processor to do work for deletion from dynamo tables.
+  */
+ public class DeleteProcessor implements Processor {
+   private final ApplicationLogger logger;
+   private StyleTable styleTable;
+   private BrandTable brandTable;
+
+   /**
+    * C'tor.
+    */
+   public DeleteProcessor(
+           ApplicationLogger logger,
+           BrandTable brandTable,
+           StyleTable styleTable
+   ) {
+     this.logger = logger;
+     this.styleTable = styleTable;
+     this.brandTable = brandTable;
+   }
+
+   @Override
+   public void execute(ConsumerRecord<String, StyleOffer> consumerRecord) {
+     var offer = consumerRecord.value();
+     var ids = offer.getIds();
+     var prodAttr = offer.getProductAttributes();
+
+     // Style Data
+     if (ids != null) {
+       var webStyle = ids.getWebStyle().getId();
+       var styleGroup = ids.getRmsStyleGroup().getId();
+
+       deleteFromStyleMap(webStyle.toString(), styleGroup.toString());
+     } else {
+       logger.error("ID data is null.");
+     }
+
+     // Brand Data do with follow-up. Define at grooming.
+   }
+
+   private void deleteFromStyleMap(String styleId, String styleGroup) {
+     styleTable.delete(styleGroup, Integer.valueOf(styleId));
+   }
+
+   private void deleteFromBrandMap(int brandId, String brandName) {
+     brandTable.delete(brandId, brandName);
+   }
+ }
```

src/main/java/com/nordstrom/indexer/processor/Processor.java +11 -0 Viewed

```java
+ package com.nordstrom.indexer.processor;
+
+ import com.nordstrom.offer.StyleOffer;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+
+ /**
+  * Processor interface for dynamo.
+  */
+ public interface Processor {
+   void execute(ConsumerRecord<String, StyleOffer> consumerRecord);
+ }
```

src/main/java/com/nordstrom/indexer/processor/ProcessorFactory.java +65 -0 Viewed

```java
+ package com.nordstrom.indexer.processor;
+
+ import com.nordstrom.dynamo.tables.BrandTable;
+ import com.nordstrom.dynamo.tables.StyleTable;
+ import com.nordstrom.indexer.config.AwsConfig;
+ import com.nordstrom.offer.StyleOffer;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import com.nordstrom.utils.DateTimeUtils;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+ import org.springframework.beans.factory.annotation.Autowired;
+ import org.springframework.stereotype.Service;
+
+ /**
+  * Processor delegator.
+  */
+ @Service
+ public class ProcessorFactory {
+   private final ApplicationLogger logger;
+   private final DeleteProcessor deleteProcessor;
+   private final UpdateProcessor updateProcessor;
+
+   /**
+    * Determining which processor to use.
+    *
+    * @param logger for logging.
+    */
+   @Autowired
+   public ProcessorFactory(ApplicationLogger logger, AwsConfig awsConfig) {
+     this.logger = logger;
+     awsConfig.setAmazonDynamoDb();
+
+     DateTimeUtils dateTimeUtils = new DateTimeUtils();
+     BrandTable brandTable = new BrandTable(awsConfig, awsConfig.getAmazonDynamoDb(), dateTimeUtils);
+     StyleTable styleTable = new StyleTable(awsConfig, awsConfig.getAmazonDynamoDb(), dateTimeUtils);
+
+     this.deleteProcessor = new DeleteProcessor(logger, brandTable, styleTable);
+     this.updateProcessor = new UpdateProcessor(logger, brandTable, styleTable);
+   }
+
+   /**
+    * Returns the matching Processor for the EventAction of the supplied EventRecord.
+    */
+   public Processor getProcessor(ConsumerRecord<String, StyleOffer> consumerRecord) {
+     String recordType = getHeader(consumerRecord);
+     switch (recordType) {
+       case "Deleted" :
+         return deleteProcessor;
+       case "Updated" :
+         return updateProcessor;
+       default:
+         logger.warn("Record is null.");
+         return null;
+     }
+   }
+
+   private String getHeader(ConsumerRecord<String, StyleOffer> consumerRecord) {
+     for (var header : consumerRecord.headers()) {
+       var headerKey = header.key();
+       if ("Type".equals(headerKey)) {
+         return new String(header.value());
+       }
+     }
+     return null;
+   }
+ }
```

src/main/java/com/nordstrom/indexer/processor/UpdateProcessor.java +76 -0 Viewed

```
+ package com.nordstrom.indexer.processor;
+
+ import com.nordstrom.dynamo.tables.BrandTable;
+ import com.nordstrom.dynamo.tables.StyleTable;
+ import com.nordstrom.indexer.model.Brand;
+ import com.nordstrom.indexer.model.Style;
+ import com.nordstrom.offer.StyleOffer;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+
+ /**
+  * Processor to do work for update from dynamo tables.
+  */
+ public class UpdateProcessor implements Processor {
+   private final ApplicationLogger logger;
+   private StyleTable styleTable;
+   private BrandTable brandTable;
+
+   /**
+    * C'tor.
+    */
+   public UpdateProcessor(
+       ApplicationLogger logger,
+       BrandTable brandTable,
+       StyleTable styleTable
+   ) {
+     this.logger = logger;
+     this.styleTable = styleTable;
+     this.brandTable = brandTable;
+   }
+
+   @Override
+   public void execute(ConsumerRecord<String, StyleOffer> consumerRecord) {
+     var offer = consumerRecord.value();
+     var ids = offer.getIds();
+     var prodAttr = offer.getProductAttributes();
+
+     // Style Data
+     if (ids != null) {
+       var webStyle = ids.getWebStyle().getId();
+       var styleGroup = ids.getRmsStyleGroup().getId();
+
+       writeToStyleMap(webStyle.toString(), styleGroup.toString());
+     } else {
+       // Metric for null ID data
+     }
+
+     // Brand Data
+     if (prodAttr != null) {
+       var brand = prodAttr.getBrand();
+       if (brand != null) {
+         writeToBrandMap(brand.getId(), brand.getName().toString());
+       } else {
+         logger.error("Brand data is null.");
+       }
+     } else {
+       logger.warn("Product Attribute is null.");
+     }
+   }
+
+   private void writeToStyleMap(String styleId, String styleGroup) {
+     Style style = new Style();
+     style.setStyleId(Integer.valueOf(styleId));
+     style.setStyleGroupId(styleGroup);
+
+     styleTable.save(style);
+   }
+
+   private void writeToBrandMap(int brandId, String brandName) {
+     Brand brand = new Brand();
+     brand.setBrandId(brandId);
+     brand.setBrandName(brandName);
+
+     brandTable.save(brand);
+   }
+ }
```

src/main/java/com/nordstrom/indexer/Application.java +3 -1 Viewed

↕ Show all unchanged lines ↥ Show 20 lines

