|  |
| --- |
| QIBOL |
| **DeepSee BI Dashboard HOW-TO** |
|  |

|  |
| --- |
|  |

Оглавление

[Постановка Задачи 2](#_Toc343639046)

[Этапы 2](#_Toc343639047)

[Требования к технологиям 2](#_Toc343639048)

[Описание проектных решений 2](#_Toc343639049)

[1. Загрузка данных с Finam 2](#_Toc343639050)

[2. KPI Классы и фильтры 2](#_Toc343639051)

[3. Классы виджетов (представление данных) 2](#_Toc343639052)

[Поэтапная разработка 3](#_Toc343639053)

[1. Развёртывание сервера разработки 3](#_Toc343639054)

[2. Загрузка данных с Finam 3](#_Toc343639055)

[3. KPI Классы и фильтры 4](#_Toc343639056)

[ Finam.Data: 4](#_Toc343639057)

[ Finam.Ticker: 5](#_Toc343639058)

[ JQueryWidgets.CandleKPI: 6](#_Toc343639059)

[ JQueryWidgets.TickerData: 7](#_Toc343639060)

[4. Классы виджетов (представление данных) 8](#_Toc343639061)

[ JQueryWidgets.CandleView: 8](#_Toc343639062)

[ JQueryWidgets.CandleWidjet: 14](#_Toc343639063)

[ JQueryWidgets.HorizontalBarView: 16](#_Toc343639064)

[ JQueryWidgets.HorizontalBarWidget: 16](#_Toc343639065)

[ JQueryWidgets.VerticalBarView: 16](#_Toc343639066)

[ JQueryWidgets.VerticalBarWidget: 17](#_Toc343639067)

# Постановка Задачи

Разработать библиотеку бизнес-виджетов для возможности использовать в DeepSee отображения данных KPI. Виджеты должны поддерживать основные браузеры (IE, FireFox, Chrome, Opera, Safari) и мобильные версии браузеров с использованием особенностей touch-навигации. Данные для виджетов поступают ежедневно с небольшим интервалом времени с ресурса Finam, преобразуются в KPI-запрос для построения диаграмм.

# Этапы

1. Разработать средства загрузки данных с ресурса fiman.ru. Зараработать классы Data и Ticker для хранения истории данных;
2. Реализовать KPI классы через SQL запросы к классам Data и Ticker. Таким же образом определить фильтры к KPI  по наименованию компании, году и месяцу биржевых данных;
3. Разработать классы виджетов, которые на основе KPI запросов формируют представление данных.

# Требования к технологиям

* основные браузеры (IE, FireFox, Chrome, Opera, Safari);
* мобильные версии браузеров;
* использованием особенностей touch-навигации;
* быстрота отображения даже на плохом соединении;
* полноценное использование возможностей технологии DeepSee dashboard.

# Описание проектных решений

## Загрузка данных с Finam

Данные загружаются с ресурса finam.ru ежедневно через файловый экспорт (возможен любой другой интервал на усмотрение устанавливающего). Список загружаемых тикеров определен изначально, загружен вручную и не изменяется (берутся все тикеры, торгуемые на ММВБ). Интервал загрузки новых данных - 1 день, интервал актуальности данных - также 1 день (котировкой за день считается котировка на момент закрытия биржи).

## KPI Классы и фильтры

KPI классы описаны SQL запросами к Cache’ классам Data и Ticker, в которых хранятся данные экспорта с ресурса finam.ru. Таким же образом определяются фильтры к KPI. Реализованы фильтры по наименованию компании, году и месяцу биржевых данных.

## Классы виджетов (представление данных)

В качестве фреймворка для написания клиентской части виджетов был выбран JQuery. Данный выбор обусловлен следующими факторами:

1. Высокая скорость разработки: одной командой в JQuery можно сделать то, на что понадобится десятки строк простого JavaScript;
2. Небольшой размер и удобочитаемость кода: вытекает из предыдущего пункта;
3. Кроссбраузерность: в jquery уже реализована поддержка популярных браузеров, это не нужно реализовывать самому;
4. Потенциально высокая скорость отображения написанных на JQuery приложений;
5. Огромный выбор готовых плагинов (расширений), среди которых:
   * Необходимые нам виджеты: японские свечи, столбиковые диаграммы, круговые диаграммы
   * Плагин для поддержки мобильных устройств, в котором уже есть поддержка touch-интерфейсов
6. Хорошо реализованная событийная модель;
7. Возможность легко и быстро конфигурировать внешний вид интерфейса.

