

Merged · Opened 2 weeks ago by  [Angelo Flores](#)

Resolve BROW-2785

✓ All threads resolvedCollapse all threads

Compare and

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

+24 -0 Viewed

```
1+ package com.nordstrom.dynamo.tables;
2+
3+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
4+ import com.nordstrom.indexer.config.AwsConfig;
5+ import com.nordstrom.indexer.model.Brand;
6+ import com.nordstrom.utils.DateTimeUtils;
7+ import org.springframework.beans.factory.annotation.Autowired;
8+ import org.springframework.stereotype.Component;
9+ import org.springframework.web.context.annotation.RequestScope;
10+
11+ /**
12+ * Brand table inheritance of HashKeyTable.
13+ */
14+ @Component
15+ public class BrandTable extends HashKeyTable<Integer, Brand> {
16+
17+ /**
18+ * C'tor.
19+ */
20+ @Autowired
21+ public BrandTable(AwsConfig awsConfig, AmazonDynamoDB dynamoDb, DateTimeUtils dateTimeUtils) {
22+     super(awsConfig, dynamoDb, dateTimeUtils, "brandname-brandid", Brand.class);
23+ }
24+ }
```

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

+241 -0 Viewed

```
1+ package com.nordstrom.dynamo.tables;
2+
3+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
4+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
5+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig;
6+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.Builder;
7+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.TableNameOverride;
8+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBScanExpression;
9+ import com.amazonaws.services.dynamodbv2.datamodeling.ScanResultPage;
10+
11+ import com.nordstrom.indexer.config.AwsConfig;
12+ import com.nordstrom.utils.DateTimeUtils;
13+ import com.nordstrom.utils.interfaces.HasEndDateTime;
14+ import com.nordstrom.utils.interfaces.HasExpirationTime;
15+ import com.nordstrom.utils.interfaces.HasModified;
16+ import java.util.List;
17+ import org.eclipse.jetty.util.StringUtil;
18+
19+ /**
20+ * Abstract class for hash key dynamo tables.
21+ *
22+ * @param <HashKeyT> Hash key type of the tuple (primary key).
23+ * @param <ItemT> object type of DB.
24+ */
25+ public abstract class HashKeyTable<HashKeyT, ItemT> {
26+     protected final AmazonDynamoDB dynamoDb;
27+     protected final DateTimeUtils dateTimeUtils;
28+     protected DynamoDBMapper mapper;
29+     protected DynamoDBMapperConfig mapperConfig;
30+     protected final Class<ItemT> itemType;
31+     private final boolean hasModified;
32+     private final boolean hasEndDateTime;
33+     private final boolean hasExpirationTime;
34+     final String hashKeyError = "Hash key is null";
35+     final String rangeKeyError = "Range key is null";
36+
37+ /**
38+ * Abstract class for hash key dynamo tables.
39+ *
40+ * @param dynamoDb specifies dynamo db client.
41+ * @param itemType specifies item class type.
42+ */
43+ public HashKeyTable(
44+     AmazonDynamoDB dynamoDb,
45+     DateTimeUtils dateTimeUtils,
46+     Class<ItemT> itemType
47+ ) {
48+     this.dynamoDb = dynamoDb;
49+     this.dateTimeUtils = dateTimeUtils;
50+     this.itemType = itemType;
51+     this.hasModified = HasModified.class.isAssignableFrom(itemType);
52+     this.hasEndDateTime = HasEndDateTime.class.isAssignableFrom(itemType);
53+     this.hasExpirationTime = HasExpirationTime.class.isAssignableFrom(itemType);
54+ }
55+
56+ /**
57+ * C'tor for hash key table using standard {app}-{env}-{name} convention.
58+ *
59+ * @param dynamoDb specifies dynamo db client.
60+ *
```



```
59  /*
60  * public HashKeyTable(
61  *     AwsConfig awsConfig,
62  *     AmazonDynamoDB dynamoDb,
63  *     DateTimeUtils dateTimeUtils,
64  *     String tableIdentifier,
65  *     Class<ItemT> itemType
66  * ) {
67  *     this(dynamoDb, dateTimeUtils, itemType);
68  *
69  *     String tableName = String.format(
70  *         "%s-%s-%s",
71  *         "product",
72  *         awsConfig.getEnvironment(),
73  *         tableIdentifier
74  *     );
75  *
76  *     this.initialize(tableName);
77  * }
78 *
79 /**
80 * Method gets an item from table by hash key.
81 *
82 * @param hashKey specifies hash key of the item object.
83 * @param rangeKey specifies range key of the item object.
84 * @return Item object.
85 * @throws IllegalArgumentException in case of illegal arguments.
86 */
87 public ItemT get(HashKeyT hashKey, String rangeKey) throws IllegalArgumentException {
88     return mapper.load(itemType, hashKey, rangeKey, this.mapperConfig);
89 }
90
91 /**
92 * Method gets an item from table by hash key.
93 *
94 * @param hashKey specifies hash key of the item object.
95 * @param rangeKey specifies range key of the item object.
96 * @return Item object.
97 * @throws IllegalArgumentException in case of illegal arguments.
98 */
99 public ItemT get(HashKeyT hashKey, Integer rangeKey) throws IllegalArgumentException {
100    return mapper.load(itemType, hashKey, rangeKey, this.mapperConfig);
101 }
102
103 /**
104 * Gets all items from table.
105 *
106 * @return List of objects.
107 */
108 public List<ItemT> getAll() {
109     return paginatedScan(new DynamoDBScanExpression());
110 }
111
112 /**
113 * Method saves an item.
114 *
115 * @param item specifies the item object to save.
116 * @throws IllegalArgumentException in case of illegal arguments.
117 */
118 public void save(ItemT item, String modified) throws IllegalArgumentException {
119     if (item == null) {
120         throw new IllegalArgumentException("Item object is null");
121     }
122     if (this.hasModified) {
123         ((HasModified) item).setModified(modified);
124     }
125     if (this.hasExpirationTime && this.hasEndDateTime) {
126         HasEndDateTime itemWithEndDateTime = (HasEndDateTime) item;
127         HasExpirationTime itemWithExpirationTime = (HasExpirationTime) item;
128
129         String endDateTime = itemWithEndDateTime.getEndDateTime();
130         long expirationTime = DateTimeUtils.calculateExpirationTime(endDateTime);
131
132         itemWithExpirationTime.setExpirationTime(expirationTime);
133     }
134
135     this.mapper.save(item, this.mapperConfig);
136 }
137
138 public void save(ItemT item) throws IllegalArgumentException {
139     save(item, dateTimeUtils.getDateNowUtcAsString());
140 }
141
142 /**
143 * Method deletes an item by hash key.
144 *
145 * @param hashKey specifies hash key of the item object.
146 * @param rangeKey specifies range key of the item object.
147 * @throws IllegalArgumentException in case of illegal arguments.
148 */
149 public void delete(HashKeyT hashKey, String rangeKey) throws IllegalArgumentException {
150     if (hashKey == null) {
151         throw new IllegalArgumentException(hashKeyError);
152     }
153     if (StringUtil.isBlank(rangeKey)) {
154         throw new IllegalArgumentException(rangeKeyError);
155     }
156     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

```

1  + package com.nordstrom.dynamo.tables;
2  +
3  + import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
4  + import com.nordstrom.indexer.config.AwsConfig;
5  + import com.nordstrom.indexer.model.Style;
6  + import com.nordstrom.utils.DateTimeUtils;
7  + import org.springframework.beans.factory.annotation.Autowired;
8  + import org.springframework.stereotype.Component;
9  + import org.springframework.web.context.annotation.RequestScope;
10 +
11 + /**
12 + * Brand table inheritance of HashKeyTable.
13 + */
14 @Component
15 public class StyleTable extends HashKeyTable<String, Style> {
16 +
17 + /**
18 + * C'tor.
19 + */
20 @Autowired
21 public StyleTable(AwsConfig awsConfig, AmazonDynamoDB dynamoDb, DateTimeUtils dateTimeUtils) {
22     super(awsConfig, dynamoDb, dateTimeUtils, "stylegroup-styleid", Style.class);
23 }
24

```

src/main/java/com/nordstrom/indexer/config/ApplicationConfig.java

+4 -33 Viewed

```

1  1 package com.nordstrom.indexer.config;
2  2
3  3 - import com.nordstrom.indexer.model.Market;
4  4 - import org.springframework.beans.factory.annotation.Autowired;
5  5 import org.springframework.beans.factory.annotation.Value;
6  6 import org.springframework.context.annotation.Configuration;
7
8  7 + /**
9  8 + * Configuration for application data.
10 + */
11 @Configuration
12 public class ApplicationConfig {
13
14 -     @Value("${ENVIRONMENT}")
15 -     @Value("${nordstrom.aws.env}")
16     private String environment;
17
18 -     @Value("${scope}")
19 -     private String scope;
20
21 -     private final KafkaConfig kafkaConfig;
22
23 -     @Value("${logging.loggingApplicationId}")
24     private String loggingApplicationId;
25
26     @Value("${logging.loggingApplicationType}")
27     private String loggingApplicationType;
28
29 -     @Value("${market}")
30 -     private Market market;
31
32 -     @Autowired
33 -     public ApplicationConfig(final KafkaConfig kafkaConfig) {
34 -         this.kafkaConfig = kafkaConfig;
35 -     }
36
37     public String getEnvironment() {
38         return environment;
39     }
40
41     public String getScope() {
42         return scope;
43     }
44
45     public String getKafkaSourceTopic() {
46         return this.kafkaConfig.getSourceTopic();
47     }
48
49     public String getKafkaGroupId() {
50         return kafkaConfig.getGroupId();
51     }
52
53     public String getLoggingApplicationId() {
54         return loggingApplicationId;
55     }

```

[Show 20 lines](#) [Show all unchanged lines](#) [Show 20 lines](#)

```

56 -     @Value("${logging.loggingApplicationType}")
57     private String loggingApplicationType;
58
59 -     @Value("${market}")
60 -     private Market market;
61
62 -     @Autowired
63 -     public ApplicationConfig(final KafkaConfig kafkaConfig) {
64 -         this.kafkaConfig = kafkaConfig;
65 -     }
66
67     public String getLoggingApplicationType() {
68         return loggingApplicationType;
69     }
70
71     public Market getMarket() {
72         return market;
73     }
74
75 }

```

[Show 20 lines](#) [Show all unchanged lines](#) [Show 20 lines](#)