```
  import org.springframework.retry.annotation.EnableRetry;
  import org.springframework.scheduling.annotation.EnableScheduling;

+ /**
+  * Application entry point.
+  */
  @SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
  @EnableRetry
  @EnableScheduling
  @EnableKafka
  public class Application {
-
    public static void main(String[] args) {
      SpringApplication.run(Application.class, args);
    }
  }
```

src/main/java/com/nordstrom/sharedlib/config/ApplicationOptions.java +3 -0 Viewed

```
1  1   package com.nordstrom.sharedlib.config;
2  2
   3 + /**
   4 +  * Application Options.
   5 +  */
3  6   public class ApplicationOptions {
4  7
5  8     String environment = "undefinedEnv";
6  9
7  10    public String getEnvironment() {
8  11      return environment;
9  12    }
10 13  }
```

src/main/java/com/nordstrom/sharedlib/config/KafkaConsumerOptions.java deleted +0 -177 Viewed

```
1  -  package com.nordstrom.sharedlib.config;
2  -
3  -  public class KafkaConsumerOptions extends KafkaOptions {
4  -    private static final String DEFAULT_TRUSTSTORE_LOCATION = "/usr/lib/jvm/java-8-openjdk-
   -  amd64/jre/lib/security/cacerts";
5  -    private static final String DEFAULT_TRUSTSTORE_PASSWORD = "changeit";
6  -
7  -    private String groupId;
8  -    private boolean autoCommitEnabled = false;
9  -    private boolean authenticationEnabled = false;
10 -    private int autoCommitIntervalMs = 100000;
11 -    private long pollIntervalMs = 100;
12 -    private int sessionTimeoutMs = 60000;
13 -    private int requestTimeoutMs = 30000;
14 -    private int maxPollRecords = 500;
15 -    private int heartbeatIntervalMs = 6000;
16 -    private String autoOffsetResetConfig = "earliest";
17 -    private boolean sslEnabled = false;
18 -    private String kafkaUsername;
19 -    private String kafkaPassword;
20 -    private String truststoreLocation;
21 -    private String truststorePassword;
22 -
23 -
24 -    public KafkaConsumerOptions withEnvironment(final String enviroment) {
25 -      this.environment = enviroment;
26 -      return this;
27 -    }
28 -
29 -    public KafkaConsumerOptions withBootstrapServer(final String bootstrapServer) {
30 -      this.bootstrapServer = bootstrapServer;
31 -      return this;
32 -    }
33 -
34 -    public KafkaConsumerOptions withTopic(final String topic) {
35 -      this.topic = topic;
36 -      return this;
37 -    }
38 -
39 -    public KafkaConsumerOptions withGroupId(final String groupId) {
40 -      this.groupId = groupId;
41 -      return this;
42 -    }
43 -
44 -    public KafkaConsumerOptions withAutoCommitEnalbed(final boolean autoCommitEnabled) {
45 -      this.autoCommitEnabled = autoCommitEnabled;
46 -      return this;
47 -    }
48 -
49 -    public KafkaConsumerOptions withAutoCommitIntervalMs(final int autoCommitIntervalMs) {
50 -      this.autoCommitIntervalMs = autoCommitIntervalMs;
51 -      return this;
52 -    }
53 -
54 -    public KafkaConsumerOptions withPollIntervalMs(final long pollIntervalMs) {
55 -      this.pollIntervalMs = pollIntervalMs;
56 -      return this;
57 -    }
58 -
59 -    public KafkaConsumerOptions withSessionTimeoutMs(final int sessionTimeoutMs) {
60 -      this.sessionTimeoutMs = sessionTimeoutMs;
61 -      return this;
62 -    }
63 -
64 -    public KafkaConsumerOptions withMaxPollRecords(final int maxPollRecords) {
65 -      this.maxPollRecords = maxPollRecords;
66 -      return this;
67 -    }
68 -
69 -    public KafkaConsumerOptions withRequestTimeoutMs(final int requestTimeoutMs) {
70 -      this.requestTimeoutMs = requestTimeoutMs;
71 -      return this;
72 -    }
73 -
74 -    public KafkaConsumerOptions withHeartbeatIntervalMs(final int heartbeatIntervalMs) {
75 -      this.heartbeatIntervalMs = heartbeatIntervalMs;
76 -      return this;
77 -    }
78 -
79 -    public KafkaConsumerOptions withAutoOffsetResetConfig(final String autoOffsetResetConfig) {
```

```
80      this.autoOffsetResetConfig = autoOffsetResetConfig;
81      return this;
82    }
83
84    /**
85     * @param username The username
86     * @param password The password
87     * @return this KafkaConsumerOptions for chaining.
88     */
89    public KafkaConsumerOptions withAuthentication(final String username, final String password) {
90      this.authenticationEnabled = true;
91      this.kafkaUsername = username;
92      this.kafkaPassword = password;
93      return this;
94    }
95
96    /**
97     * @return this KafkaConsumerOptions for chaining.
98     */
99    public KafkaConsumerOptions withSsl() {
100     this.sslEnabled = true;
101     this.truststoreLocation = DEFAULT_TRUSTSTORE_LOCATION;
102     this.truststorePassword = DEFAULT_TRUSTSTORE_PASSWORD;
103     return this;
104   }
105
106   /**
107    * @param truststoreLocation The truststoreLocation
108    * @param truststorePassword The truststorePassword
109    * @return this KafkaConsumerOptions for chaining.
110    */
111   public KafkaConsumerOptions withSsl(String truststoreLocation, String truststorePassword) {
112     this.sslEnabled = true;
113     this.truststoreLocation = truststoreLocation;
114     this.truststorePassword = truststorePassword;
115     return this;
116   }
117
118   public String getGroupId() {
119     return groupId;
120   }
121
122   public boolean isAutoCommitEnabled() {
123     return autoCommitEnabled;
124   }
125
126   public int getAutoCommitIntervalMs() {
127     return autoCommitIntervalMs;
128   }
129
130   public long getPollIntervalMs() {
131     return pollIntervalMs;
132   }
133
134   public int getSessionTimeoutMs() {
135     return sessionTimeoutMs;
136   }
137
138   public int getMaxPollRecords() {
139     return maxPollRecords;
140   }
141
142   public int getRequestTimeoutMs() {
143     return requestTimeoutMs;
144   }
145
146   public int getHeartbeatIntervalMs() {
147     return heartbeatIntervalMs;
148   }
149
150   public String getAutoOffsetResetConfig() {
151     return autoOffsetResetConfig;
152   }
153
154   public boolean getSslEnabled() {
155     return sslEnabled;
156   }
157
158   public boolean isAuthenticationEnabled() {
159     return authenticationEnabled;
160   }
161
162   public String getKafkaUsername() {
163     return kafkaUsername;
164   }
165
166   public String getKafkaPassword() {
167     return kafkaPassword;
168   }
169
170   public String getTruststoreLocation() {
171     return truststoreLocation;
172   }
173
174   public String getTruststorePassword() {
175     return truststorePassword;
176   }
177 }
```

src/main/java/com/nordstrom/sharedlib/config/KafkaOptions.java deleted +0 -15 Viewed