# Поэтапная разработка

## Развёртывание сервера разработки

Для разработки наиболее комфортно использовать сервер на Linux, но это дело вкуса. На сервере должны быть установлены:

1) Cache 2012.1

2) JRE

Для работы с Finam необходимо создать область, в которую помещаются данные. Можно использовать и любую из имеемых областей. Для работы с Finam области обязательно должны присутствовать классы Finam.Ticker и Finam.Data (см. часть 3).

Для ежедневной загрузки данных с Finam необходимо установить крон примерно такого содержания:

30 21 \* \* \* java -cp /home/dev/finamobtainer.jar finamobt.Main

Для Windows это называется Scheduled Task.

## Загрузка данных с Finam

Загрузка данных осуществляется с помощью JAVA-приложения finamobtainer.jar. Приложение запускается как

java -cp finamobtainer.jar finamobt.Main [конфигурационный файл]

В качестве параметра задается путь к файлу конфигурации. По умолчанию приложение ищет файл с именем config. Доступные параметры:

1) path: каталог, в который складывать загруженные файлы (по умолчанию /home/dev/finam\_import), обязательно без слэша на конце;

2) dateFrom: дата начала интервала в формате ГГГГ-ММ-ДД (по умолчанию текущая дата);

3) dateTo: дата окончания интервала в формате ГГГГ-ММ-ДД (по умолчанию текущая дата);

4) dbNamespace: область Cache (по умолчанию USER, мы можем поставить здесь FINANCE);

5) dbLogin: пользователь Cache (по умолчанию \_SYSTEM);

6) dbPwd: парот\ль Cache (по умолчанию sys).

Значения конфигурации задаются построчно в формате (ключ):(значение).

Важно принять во внимание, что приложение не проверяет уже загруженные в систему данные, т. е., если Вы 2 раза запустили импорт на одни и те же даты, данные в системе появятся 2 раза.

Также интересной особенностью финама является то, что если просить данные за вчера, он отдает данные за вчера и сегодня (за сегодня пустые). Возможно, этот баг уже исправлен.

## KPI Классы и фильтры

* 1. Запустим Cache Studio;
  2. Выполним настройку соединения с сервером разработки;
  3. Создадим новую область. Название может быть любым, например, “FINANCE”;
  4. Создадим следующие классы:

### Finam.Data

Данный класс описывает отдельную запись источника данных, соответствующего данным, выгружаемым с Finam. Все поля, присутствующие в данных с Finam, представлены в данном классе.

Class Finam.Data Extends %Persistent  
{  
  
Property Open As %Float [ Required, SqlFieldName = openCol ];  
  
Property High As %Float [ Required, SqlFieldName = highCol ];  
  
Property Low As %Float [ Required, SqlFieldName = lowCol ];  
  
Property Close As %Float [ Required, SqlFieldName = closeCol ];  
  
Property Volume As %Float;  
  
Property CreatedDate As %Date [ Required, SqlFieldName = dateCol ];  
  
Property CreatedTime As %Time;  
  
Index CreatedIndex On (CreatedDate, CreatedTime) [ Data = (CreatedDate, CreatedTime) ];  
  
Relationship Ticker As Finam.Ticker [ Cardinality = one, Inverse = hrefIO, Required ];  
  
Index TickerIndex On Ticker;  
  
Projection JavaInternalIO As %Projection.Java;  
  
ClassMethod ParseFile(file As %String) As %Status  
{  
 s stream=##class(%FileCharacterStream).%New()  
 s stream.Filename = file  
   
 While 'stream.AtEnd {  
 s line = stream.ReadLine()  
 s list=$listfromstring(line,",")  
   
 s tid = ##class(Finam.Ticker).FindTicker($lg(list,1))  
 // если 0 или "", то такого тиккера еще нет. Добавляем  
 if ((tid = 0) || (tid = "")){  
 s tick = ##class(Finam.Ticker).%New()   
 s tick.Code = $lg(list,1), tick.Cost = $lg(list,8)  
 d tick.%Save()  
 }  
 else{  
 s tick = ##class(Finam.Ticker).%OpenId(tid)  
 }  
   