```

60  35     public String getLoggingApplicationType() {
61  36         return loggingApplicationType;
62  37     }
63
64 -     public Market getMarket() {
65 -         return market;
66 -     }
67
68 }

```

+78 -0 Viewed

src/main/java/com/nordstrom/indexer/config/AwsConfig.java

```
1+ package com.nordstrom.indexer.config;
2+
3+ import com.amazonaws.auth.AWS CredentialsProvider;
4+ import com.amazonaws.auth.AWS CredentialsProviderChain;
5+ import com.amazonaws.auth.DefaultAWS CredentialsProviderChain;
6+ import com.amazonaws.auth.ProcessCredentialsProvider;
7+ import com.amazonaws.auth.profile.ProfileCredentialsProvider;
8+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
9+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
10+ import org.springframework.boot.context.properties.ConfigurationProperties;
11+ import org.springframework.stereotype.Component;
12+
13+ /**
14+ * Configuration for AWS account data.
15+ */
16+ @Component
17+ @ConfigurationProperties(prefix = "nordstrom.aws")
18+ public class AwsConfig {
19+     private AmazonDynamoDB amazonDynamoDb;
20+     private String env;
21+     private String region;
22+     private String profileCredentialsProviderCommand;
23+     private String profileCredentialsProvider;
24+
25+     public AmazonDynamoDB getAmazonDynamoDb() {
26+         return amazonDynamoDb;
27+     }
28+
29+ /**
30+ * Configuring Amazon connection after prop files are filled.
31+ */
32+ public void setAmazonDynamoDb() {
33+     AWS CredentialsProvider awsCredentials = new AWS CredentialsProviderChain(
34+         ProcessCredentialsProvider.builder()
35+             .withCommand(getProfileCredentialsProviderCommand()).build(),
36+             new ProfileCredentialsProvider(getProfileCredentialsProvider()),
37+             new DefaultAWS CredentialsProviderChain()
38+     );
39+
40+     this.amazonDynamoDb = AmazonDynamoDBClientBuilder.standard()
41+         .withCredentials(awsCredentials)
42+         .withRegion(getRegion())
43+         .build();
44+ }
45+
46+ public String getEnvironment() {
47+     return env;
48+ }
49+
50+ public void setEnv(String env) {
51+     this.env = env;
52+ }
53+
54+ public String getRegion() {
55+     return region;
56+ }
57+
58+ public void setRegion(String region) {
59+     this.region = region;
60+ }
61+
62+ public String getProfileCredentialsProviderCommand() {
63+     return profileCredentialsProviderCommand;
64+ }
65+
66+ public void setProfileCredentialsProviderCommand(String profileCredentialsProviderCommand) {
67+     this.profileCredentialsProviderCommand = profileCredentialsProviderCommand;
68+ }
69+
70+ public String getProfileCredentialsProvider() {
71+     return profileCredentialsProvider;
72+ }
73+
74+ public void setProfileCredentialsProvider(String profileCredentialsProvider) {
75+     this.profileCredentialsProvider = profileCredentialsProvider;
76+ }
77+ }
78+
```

src/main/java/com/nordstrom/indexer/config/KafkaConfig.java deleted

+0 -31 Viewed

```

1 package com.nordstrom.indexer.config;
2
3 import org.springframework.beans.factory.annotation.Value;
4 import org.springframework.boot.context.properties.ConfigurationProperties;
5 import org.springframework.context.annotation.Configuration;
6
7
8 @Configuration
9 @ConfigurationProperties(prefix = "kafka")
10 public class KafkaConfig {
11
12     private String sourceTopic;
13
14     @Value("${spring.kafka.consumer.group-id}")
15     private String groupId;
16
17     public String getSourceTopic() {
18         return sourceTopic;
19     }
20
21     public void setSourceTopic(final String sourceTopic) {
22         this.sourceTopic = sourceTopic;
23     }
24
25     public String getGroupId() {
26         return groupId;
27     }
28
29     public void setGroupId(String groupId) {
30         this.groupId = groupId;
31     }
32 }
```

src/main/java/com/nordstrom/indexer/config/MetricsConfig.java

+3 -0 Viewed

```

1 package com.nordstrom.indexer.config;
2
3 import org.springframework.beans.factory.annotation.Value;
4 import org.springframework.boot.context.properties.ConfigurationProperties;
5 import org.springframework.context.annotation.Configuration;
6
7 /**
8 * Configuration for Metrics.
9 */
10 @Configuration
11 @ConfigurationProperties("statsd")
12 public class MetricsConfig {
13
14     @Value("${STATSD_HOST:localhost}")
15     private String host;
16
17     @Value("${STATSD_PORT:8128}")
18     private int port;
19
20     @Value("${POD_NAME:oid-indexer}")
21     private String podName;
22
23     private String prefix;
24
25     public String getHost() {
26         return host;
27     }
28
29     public void setHost(String host) {
30         this.host = host;
31     }
32
33     public int getPort() {
34         return port;
35     }
36
37     public void setPort(int port) {
38         this.port = port;
39     }
40
41     public String getPodName() {
42         return podName;
43     }
44
45     public void setPodName(String podName) {
46         this.podName = podName;
47     }
48
49     public String getPrefix() {
50         return prefix;
51     }
52
53     public void setPrefix(String prefix) {
54         this.prefix = prefix;
55     }
56 }
```

src/main/java/com/nordstrom/indexer/consumer/EventProcessingException.java

+8 -0 Viewed

```
1 package com.nordstrom.indexer.consumer;
2
3 + class EventProcessingException extends RuntimeException {
4 +
5 +     public EventProcessingException(final Throwable cause) {
6 +         super(cause);
7 +     }
8 + }
```

src/main/java/com/nordstrom/indexer/consumer/OisStreamConsumer.java

+48 -14 Viewed

```
1 package com.nordstrom.indexer.consumer;
2
3 + import com.nordstrom.indexer.processor.Processor;
4 + import com.nordstrom.indexer.processor.ProcessorFactory;
5 + import com.nordstrom.offer.StyleOffer;
6 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
7 + import com.nordstrom.utils.ValidationHelper;
8 + import java.util.List;
9
10 - import org.apache.avro.generic.GenericData;
11 + import org.apache.commons.lang3.time.StopWatch;
12 import org.apache.kafka.clients.consumer.ConsumerRecord;
13 import org.springframework.beans.factory.annotation.Autowired;
14 import org.springframework.kafka.annotation.KafkaListener;
15 import org.springframework.stereotype.Service;
16
17 + /**
18 + * Kafka Consumer for Ois Topic.
19 + */
20 @Service
21 public class OisStreamConsumer {
22
23     - private final ApplicationLogger logger;
24     + private final ProcessorFactory processorFactory;
25
26     + private final String topicName = "ois-rolling-us-avro";
27
28     /**
29     * Creates a consumer to read messages from the OIS stream.
30     * @param logger logger to log messages
31     * C'tor.
32     */
33     @Autowired
34     public OisStreamConsumer(
35         - final ApplicationLogger logger
36         ) {
37         + final ApplicationLogger logger,
38         + ProcessorFactory processorFactory) {
39         this.logger = logger;
40         this.processorFactory = processorFactory;
41     }
42
43     /**
44     * Processes a batch of messages from the OIS stream.
45     * @param consumerRecords generic representation of the kafka messages
46     * Processes stream records.
47     */
48     @KafkaListener(topics = "${kafka.sourceTopic}", containerFactory = "batchConsumerFactory", groupId =
49     "${spring.kafka.consumer.group-id}")
50     public void process(List<ConsumerRecord<String, GenericData.Record>> consumerRecords) {
51         // Do nothing
52
53         @KafkaListener(
54             topics = "${spring.kafka.sourceTopic}", containerFactory = "batchConsumerFactory",
55             groupId = "${spring.kafka.consumer.group-id}"
56         )
57         public void process(List<ConsumerRecord<String, StyleOffer>> consumerRecords) {
58             StopWatch stopWatch = new StopWatch();
59             stopWatch.start();
60             if (consumerRecords == null) {
61                 logger.warn(String.format("%s has served a null record, and will not be processed.",
62                     topicName));
63             } else {
64                 stopWatch.split();
65                 logger.warn(String.format("Processing OIS EventRecords completed in %s ns. Batch size=%s."
66                     + "Filtered batch size=%s",
67                     stopWatch.getSplitTime(), consumerRecords.size(), consumerRecords.size()));
68             }
69             try {
70                 consumerRecords.parallelStream().forEach(this::processRecord);
71             } catch (Exception exception) {
72                 logger.error(exception, String.format("Failed to process records %s.", consumerRecords));
73                 throw new EventProcessingException(exception);
74             } finally {
75                 stopWatch.stop();
76             }
77         }
78
79         protected void processRecord(ConsumerRecord<String, StyleOffer> consumerRecord) {
80             ValidationHelper.validateArgumentForNull(consumerRecord);
81
82             Processor processor = processorFactory.getProcessor(consumerRecord);
83             processor.execute(consumerRecord);
84         }
85     }
86 }
```

src/main/java/com/nordstrom/indexer/dependency/KafkaBootstrapper.java

+11 -6 Viewed

```

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

```

src/main/java/com/nordstrom/indexer/model/Brand.java

+31 -0 Viewed

```

1  + package com.nordstrom.indexer.model;
2  +
3  + import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
4  + import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
5  +
6  + /**
7  * Brand POJO for item in dynamo.
8  */
9  public class Brand {
10 +     private Integer brandId;
11 +     private String brandName;
12 +
13 +     @DynamoDBHashKey(attributeName = "brandId")
14 +     public Integer getBrandId() {
15 +         return brandId;
16 +
17     }
18 +
19 +     public void setBrandId(Integer brandId) {
20 +         this.brandId = brandId;
21     }
22 +
23 +     @DynamoDBRangeKey(attributeName = "brandName")
24 +     public String getBrandName() {
25 +         return brandName;
26     }
27 +
28 +     public void setBrandName(String brandName) {
29 +         this.brandName = brandName;
30     }
31 }
```

src/main/java/com/nordstrom/indexer/model/Market.java deleted

+0 -7 Viewed

```

1  - package com.nordstrom.indexer.model;
2  -
3  - public enum Market {
4  -     US,
5  -     CA,
6  -     NONE
7  }
```

src/main/java/com/nordstrom/indexer/model/Style.java

+32 -0 Viewed

