**配置中心客户端使用范例**

依赖项配置（以Maven为例）： pom.xml

<dependency>

<groupId>com.taobao.configserver</groupId>

<artifactId>config-client</artifactId>

<version>1.3.1</version>

</dependency>

代码范例： config-client-sample.java

package com.taobao.config.client.sample;

import java.io.Serializable;

import java.util.List;

import com.taobao.config.client.Publisher;

import com.taobao.config.client.PublisherRegistrar;

import com.taobao.config.client.PublisherRegistration;

import com.taobao.config.client.Subscriber;

import com.taobao.config.client.SubscriberDataObserver;

import com.taobao.config.client.SubscriberRegistrar;

import com.taobao.config.client.SubscriberRegistration;

import com.taobao.config.client.Subscription;

public class ConfigClientDemo {

static String publication = "com.playboy.magzine";

public void publish() {

String publisherName = "Playboy (HK) LTD."; // Identity of publisher

PublisherRegistration<PlayboyMagzine> registration // Registration of publisher

= new PublisherRegistration<PlayboyMagzine>(publisherName, publication);

Publisher<PlayboyMagzine> publisher = PublisherRegistrar.register(registration);

// Do register

publisher.publish(new PlayboyMagzine("2008 Supplement")); // Do publish now

}

public void subscribe() {

String subscriberName = "Mr. Wish"; // Identity of subscriber

SubscriberRegistration registration = new SubscriberRegistration(subscriberName, publication);

// Registration of subscriber

Subscriber subscriber = SubscriberRegistrar.register(registration);

// Do register

subscriber.setDataObserver(new NewIssueNotifier()); // Get notified upon new publication

}

public void subscribeAsFuture() {

String subscriberName = "Mr. Yew";

SubscriberRegistration registration = new SubscriberRegistration(subscriberName, publication);

Subscriber subscriber = SubscriberRegistrar.register(registration);

Subscription subscription = subscriber.getSubscription(); // Get a subscription (future style)

List<Object> issues;

if (subscription.hasNext()) {

try {

issues = subscription.waitNext(); // Publication will be returned immediately since

// last invocation of hasNext() returned true.

System.out.println("Issues were already there: " + issues);

} catch (Exception e) {

e.printStackTrace();

}

} else {

System.out.println("No issues yet, waiting for incoming issues...");

}

for (;;) {

try {

issues = subscription.waitNext(); // Block until next publication arrives.

} catch (Exception e) {

e.printStackTrace();

break;

}

System.out.println("I get new issue(s): " + issues);

}

}

}

class PlayboyMagzine implements Serializable {

public PlayboyMagzine(String issue) {}

private static final long serialVersionUID = 1L;

}

class NewIssueNotifier implements SubscriberDataObserver {

public void handleData(String dataId, List<Object> data) {

System.out.println("New issue(s) arrived: " + data);

}

}