 s nio = ##class(Finam.Data).%New()  
 // DD/MM/YY  
 s nio.CreatedDate = $zdh($lg(list, 3), 4, -1, 6)  
 // HH:MM:SS  
 s nio.CreatedTime = $zth($lg(list, 4), 1)  
 s nio.Open = $lg(list, 5), nio.High = $lg(list,6)  
 s nio.Low = $lg(list,7), nio.Close = $lg(list,8)  
 s nio.Volume = $lg(list,9), nio.Ticker = tick  
   
 d nio.%Save()  
 d tick.%Save()  
 }  
 q $$$OK  
}  
  
Method %OnBeforeSave(insert As %Boolean) As %Status [ Private, ServerOnly = 1 ]  
{  
 //вставляем отметку времени, если она не была проставлена  
 if (..CreatedDate = ""){  
 s ..CreatedDate = $p($h,",",1)  
 }  
 if (..CreatedTime = ""){  
 s ..CreatedTime = $p($h,",",2)  
 }  
 Quit $$$OK  
}  
  
}

### Finam.Ticker

Данный класс описывает тикер, который торгуется на бирже ММВБ.

Class Finam.Ticker Extends %Persistent  
{  
  
Property Code As %String;  
  
Index CodeIndex On Code [ Data = Code, Unique ];  
  
Property Title As %String;  
  
Property Cost As %Float;  
  
Relationship hrefIO As Finam.Data [ Cardinality = many, Inverse = Ticker ];  
  
Projection JavaTicker As %Projection.Java;  
  
ClassMethod FindTicker(code) As %Integer  
{  
 &sql(SELECT ID INTO :res FROM Finam.Ticker WHERE Code = :code)  
 if (SQLCODE'=0) {q 0}  
 q res  
}  
  
}

### JQueryWidgets.CandleKPI

Данный класс описывает KPI для источника данных, используемого для отображения японских свечей.

/// JQueryWidgets.CandleKPI  
Class JQueryWidgets.CandleKPI Extends %DeepSee.KPI  
{  
  
/// Этот блок XData содержит определение KPI  
XData KPI [ XMLNamespace = "http://www.intersystems.com/deepsee/kpi" ]  
{  
<kpi xmlns="http://www.intersystems.com/deepsee/kpi"  
 name="JQueryWidgets/CandleKPI" sourceType="sql"  
 caption="KPI title"  
 sql="select TO\_CHAR(D.dateCol,'DD/MM/YYYY'), D.openCol, D.closeCol, D.highCol, D.lowCol, T.Code, TO\_CHAR(D.dateCol,'YYYY') as yearCol, TO\_CHAR(D.dateCol,'MONTH') as monthCol from Finam.Data AS D, Finam.Ticker AS T WHERE D.Ticker = T.%ID"  
>  
  
<property name="dateCol" displayName="date" columnNo="1" />  
<property name="openCol" displayName="open" columnNo="2" />  
<property name="closeCol" displayName="close" columnNo="3" />  
<property name="highCol" displayName="high" columnNo="4" />  
<property name="lowCol" displayName="low" columnNo="5" />  
  
<filter name="Year" sql="select DISTINCT(TO\_CHAR(D.dateCol, 'YYYY')) from Finam.Data as D"/>  
<filter name="Month" sql="select DISTINCT(TO\_CHAR(D.dateCol, 'MONTH')) from Finam.Data as D" dependsOn="Year"/>  
<filter name="Ticker" sql="select T.Code, T.Title from Finam.Ticker as T" />  
</kpi>  
}  
  
/// Notification that this KPI is being executed.  
/// This is a good place to override properties, such as range and threshold.  
Method %OnLoadKPI() As %Status  
{  
 Quit $$$OK  
}  
  
/// This callback is invoked from a dashboard when an action defined by this dashboard is invoked.  
ClassMethod %OnDashboardAction(pAction As %String, pContext As %ZEN.proxyObject) As %Status  
{  
 // pAction is the name of the action (as defined in the XML list).  
 // pContext contains information from the client  
 // and can be used to return information.  
 Quit $$$OK  
}  
  
Method %OnGetSQL(ByRef pSQL As %String) As %Status  
{  
   //this is the start of the SQL query for this KPI  
    Set pSQL = "select TO\_CHAR(D.dateCol,'DD/MM/YYYY'), D.openCol, D.closeCol, D.highCol, D.lowCol, T.Title, T.Code, TO\_CHAR(D.dateCol,'YYYY') as yearCol, TO\_CHAR(D.dateCol,'MONTH') as monthCol from Finam.Data AS D, Finam.Ticker AS T "  
      