```

1  + package com.nordstrom.indexer.model;
2  +
3  + import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
4  + import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
5  + import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;
6  +
7  + /**
8  * Style POJO for item in dynamo.
9  */
10 + @DynamoDBTable(tableName = "product-dev-stylegroup-styleid")
11 + public class Style {
12 +     private String styleGroupId;
13 +     private Integer styleId;
14 +
15 +     @DynamoDBHashKey(attributeName = "styleGroupId")
16 +     public String getStyleGroupId() {
17 +         return styleGroupId;
18     }
19 +
20 +     public void setStyleGroupId(String styleGroupId) {
21 +         this.styleGroupId = styleGroupId;
22     }
23 +
24 +     @DynamoDBRangeKey(attributeName = "styleId")
25 +     public Integer getStyleId() {
26 +         return styleId;
27     }
28 +
29 +     public void setStyleId(Integer styleId) {
30 +         this.styleId = styleId;
31     }
32 }
```

src/main/java/com/nordstrom/indexer/processor/DeleteProcessor.java

+56 -0 Viewed

```

1 + package com.nordstrom.indexer.processor;
2 +
3 + import com.nordstrom.dynamo.tables.BrandTable;
4 + import com.nordstrom.dynamo.tables.StyleTable;
5 + import com.nordstrom.offer.StyleOffer;
6 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
7 + import org.apache.kafka.clients.consumer.ConsumerRecord;
8 +
9 + /**
10 + * Processor to do work for deletion from dynamo tables.
11 + */
12 + public class DeleteProcessor implements Processor {
13 +     private final ApplicationLogger logger;
14 +     private StyleTable styleTable;
15 +     private BrandTable brandTable;
16 +
17 + /**
18 + * C'tor.
19 + */
20 + public DeleteProcessor(
21 +     ApplicationLogger logger,
22 +     BrandTable brandTable,
23 +     StyleTable styleTable
24 + ) {
25 +     this.logger = logger;
26 +     this.styleTable = styleTable;
27 +     this.brandTable = brandTable;
28 + }
29 +
30 + @Override
31 + public void execute(ConsumerRecord<String, StyleOffer> consumerRecord) {
32 +     var offer = consumerRecord.value();
33 +     var ids = offer.getIds();
34 +     var prodAttr = offer.getProductAttributes();
35 +
36 +     // Style Data
37 +     if (ids != null) {
38 +         var webStyle = ids.getWebStyle().getId();
39 +         var styleGroup = ids.getRmsStyleGroup().getId();
40 +
41 +         deleteFromStyleMap(webStyle.toString(), styleGroup.toString());
42 +     } else {
43 +         logger.error("ID data is null.");
44 +     }
45 +
46 +     // Brand Data do with follow-up. Define at grooming.
47 + }
48 +
49 + private void deleteFromStyleMap(String styleId, String styleGroup) {
50 +     styleTable.delete(styleGroup, Integer.valueOf(styleId));
51 + }
52 +
53 + private void deleteFromBrandMap(int brandId, String brandName) {
54 +     brandTable.delete(brandId, brandName);
55 + }
56 + }

```

src/main/java/com/nordstrom/indexer/processor/Processor.java

+11 -0 Viewed

```

1 + package com.nordstrom.indexer.processor;
2 +
3 + import com.nordstrom.offer.StyleOffer;
4 + import org.apache.kafka.clients.consumer.ConsumerRecord;
5 +
6 + /**
7 + * Processor interface for dynamo.
8 + */
9 + public interface Processor {
10 +     void execute(ConsumerRecord<String, StyleOffer> consumerRecord);
11 + }

```

src/main/java/com/nordstrom/indexer/processor/ProcessorFactory.java

+65 -0 Viewed

```
1 + package com.nordstrom.indexer.processor;
2 +
3 + import com.nordstrom.dynamo.tables.BrandTable;
4 + import com.nordstrom.dynamo.tables.StyleTable;
5 + import com.nordstrom.indexer.config.AwsConfig;
6 + import com.nordstrom.offer.StyleOffer;
7 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
8 + import com.nordstrom.utils.DateTimeUtils;
9 + import org.apache.kafka.clients.consumer.ConsumerRecord;
10 + import org.springframework.beans.factory.annotation.Autowired;
11 + import org.springframework.stereotype.Service;
12 +
13 + /**
14 + * Processor delegator.
15 + */
16 + @Service
17 + public class ProcessorFactory {
18 +     private final ApplicationLogger logger;
19 +     private final DeleteProcessor deleteProcessor;
20 +     private final UpdateProcessor updateProcessor;
21 +
22 +     /**
23 +      * Determining which processor to use.
24 +      *
25 +      * @param logger for logging.
26 +      */
27 +     @Autowired
28 +     public ProcessorFactory(ApplicationLogger logger, AwsConfig awsConfig) {
29 +         this.logger = logger;
30 +         awsConfig.setAmazonDynamoDb();
31 +
32 +         DateTimeUtils dateTimeUtils = new DateTimeUtils();
33 +         BrandTable brandTable = new BrandTable(awsConfig, awsConfig.getAmazonDynamoDb(), dateTimeUtils);
34 +         StyleTable styleTable = new StyleTable(awsConfig, awsConfig.getAmazonDynamoDb(), dateTimeUtils);
35 +
36 +         this.deleteProcessor = new DeleteProcessor(logger, brandTable, styleTable);
37 +         this.updateProcessor = new UpdateProcessor(logger, brandTable, styleTable);
38 +     }
39 +
40 +     /**
41 +      * Returns the matching Processor for the EventAction of the supplied EventRecord.
42 +      */
43 +     public Processor getProcessor(ConsumerRecord<String, StyleOffer> consumerRecord) {
44 +         String recordType = getHeader(consumerRecord);
45 +         switch (recordType) {
46 +             case "Deleted" :
47 +                 return deleteProcessor;
48 +             case "Updated" :
49 +                 return updateProcessor;
50 +             default:
51 +                 logger.warn("Record is null.");
52 +                 return null;
53 +         }
54 +     }
55 +
56 +     private String getHeader(ConsumerRecord<String, StyleOffer> consumerRecord) {
57 +         for (var header : consumerRecord.headers()) {
58 +             var headerKey = header.key();
59 +             if ("Type".equals(headerKey)) {
60 +                 return new String(header.value());
61 +             }
62 +         }
63 +         return null;
64 +     }
65 + }
```

src/main/java/com/nordstrom/indexer/processor/UpdateProcessor.java

+76 -0 Viewed

```

1 + package com.nordstrom.indexer.processor;
2 +
3 + import com.nordstrom.dynamo.tables.BrandTable;
4 + import com.nordstrom.dynamo.tables.StyleTable;
5 + import com.nordstrom.indexer.model.Brand;
6 + import com.nordstrom.indexer.model.Style;
7 + import com.nordstrom.offer.StyleOffer;
8 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
9 + import org.apache.kafka.clients.consumer.ConsumerRecord;
10 +
11 + /**
12 + * Processor to do work for update from dynamo tables.
13 + */
14 + public class UpdateProcessor implements Processor {
15 +     private final ApplicationLogger logger;
16 +     private StyleTable styleTable;
17 +     private BrandTable brandTable;
18 +
19 +     /**
20 +      * C'tor.
21 +     */
22 +     public UpdateProcessor(
23 +         ApplicationLogger logger,
24 +         BrandTable brandTable,
25 +         StyleTable styleTable
26 +     ) {
27 +         this.logger = logger;
28 +         this.styleTable = styleTable;
29 +         this.brandTable = brandTable;
30 +     }
31 +
32 +     @Override
33 +     public void execute(ConsumerRecord<String, StyleOffer> consumerRecord) {
34 +         var offer = consumerRecord.value();
35 +         var ids = offer.getIds();
36 +         var prodAttr = offer.getProductAttributes();
37 +
38 +         // Style Data
39 +         if (ids != null) {
40 +             var webStyle = ids.getWebStyle().getId();
41 +             var styleGroup = ids.getRmsStyleGroup().getId();
42 +
43 +             writeToStyleMap(webStyle.toString(), styleGroup.toString());
44 +         } else {
45 +             // Metric for null ID data
46 +         }
47 +
48 +         // Brand Data
49 +         if (prodAttr != null) {
50 +             var brand = prodAttr.getBrand();
51 +             if (brand != null) {
52 +                 writeToBrandMap(brand.getId(), brand.getName().toString());
53 +             } else {
54 +                 logger.error("Brand data is null.");
55 +             }
56 +         } else {
57 +             logger.warn("Product Attribute is null.");
58 +         }
59 +     }
60 +
61 +     private void writeToStyleMap(String styleId, String styleGroup) {
62 +         Style style = new Style();
63 +         style.setStyleId(Integer.valueOf(styleId));
64 +         style.setStyleGroupId(styleGroup);
65 +
66 +         styleTable.save(style);
67 +     }
68 +
69 +     private void writeToBrandMap(int brandId, String brandName) {
70 +         Brand brand = new Brand();
71 +         brand.setBrandId(brandId);
72 +         brand.setBrandName(brandName);
73 +
74 +         brandTable.save(brand);
75 +     }
76 + }
```



src/main/java/com/nordstrom/indexer/Application.java

+3 -1 Viewed

[>Show all unchanged lines](#) [Show 20 lines](#)

```

7 7 import org.springframework.retry.annotation.EnableRetry;
8 8 import org.springframework.scheduling.annotation.EnableScheduling;
9 9
10+ /**
11+ * Application entry point.
12+ */
13 @SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
14 @EnableRetry
15 @EnableScheduling
16 @EnableKafka
17 public class Application {
18
19     public static void main(String[] args) {
20         SpringApplication.run(Application.class, args);
21     }
22 }
```

src/main/java/com/nordstrom/sharedlib/config/ApplicationOptions.java

+3 -0 Viewed

