package com.twitter.search.earlybird.common.config;

import java.util.Date;

import java.util.List;

import java.util.Map;

import javax.annotation.Nullable;

import com.google.common.collect.ImmutableMap;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.common\_internal.text.version.PenguinVersion;

import com.twitter.search.common.aurora.AuroraInstanceKey;

import com.twitter.search.common.config.Config;

import com.twitter.search.common.config.ConfigFile;

import com.twitter.search.common.config.ConfigurationException;

import com.twitter.search.common.config.SearchPenguinVersionsConfig;

public final class EarlybirdConfig {

private static final Logger LOG = LoggerFactory.getLogger(EarlybirdConfig.class);

private static final String DEFAULT\_CONFIG\_FILE = "earlybird-search.yml";

private static final String LATE\_TWEET\_BUFFER\_KEY = "late\_tweet\_buffer";

public static final String EARLYBIRD\_ZK\_CONFIG\_DIR = "/twitter/search/production/earlybird/";

public static final String EARLYBIRD\_CONFIG\_DIR = "earlybird/config";

public static final String USER\_SNAPSHOT\_BASE\_DIR = "user\_snapshot\_base\_dir";

private static volatile ConfigFile earlybirdConfig = null;

private static volatile Map<String, Object> overrideValueMap = ImmutableMap.of();

private static String logDirOverride = null;

private static AuroraInstanceKey auroraInstanceKey = null;

private static int adminPort;

private EarlybirdConfig() { }

private static final class PenguinVersionHolder {

private static final PenguinVersion PENGUIN\_VERSION\_SINGLETON =

SearchPenguinVersionsConfig.getSingleSupportedVersion(

EarlybirdProperty.PENGUIN\_VERSION.get());

private static final byte PENGUIN\_VERSION\_BYTE\_VALUE =

PENGUIN\_VERSION\_SINGLETON.getByteValue();

}

public static byte getPenguinVersionByte() {

return PenguinVersionHolder.PENGUIN\_VERSION\_BYTE\_VALUE;

}

public static PenguinVersion getPenguinVersion() {

return PenguinVersionHolder.PENGUIN\_VERSION\_SINGLETON;

}

/\*\*

\* Reads the earlybird configuration from the given file.

\*/

public static synchronized void init(@Nullable String configFile) {

if (earlybirdConfig == null) {

String file = configFile == null ? DEFAULT\_CONFIG\_FILE : configFile;

earlybirdConfig = new ConfigFile(EARLYBIRD\_CONFIG\_DIR, file);

}

}

public static synchronized void setOverrideValues(Map<String, Object> overrideValues) {

overrideValueMap = ImmutableMap.copyOf(overrideValues);

}

/\*\*

\* Pack all values in a string that can be printed for informational purposes.

\* @return the string.

\*/

public static String allValuesAsString() {

Map<String, String> stringMap = earlybirdConfig.getStringMap();

StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append("Config environment: " + Config.getEnvironment() + "\n\n");

stringBuilder.append(

String.format("Values from earlybird-search.yml (total %d):\n", stringMap.size()));

stringMap.forEach((key, value) -> {

stringBuilder.append(String.format(" %s: %s\n", key, value.toString()));

if (overrideValueMap.containsKey(key)) {

stringBuilder.append(String.format(

" override value: %s\n", overrideValueMap.get(key).toString()));

}

});

stringBuilder.append(String.format(

"\n\nAll command-line overrides (total: %d):\n", overrideValueMap.size()));

overrideValueMap.forEach((key, value) -> {

stringBuilder.append(String.format(" %s: %s\n", key, value.toString()));

});

return stringBuilder.toString();

}

/\*\*

\* Returns the value of the given property as a string. If the property is not set, a runtime

\* exception is thrown.

\*/

public static String getString(String property) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (String) overrideValue;

}

try {

return earlybirdConfig.getString(property);

} catch (ConfigurationException e) {

LOG.error("Fatal error: could not get config string " + property, e);

throw new RuntimeException(e);

}

}

/\*\*

\* Returns the value of the given property as a string.

\*/

public static String getString(String property, String defaultValue) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (String) overrideValue;

}

return earlybirdConfig.getString(property, defaultValue);

}

/\*\*

\* Returns the value of the given property as an integer. If the property is not set, a runtime

\* exception is thrown.

\*/

public static int getInt(String property) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (int) overrideValue;

}

try {

return earlybirdConfig.getInt(property);

} catch (ConfigurationException e) {

LOG.error("Fatal error: could not get config int " + property, e);

throw new RuntimeException(e);

}

}

/\*\*

\* Returns the value of the given property as an integer.

\*/

public static int getInt(String property, int defaultValue) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (int) overrideValue;

}

return earlybirdConfig.getInt(property, defaultValue);

}

/\*\*

\* Returns the value of the given property as a double.