    Set where = "WHERE D.Ticker = T.%ID "  
    //look at %filterValues to see if a filter has been applied to this KPI instance  
    If $IsObject(..%filterValues) {  
        If (..%filterValues.Ticker'="")   
        {  
            // Call utility method that returns filter data in convenient format  
            Set sqlstring=..%GetSQLForFilter("T.Code", "Ticker")  
            Set where = where\_"AND "\_sqlstring  
        }  
        If (..%filterValues.Year'="")  
        {  
         Set sqlstring = " AND "\_..%GetSQLForFilter("TO\_CHAR(D.dateCol, 'YYYY')","Year")  
         Set where = where\_sqlstring  
         If (..%filterValues.Month'="")   
        {  
             Set sqlstring = " AND "\_..%GetSQLForFilter("TO\_CHAR(D.dateCol,'MONTH')","Month")  
             Set where = where\_sqlstring  
            }  
    }  
  
    // assemble the SQL statement  
    Set pSQL=pSQL\_where  //\_groupby\_orderby  
 Quit $$$OK  
}  
}  
  
}

### JQueryWidgets.TickerData

Данный класс описывает отдельную запись источника данных, используемого для виджета, отображающего японские свечи. Каждая запись (экземпляр данного класса) содержит следующие поля: open (стоимость акции на момент открытия торговой сессии), close (стоимость акции на момент закрытия торговой сессии), high (максимальная стоимость акции), low (минимальная стоимость акции) и createdDate (дата торговой сессии).

Class JQueryWidgets.TickerData Extends (%Persistent, %Populate, %ZEN.DataModel.Adaptor) [ SqlTableName = Ticker ]  
{  
  
/// Date  
Property Date As %Date(POPSPEC = "Date()") [ Required, SqlFieldName = dateCol ];  
  
Index DateIndex On Date [ Unique ];  
  
Property open As %Double [ Required, SqlFieldName = openCol ];  
  
Property close As %Double [ Required, SqlFieldName = closeCol ];  
  
Property high As %Double [ Required, SqlFieldName = highCol ];  
  
Property low As %Double [ Required, SqlFieldName = lowCol ];  
  
}

## Классы виджетов (представление данных)

Теперь займёмся созданием классов представления данных (ВАЖНО: для корректной работы класса в дальнейшем снимем флаг read-only c CACHELIB):

### JQueryWidgets.CandleView

Первый класс будет включать методы отрисовки виджета японских свеч. Начнём с объявления класса:

Class JQueryWidgets.CandleView Extends (%ZEN.Component.component, %ZEN.Component.dataView) {

Определим namespace:

Parameter NAMESPACE = "http://www.intersystems.com/deepsee";

Далее подключим требуемые библиотеки jqplot:

Parameter INCLUDEFILES = "script:jquery/jquery-1.7.2.min.js, jquery/jquery.mobile-1.1.1.min.js, jquery/jquery.jqplot.min.js, jquery/plugins/jqplot.dateAxisRenderer.min.js, jquery/plugins/jqplot.ohlcRenderer.min.js, jquery/plugins/jqplot.highlighter.min.js, jquery/plugins/jqplot.cursor.min.js, jquery/plugins/jqplot.categoryAxisRenderer.min.js, jquery/plugins/jqplot.canvasAxisTickRenderer.min.js, jquery/plugins/jqplot.canvasTextRenderer.min.js, jquery/plugins/jqplot.mobile.min.js, jquery/jquery.jqplot.min.css";

Теперь инициализируем поля класса:

**/// Определим исходные свойства отображения виджета**  
Property chartType As %ZEN.Datatype.string(VALUELIST = ",CANDLESTICK,SIMPLE CANDLE") [ InitialExpression = "CANDLE" ];  
**/// Свойства, соответствующие мерам**  
Property openName As %ZEN.Datatype.string;  
Property closeName As %ZEN.Datatype.string;  
Property highName As %ZEN.Datatype.string;  
Property lowName As %ZEN.Datatype.string;  
**/// Свойство, соответствующее подписям осей**  
Property labelProp As %ZEN.Datatype.string;  
**/// Свойство, соответствующее тому, является ли ось абсцисс - датой**  
Property xAxisIsDate As %ZEN.Datatype.boolean;

Далее опишем методы класса:

Первый метод будет отвечать за инициализацию javascript-объекта нашего виджета.