```
1 package com.nordstrom.sharedlib.config;
2
3 /**
4 * Application Options.
5 */
6 public class ApplicationOptions {
7
8     String environment = "undefinedEnv";
9
10    public String getEnvironment() {
11        return environment;
12    }
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
5     private static final String DEFAULT_TRUSTSTORE_LOCATION = "/usr/lib/jvm/java-8-openjdk-
6         amd64/jre/lib/security/cacerts";
7
8     private static final String DEFAULT_TRUSTSTORE_PASSWORD = "changeit";
9
10    private String groupId;
11    private boolean autoCommitEnabled = false;
12    private boolean authenticationEnabled = false;
13    private int autoCommitIntervalMs = 100000;
14    private long pollIntervalMs = 100;
15    private int sessionTimeoutMs = 60000;
16    private int requestTimeoutMs = 30000;
17    private int maxPollRecords = 500;
18    private int heartbeatIntervalMs = 6000;
19    private String autoOffsetResetConfig = "earliest";
20    private boolean sslEnabled = false;
21    private String kafkaUsername;
22    private String kafkaPassword;
23    private String truststoreLocation;
24    private String truststorePassword;
25
26    public KafkaConsumerOptions withEnvironment(final String enviroment) {
27        this.environment = enviroment;
28        return this;
29    }
30
31    public KafkaConsumerOptions withBootstrapServer(final String bootstrapServer) {
32        this.bootstrapServer = bootstrapServer;
33        return this;
34    }
35
36    public KafkaConsumerOptions withTopic(final String topic) {
37        this.topic = topic;
38        return this;
39    }
40
41    public KafkaConsumerOptions withGroupId(final String groupId) {
42        this.groupId = groupId;
43        return this;
44    }
45
46    public KafkaConsumerOptions withAutoCommitEnalbed(final boolean autoCommitEnabled) {
47        this.autoCommitEnabled = autoCommitEnabled;
48        return this;
49    }
50
51    public KafkaConsumerOptions withAutoCommitIntervalMs(final int autoCommitIntervalMs) {
52        this.autoCommitIntervalMs = autoCommitIntervalMs;
53        return this;
54    }
55
56    public KafkaConsumerOptions withPollIntervalMs(final long pollIntervalMs) {
57        this.pollIntervalMs = pollIntervalMs;
58        return this;
59    }
60
61    public KafkaConsumerOptions withSessionTimeoutMs(final int sessionTimeoutMs) {
62        this.sessionTimeoutMs = sessionTimeoutMs;
63        return this;
64    }
65
66    public KafkaConsumerOptions withMaxPollRecords(final int maxPollRecords) {
67        this.maxPollRecords = maxPollRecords;
68        return this;
69    }
70
71    public KafkaConsumerOptions withRequestTimeoutMs(final int requestTimeoutMs) {
72        this.requestTimeoutMs = requestTimeoutMs;
73        return this;
74    }
75
76    public KafkaConsumerOptions withHeartbeatIntervalMs(final int heartbeatIntervalMs) {
77        this.heartbeatIntervalMs = heartbeatIntervalMs;
78        return this;
79    }
80
81    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 | package com.nordstrom.sharedlib.config;
2 |
3 | + /**
4 | + * Logging options.
5 | + */
6 | public class LoggingOptions extends ApplicationOptions {
7 |
8 |     private String loggingApplicationId = "undefinedApplicationID";
9 |     private String loggingNameSpace = "undefinedLambdaNameSpace";
10 |
11 |     private String loggingAppTypeDefault = "streamConsumer";
12 |     private String market = "US";
13 |
14 |     public LoggingOptions withEnvironment(String environment) {
15 |         this.environment = environment;
16 |         return this;
17 |     }
18 |
19 |     public LoggingOptions withLoggingApplicationId(String loggingApplicationId) {
20 |         this.loggingApplicationId = loggingApplicationId;
21 |         return this;
22 |     }
23 |
24 |     public LoggingOptions withLoggingNameSpace(String loggingNameSpace) {
25 |         this.loggingNameSpace = loggingNameSpace;
26 |         return this;
27 |     }
28 |
29 |     public LoggingOptions withApplicationType(String loggingApplicationType) {
30 |         this.loggingAppTypeDefault = loggingApplicationType;
31 |         return this;
32 |     }
33 |
34 |     public LoggingOptions withMarket(String market) {
35 |         this.market = market;
36 |         return this;
37 |     }
38 |
39 |     public String getLoggingApplicationId() {
40 |         return loggingApplicationId;
41 |     }
42 |
43 |     public String getLoggingNameSpace() {
44 |         return loggingNameSpace;
45 |     }
46 |     public String getLoggingApplicationType() {
47 |         return loggingAppTypeDefault;
48 |     }
49 |
50 |
51 |
52 |
53 |
54 |
55 |
56 |
57 |
58 |
59 |
60 |
61 |
62 |
63 |
64 |
65 |
66 |
67 |
68 |
69 |
70 |
71 |
72 |
73 |
74 |
75 |
76 |
77 |
78 |
79 |
80 |
81 |
82 |
83 |
84 |
85 |
86 |
87 |
88 |
89 |
90 |
91 |
92 |
93 |
94 |
95 |
96 |
97 |
98 |
99 |
100 |
101 |
102 |
103 |
104 |
105 |
106 |
107 |
108 |
109 |
110 |
111 |
112 |
113 |
114 |
115 |
116 |
117 |
118 |
119 |
120 |
121 |
122 |
123 |
124 |
125 |
126 |
127 |
128 |
129 |
130 |
131 |
132 |
133 |
134 |
135 |
136 |
137 |
138 |
139 |
140 |
141 |
142 |
143 |
144 |
145 |
146 |
147 |
148 |
149 |
150 |
151 |
152 |
153 |
154 |
155 |
156 |
157 |
158 |
159 |
160 |
161 |
162 |
163 |
164 |
165 |
166 |
167 |
168 |
169 |
170 |
171 |
172 |
173 |
174 |
175 |
176 |
177 |
178 |
179 |
180 |
181 |
182 |
183 |
184 |
185 |
186 |
187 |
188 |
189 |
190 |
191 |
192 |
193 |
194 |
195 |
196 |
197 |
198 |
199 |
200 |
201 |
202 |
203 |
204 |
205 |
206 |
207 |
208 |
209 |
210 |
211 |
212 |
213 |
214 |
215 |
216 |
217 |
218 |
219 |
220 |
221 |
222 |
223 |
224 |
225 |
226 |
227 |
228 |
229 |
230 |
231 |
232 |
233 |
234 |
235 |
236 |
237 |
238 |
239 |
239 |
240 |
241 |
242 |
243 |
244 |
245 |
246 |
247 |
248 |
249 |
249 |
250 |
251 |
252 |
253 |
254 |
255 |
256 |
257 |
258 |
259 |
259 |
260 |
261 |
262 |
263 |
264 |
265 |
266 |
267 |
268 |
269 |
269 |
270 |
271 |
272 |
273 |
274 |
275 |
276 |
277 |
278 |
279 |
279 |
280 |
281 |
282 |
283 |
284 |
285 |
286 |
287 |
288 |
289 |
289 |
290 |
291 |
292 |
293 |
294 |
295 |
296 |
297 |
298 |
299 |
299 |
300 |
301 |
302 |
303 |
304 |
305 |
306 |
307 |
308 |
309 |
309 |
310 |
311 |
312 |
313 |
314 |
315 |
316 |
317 |
318 |
319 |
319 |
320 |
321 |
322 |
323 |
324 |
325 |
326 |
327 |
328 |
329 |
329 |
330 |
331 |
332 |
333 |
334 |
335 |
336 |
337 |
338 |
339 |
339 |
340 |
341 |
342 |
343 |
344 |
345 |
346 |
347 |
348 |
349 |
349 |
350 |
351 |
352 |
353 |
354 |
355 |
356 |
357 |
358 |
359 |
359 |
360 |
361 |
362 |
363 |
364 |
365 |
366 |
367 |
368 |
369 |
369 |
370 |
371 |
372 |
373 |
374 |
375 |
376 |
377 |
378 |
379 |
379 |
380 |
381 |
382 |
383 |
384 |
385 |
386 |
387 |
388 |
389 |
389 |
390 |
391 |
392 |
393 |
394 |
395 |
396 |
397 |
398 |
399 |
399 |
400 |
401 |
402 |
403 |
404 |
405 |
406 |
407 |
408 |
409 |
409 |
410 |
411 |
412 |
413 |
414 |
415 |
416 |
417 |
418 |
419 |
419 |
420 |
421 |
422 |
423 |
424 |
425 |
426 |
427 |
428 |
429 |
429 |
430 |
431 |
432 |
433 |
434 |
435 |
436 |
437 |
438 |
439 |
439 |
440 |
441 |
442 |
443 |
444 |
445 |
446 |
447 |
448 |
449 |
449 |
450 |
451 |
452 |
453 |
454 |
455 |
456 |
457 |
458 |
459 |
459 |
460 |
461 |
462 |
463 |
464 |
465 |
466 |
467 |
468 |
469 |
469 |
470 |
471 |
472 |
473 |
474 |
475 |
476 |
477 |
478 |
479 |
479 |
480 |
481 |
482 |
483 |
484 |
485 |
486 |
487 |
488 |
489 |
489 |
490 |
491 |
492 |
493 |
494 |
495 |
496 |
497 |
498 |
499 |
499 |
500 |
501 |
502 |
503 |
504 |
505 |