```
1  - package com.nordstrom.sharedlib.config;
2  -
3  - public class KafkaOptions extends ApplicationOptions {
4  -
5  -   String bootstrapServer;
6  -   String topic;
7  -
8  -   public String getKafkaTopic() {
9  -     return topic;
10 -   }
11 -
12 -   public String getBootstrapServer() {
13 -     return bootstrapServer;
14 -   }
15 - }
```

src/main/java/com/nordstrom/sharedlib/config/LoggingOptions.java +3 -5 Viewed

```
1  1   package com.nordstrom.sharedlib.config;
2  2
   3 + /**
   4 +  * Logging options.
   5 +  */
3  6   public class LoggingOptions extends ApplicationOptions {
4  7
5  8     private String loggingApplicationId  = "undefinedApplicationID";
6  9     private String loggingNameSpace = "undefinedLambdaNameSpace";
7  10    private String loggingAppTypeDefault = "streamConsumer";
8  11    private String market = "US";
9  12
10 13    public LoggingOptions withEnvironment(String enviroment) {
11 14      this.environment = enviroment;
12 15      return this;
13 16    }
14 17
15 18    public LoggingOptions withLoggingApplicationId(String loggingApplicationId) {
16 19      this.loggingApplicationId = loggingApplicationId;
17 20      return this;
18 21    }
19 22
20 23    public LoggingOptions withLoggingNameSpace(String loggingNameSpace) {
21 24      this.loggingNameSpace = loggingNameSpace;
22 25      return this;
23 26    }
24 27
25 28    public LoggingOptions withApplicationType(String loggingApplicationType) {
26 29      this.loggingAppTypeDefault = loggingApplicationType;
27 30      return this;
28 31    }
29 32
30 -    public LoggingOptions withMarket(String market) {
31 -      this.market = market;
32 -      return this;
33 -    }
34 -
35 33    public String getLoggingApplicationId() {
36 34      return loggingApplicationId;
37 35    }
38 36
39 37    public String getLoggingNameSpace() {
40 38      return loggingNameSpace;
41 39    }
42 40
43 41    public String getLoggingApplicationType() {
44 42      return loggingAppTypeDefault;
45 43    }
46 44  }
```

src/main/java/com/nordstrom/utils/interfaces/HasEndDateTime.java +10 -0 Viewed

```
1  + package com.nordstrom.utils.interfaces;
2  +
3  + /**
4  +  * HasEndDateTime Interface.
5  +  */
6  + public interface HasEndDateTime {
7  +   void setEndDateTime(String var1);
8  +
9  +   String getEndDateTime();
10 + }
```

src/main/java/com/nordstrom/utils/interfaces/HasExpirationTime.java +10 -0 Viewed

```
1  + package com.nordstrom.utils.interfaces;
2  +
3  + /**
4  +  * HasExpirationTime Interface.
5  +  */
6  + public interface HasExpirationTime {
7  +   void setExpirationTime(Long var1);
8  +
9  +   Long getExpirationTime();
10 + }
```

src/main/java/com/nordstrom/utils/interfaces/HasModified.java +10 -0 Viewed

```
+ package com.nordstrom.utils.interfaces;
+
+ /**
+  * HasModified Interface.
+  */
+ public interface HasModified {
+   void setModified(String var1);
+
+   String getModified();
+ }
```

src/main/java/com/nordstrom/utils/DateTimeUtils.java +46 -0 Viewed

```
+ package com.nordstrom.utils;
+
+ import java.text.DateFormat;
+ import java.text.ParseException;
+ import java.text.SimpleDateFormat;
+ import java.time.Clock;
+ import java.time.Instant;
+ import java.time.ZoneOffset;
+ import java.time.ZonedDateTime;
+ import java.time.format.DateTimeFormatter;
+ import java.util.Date;
+ import java.util.TimeZone;
+ import org.apache.commons.lang3.StringUtils;
+ import org.springframework.stereotype.Component;
+
+ /**
+  * Deals with date and time by ISO 8601.
+  */
+ @Component
+ public class DateTimeUtils {
+   public static final String DATETIME_FORMAT = "yyyy-MM-dd'T'HH:mm:ssX";
+   private static final DateTimeFormatter DTF = DateTimeFormatter
+       .ofPattern(DATETIME_FORMAT);
+   private static final long DEFAULT_EXPIRATION_TIME_OFFSET_DAYS = 7L;
+
+   public ZonedDateTime getDateTimeNowUtc() {
+     return ZonedDateTime.now(Clock.systemUTC());
+   }
+
+   public String dateTimeToString(ZonedDateTime dateTime) {
+     return dateTime.format(DTF);
+   }
+
+   public String getDateTimeNowUtcAsString() {
+     return dateTimeToString(getDateTimeNowUtc());
+   }
+
+   /**
+    * Given a base date expressed as a String, add an offset and return it as an epoch long.
+    */
+   public static long calculateExpirationTime(String baseDate) {
+     ZonedDateTime zonedBaseDate = ZonedDateTime.parse(baseDate, DTF);
+     return zonedBaseDate.plusDays(DEFAULT_EXPIRATION_TIME_OFFSET_DAYS).toEpochSecond();
+   }
+ }
+
```

src/main/java/com/nordstrom/utils/ValidationHelper.java +15 -0 Viewed