ClientMethod initChart(chartDiv, labelProp) [ Language = javascript ]  
{

В нём будем выполнять следующие действия:

- удалим предыдущий объект, если он существовал:

if (this.\_chartObject)  
 this.\_chartObject.destroy();  
 this.\_chartObject = null;

- создадим новый объект chartObject:

var chartDiv = this.getEnclosingDiv();  
 chartDiv.innerHTML = "";

- инициализируем оси:

$(document).ready(function(){   
 $.jqplot.config.enablePlugins = true;   
 this.xAxisIsDate = true;  
   
 if (this.xAxisIsDate)  
 xAxisObj = {  
 renderer:$.jqplot.DateAxisRenderer,  
 tickOptions:{   
       fontSize:'10pt',   
        fontFamily:'Tahoma',   
        angle:-40  
      }  
 }  
 else  
 xAxisObj = {  
 renderer:$.jqplot.CategoryAxisRenderer,  
 ticks:this.\_chartTicks,   
 rendererOptions:{  
 tickRenderer:$.jqplot.CanvasAxisTickRenderer  
      },  
 tickOptions:{   
     fontSize:'10pt',   
     fontFamily:'Tahoma',   
     angle:-40  
     }  
 }

- инициализируем виджет:

var plot2 = $.jqplot(chartDiv.getAttribute('id').replace(/([\$\'\"\/])/g, "\\$1"),  
 [this.\_chartData],  
 {

   title: labelProp,  
   axesDefaults:{},  
   axes: {  
   xaxis: xAxisObj,  
     yaxis: {  
     tickOptions:{ prefix: '$' }  
     }  
   },  
   series: [  
   {  
 renderer:$.jqplot.OHLCRenderer,   
 rendererOptions:{candleStick:true,upBodyColor:"#68D332"}  
 }   
 ],  
   cursor:{  
     zoom:true,  
   tooltipOffset: 10,  
     tooltipLocation: 'nw'  
   },  
   highlighter: {  
     showMarker:false,  
   tooltipAxes: 'xy',  
     yvalues: 4,  
     formatString:'<table class="jqplot-highlighter"> \  
     <tr><td>dimension:</td><td>%s</td></tr> \  
     <tr><td>open:</td><td>%s</td></tr> \  
     <tr><td>hi:</td><td>%s</td></tr> \  
     <tr><td>low:</td><td>%s</td></tr> \  
     <tr><td>close:</td><td>%s</td></tr></table>'  
   }  
 });

this.\_chartObject = plot2;  
 });  
 }

Теперь создадим метод, который устанавливает размер области, стили:

/// Client-side method to render control.  
ClientMethod renderContents() [ Language = javascript ]  
{  
 var chartDiv = this.getEnclosingDiv();  
   
 if (chartDiv == null) return;  
 this.getContainerSizes();  
 chartDiv.style.width = this.width;  
 chartDiv.style.height = this.height;

вызывает метод получения данных от конроллера:

// get data from controller, if present.  
 this.acquireData(this.openName, this.closeName, this.highName, this.lowName);

и инициализации виджета:

// create chart  
 this.initChart(chartDiv, this.labelProp);   
}

Теперь напишем метод, который получает размеры области, в которой виджету необходимо отрисоваться:

ClientMethod getContainerSizes() [ Language = javascript ]  
{  
 var chartDivId = this.getEnclosingDiv().id.split("/");  
 delete chartDivId[chartDivId.length - 1];  
 --chartDivId.length;  
 var search = chartDivId.join("/");  
 var tgt = document.getElementById(search);  
 if ((typeof(tgt) === "object") && (tgt !== null) && (typeof(tgt.style) === "object")) {  
 this.width = (parseInt(tgt.style.width) - 10) + "px";  
 this.height = (parseInt(tgt.style.height) - 80) + "px";  
 } else {  
 if (typeof(console) === "object")  
 console.log("Chart container is undefined");  
 }  
}

Следующий метод будет возвращать объект виджета:

ClientMethod getChartObject() [ Language = javascript ]  
{  
 return this.\_chartObject;  
}

Следующий метод будет вызываться после загрузки страницы или перерисовки виджета. В него передаются новые параметры высоты и ширины окна.