506 |
507 |
508 |
509 |
509 |
510 |
511 |
512 |
513 |
514 |
515 |
516 |
517 |
518 |
519 |
519 |
520 |
521 |
522 |
523 |
524 |
525 |
526 |
527 |
528 |
529 |
529 |
530 |
531 |
532 |
533 |
534 |
535 |
536 |
537 |
538 |
539 |
539 |
540 |
541 |
542 |
543 |
544 |
545 |
546 |
547 |
548 |
549 |
549 |
550 |
551 |
552 |
553 |
554 |
555 |
556 |
557 |
558 |
559 |
559 |
560 |
561 |
562 |
563 |
564 |
565 |
566 |
567 |
568 |
569 |
569 |
570 |
571 |
572 |
573 |
574 |
575 |
576 |
577 |
578 |
579 |
579 |
580 |
581 |
582 |
583 |
584 |
585 |
586 |
587 |
588 |
589 |
589 |
590 |
591 |
592 |
593 |
594 |
595 |
596 |
597 |
598 |
599 |
599 |
600 |
601 |
602 |
603 |
604 |
605 |
606 |
607 |
608 |
609 |
609 |
610 |
611 |
612 |
613 |
614 |
615 |
616 |
617 |
618 |
619 |
619 |
620 |
621 |
622 |
623 |
624 |
625 |
626 |
627 |
628 |
629 |
629 |
630 |
631 |
632 |
633 |
634 |
635 |
636 |
637 |
638 |
639 |
639 |
640 |
641 |
642 |
643 |
644 |
645 |
646 |
647 |
648 |
649 |
649 |
650 |
651 |
652 |
653 |
654 |
655 |
656 |
657 |
658 |
659 |
659 |
660 |
661 |
662 |
663 |
664 |
665 |
666 |
667 |
668 |
669 |
669 |
670 |
671 |
672 |
673 |
674 |
675 |
676 |
677 |
678 |
679 |
679 |
680 |
681 |
682 |
683 |
684 |
685 |
686 |
687 |
688 |
689 |
689 |
690 |
691 |
692 |
693 |
694 |
695 |
696 |
697 |
698 |
698 |
699 |
700 |
701 |
702 |
703 |
704 |
705 |
706 |
707 |
708 |
709 |
709 |
710 |
711 |
712 |
713 |
714 |
715 |
716 |
717 |
718 |
719 |
719 |
720 |
721 |
722 |
723 |
724 |
725 |
726 |
727 |
728 |
729 |
729 |
730 |
731 |
732 |
733 |
734 |
735 |
736 |
737 |
738 |
739 |
739 |
740 |
741 |
742 |
743 |
744 |
745 |
746 |
747 |
748 |
749 |
749 |
750 |
751 |
752 |
753 |
754 |
755 |
756 |
757 |
758 |
759 |
759 |
760 |
761 |
762 |
763 |
764 |
765 |
766 |
767 |
768 |
769 |
769 |
770 |
771 |
772 |
773 |
774 |
775 |
776 |
777 |
778 |
779 |
779 |
780 |
781 |
782 |
783 |
784 |
785 |
786 |
787 |
788 |
788 |
789 |
789 |
790 |
791 |
792 |
793 |
794 |
795 |
796 |
797 |
798 |
798 |
799 |
800 |
801 |
802 |
803 |
804 |
805 |
806 |
807 |
808 |
809 |
809 |
810 |
811 |
812 |
813 |
814 |
815 |
816 |
817 |
818 |
819 |
819 |
820 |
821 |
822 |
823 |
824 |
825 |
826 |
827 |
828 |
829 |
829 |
830 |
831 |
832 |
833 |
834 |
835 |
836 |
837 |
838 |
839 |
839 |
840 |
841 |
842 |
843 |
844 |
845 |
846 |
847 |
848 |
849 |
849 |
850 |
851 |
852 |
853 |
854 |
855 |
856 |
857 |
858 |
859 |
859 |
860 |
861 |
862 |
863 |
864 |
865 |
866 |
867 |
868 |
869 |
869 |
870 |
871 |
872 |
873 |
874 |
875 |
876 |
877 |
878 |
878 |
879 |
880 |
881 |
882 |
883 |
884 |
885 |
886 |
887 |
888 |
888 |
889 |
889 |
890 |
891 |
892 |
893 |
894 |
895 |
896 |
897 |
898 |
898 |
899 |
900 |
901 |
902 |
903 |
904 |
905 |
906 |
907 |
908 |
909 |
909 |
910 |
911 |
912 |
913 |
914 |
915 |
916 |
917 |
918 |
919 |
919 |
920 |
921 |
922 |
923 |
924 |
925 |
926 |
927 |
928 |
929 |
929 |
930 |
931 |
932 |
933 |
934 |
935 |
936 |
937 |
938 |
939 |
939 |
940 |
941 |
942 |
943 |
944 |
945 |
946 |
947 |
948 |
949 |
949 |
950 |
951 |
952 |
953 |
954 |
955 |
956 |
957 |
958 |
959 |
959 |
960 |
961 |
962 |
963 |
964 |
965 |
966 |
967 |
968 |
969 |
969 |
970 |
971 |
972 |
973 |
974 |
975 |
976 |
977 |
978 |
978 |
979 |
980 |
981 |
982 |
983 |
984 |
985 |
986 |
987 |
988 |
988 |
989 |
989 |
990 |
991 |
992 |
993 |
994 |
995 |
995 |
996 |
997 |
997 |
998 |
999 |
999 |
1000 |
1000 |
1001 |
1001 |
1002 |
1002 |
1003 |
1003 |
1004 |
1004 |
1005 |
1005 |
1006 |
1006 |
1007 |
1007 |
1008 |
1008 |
1009 |
1009 |
1010 |
1010 |
1011 |
1011 |
1012 |
1012 |
1013 |
1013 |
1014 |
1014 |
1015 |
1015 |
1016 |
1016 |
1017 |
1017 |
1018 |
1018 |
1019 |
1019 |
1020 |
1020 |
1021 |
1021 |
1022 |
1022 |
1023 |
1023 |
1024 |
1024 |
1025 |
1025 |
1026 |
1026 |
1027 |
1027 |
1028 |
1028 |
1029 |
1029 |
1030 |
1030 |
1031 |
1031 |
1032 |
1032 |
1033 |
1033 |
1034 |
1034 |
1035 |
1035 |
1036 |
1036 |
1037 |
1037 |
1038 |
1038 |
1039 |
1039 |
1040 |
1040 |
1041 |
1041 |
1042 |
1042 |
1043 |
1043 |
1044 |
1044 |
1045 |
1045 |
1046 |
1046 |
1047 |
1047 |
1048 |
1048 |
1049 |
1049 |
1050 |
1050 |
1051 |
1051 |
1052 |
1052 |
1053 |
1053 |
1054 |
1054 |
1055 |
1055 |
1056 |
1056 |
1057 |
1057 |
1058 |
1058 |
1059 |
1059 |
1060 |
1060 |
1061 |
1061 |
1062 |
1062 |
1063 |
1063 |
1064 |
1064 |
1065 |
1065 |
1066 |
1066 |
1067 |
1067 |
1068 |
1068 |
1069 |
1069 |
1070 |
1070 |
1071 |
1071 |
1072 |
1072 |
1073 |
1073 |
1074 |
1074 |
1075 |
1075 |
1076 |
1076 |
1077 |
1077 |
1078 |
1078 |
1079 |
1079 |
1080 |
1080 |
1081 |
1081 |
1082 |
1082 |
1083 |
1083 |
1084 |
1084 |
1085 |
1085 |
1086 |
1086 |
1087 |
1087 |
1088 |
1088 |
1089 |
1089 |
1090 |
1090 |
1091 |
1091 |
1092 |
1092 |
1093 |
1093 |
1094 |
1094 |
1095 |
1095 |
1096 |
1096 |
1097 |
1097 |
1098 |
1098 |
1099 |
1099 |
1100 |
1100 |
1101 |
1101 |
1102 |
1102 |
1103 |
1103 |
1104 |
1104 |
1105 |
1105 |
1106 |
1106 |
1107 |
1107 |
1108 |
1108 |
1109 |
1109 |
1110 |
1110 |
1111 |
1111 |
1112 |
1112 |
1113 |
1113 |
1114 |
1114 |
1115 |
1115 |
1116 |
1116 |
1117 |
1117 |
1118 |
1118 |
1119 |
1119 |
1120 |
1120 |
1121 |
1121 |
1122 |
1122 |
1123 |
1123 |
1124 |
1124 |
1125 |
1125 |
1126 |
1126 |
1127 |
1127 |
1128 |
1128 |
1129 |
1129 |
1130 |
1130 |
1131 |
1131 |
1132 |
1132 |
1133 |
1133 |
1134 |
1134 |
1135 |
1135 |
1136 |
1136 |
1137 |
1137 |
1138 |
1138 |
1139 |
1139 |
1140 |
1140 |
1141 |
1141 |
1142 |
1142 |
1143 |
1143 |
1144 |
1144 |
1145 |
1145 |
1146 |
1146 |
1147 |
1147 |
1148 |
1148 |
1149 |
1149 |
1150 |
1150 |
1151 |
1151 |
1152 |
1152 |
1153 |
1153 |
1154 |
1154 |
1155 |
1155 |
1156 |
1156 |
1157 |
1157 |
1158 |
1158 |
1159 |
1159 |
1160 |
1160 |
1161 |
1161 |
1162 |
1162 |
1163 |
1163 |
1164 |
1164 |
1165 |
1165 |
1166 |
1166 |
1167 |
1167 |
1168 |
1168 |
1169 |
1169 |
1170 |
1170 |
1171 |
1171 |
1172 |
1172 |
1173 |
1173 |
1174 |
1174 |
1175 |
1175 |
1176 |
1176 |
1177 |
1177 |
1178 |
1178 |
1179 |
1179 |
1180 |
1180 |
1181 |
1181 |
1182 |
1182 |
1183 |
1183 |
1184 |
1184 |
1185 |
1185 |
1186 |
1186 |
1187 |
1187 |
1188 |
1188 |
1189 |
1189 |
1190 |
1190 |
1191 |
1191 |
1192 |
1192 |
1193 |
1193 |
1194 |
1194 |
1195 |
1195 |
1196 |
1196 |
1197 |
1197 |
1198 |
1198 |
1199 |
1199 |
1200 |
1200 |
1201 |
1201 |
1202 |
1202 |
1203 |
1203 |
1204 |
1204 |
1205 |
1205 |
1206 |
1206 |
1207 |
1207 |
1208 |
1208 |
1209 |
1209 |
1210 |
1210 |
1211 |
1211 |
1212 |
1212 |
1213 |
1213 |
1214 |
1214 |
1215 |
1215 |
1216 |
1216 |
1217 |
1217 |
1218 |
1218 |
1219 |
1219 |
1220 |
1220 |
1221 |
1221 |
1222 |
1222 |
1223 |
1223 |
1224 |
1224 |
1225 |
1225 |
1226 |
1226 |
1227 |
1227 |
1228 |
1228 |
1229 |
1229 |
1230 |
1230 |
1231 |
1231 |
1232 |
1232 |
1233 |
1233 |
1234 |
1234 |
1235 |
1235 |
1236 |
1236 |
1237 |
1237 |
1238 |
1238 |
1239 |
1239 |
1240 |
1240 |
1241 |
1241 |
1242 |
1242 |
1243 |
1243 |
1244 |
1244 |
1245 |
1245 |
1246 |
1246 |
1247 |
1247 |
1248 |
1248 |
1249 |
1249 |
1250 |
1250 |
1251 |
1251 |
1252 |
1252 |
1253 |
1253 |
1254 |
1254 |
1255 |
1255 |
1256 |
1256 |
1257 |
1257 |
1258 |
1258 |
1259 |
1259 |
1260 |
1260 |
1261 |
1261 |
1262 |
1262 |
1263 |
1263 |
1264 |
1264 |
1265 |
1265 |
1266 |
1266 |
1267 |
1267 |
1268 |
1268 |
1269 |
1269 |
1270 |
1270 |
1271 |
1271 |
1272 |
1272 |
1273 |
1273 |
1274 |
1274 |
1275 |
1275 |
1276 |
1276 |
1277 |
1277 |
1278 |
1278 |
1279 |
1279 |
1280 |
1280 |
1281 |
1281 |
1282 |
1282 |
1283 |
1283 |
1284 |
1284 |
1285 |
1285 |
1286 |
1286 |
1287 |
1287 |
1288 |
1288 |
1289 |
1289 |
1290 |
1290 |
1291 |
1291
```

src/main/java/com/nordstrom/utils/interfaces/HasModified.java

+10 -0 Viewed