```
+ package com.nordstrom.utils;
+
+ public class ValidationHelper {
+
+   /**
+    * @param argument The argument value.
+    * @throws IllegalArgumentException An IllegalArgumentException.
+    */
+   public static void validateArgumentForNull(Object argument)
+       throws IllegalArgumentException {
+     if (argument == null) {
+       throw new IllegalArgumentException("The consumer record argument cannot be null.");
+     }
+   }
+ }
```

src/main/resources/META-INF/additional-spring-configuration-metadata.json +8 -0 Viewed

src/main/resources/application-ca-dev.yml +4 -1 Viewed

src/main/resources/application-ca-prod.yml +8 -1 Viewed

src/main/resources/application-ca.yml +1 -0 Viewed

src/main/resources/application-us-dev.yml +4 -1 Viewed

src/main/resources/application-us-prod.yml +8 -1 Viewed

src/main/resources/application-us.yml +1 -3 Viewed

src/main/resources/application.yml +13 -5 Viewed

src/test/java/com/nordstrom/dynamo/tables/HashKeyTableTest.java +196 -0 Viewed

```
+ package com.nordstrom.dynamo.tables;
+
+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBDeleteExpression;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.TableNameOverride;
+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.Builder;
+ import com.amazonaws.services.dynamodbv2.datamodeling.ScanResultPage;
+ import com.amazonaws.services.dynamodbv2.model.AttributeValue;
+ import com.nordstrom.indexer.config.AwsConfig;
+ import com.nordstrom.indexer.model.Brand;
+ import com.nordstrom.indexer.model.Style;
+ import com.nordstrom.utils.DateTimeUtils;
+ import java.util.Map;
+ import junit.framework.TestCase;
+ import static org.mockito.Matchers.any;
+ import static org.mockito.Mockito.doNothing;
+ import static org.mockito.Mockito.times;
+ import static org.mockito.Mockito.verify;
+ import static org.mockito.Mockito.when;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import org.mockito.Mock;
+ import org.mockito.runners.MockitoJUnitRunner;
+ import java.util.List;
+ import org.mockito.stubbing.Answer;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class HashKeyTableTest extends TestCase {
+   private final String TEST_NAME = "testName";
+   private final String TEST_GROUP = "testGroup";
+   private final Integer TEST_ID = 123;
+
+   @Mock
+   DynamoDBMapper mapper;
+   @Mock
+   AwsConfig awsConfig;
+   @Mock
+   AmazonDynamoDB amazonDynamoDB;
+   @Mock
+   DateTimeUtils dateTimeUtils;
+   @Mock
+   ScanResultPage scanResultPage;
+
+   private BrandTable brandTable;
+   private StyleTable styleTable;
+   private Brand brand;
+   private Style style;
+
+   @Before
+   public void before() {
+     brand = new Brand();
+     style = new Style();
+
+     brandTable = new BrandTable(awsConfig, amazonDynamoDB, dateTimeUtils);
+     brandTable.setMapper(mapper);
+     styleTable = new StyleTable(awsConfig, amazonDynamoDB, dateTimeUtils);
+     styleTable.setMapper(mapper);
+
+     Builder builder = new DynamoDBMapperConfig.Builder();
+     builder.setTableNameOverride(new TableNameOverride(""));
+   }
+
+   @Test
+   public void brandGetTest() {
+     when(mapper.scanPage(any(), any(), any()))
+         .then((Answer<?>) invocation -> {
+           brand.setBrandId(TEST_ID);
+           brand.setBrandName(TEST_NAME);
+           scanResultPage =  new ScanResultPage();
+           scanResultPage.setResults(List.of(brand));
+
+           return scanResultPage;
+         });
+
+     List<Brand> result = brandTable.getAll();
+     verify(mapper, times(1)).scanPage(any(), any(), any());
+     assertEquals(result.get(0).getBrandId(), TEST_ID);
+     assertEquals(result.get(0).getBrandName(), TEST_NAME);
+   }
+
+   @Test
+   public void styleGetTest() {
+     when(mapper.scanPage(any(), any(), any()))
+         .then((Answer<?>) invocation -> {
+           style.setStyleId(TEST_ID);
+           style.setStyleGroupId(TEST_GROUP);
+           scanResultPage =  new ScanResultPage();
```

