COMP 3111: Software Engineering

ExternalFunctions.cs Documentation

TO USE THE METHODS IN EXTERNALFUNCTIONS.CS

1. Add the following directive to your code file.

```
using HKeInvestWebApplication.ExternalSystems.Code_File;
```

2. Declare an instance of the ExternalFunctions class, such as

```
ExternalFunctions myExternalFunctions = new ExternalFunctions();
```

CURRENCY DATA METHODS

Retrieve All Available Currencies and their Exchange Rates

```
public DataTable getCurrencyData()
```

Parameters: none.

Return value: a `DataTable` containing all the available currencies and their exchange rates.
`currency` column contains the three-character currency code.
`rate` column contains the corresponding exchange rate to the Hong Kong dollar.

`null` - an error occurred in retrieving the currency information.

Example usage: `DataTable dtCurrency = myExternalFunctions.getCurrencyData();`

Retrieve All Available Currencies Only

```
public DataTable getCurrencies()
```

Parameters: none.

Return value: a `DataTable` containing all the available currencies.
`currency` column contains the three-character currency code.

`null` - an error occurred in retrieving the currency information.

Example usage: `DataTable dtCurrencies = myExternalFunctions.getCurrencies();`

Retrieve the Exchange Rate for a Specific Currency

```
public decimal getCurrencyRate(string currency)
```

Parameters: `currency` - the three-character currency code for which the exchange rate should be returned.

Return value: a `positive number` representing the exchange rate of the specified currency to the Hong Kong dollar.

`-1` an error occurred in retrieving the currency information.

Example usage: `decimal usdRate = myExternalFunctions.getCurrencyRate("USD");`

SECURITIES DATA METHODS

Retrieve All the Data for a Specific Security Type

```
public DataTable getSecuritiesData(string securityType)
```

Parameters: `securityType` - the security type; valid values are `bond`, `stock`, `unit trust`.

Return value: a `DataTable` containing the information for all of the securities of the specified type.

The specific columns in the DataTable depend on the specified security type.

if the security type is "bond", the columns returned are `code`, `name`, `launchDate`, `base`, `size`, `price`, `sixMonths`, `oneYear`, `threeYears`, `sinceLaunch`.

if the security type is "stock", the columns returned are `code`, `name`, `close`, `changePercent`, `changeDollar`, `volume`, `high`, `low`, `peRatio`, `yield`.

if the security type is "unit trust", the columns returned are `code`, `name`, `launchDate`, `base`, `size`, `price`, `riskReturn`, `sixMonths`, `oneYear`, `threeYears`, `sinceLaunch`.

`null` - the specified security type is not a valid security type or an error occurred in retrieving the security information.

Example usage:
```
DataTable dtBond =
myExternalFunctions.getSecuritiesData("bond");
```

Retrieve All the Data for a Specific Security Type Where the Name Matches All or Part of a Specified String

```
public DataTable getSecuritiesByName(string securityType, string securityName)
```

Parameters: `securityType` - the security type; valid values are `bond`, `stock`, `unit trust`

`securityName` - a string used to match against the names of the specified security type using the SQL "like" operator. The matching is case insensitive. The wild card character "%" is automatically prepended and appended to the input string.

Return value: a `DataTable` containing the information for the securities of the specified type whose name contains `securityName` (see also "Retrieve All the Data for a Specific Security Type").

`null` - the specified security type is not a valid security type, or there is no security whose name contains the specified string, or an error occurred in retrieving the security information

Example usage:
```
DataTable dtBond =
myExternalFunctions.getSecuritiesByName("bond", "global");
```

Retrieve All the Data for a Specific Security Where the Code Matches a Specified Code

```
public DataTable getSecuritiesByCode(string securityType, string securityCode)
```

Parameters: `securityType` - the security type; valid values are `bond`, `stock`, `unit trust`.

`securityCode` - a string containing the security code to match against the codes of the specified security type.

Return value: a `DataTable` containing the information for the securities of the specified type whose code matches exactly `securityCode` (see also "Retrieve All the Data for a Specific Security Type").

`null` - the specified security type is not a valid security type, or the security code does not exist, or an error occurred in retrieving the security information.

Example usage:
```
DataTable dtBond =
myExternalFunctions.getSecuritiesByCode("bond", "22");
```

Retrieve the Current Price of a Specific Security

```
public decimal getSecuritiesPrice(string securityType, string securityCode)
```

Parameters: `securityType` - the security type; valid values are `bond`, `stock`, `unit trust`

`securityCode` - the code of the security

Return value: a `positive number` representing the current price of the specified security

`-1` the specified security type or security code do not exist or an error occurred in retrieving the security

Example usage:
```
decimal price = myExternalFunctions.getSecuritiesPrice(“bond”, “22”);
```

SECURITIES BUY ORDER METHODS

Bond Buy Order

```
public string submitBondBuyOrder(string code, string amount)
```

Parameters: `code` - the security code of the bond to buy.

`amount` - the Hong Kong dollar amount to buy.