```

1 + package com.nordstrom.utils.interfaces;
2 +
3 + /**
4 + * HasModified Interface.
5 + */
6 + public interface HasModified {
7 +     void setModified(String var1);
8 +
9 +     String getModified();
10+

```

src/main/java/com/nordstrom/utils/DateTimeUtils.java

+46 -0 Viewed

```

1 + package com.nordstrom.utils;
2 +
3 + import java.text.DateFormat;
4 + import java.text.ParseException;
5 + import java.text.SimpleDateFormat;
6 + import java.time.Clock;
7 + import java.time.Instant;
8 + import java.time.ZoneOffset;
9 + import java.time.ZonedDateTime;
10+ import java.time.format.DateTimeFormatter;
11+ import java.util.Date;
12+ import java.util.TimeZone;
13+ import org.apache.commons.lang3.StringUtils;
14+ import org.springframework.stereotype.Component;
15+
16+ /**
17+ * Deals with date and time by ISO 8601.
18+ */
19+ @Component
20+ public class DateTimeUtils {
21+     public static final String DATETIME_FORMAT = "yyyy-MM-dd'T'HH:mm:ssX";
22+     private static final DateTimeFormatter DTF = DateTimeFormatter
23+         .ofPattern(DATETIME_FORMAT);
24+     private static final long DEFAULT_EXPIRATION_TIME_OFFSET_DAYS = 7L;
25+
26+     public ZonedDateTime getDateNowUtc() {
27+         return ZonedDateTime.now(Clock.systemUTC());
28+     }
29+
30+     public String dateTimeToString(ZonedDateTime dateTime) {
31+         return dateTime.format(DTF);
32+     }
33+
34+     public String getDateNowUtcAsString() {
35+         return dateTimeToString(getDateNowUtc());
36+     }
37+
38+ /**
39+ * Given a base date expressed as a String, add an offset and return it as an epoch long.
40+ */
41+ public static long calculateExpirationTime(String baseDate) {
42+     ZonedDateTime zonedBaseDate = ZonedDateTime.parse(baseDate, DTF);
43+     return zonedBaseDate.plusDays(DEFAULT_EXPIRATION_TIME_OFFSET_DAYS).toEpochSecond();
44+ }
45+
46+

```

src/main/java/com/nordstrom/utils/ValidationHelper.java

+15 -0 Viewed

```

1 + package com.nordstrom.utils;
2 +
3 + public class ValidationHelper {
4 +
5+ /**
6+ * @param argument The argument value.
7+ * @throws IllegalArgumentException An IllegalArgumentException.
8+ */
9+ public static void validateArgumentForNull(Object argument)
10+     throws IllegalArgumentException {
11+     if (argument == null) {
12+         throw new IllegalArgumentException("The consumer record argument cannot be null.");
13+     }
14+ }
15+

```

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

```
1+ package com.nordstrom.dynamo.tables;
2+
3+ import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
4+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBDeleteExpression;
5+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
6+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig;
7+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.TableNameOverride;
8+ import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig.Builder;
9+ import com.amazonaws.services.dynamodbv2.datamodeling.ScanResultPage;
10+ import com.amazonaws.services.dynamodbv2.model.AttributeValue;
11+ import com.nordstrom.indexer.config.AwsConfig;
12+ import com.nordstrom.indexer.model.Brand;
13+ import com.nordstrom.indexer.model.Style;
14+ import com.nordstrom.utils.DateTimeUtils;
15+ import java.util.Map;
16+ import junit.framework.TestCase;
17+ import static org.mockito.Matchers.any;
18+ import static org.mockito.Mockito.doNothing;
19+ import static org.mockito.Mockito.times;
20+ import static org.mockito.Mockito.verify;
21+ import static org.mockito.Mockito.when;
22+ import org.junit.Before;
23+ import org.junit.Test;
24+ import org.junit.runner.RunWith;
25+ import org.mockito.Mock;
26+ import org.mockito.runners.MockitoJUnitRunner;
27+ import java.util.List;
28+ import org.mockito.stubbing.Answer;
29+
30+ @RunWith(MockitoJUnitRunner.class)
31+ public class HashKeyTableTest extends TestCase {
32+     private final String TEST_NAME = "testName";
33+     private final String TEST_GROUP = "testGroup";
34+     private final Integer TEST_ID = 123;
35+
36+     @Mock
37+     DynamoDBMapper mapper;
38+     @Mock
39+     AwsConfig awsConfig;
40+     @Mock
41+     AmazonDynamoDB amazonDynamoDB;
42+     @Mock
43+     DateTimeUtils dateTimeUtils;
44+     @Mock
45+     ScanResultPage scanResultPage;
46+
47+     private BrandTable brandTable;
48+     private StyleTable styleTable;
49+     private Brand brand;
50+     private Style style;
51+
52+     @Before
53+     public void before() {
54+         brand = new Brand();
55+         style = new Style();
56+
57+         brandTable = new BrandTable(awsConfig, amazonDynamoDB, dateTimeUtils);
58+         brandTable.setMapper(mapper);
59+         styleTable = new StyleTable(awsConfig, amazonDynamoDB, dateTimeUtils);
60+         styleTable.setMapper(mapper);
61+
62+         Builder builder = new DynamoDBMapperConfig.Builder();
63+         builder.setTableNameOverride(new TableNameOverride(""));
64+     }
65+
66+     @Test
67+     public void brandGetTest() {
68+         when(mapper.scanPage(any(), any(), any()))
69+             .then((Answer<?>) invocation -> {
70+                 brand.setId(TEST_ID);
71+                 brand.setName(TEST_NAME);
72+                 scanResultPage = new ScanResultPage();
73+                 scanResultPage.setResults(List.of(brand));
74+
75+                 return scanResultPage;
76+             });
77+
78+         List<Brand> result = brandTable.getAll();
79+         verify(mapper, times(1)).scanPage(any(), any(), any());
80+         assertEquals(result.get(0).getId(), TEST_ID);
81+         assertEquals(result.get(0).getName(), TEST_NAME);
82+     }
83+
84+     @Test
85+     public void styleGetTest() {
86+         when(mapper.scanPage(any(), any(), any()))
87+             .then((Answer<?>) invocation -> {
88+                 style.setId(TEST_ID);
89+                 style.setGroup(TEST_GROUP);
90+                 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
110             AttributeValue().withN(TEST_ID.toString())));
111         scanResultPage.setResults(List.of(style));
112
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         assertThat(result, isNotNull());
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         assertThat(result, isNotNull());
43         assertEquals(env, result);
44     }
45
46     @Test
47     public void profileCredentialsProviderTest() {
48         awsConfig.setProfileCredentialsProvider(testCredsProvider);
49         String result = awsConfig.getProfileCredentialsProvider();
50
51         assertThat(result, isNotNull());
52         assertEquals(testCredsProvider, result);
53     }
54
\ No newline at end of file

```

▼ [src/test/java/com/nordstrom/indexer/consumer/OisStreamConsumerTest.java](#)

+76 -0 Viewed

```
1 + package com.nordstrom.indexer.consumer;
2 +
3 + import com.nordstrom.indexer.processor.DeleteProcessor;
4 + import com.nordstrom.indexer.processor.ProcessorFactory;
5 + import com.nordstrom.indexer.processor.UpdateProcessor;
6 + import com.nordstrom.offer.StyleOffer;
7 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
8 + import java.util.List;
9 + import junit.framework.TestCase;
10 + import org.apache.kafka.clients.consumer.ConsumerRecord;
11 + import org.junit.Before;
12 + import org.junit.Test;
13 + import org.junit.runner.RunWith;
14 + import static org.mockito.Matchers.any;
15 + import static org.mockito.Matchers.anyString;
16 + import org.mockito.Mock;
17 + import static org.mockito.Mockito.doNothing;
18 + import static org.mockito.Mockito.times;
19 + import static org.mockito.Mockito.verify;
20 + import static org.mockito.Mockito.when;
21 + import org.mockito.runners.MockitoJUnitRunner;
22 +
23 + @RunWith(MockitoJUnitRunner.class)
24 + public class OisStreamConsumerTest extends TestCase {
25 +     private OisStreamConsumer consumer;
26 +
27 +     @Mock
28 +     private ApplicationLogger applicationLogger;
29 +     @Mock
30 +     private ProcessorFactory processorFactory;
31 +     @Mock
32 +     private ConsumerRecord<String, StyleOffer> consumerRecord1;
33 +     @Mock
34 +     private ConsumerRecord<String, StyleOffer> consumerRecord2;
35 +     @Mock
36 +     private UpdateProcessor updateProcessor;
37 +     @Mock
38 +     private DeleteProcessor deleteProcessor;
39 +
40 +     @Before
41 +     public void before() {
42 +         consumer = new OisStreamConsumer(applicationLogger, processorFactory);
43 +     }
44 +
45 +     @Test
46 +     public void processUpdateTest() {
47 +         when(processorFactory.getProcessor(any())).thenAnswer(invocation -> updateProcessor);
48 +         doNothing().when(updateProcessor).execute(any());
49 +         consumer.process(List.of(consumerRecord1, consumerRecord2));
50 +
51 +         verify(processorFactory, times(2)).getProcessor(any());
52 +         verify(updateProcessor, times(2)).execute(any());
53 +     }
54 +
55 +     @Test
56 +     public void processDeleteTest() {
57 +         when(processorFactory.getProcessor(any())).thenAnswer(invocation -> deleteProcessor);
58 +         doNothing().when(deleteProcessor).execute(any());
59 +         consumer.process(List.of(consumerRecord1, consumerRecord2));
60 +
61 +         verify(processorFactory, times(2)).getProcessor(any());
62 +         verify(deleteProcessor, times(2)).execute(any());
63 +     }
64 +
65 +     @Test
66 +     public void nullConsumerRecordsTest() {
67 +         consumer.process(null);
68 +         verify(applicationLogger, times(1)).warn(anyString());
69 +         verify(processorFactory, times(0)).getProcessor(any());
70 +     }
71 +
72 +     @Test(expected = EventProcessingException.class)
73 +     public void eventProcessingExceptionTest() {
74 +         consumer.process(List.of(consumerRecord1, consumerRecord2));
75 +     }
76 + }
```

\ No newline at end of file

src/test/java/com/nordstrom/indexer/dependency/KafkaBootstrapperTest.java

+51 -0 Viewed