```
91  +           scanResultPage.setResults(List.of(style));
92  +
93  +           return scanResultPage;
94  +         });
95  +
96  +     List<Style> result = styleTable.getAll();
97  +     verify(mapper, times(1)).scanPage(any(), any(), any());
98  +     assertEquals(result.get(0).getStyleId(), TEST_ID);
99  +     assertEquals(result.get(0).getStyleGroupId(), TEST_GROUP);
100 +   }
101 +
102 +   @Test
103 +   public void styleGetWithExclusiveStartKeyTest() {
104 +     when(mapper.scanPage(any(), any(), any()))
105 +       .then((Answer<?>) invocation -> {
106 +         style.setStyleId(TEST_ID);
107 +         style.setStyleGroupId(TEST_GROUP);
108 +         scanResultPage =  new ScanResultPage();
109 +         scanResultPage.setLastEvaluatedKey(Map.of(TEST_GROUP, new
    AttributeValue().withN(TEST_ID.toString())));
110 +         scanResultPage.setResults(List.of(style));
111 +
112 +         return scanResultPage;
113 +       });
114 +
115 +     List<Style> result = styleTable.getAll();
116 +     verify(mapper, times(2)).scanPage(any(), any(), any());
117 +     assertEquals(result.get(0).getStyleId(), TEST_ID);
118 +     assertEquals(result.get(0).getStyleGroupId(), TEST_GROUP);
119 +   }
120 +
121 +   @Test
122 +   public void brandSaveTest() {
123 +     doNothing().when(mapper).save(brand);
124 +
125 +     brandTable.save(brand);
126 +     verify(mapper, times(1)).save(any(), (DynamoDBMapperConfig) any());
127 +   }
128 +
129 +   @Test(expected = IllegalArgumentException.class)
130 +   public void brandSaveNullItemTest() {
131 +     brandTable.save(null);
132 +   }
133 +
134 +   @Test
135 +   public void styleSaveTest() {
136 +     doNothing().when(mapper).save(style);
137 +
138 +     styleTable.save(style);
139 +     verify(mapper, times(1)).save(any(), (DynamoDBMapperConfig) any());
140 +   }
141 +
142 +   @Test(expected = IllegalArgumentException.class)
143 +   public void styleSaveNullItemTest() {
144 +     styleTable.save(null);
145 +   }
146 +
147 +   @Test
148 +   public void brandDeleteTest() {
149 +     doNothing().when(mapper).delete(any(), (DynamoDBDeleteExpression) any());
150 +     when(mapper.load(any(), any(), any(), any()))
151 +         .then((Answer<Brand>) invocation -> brand);
152 +
153 +     brandTable.delete(TEST_ID, TEST_NAME);
154 +     verify(mapper, times(1)).load(any(), any(), any(), any());
155 +   }
156 +
157 +   @Test(expected = IllegalArgumentException.class)
158 +   public void brandDeleteNullPartitionKeyTest() {
159 +     brandTable.delete(null, TEST_ID);
160 +   }
161 +
162 +   @Test(expected = IllegalArgumentException.class)
163 +   public void brandDeleteNullSortKeyTest() {
164 +     brandTable.delete(TEST_ID, (Integer) null);
165 +   }
166 +
167 +   @Test(expected = IllegalArgumentException.class)
168 +   public void nullItemBrandDeleteTest() {
169 +     brandTable.delete(null);
170 +   }
171 +
172 +   @Test
173 +   public void styleDeleteTest() {
174 +     doNothing().when(mapper).delete(any(), (DynamoDBDeleteExpression) any());
175 +     when(mapper.load(any(), any(), any(), any()))
176 +         .then((Answer<Style>) invocation -> style);
177 +
178 +     styleTable.delete(TEST_GROUP, TEST_ID);
179 +     verify(mapper, times(1)).load(any(), any(), any(), any());
180 +   }
181 +
182 +   @Test(expected = IllegalArgumentException.class)
183 +   public void styleDeleteNullPartitionKeyTest() {
184 +     styleTable.delete(null, TEST_NAME);
185 +   }
186 +
187 +   @Test(expected = IllegalArgumentException.class)
188 +   public void styleDeleteBlankSortKeyTest() {
```

```
189 +      styleTable.delete(TEST_GROUP, "");
190 +    }
191 +
192 +    @Test(expected = IllegalArgumentException.class)
193 +    public void nullItemStyleDeleteTest() {
194 +      styleTable.delete(null);
195 +    }
196 + }
    \ No newline at end of file
```

src/test/java/com/nordstrom/indexer/config/AwsConfigTest.java +54 -0 Viewed

```
1  + package com.nordstrom.indexer.config;
2  +
3  + import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
4  + import junit.framework.TestCase;
5  + import static org.hamcrest.core.IsInstanceOf.instanceOf;
6  + import static org.junit.Assert.assertThat;
7  + import org.junit.Before;
8  + import org.junit.Test;
9  + import org.junit.runner.RunWith;
10 + import org.mockito.runners.MockitoJUnitRunner;
11 +
12 + @RunWith(MockitoJUnitRunner.class)
13 + public class AwsConfigTest extends TestCase {
14 +   private final String testCredsProvider = "testCredProvider";
15 +   private final String testCredsCommand = "testCredProviderCmd";
16 +   private final String region = "us-west-2";
17 +   private final String env = "dev";
18 +   private AwsConfig awsConfig;
19 +
20 +   @Before
21 +   public void before() {
22 +     awsConfig = new AwsConfig();
23 +   }
24 +
25 +   @Test
26 +   public void amazonDynamoDbTest() {
27 +     awsConfig.setProfileCredentialsProviderCommand(testCredsCommand);
28 +     awsConfig.setRegion(region);
29 +     awsConfig.setAmazonDynamoDb();
30 +
31 +     AmazonDynamoDB result = awsConfig.getAmazonDynamoDb();
32 +
33 +     assertNotNull(result);
34 +     assertThat(result, instanceOf(AmazonDynamoDB.class));
35 +   }
36 +
37 +   @Test
38 +   public void environmentTest() {
39 +     awsConfig.setEnv(env);
40 +     String result = awsConfig.getEnvironment();
41 +
42 +     assertNotNull(result);
43 +     assertEquals(env, result);
44 +   }
45 +
46 +   @Test
47 +   public void profileCredentialsProviderTest() {
48 +     awsConfig.setProfileCredentialsProvider(testCredsProvider);
49 +     String result = awsConfig.getProfileCredentialsProvider();
50 +
51 +     assertNotNull(result);
52 +     assertEquals(testCredsProvider, result);
53 +   }
54 + }
    \ No newline at end of file
```

src/test/java/com/nordstrom/indexer/consumer/OisStreamConsumerTest.java +76 -0 Viewed

```
+ package com.nordstrom.indexer.consumer;
+
+ import com.nordstrom.indexer.processor.DeleteProcessor;
+ import com.nordstrom.indexer.processor.ProcessorFactory;
+ import com.nordstrom.indexer.processor.UpdateProcessor;
+ import com.nordstrom.offer.StyleOffer;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import java.util.List;
+ import junit.framework.TestCase;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import static org.mockito.Matchers.any;
+ import static org.mockito.Matchers.anyString;
+ import org.mockito.Mock;
+ import static org.mockito.Mockito.doNothing;
+ import static org.mockito.Mockito.times;
+ import static org.mockito.Mockito.verify;
+ import static org.mockito.Mockito.when;
+ import org.mockito.runners.MockitoJUnitRunner;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class OisStreamConsumerTest extends TestCase {
+   private OisStreamConsumer consumer;
+
+   @Mock
+   private ApplicationLogger applicationLogger;
+   @Mock
+   private ProcessorFactory processorFactory;
+   @Mock
+   private ConsumerRecord<String, StyleOffer> consumerRecord1;
+   @Mock
+   private ConsumerRecord<String, StyleOffer> consumerRecord2;
+   @Mock
+   private UpdateProcessor updateProcessor;
+   @Mock
+   private DeleteProcessor deleteProcessor;
+
+   @Before
+   public void before() {
+     consumer = new OisStreamConsumer(applicationLogger, processorFactory);
+   }
+
+   @Test
+   public void processUpdateTest() {
+     when(processorFactory.getProcessor(any())).thenAnswer(invocation -> updateProcessor);
+     doNothing().when(updateProcessor).execute(any());
+     consumer.process(List.of(consumerRecord1, consumerRecord2));
+
+     verify(processorFactory, times(2)).getProcessor(any());
+     verify(updateProcessor, times(2)).execute(any());
+   }
+
+   @Test
+   public void processDeleteTest() {
+     when(processorFactory.getProcessor(any())).thenAnswer(invocation -> deleteProcessor);
+     doNothing().when(deleteProcessor).execute(any());
+     consumer.process(List.of(consumerRecord1, consumerRecord2));
+
+     verify(processorFactory, times(2)).getProcessor(any());
+     verify(deleteProcessor, times(2)).execute(any());
+   }
+
+   @Test
+   public void nullConsumerRecordsTest() {
+     consumer.process(null);
+     verify(applicationLogger, times(1)).warn(anyString());
+     verify(processorFactory, times(0)).getProcessor(any());
+   }
+
+   @Test(expected = EventProcessingException.class)
+   public void eventProcessingExceptionTest() {
+     consumer.process(List.of(consumerRecord1, consumerRecord2));
+   }
+ }
\ No newline at end of file
```

src/test/java/com/nordstrom/indexer/dependency/KafkaBootstrapperTest.java +51 -0 Viewed