Return value:

`!null`	the order reference number (the order is successfully submitted and pending execution).
`null`	the order is not successfully submitted because it is not valid (i.e., the specified code does not exist or the specified amount is not numeric).

Example usage:

```
string result = myExternalFunctions.submitBondBuyOrder("22",
"10000");
```

Stock Buy Order

```
public string submitStockBuyOrder(string code, string shares, string orderType,
      string expiryDay, string allOrNone, string highPrice, string stopPrice)
```

Parameters: `code` - the security code of the stock to buy.

`shares` - the number of shares of the stock to buy (must be a multiple of 100).

`orderType` - the type of stock buy order; valid values are `market`, `limit`, `stop`, `stop limit`.

`expiryDay` - the expiry day of the buy order; valid values are `1`, `2`, `3`, `4`, `5`, `6`, `7`.

`allOrNone` - valid values are `Y` (order is all or none) or `N` (order is not all or none).

`highPrice` - the highest buying price in Hong Kong dollars or the empty string.

`stopPrice` - the stop price in Hong Kong dollars or the empty string.

Return value:

`!null`	the order reference number (the order is successfully submitted and pending execution).
`null`	the order is not successfully submitted because it is not valid (e.g., the specified code does not exist, the specified amount is not numeric, etc.).

Example usage:

```
string result = myExternalFunctions.submitStockBuyOrder("22",
"900", "stop limit", "2", "Y", "10", "9");
```

Unit Trust Buy Order

```
public string submitUnitTrustBuyOrder(string code, string amount)
```

Parameters: `code` - the security code of the unit trust to buy.

`amount` - the Hong Kong dollar amount to buy.

Return value:

`!null`	the order reference number (the order is successfully submitted and pending execution).
`null`	the order is not successfully submitted because it is not valid (i.e., the specified code does not exist or the specified amount is not numeric).

Example usage:

```
string result =
myExternalFunctions.submitUnitTrustBuyOrder("22", "10000");
```

SECURITIES SELL ORDER METHODS

Bond Sell Order

```
public string submitBondSellOrder(string code, string shares)
```

Parameters: `code` - the security code of the bond to sell.

`shares` - the number of shares of the bond to sell.

Return value: `!null` the order reference number (the order is successfully submitted and pending execution).

`null` the order is not successfully submitted because it is not valid (i.e., the specified code does not exist or the specified shares to sell is not numeric).

Example usage:
```
string result = myExternalFunctions.submitBondSellOrder("22", "20.1");
```

Stock Sell Order

```
public string submitStockSellOrder(string code, string shares, string orderType,
      string expiryDay, string allOrNone, string lowPrice, string stopPrice)
```

Parameters: `code` - the security code of the stock to sell.

`shares` - the number of shares of the stock to sell.

`orderType` - the type of stock sell order; valid values are `market`, `limit`, `stop`, `stop limit`.

`expiryDay` - the expiry day of the sell order; valid values are `1`, `2`, `3`, `4`, `5`, `6`, `7`.

`allOrNone` - valid values are `Y` (order is all or none) or `N` (order is not all or none).

`lowPrice` - the lowest selling price in Hong Kong dollars or the empty string.

`stopPrice` - the stop price in Hong Kong dollars or the empty string.

Return value: `!null` the order reference number (the order is successfully submitted and pending execution).

`null` the order is not successfully submitted because it is not valid (e.g., the specified code does not exist, the specified shares to sell is not numeric, etc.).

Example usage:
```
string result = myExternalFunctions.submitStockSellOrder("22", "800", "stop limit", "7", "N", "8", "9");
```

Unit Trust Sell Order

```
public string submitUnitTrustSellOrder(string code, string shares)
```

Parameters: `code` - the security code of the unit trust to sell.

`shares` - the number of shares of the unit trust to sell.

Return value: `!null` the order reference number (the order is successfully submitted and pending execution).

`null` the order is not successfully submitted because it is not valid (i.e., the specified code does not exist or the specified shares to sell is not numeric).

Example usage:
```
string result = myExternalFunctions.submitUnitTrustSellOrder("22", "300.4");
```

ORDER AND TRANSACTION METHODS

Order Status

```
public string getOrderStatus(string referenceNumber)
```

Parameters: `referenceNumber` - the order reference number.

Return value: the `status` of the order, which is one of `pending`, `partial`, `completed`, `cancelled`.

`null` - the order reference number is not valid.

Example usage:
```
string status = myExternalFunctions.getOrderStatus(“00000055”);
```

Order Transactions

```
public DataTable getOrderTransaction(string referenceNumber)
```

Parameters: `referenceNumber` - the order reference number.

Return value: a `DataTable` containing the transactions for the order.
- `transactionNumber` column contains the transaction number.
- `referenceNumber` column contains the order reference number.
- `executeDate` column contains the date the transaction was executed.
- `executeShares` column contains the quantity of shares executed by the transaction.
- `executePrice` column contains the price per share in the base currency at which the transaction was executed.

`null` - the order reference number is not valid.

Example usage:
```
DataTable transactions =
myExternalFunctions.getOrderTransaction(“00000055”);
```