```
1 + package com.nordstrom.indexer.dependency;
2 +
3 + import com.nordstrom.offer.StyleOffer;
4 + import java.util.Map;
5 + import junit.framework.TestCase;
6 + import static org.hamcrest.core.IsInstanceOf.instanceOf;
7 + import static org.junit.Assert.assertThat;
8 + import org.junit.Before;
9 + import org.junit.Test;
10 + import org.junit.runner.RunWith;
11 + import org.mockito.Mock;
12 + import static org.mockito.Mockito.when;
13 + import org.mockito.runners.MockitoJUnitRunner;
14 + import org.springframework.boot.autoconfigure.kafka.KafkaProperties;
15 + import org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;
16 + import org.springframework.kafka.core.ConsumerFactory;
17 +
18 + @RunWith(MockitoJUnitRunner.class)
19 + public class KafkaBootstrapperTest extends TestCase {
20 +     private final String testKey = "key";
21 +     private final StyleOffer testOffer = new StyleOffer();
22 +     private KafkaBootstrapper kafkaBootstrapper;
23 +
24 +     @Mock
25 +     KafkaProperties kafkaProperties;
26 +     @Mock
27 +     ConsumerFactory<String, StyleOffer> consumerFactory;
28 +
29 +     @Before
30 +     public void before() {
31 +         kafkaBootstrapper = new KafkaBootstrapper(kafkaProperties);
32 +     }
33 +
34 +     @Test
35 +     public void kafkaListenerContainerFactoryTest() {
36 +         ConcurrentKafkaListenerContainerFactory<String, StyleOffer> result;
37 +         result = kafkaBootstrapper.kafkaListenerContainerFactory(consumerFactory);
38 +
39 +         assertThat(result, instanceOf(ConcurrentKafkaListenerContainerFactory.class));
40 +     }
41 +
42 +     @Test
43 +     public void retryKafkaListenerContainerFactoryTest() {
44 +         when(kafkaProperties.buildConsumerProperties()).thenAnswer(invocation -> Map.of(testKey, testOffer));
45 +
46 +         ConcurrentKafkaListenerContainerFactory<String, String> result;
47 +         result = kafkaBootstrapper.retryKafkaListenerContainerFactory();
48 +
49 +         assertThat(result, instanceOf(ConcurrentKafkaListenerContainerFactory.class));
50 +     }
51 + }
```

\ No newline at end of file

src/test/java/com/nordstrom/indexer/dependency/SharedLibraryBeanBootstrapperTest.java

+43 -0 Viewed

```
1 + package com.nordstrom.indexer.dependency;
2 +
3 + import com.nordstrom.indexer.config.ApplicationConfig;
4 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
5 + import junit.framework.TestCase;
6 + import static org.hamcrest.core.IsInstanceOf.instanceOf;
7 + import static org.junit.Assert.assertThat;
8 + import org.junit.Before;
9 + import org.junit.Test;
10 + import org.junit.runner.RunWith;
11 + import org.mockito.Mock;
12 + import static org.mockito.Mockito.when;
13 + import org.mockito.runners.MockitoJUnitRunner;
14 +
15 + @RunWith(MockitoJUnitRunner.class)
16 + public class SharedLibraryBeanBootstrapperTest extends TestCase {
17 +     private final String env = "dev";
18 +     private final String loggingType = "testLogger";
19 +     private final String loggingId = "testLoggerId";
20 +     private final String loggingNamespace = "testNamespace";
21 +     private SharedLibraryBeanBootstrapper sharedLibraryBeanBootstrapper;
22 +
23 +     @Mock
24 +     ApplicationConfig applicationConfig;
25 +
26 +     @Before
27 +     public void before() {
28 +         sharedLibraryBeanBootstrapper = new SharedLibraryBeanBootstrapper();
29 +     }
30 +
31 +     @Test
32 +     public void configureLoggerTest() {
33 +         ApplicationLogger result;
34 +
35 +         when(applicationConfig.getEnvironment()).thenReturn(invocation -> env);
36 +         when(applicationConfig.getLoggingApplicationType()).thenReturn(invocation -> loggingType);
37 +         when(applicationConfig.getLoggingApplicationId()).thenReturn(invocation -> loggingId);
38 +         when(applicationConfig.getLoggingNamespace()).thenReturn(invocation -> loggingNamespace);
39 +
40 +         result = sharedLibraryBeanBootstrapper.configureLogger(applicationConfig);
41 +         assertThat(result, instanceOf(ApplicationLogger.class));
42 +     }
43 + }
```

\ No newline at end of file

src/test/java/com/nordstrom/indexer/processor/DeleteProcessorTest.java

+95 -0 Viewed

```
1 + package com.nordstrom.indexer.processor;
2 +
3 + import com.nordstrom.dynamo.tables.BrandTable;
4 + import com.nordstrom.dynamo.tables.StyleTable;
5 + import com.nordstrom.event.rosettastone.ProductStyle;
6 + import com.nordstrom.offer.Brand;
7 + import com.nordstrom.offer.StyleIdentifiers;
8 + import com.nordstrom.offer.StyleOffer;
9 + import com.nordstrom.offer.StyleProductAttributes;
10 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
11 + import junit.framework.TestCase;
12 + import org.apache.kafka.clients.consumer.ConsumerRecord;
13 + import org.junit.Before;
14 + import org.junit.Test;
15 + import org.junit.runner.RunWith;
16 + import static org.mockito.Matchers.any;
17 + import static org.mockito.Matchers.anyInt;
18 + import static org.mockito.Matchers.anyString;
19 + import org.mockito.Mock;
20 + import static org.mockito.Mockito.doNothing;
21 + import static org.mockito.Mockito.times;
22 + import static org.mockito.Mockito.verify;
23 + import static org.mockito.Mockito.when;
24 + import org.mockito.runners.MockitoJUnitRunner;
25 +
26 + @RunWith(MockitoJUnitRunner.class)
27 + public class DeleteProcessorTest extends TestCase {
28 +     private final int testBrandId = 1234567;
29 +     private StringBuffer testBrandName;
30 +     private StringBuffer testId;
31 +     private DeleteProcessor deleteProcessor;
32 +
33 +     @Mock
34 +     private ApplicationLogger logger;
35 +     @Mock
36 +     private BrandTable brandTable;
37 +     @Mock
38 +     private StyleTable styleTable;
39 +     @Mock
40 +     private ConsumerRecord<String, StyleOffer> consumerRecord;
41 +     @Mock
42 +     private StyleOffer styleOffer;
43 +     @Mock
44 +     private StyleIdentifiers styleIdentifiers;
45 +     @Mock
46 +     private StyleProductAttributes styleProductAttributes;
47 +     @Mock
48 +     private ProductStyle productStyle;
49 +     @Mock
50 +     private Brand brand;
51 +
52 +     @Before
53 +     public void before() {
54 +         testId = new StringBuffer("1234567");
55 +         testBrandName = new StringBuffer("testName");
56 +         deleteProcessor = new DeleteProcessor(logger, brandTable, styleTable);
57 +     }
58 +
59 +     @Test
60 +     public void executeDeletesStyleTest() {
61 +         when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
62 +         when(styleOffer.getIds()).thenAnswer(invocation -> styleIdentifiers);
63 +         when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
64 +         when(styleIdentifiers.getWebStyle()).thenAnswer(invocation -> productStyle);
65 +         when(styleIdentifiers.getRmsStyleGroup()).thenAnswer(invocation -> productStyle);
66 +         when(productStyle.getId()).thenAnswer(invocation -> testId);
67 +
68 +         doNothing().when(styleTable).delete(any(), anyString());
69 +         deleteProcessor.execute(consumerRecord);
70 +
71 +         verify(styleTable, times(1)).delete(anyString(), anyInt());
72 +     }
73 +
74 +     @Test
75 +     public void executeDoesNotDeleteBrandTest() {
76 +         when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
77 +         when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
78 +         when(styleProductAttributes.getBrand()).thenAnswer(invocation -> brand);
79 +         when(brand.getId()).thenAnswer(invocation -> testBrandId);
80 +         when(brand.getName()).thenAnswer(invocation -> testBrandName);
81 +
82 +         doNothing().when(brandTable).delete(any(), anyInt());
83 +         deleteProcessor.execute(consumerRecord);
84 +
85 +         verify(brandTable, times(0)).delete(anyInt(), anyString());
86 +     }
87 +
88 +     @Test
89 +     public void nullStyleIdTest() {
90 +         when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
91 +         deleteProcessor.execute(consumerRecord);
92 +
93 +         verify(logger, times(1)).error(anyString());
94 +     }
95 + }
```

\ No newline at end of file

src/test/java/com/nordstrom/indexer/processor/ProcessorFactoryTest.java

+67 -0 Viewed

```

1  + package com.nordstrom.indexer.processor;
2  +
3  + import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
4  + import com.nordstrom.indexer.config.AwsConfig;
5  + import com.nordstrom.offer.StyleOffer;
6  + import com.nordstrom.sharedlib.logging.ApplicationLogger;
7  + import junit.framework.TestCase;
8  + import org.apache.kafka.clients.consumer.ConsumerRecord;
9  + import org.apache.kafka.common.header.Headers;
10 + import org.apache.kafka.common.header.internals.RecordHeader;
11 + import org.apache.kafka.common.header.internals.RecordHeaders;
12 + import static org.hamcrest.core.IsInstanceOf.instanceOf;
13 + import static org.junit.Assert.assertThat;
14 + import org.junit.Before;
15 + import org.junit.Test;
16 + import org.junit.runner.RunWith;
17 + import org.mockito.Mock;
18 + import static org.mockito.Mockito.when;
19 + import org.mockito.runners.MockitoJUnitRunner;
20 + import org.mockito.stubbing.Answer;
21 +
22 + @RunWith(MockitoJUnitRunner.class)
23 + public class ProcessorFactoryTest extends TestCase {
24 +     private final String testKey = "Type";
25 +     private final String updateValue = "Updated";
26 +     private final String deleteValue = "Deleted";
27 +     private Processor processor;
28 +     private ProcessorFactory processorFactory;
29 +     private Headers headers;
30 +
31 +     @Mock
32 +     private ApplicationLogger applicationLogger;
33 +     @Mock
34 +     private AwsConfig awsConfig;
35 +     @Mock
36 +     private AmazonDynamoDB amazonDynamoDB;
37 +     @Mock
38 +     private ConsumerRecord<String, StyleOffer> consumerRecord;
39 +
40 +     @Before
41 +     public void before() {
42 +         headers = new RecordHeaders();
43 +         processorFactory = new ProcessorFactory(applicationLogger, awsConfig);
44 +     }
45 +
46 +     @Test
47 +     public void getProcessorUpdateTest() {
48 +         headers.add(new RecordHeader(testKey, updateValue.getBytes()));
49 +
50 +         when(awsConfig.getAmazonDynamoDb()).thenAnswer(invocation -> amazonDynamoDB);
51 +         when(consumerRecord.headers()).thenAnswer((Answer<Headers>) invocation -> headers);
52 +         processor = processorFactory.getProcessor(consumerRecord);
53 +
54 +         assertThat(processor, instanceOf(UpdateProcessor.class));
55 +     }
56 +
57 +     @Test
58 +     public void getProcessorDeleteTest() {
59 +         headers.add(new RecordHeader(testKey, deleteValue.getBytes()));
60 +
61 +         when(awsConfig.getAmazonDynamoDb()).thenAnswer(invocation -> amazonDynamoDB);
62 +         when(consumerRecord.headers()).thenAnswer((Answer<Headers>) invocation -> headers);
63 +         processor = processorFactory.getProcessor(consumerRecord);
64 +
65 +         assertThat(processor, instanceOf(DeleteProcessor.class));
66 +     }
67 + }
```

\ No newline at end of file

src/test/java/com/nordstrom/indexer/processor/UpdateProcessorTest.java

+120 -0 Viewed