```
+ package com.nordstrom.indexer.dependency;
+
+ import com.nordstrom.offer.StyleOffer;
+ import java.util.Map;
+ import junit.framework.TestCase;
+ import static org.hamcrest.core.IsInstanceOf.instanceOf;
+ import static org.junit.Assert.assertThat;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import org.mockito.Mock;
+ import static org.mockito.Mockito.when;
+ import org.mockito.runners.MockitoJUnitRunner;
+ import org.springframework.boot.autoconfigure.kafka.KafkaProperties;
+ import org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;
+ import org.springframework.kafka.core.ConsumerFactory;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class KafkaBootstrapperTest extends TestCase {
+   private final String testKey = "key";
+   private final StyleOffer testOffer = new StyleOffer();
+   private KafkaBootstrapper kafkaBootstrapper;
+
+   @Mock
+   KafkaProperties kafkaProperties;
+   @Mock
+   ConsumerFactory<String, StyleOffer> consumerFactory;
+
+   @Before
+   public void before() {
+     kafkaBootstrapper = new KafkaBootstrapper(kafkaProperties);
+   }
+
+   @Test
+   public void kafkaListenerContainerFactoryTest() {
+     ConcurrentKafkaListenerContainerFactory<String, StyleOffer> result;
+     result = kafkaBootstrapper.kafkaListenerContainerFactory(consumerFactory);
+
+     assertThat(result, instanceOf(ConcurrentKafkaListenerContainerFactory.class));
+   }
+
+   @Test
+   public void retryKafkaListenerContainerFactoryTest() {
+     when(kafkaProperties.buildConsumerProperties()).thenAnswer(invocation -> Map.of(testKey, testOffer));
+
+     ConcurrentKafkaListenerContainerFactory<String, String> result;
+     result = kafkaBootstrapper.retryKafkaListenerContainerFactory();
+
+     assertThat(result, instanceOf(ConcurrentKafkaListenerContainerFactory.class));
+   }
+ }
\ No newline at end of file
```

src/test/java/com/nordstrom/indexer/dependency/SharedLibraryBeanBootstrapperTest.java +43 -0 Viewed

```
+ package com.nordstrom.indexer.dependency;
+
+ import com.nordstrom.indexer.config.ApplicationConfig;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import junit.framework.TestCase;
+ import static org.hamcrest.core.IsInstanceOf.instanceOf;
+ import static org.junit.Assert.assertThat;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import org.mockito.Mock;
+ import static org.mockito.Mockito.when;
+ import org.mockito.runners.MockitoJUnitRunner;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class SharedLibraryBeanBootstrapperTest extends TestCase {
+   private final String env = "dev";
+   private final String loggingType = "testLogger";
+   private final String loggingId = "testLoggerId";
+   private final String loggingNamespace = "testNamespace";
+   private SharedLibraryBeanBootstrapper sharedLibraryBeanBootstrapper;
+
+   @Mock
+   ApplicationConfig applicationConfig;
+
+   @Before
+   public void before() {
+     sharedLibraryBeanBootstrapper = new SharedLibraryBeanBootstrapper();
+   }
+
+   @Test
+   public void configureLoggerTest() {
+     ApplicationLogger result;
+
+     when(applicationConfig.getEnvironment()).thenAnswer(invocation -> env);
+     when(applicationConfig.getLoggingApplicationType()).thenAnswer(invocation -> loggingType);
+     when(applicationConfig.getLoggingApplicationId()).thenAnswer(invocation -> loggingId);
+     when(applicationConfig.getLoggingNamespace()).thenAnswer(invocation -> loggingNamespace);
+
+     result = sharedLibraryBeanBootstrapper.configureLogger(applicationConfig);
+     assertThat(result, instanceOf(ApplicationLogger.class));
+   }
+ }
\ No newline at end of file
```

src/test/java/com/nordstrom/indexer/processor/DeleteProcessorTest.java +95 -0 Viewed