\*/

public static double getDouble(String property, double defaultValue) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (double) overrideValue;

}

return earlybirdConfig.getDouble(property, defaultValue);

}

/\*\*

\* Returns the value of the given property as a long. If the property is not set, a runtime

\* exception is thrown.

\*/

public static long getLong(String property) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (long) overrideValue;

}

try {

return earlybirdConfig.getLong(property);

} catch (ConfigurationException e) {

LOG.error("Fatal error: could not get config long " + property, e);

throw new RuntimeException(e);

}

}

/\*\*

\* Returns the value of the given property as a long.

\*/

public static long getLong(String property, long defaultValue) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (long) overrideValue;

}

return earlybirdConfig.getLong(property, defaultValue);

}

/\*\*

\* Returns the value of the given property as a boolean. If the property is not set, a runtime

\* exception is thrown.

\*/

public static boolean getBool(String property) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (boolean) overrideValue;

}

try {

return earlybirdConfig.getBool(property);

} catch (ConfigurationException e) {

LOG.error("Fatal error: could not get config boolean " + property, e);

throw new RuntimeException(e);

}

}

/\*\*

\* Returns the value of the given property as a boolean.

\*/

public static boolean getBool(String property, boolean defaultValue) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (boolean) overrideValue;

}

return earlybirdConfig.getBool(property, defaultValue);

}

/\*\*

\* Returns the value of the given property as a date.

\*/

public static Date getDate(String property) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (Date) overrideValue;

}

Date date = (Date) earlybirdConfig.getObject(property, null);

if (date == null) {

throw new RuntimeException("Could not get config date: " + property);

}

return date;

}

/\*\*

\* Returns the value of the given property as a list of strings.

\*/

public static List<String> getListOfStrings(String property) {

Object overrideValue = overrideValueMap.get(property);

if (overrideValue != null) {

return (List<String>) overrideValue;

}

List<String> list = (List<String>) earlybirdConfig.getObject(property, null);

if (list == null) {

throw new RuntimeException("Could not get list of strings: " + property);

}

return list;

}

/\*\*

\* Returns the value of the given property as a map.

\*/

@SuppressWarnings("unchecked")

public static Map<String, Object> getMap(String property) {

Map<String, Object> map = (Map<String, Object>) earlybirdConfig.getObject(property, null);

if (map == null) {

throw new RuntimeException("Could not find config property: " + property);

}

return map;

}

public static int getMaxSegmentSize() {

return EarlybirdConfig.getInt("max\_segment\_size", 1 << 16);

}

/\*\*

\* Returns the log properties file.

\*/

public static String getLogPropertiesFile() {

try {

String filename = earlybirdConfig.getString("log\_properties\_filename");

return earlybirdConfig.getConfigFilePath(filename);

} catch (ConfigurationException e) {

// Print here rather than use LOG - log was probably not initialized yet.

LOG.error("Fatal error: could not get log properties file", e);

throw new RuntimeException(e);

}

}

/\*\*

\* Returns the log directory.

\*/

public static String getLogDir() {

if (logDirOverride != null) {

return logDirOverride;

} else {

return EarlybirdConfig.getString("log\_dir");

}

}

public static void overrideLogDir(String logDir) {

EarlybirdConfig.logDirOverride = logDir;

}

public static int getThriftPort() {

return EarlybirdProperty.THRIFT\_PORT.get();

}

public static int getWarmUpThriftPort() {

return EarlybirdProperty.WARMUP\_THRIFT\_PORT.get();

}

public static int getSearcherThreads() {

return EarlybirdProperty.SEARCHER\_THREADS.get();

}

public static int getLateTweetBuffer() {

return getInt(LATE\_TWEET\_BUFFER\_KEY);

}

public static int getAdminPort() {

return adminPort;

}

public static void setAdminPort(int adminPort) {

EarlybirdConfig.adminPort = adminPort;

}

public static boolean isRealtimeOrProtected() {

String earlybirdName = EarlybirdProperty.EARLYBIRD\_NAME.get();

return earlybirdName.contains("realtime") || earlybirdName.contains("protected");

}

public static boolean consumeUserScrubGeoEvents() {

return EarlybirdProperty.CONSUME\_GEO\_SCRUB\_EVENTS.get();

}

@Nullable

public static AuroraInstanceKey getAuroraInstanceKey() {

return auroraInstanceKey;

}

public static void setAuroraInstanceKey(AuroraInstanceKey auroraInstanceKey) {

EarlybirdConfig.auroraInstanceKey = auroraInstanceKey;

}

public static boolean isAurora() {

return auroraInstanceKey != null;

}

public static void setForTests(String property, Object value) {

earlybirdConfig.setForTests(DEFAULT\_CONFIG\_FILE, property, value);

}

public static synchronized void clearForTests() {

earlybirdConfig = new ConfigFile(EARLYBIRD\_CONFIG\_DIR, DEFAULT\_CONFIG\_FILE);

}

}