```

1  + package com.nordstrom.indexer.processor;
2  +
3  + import com.nordstrom.dynamo.tables.BrandTable;
4  + import com.nordstrom.dynamo.tables.StyleTable;
5  + import com.nordstrom.event.rosettastone.ProductStyle;
6  + import com.nordstrom.indexer.model.Style;
7  + import com.nordstrom.offer.Brand;
8  + import com.nordstrom.offer.StyleIdentifiers;
9  + import com.nordstrom.offer.StyleOffer;
10 + import com.nordstrom.offer.StyleProductAttributes;
11 + import com.nordstrom.sharedlib.logging.ApplicationLogger;
12 + import junit.framework.TestCase;
13 + import org.apache.kafka.clients.consumer.ConsumerRecord;
14 + import org.junit.Before;
15 + import org.junit.Test;
16 + import org.junit.runner.RunWith;
17 + import org.mockito.ArgumentMatcher;
18 + import static org.mockito.Matchers.any;
19 + import static org.mockito.Matchers.anyString;
20 + import static org.mockito.Matchers.argThat;
21 + import org.mockito.Mock;
22 + import static org.mockito.Mockito.doNothing;
23 + import static org.mockito.Mockito.times;
24 + import static org.mockito.Mockito.verify;
25 + import static org.mockito.Mockito.when;
26 + import org.mockito.runners.MockitoJUnitRunner;
```

```
27+
28+ @RunWith(MockitoJUnitRunner.class)
29+ public class UpdateProcessorTest extends TestCase {
30+   private final int testBrandId = 1234567;
31+   private StringBuffer testBrandName;
32+   private StringBuffer testStyleId;
33+   private StringBuffer testStyleGroup;
34+   private UpdateProcessor updateProcessor;
35+
36+   @Mock
37+   private ApplicationLogger logger;
38+   @Mock
39+   private BrandTable brandTable;
40+   @Mock
41+   private StyleTable styleTable;
42+   @Mock
43+   private ConsumerRecord<String, StyleOffer> consumerRecord;
44+   @Mock
45+   private StyleOffer styleOffer;
46+   @Mock
47+   private StyleIdentifiers styleIdentifiers;
48+   @Mock
49+   private StyleProductAttributes styleProductAttributes;
50+   @Mock
51+   private ProductStyle webStyle;
52+   @Mock
53+   private ProductStyle styleGroup;
54+   @Mock
55+   private Brand brand;
56+
57+   @Before
58+   public void before() {
59+     testStyleId = new StringBuffer("1234567");
60+     testStyleGroup = new StringBuffer("1111111");
61+     testBrandName = new StringBuffer("testName");
62+     updateProcessor = new UpdateProcessor(logger, brandTable, styleTable);
63+   }
64+
65+   @Test
66+   public void executeTest() {
67+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
68+     when(styleOffer.getIds()).thenAnswer(invocation -> styleIdentifiers);
69+     when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
70+     when(styleIdentifiers.getWebStyle()).thenAnswer(invocation -> webStyle);
71+     when(styleIdentifiers.getRmsStyleGroup()).thenAnswer(invocation -> styleGroup);
72+     when(webStyle.getId()).thenAnswer(invocation -> testStyleId);
73+     when(styleGroup.getId()).thenAnswer(invocation -> testStyleGroup);
74+     when(styleProductAttributes.getBrand()).thenAnswer(invocation -> brand);
75+     when(brand.getId()).thenAnswer(invocation -> testBrandId);
76+     when(brand.getName()).thenAnswer(invocation -> testBrandName);
77+
78+     doNothing().when(styleTable).save(any());
79+     doNothing().when(brandTable).save(any());
80+     updateProcessor.execute(consumerRecord);
81+
82+     verify(styleTable, times(1)).save(any(Style.class));
83+     verify(brandTable, times(1)).save(any(com.nordstrom.indexer.model.Brand.class));
84+     verify(styleTable, times(1)).save(argThat(new ArgumentMatcher<>() {
85+       @Override
86+       public boolean matches(Object argument) {
87+         var style = (Style) argument;
88+         return
89+           style.getStyleId() == Integer.parseInt(testStyleId.toString()) &&
90+           style.getStyleGroupId().equals(testStyleGroup.toString());
91+         }
92+       }));
93+     verify(brandTable, times(1)).save(argThat(new ArgumentMatcher<>() {
94+       @Override
95+       public boolean matches(Object argument) {
96+         var brand = (com.nordstrom.indexer.model.Brand) argument;
97+         return
98+           brand.getBrandId() == testBrandId &&
99+           brand.getBrandName().equals(testBrandName.toString());
100+
101+       });
102+     });
103+
104+   @Test
105+   public void null_Product_Attributes_Test() {
106+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
107+     updateProcessor.execute(consumerRecord);
108+
109+     verify(logger, times(1)).warn(anyString());
110+   }
111+
112+   @Test
113+   public void null_Brand_Test() {
114+     when(consumerRecord.value()).thenAnswer(invocation -> styleOffer);
115+     when(styleOffer.getProductAttributes()).thenAnswer(invocation -> styleProductAttributes);
116+     updateProcessor.execute(consumerRecord);
117+
118+     verify(logger, times(1)).error(anyString());
119+   }
120+ }
```

```

1 + package com.nordstrom.utils;
2 +
3 + import java.time.ZonedDateTime;
4 + import java.time.format.DateTimeFormatter;
5 + import junit.framework.TestCase;
6 + import static org.hamcrest.core.IsInstanceOf.instanceOf;
7 + import static org.junit.Assert.assertThat;
8 + import org.junit.Before;
9 + import org.junit.Test;
10 + import org.junit.runner.RunWith;
11 + import org.mockito.runners.MockitoJUnitRunner;
12 +
13 + @RunWith(MockitoJUnitRunner.class)
14 + public class DateTimeUtilsTest extends TestCase {
15 +     public static final String DATETIME_FORMAT = "yyyy-MM-dd'T'HH:mm:ssX";
16 +     private static final DateTimeFormatter DTF = DateTimeFormatter
17 +         .ofPattern(DATETIME_FORMAT);
18 +     private final String inputDatetime = "2013-09-29T18:46:19Z";
19 +
20 +     private DateTimeUtils dateTimeUtils;
21 +     private ZonedDateTime zonedBaseDate;
22 +
23 +
24 +     @Before
25 +     public void before() {
26 +         dateTimeUtils = new DateTimeUtils();
27 +         zonedBaseDate = ZonedDateTime.parse(inputDatetime, DTF);
28 +     }
29 +
30 +     @Test
31 +     public void getDateTimeNowUtcTest() {
32 +         ZonedDateTime result = dateTimeUtils.getDateTimeNowUtc();
33 +
34 +         assertThat(result);
35 +         assertThat(result, instanceOf(ZonedDateTime.class));
36 +     }
37 +
38 +     @Test
39 +     public void dateTimeToStringTest() {
40 +         String result = dateTimeUtils.dateTimeToString(zonedBaseDate);
41 +
42 +         assertThat(result);
43 +         assertThat(zonedBaseDate.format(DTF), result);
44 +     }
45 +
46 +     @Test
47 +     public void getDateTimeNowUtcAsStringTest() {
48 +         String result = dateTimeUtils.getDateTimeNowUtcAsString();
49 +         assertThat(result);
50 +     }
51 +
52 +     @Test
53 +     public void calculateExpirationTimeTest() {
54 +         long expirationTime = DateTimeUtils.calculateExpirationTime(inputDatetime);
55 +         assertThat(zonedBaseDate.plusDays(7L).toEpochSecond(), expirationTime);
56 +     }
57 +
58 +
59 + }
```

\ No newline at end of file

src/test/java/com/nordstrom/utils/ValidationHelperTest.java

+34 -0 Viewed

```

1 + package com.nordstrom.utils;
2 +
3 + import com.nordstrom.offer.StyleOffer;
4 + import junit.framework.TestCase;
5 + import org.junit.Test;
6 + import org.junit.runner.RunWith;
7 + import org.mockito.runners.MockitoJUnitRunner;
8 +
9 + @RunWith(MockitoJUnitRunner.class)
10 + public class ValidationHelperTest extends TestCase {
11 +     private final String testArg = "testArgument";
12 +     private ValidationHelper validationHelper;
13 +
14 +     @Test
15 +     public void creationTest() {
16 +         validationHelper = new ValidationHelper();
17 +         assertThat(validationHelper);
18 +     }
19 +
20 +     @Test(expected = IllegalArgumentException.class)
21 +     public void nullArgValidateArgumentForNullTest() {
22 +         ValidationHelper.validateArgumentForNull(null);
23 +     }
24 +
25 +     @Test
26 +     public void newStyleOfferValidateArgumentForNullTest() {
27 +         ValidationHelper.validateArgumentForNull(new StyleOffer());
28 +     }
29 +
30 +     @Test(expected = IllegalArgumentException.class)
31 +     public void allNullValidateArgumentForNullTest() {
32 +         ValidationHelper.validateArgumentForNull(null);
33 +     }
34 + }
```