```
+ package com.nordstrom.indexer.processor;
+
+ import com.nordstrom.dynamo.tables.BrandTable;
+ import com.nordstrom.dynamo.tables.StyleTable;
+ import com.nordstrom.event.rosettastone.ProductStyle;
+ import com.nordstrom.offer.Brand;
+ import com.nordstrom.offer.StyleIdentifiers;
+ import com.nordstrom.offer.StyleOffer;
+ import com.nordstrom.offer.StyleProductAttributes;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import junit.framework.TestCase;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import static org.mockito.Matchers.any;
+ import static org.mockito.Matchers.anyInt;
+ import static org.mockito.Matchers.anyString;
+ import org.mockito.Mock;
+ import static org.mockito.Mockito.doNothing;
+ import static org.mockito.Mockito.times;
+ import static org.mockito.Mockito.verify;
+ import static org.mockito.Mockito.when;
+ import org.mockito.runners.MockitoJUnitRunner;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class DeleteProcessorTest extends TestCase {
+   private final int testBrandId = 1234567;
+   private StringBuffer testBrandName;
+   private StringBuffer testId;
+   private DeleteProcessor deleteProcessor;
+
+   @Mock
+   private ApplicationLogger logger;
+   @Mock
+   private BrandTable brandTable;
+   @Mock
+   private StyleTable styleTable;
+   @Mock
+   private ConsumerRecord<String, StyleOffer> consumerRecord;
+   @Mock
+   private StyleOffer styleOffer;
+   @Mock
+   private StyleIdentifiers styleIdentifiers;
+   @Mock
+   private StyleProductAttributes styleProductAttributes;
+   @Mock
+   private ProductStyle productStyle;
+   @Mock
+   private Brand brand;
+
+   @Before
+   public void before() {
+     testId = new StringBuffer("1234567");
+     testBrandName = new StringBuffer("testName");
+     deleteProcessor = new DeleteProcessor(logger, brandTable, styleTable);
+   }
+
+   @Test
+   public void executeDeletesStyleTest() {
+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
+     when(styleOffer.getIds()).thenAnswer(invocation -> styleIdentifiers);
+     when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
+     when(styleIdentifiers.getWebStyle()).thenAnswer(invocation -> productStyle);
+     when(styleIdentifiers.getRmsStyleGroup()).thenAnswer(invocation -> productStyle);
+     when(productStyle.getId()).thenAnswer(invocation -> testId);
+
+     doNothing().when(styleTable).delete(any(), anyString());
+     deleteProcessor.execute(consumerRecord);
+
+     verify(styleTable, times(1)).delete(anyString(), anyInt());
+   }
+
+   @Test
+   public void executeDoesNotDeleteBrandTest() {
+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
+     when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
+     when(styleProductAttributes.getBrand()).thenAnswer(invocation -> brand);
+     when(brand.getId()).thenAnswer(invocation -> testBrandId);
+     when(brand.getName()).thenAnswer(invocation -> testBrandName);
+
+     doNothing().when(brandTable).delete(any(), anyInt());
+     deleteProcessor.execute(consumerRecord);
+
+     verify(brandTable, times(0)).delete(anyInt(), anyString());
+   }
+
+   @Test
+   public void nullStyleIdTest() {
+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
+     deleteProcessor.execute(consumerRecord);
+
+     verify(logger, times(1)).error(anyString());
+   }
+ }
\ No newline at end of file
```

src/test/java/com/nordstrom/indexer/processor/ProcessorFactoryTest.java +67 -0 Viewed

```
+ package com.nordstrom.indexer.processor;
+
+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
+ import com.nordstrom.indexer.config.AwsConfig;
+ import com.nordstrom.offer.StyleOffer;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import junit.framework.TestCase;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+ import org.apache.kafka.common.header.Headers;
+ import org.apache.kafka.common.header.internals.RecordHeader;
+ import org.apache.kafka.common.header.internals.RecordHeaders;
+ import static org.hamcrest.core.IsInstanceOf.instanceOf;
+ import static org.junit.Assert.assertThat;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import org.mockito.Mock;
+ import static org.mockito.Mockito.when;
+ import org.mockito.runners.MockitoJUnitRunner;
+ import org.mockito.stubbing.Answer;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class ProcessorFactoryTest extends TestCase {
+   private final String testKey = "Type";
+   private final String updateValue = "Updated";
+   private final String deleteValue = "Deleted";
+   private Processor processor;
+   private ProcessorFactory processorFactory;
+   private Headers headers;
+
+   @Mock
+   private ApplicationLogger applicationLogger;
+   @Mock
+   private AwsConfig awsConfig;
+   @Mock
+   private AmazonDynamoDB amazonDynamoDB;
+   @Mock
+   private ConsumerRecord<String, StyleOffer> consumerRecord;
+
+   @Before
+   public void before() {
+     headers = new RecordHeaders();
+     processorFactory = new ProcessorFactory(applicationLogger, awsConfig);
+   }
+
+   @Test
+   public void getProcessorUpdateTest() {
+     headers.add(new RecordHeader(testKey,updateValue.getBytes()));
+
+     when(awsConfig.getAmazonDynamoDb()).thenAnswer(invocation -> amazonDynamoDB);
+     when(consumerRecord.headers()).thenAnswer((Answer<Headers>) invocation -> headers);
+     processor = processorFactory.getProcessor(consumerRecord);
+
+     assertThat(processor, instanceOf(UpdateProcessor.class));
+   }
+
+   @Test
+   public void getProcessorDeleteTest() {
+     headers.add(new RecordHeader(testKey,deleteValue.getBytes()));
+
+     when(awsConfig.getAmazonDynamoDb()).thenAnswer(invocation -> amazonDynamoDB);
+     when(consumerRecord.headers()).thenAnswer((Answer<Headers>) invocation -> headers);
+     processor = processorFactory.getProcessor(consumerRecord);
+
+     assertThat(processor, instanceOf(DeleteProcessor.class));
+   }
+ }
\ No newline at end of file
```

src/test/java/com/nordstrom/indexer/processor/UpdateProcessorTest.java +120 -0 Viewed

```
+ package com.nordstrom.indexer.processor;
+
+ import com.nordstrom.dynamo.tables.BrandTable;
+ import com.nordstrom.dynamo.tables.StyleTable;
+ import com.nordstrom.event.rosettastone.ProductStyle;
+ import com.nordstrom.indexer.model.Style;
+ import com.nordstrom.offer.Brand;
+ import com.nordstrom.offer.StyleIdentifiers;
+ import com.nordstrom.offer.StyleOffer;
+ import com.nordstrom.offer.StyleProductAttributes;
+ import com.nordstrom.sharedlib.logging.ApplicationLogger;
+ import junit.framework.TestCase;
+ import org.apache.kafka.clients.consumer.ConsumerRecord;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import org.mockito.ArgumentMatcher;
+ import static org.mockito.Matchers.any;
+ import static org.mockito.Matchers.anyString;
+ import static org.mockito.Matchers.argThat;
+ import org.mockito.Mock;
+ import static org.mockito.Mockito.doNothing;
+ import static org.mockito.Mockito.times;
+ import static org.mockito.Mockito.verify;
+ import static org.mockito.Mockito.when;
+ import org.mockito.runners.MockitoJUnitRunner;
```

