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

import java.io.InputStream;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

/\*\*

\* Config file for relevance computation.

\*/

public final class TweetProcessingConfig {

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

private static final String SCORER\_CONFIG\_DIR = "common/relevance/config";

public static final String DEFAULT\_CONFIG\_FILE = "relevance.yml";

private static ConfigFile relevanceConfig = null;

private TweetProcessingConfig() {

}

/\*\* Initializes this instance from the given config file. \*/

public static void init(String configFile) {

if (relevanceConfig == null) {

synchronized (TweetProcessingConfig.class) {

if (relevanceConfig == null) {

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

relevanceConfig = new ConfigFile(SCORER\_CONFIG\_DIR, file);

}

}

}

}

/\*\* Initializes this instance from the given input stream. \*/

public static void init(InputStream inputStream, String configType) {

if (relevanceConfig == null) {

synchronized (TweetProcessingConfig.class) {

if (relevanceConfig == null) {

relevanceConfig = new ConfigFile(inputStream, configType);

}

}

}

}

/\*\* Initializes this instance. \*/

public static void init() {

init(null);

}

/\*\*

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

\*

\* @param property The property.

\* @param defaultValue The default value to return if the property is not present in the config.

\*/

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

return relevanceConfig.getDouble(property, defaultValue);

}

/\*\*

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

\*

\* @param property The property.

\* @param defaultValue The default value to return if the property is not present in the config.

\*/

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

return relevanceConfig.getString(property, defaultValue);

}

/\*\*

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

\*

\* @param property The property.

\* @param defaultValue The default value to return if the property is not present in the config.

\*/

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

return relevanceConfig.getInt(property, defaultValue);

}

/\*\*

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

\*

\* @param property The property.

\* @param defaultValue The default value to return if the property is not present in the config.

\*/

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

return relevanceConfig.getLong(property, defaultValue);

}

/\*\*

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

\*

\* @param property The property.

\* @param defaultValue The default value to return if the property is not present in the config.

\*/

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

return relevanceConfig.getBool(property, defaultValue);

}

/\*\*

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

\*

\* @param property The property.

\* @throws ConfigurationException If the given property is not found in the config.

\*/

public static String getString(String property) {

try {

return relevanceConfig.getString(property);

} catch (ConfigurationException e) {

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

throw new RuntimeException(e);

}

}

}