```
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class UpdateProcessorTest extends TestCase {
+   private final int testBrandId = 1234567;
+   private StringBuffer testBrandName;
+   private StringBuffer testStyleId;
+   private StringBuffer testStyleGroup;
+   private UpdateProcessor updateProcessor;
+
+   @Mock
+   private ApplicationLogger logger;
+   @Mock
+   private BrandTable brandTable;
+   @Mock
+   private StyleTable styleTable;
+   @Mock
+   private ConsumerRecord<String, StyleOffer> consumerRecord;
+   @Mock
+   private StyleOffer styleOffer;
+   @Mock
+   private StyleIdentifiers styleIdentifiers;
+   @Mock
+   private StyleProductAttributes styleProductAttributes;
+   @Mock
+   private ProductStyle webStyle;
+   @Mock
+   private ProductStyle styleGroup;
+   @Mock
+   private Brand brand;
+
+   @Before
+   public void before() {
+     testStyleId = new StringBuffer("1234567");
+     testStyleGroup = new StringBuffer("1111111");
+     testBrandName = new StringBuffer("testName");
+     updateProcessor = new UpdateProcessor(logger, brandTable, styleTable);
+   }
+
+   @Test
+   public void executeTest() {
+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
+     when(styleOffer.getIds()).thenAnswer(invocation -> styleIdentifiers);
+     when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
+     when(styleIdentifiers.getWebStyle()).thenAnswer(invocation -> webStyle);
+     when(styleIdentifiers.getRmsStyleGroup()).thenAnswer(invocation -> styleGroup);
+     when(webStyle.getId()).thenAnswer(invocation -> testStyleId);
+     when(styleGroup.getId()).thenAnswer(invocation -> testStyleGroup);
+     when(styleProductAttributes.getBrand()).thenAnswer(invocation -> brand);
+     when(brand.getId()).thenAnswer(invocation -> testBrandId);
+     when(brand.getName()).thenAnswer(invocation -> testBrandName);
+
+     doNothing().when(styleTable).save(any());
+     doNothing().when(brandTable).save(any());
+     updateProcessor.execute(consumerRecord);
+
+     verify(styleTable, times(1)).save(any(Style.class));
+     verify(brandTable, times(1)).save(any(com.nordstrom.indexer.model.Brand.class));
+     verify(styleTable, times(1)).save(argThat(new ArgumentMatcher<>() {
+       @Override
+       public boolean matches(Object argument) {
+         var style = (Style) argument;
+         return
+           style.getStyleId() == Integer.parseInt(testStyleId.toString()) &&
+           style.getStyleGroupId().equals(testStyleGroup.toString());
+       }
+     }));
+     verify(brandTable, times(1)).save(argThat(new ArgumentMatcher<>() {
+       @Override
+       public boolean matches(Object argument) {
+         var brand = (com.nordstrom.indexer.model.Brand) argument;
+         return
+           brand.getBrandId() == testBrandId &&
+           brand.getBrandName().equals(testBrandName.toString());
+       }
+     }));
+   }
+
+   @Test
+   public void null_Product_Attributes_Test() {
+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
+     updateProcessor.execute(consumerRecord);
+
+     verify(logger, times(1)).warn(anyString());
+   }
+
+   @Test
+   public void null_Brand_Test() {
+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
+     when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
+     updateProcessor.execute(consumerRecord);
+
+     verify(logger, times(1)).error(anyString());
+   }
+ }
```

src/test/java/com/nordstrom/utils/DateTimeUtilsTest.java +59 -0 Viewed

```
+ package com.nordstrom.utils;
+
+ import java.time.ZonedDateTime;
+ import java.time.format.DateTimeFormatter;
+ import junit.framework.TestCase;
+ import static org.hamcrest.core.IsInstanceOf.instanceOf;
+ import static org.junit.Assert.assertThat;
+ import org.junit.Before;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import org.mockito.runners.MockitoJUnitRunner;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class DateTimeUtilsTest extends TestCase {
+   public static final String DATETIME_FORMAT = "yyyy-MM-dd'T'HH:mm:ssX";
+   private static final DateTimeFormatter DTF = DateTimeFormatter
+     .ofPattern(DATETIME_FORMAT);
+   private final String inputDatetime = "2013-09-29T18:46:19Z";
+
+   private DateTimeUtils dateTimeUtils;
+   private ZonedDateTime zonedBaseDate;
+
+
+   @Before
+   public void before() {
+     dateTimeUtils = new DateTimeUtils();
+     zonedBaseDate = ZonedDateTime.parse(inputDatetime, DTF);
+   }
+
+   @Test
+   public void getDateTimeNowUtcTest() {
+     ZonedDateTime result = dateTimeUtils.getDateTimeNowUtc();
+
+     assertNotNull(result);
+     assertThat(result, instanceOf(ZonedDateTime.class));
+   }
+
+   @Test
+   public void dateTimeToStringTest() {
+     String result = dateTimeUtils.dateTimeToString(zonedBaseDate);
+
+     assertNotNull(result);
+     assertEquals(zonedBaseDate.format(DTF), result);
+   }
+
+   @Test
+   public void getDateTimeNowUtcAsStringTest() {
+     String result = dateTimeUtils.getDateTimeNowUtcAsString();
+     assertNotNull(result);
+   }
+
+   @Test
+   public void calculateExpirationTimeTest() {
+     long expirationTime = DateTimeUtils.calculateExpirationTime(inputDatetime);
+     assertEquals(zonedBaseDate.plusDays(7L).toEpochSecond(), expirationTime);
+   }
+
+
+ }
\ No newline at end of file
```

src/test/java/com/nordstrom/utils/ValidationHelperTest.java +34 -0 Viewed

```
+ package com.nordstrom.utils;
+
+ import com.nordstrom.offer.StyleOffer;
+ import junit.framework.TestCase;
+ import org.junit.Test;
+ import org.junit.runner.RunWith;
+ import org.mockito.runners.MockitoJUnitRunner;
+
+ @RunWith(MockitoJUnitRunner.class)
+ public class ValidationHelperTest extends TestCase {
+   private final String testArg = "testArgument";
+   private ValidationHelper validationHelper;
+
+   @Test
+   public void creationTest() {
+     validationHelper = new ValidationHelper();
+     assertNotNull(validationHelper);
+   }
+
+   @Test(expected = IllegalArgumentException.class)
+   public void nullArgValidateArgumentForNullTest() {
+     ValidationHelper.validateArgumentForNull(null);
+   }
+
+   @Test
+   public void newStyleOfferValidateArgumentForNullTest() {
+     ValidationHelper.validateArgumentForNull(new StyleOffer());
+   }
+
+   @Test(expected = IllegalArgumentException.class)
+   public void allNullValidateArgumentForNullTest() {
+     ValidationHelper.validateArgumentForNull(null);
+   }
+ }
```