ClientMethod adjustContentSize(load, width, height) [ Language = javascript ]  
{

Этот метод будет устанавливать эти новые параметры:

var chartDiv = this.getEnclosingDiv();  
 width = width > 10 ? width : 0;  
 chartDiv.style.width = width-10 + 'px';  
 chartDiv.style.height = height + 'px';

А также вызывать повторную инициализацию виджета:

this.initChart(chartDiv);  
}

Теперь напишем метод для установки свойств виджета. В него передаётся имя свойства для установки и его новое значение:

/// Set the value of a named property.  
ClientMethod setProperty(property, value, value2) [ Language = javascript ]  
{  
 var chart = this.getChartObject();  
  
 switch(property) {  
 case 'chartType':  
 this[property] = value;  
 if (chart) {  
 this.initChart(this.getEnclosingDiv());  
 }  
 break;  
 default:  
 return this.invokeSuper('setProperty',arguments);  
 }  
 return true;  
}

Следующий метод будет вызываться контроллером при появлении внешних событий. В аргументе reason передаётся имя события, в остальных – данные, связанные с событием, но они в нашем случае не используются:

/// Notification that the dataController associated with this dataView has raised an event.  
ClientMethod notifyViewHandler(reason, data1, data2, data3) [ Language = javascript ]  
{

Этот метод при появлении событий dataChange и modelChange будет вызывать повторную инициализацию виджета, а остальные – игнорировать:

switch(reason) {  
 case 'dataChange':  
 case 'modelChange':  
 // some change in dataController  
 this.initChart(this.getEnclosingDiv());  
 break;  
 case 'seriesChange':  
 break;  
 }  
}

Определим метод, возвращающий данные из области виджета:

/// Return the dataSet object associated with this view.  
ClientMethod getDataSet() [ Language = javascript ]  
{  
 var controller = this.getController();  
 return (controller ? controller : this.dataSet);  
}

Следующий метод получает данные от контроллера и подготавливает их для передачи в jqplot-объект, преобразуя их в формат исходных данных виджета. Полный листинг метода смотрите в приложении:

/// Internal method: acquire data for this view and store a copy in a local cache.  
ClientMethod acquireData(openName, closeName, highName, lowName) [ Internal, Language = javascript ]  
{  
 …  
}

Ещё нам необходим метод, возвращающий ссылку на контрол виджета (например, на фильтр):

/// Obtain a reference to a filter (or any other widget control).  
ClientMethod getControl(ind) [ Language = javascript ]  
{  
 console("In getControl: " + ind);  
 return zenPage.getComponent(this.parent.parent.controlIndices[ind]);  
}

Следующий метод будет возвращать массив состояний фильтров виджета:

/// Get the array of filter state information for the widget.  
ClientMethod getFilterStateArray() [ Language = javascript ]  
{  
 return this.parent.parent.getFilterStateArray();  
}

Теперь определим метод, который вызывается при выборе любого фильтра в ui. Этот метод будет вызывать заново метод получения данных от контроллера, а также метод прорисовки виджета:

ClientMethod onApplyFilters(refresh) [ Internal, Language = javascript ]  
{  
 this.acquireData();  
 this.renderContents();  
 //zenInvokeCallbackMethod(this.onApplyFilters,this,'onApplyFilters','refreshChart','refresh');  
}

Далее напишем метод для обновления виджета:

ClientMethod refreshChart() [ Language = javascript ]  
{  
 this.getContainerSizes();  
 var chartDiv = this.getEnclosingDiv();  
 chartDiv.style.width = this.width;  
 chartDiv.style.height = this.height;  
   
 // test if we have not been rendered yet  
 if (!chartDiv)   
 return;

Он получает заново данные от контроллера:  
  
 // get data from controller, if present.  
 this.acquireData(this.openName, this.closeName, this.highName, this.lowName);

И перерисовывает виджет:  
  
 // create chart  
 this.initChart(chartDiv, this.labelProp);   
}

И последний метод будет просто вызывать родительский метод для генерации кода виджета:

Method %DrawHTML()  
{  
 do ##super()  
}

}

И не забудем поставить завершающую скобку класса:

}

### JQueryWidgets.CandleWidjet

Теперь напишем класс-интерфейс для того, чтобы Cache могла обращаться к методам класса *JQueryWidgets.CandleView*:

Class JQueryWidgets.CandleWidget Extends %DeepSee.Component.Widget.widget  
{

Первый метод будет возвращает информацию, использующуюся для отображения виджета в каталоге:

/// Return information used to list this widget within the "widget catalog".  
ClassMethod %GetCatalogInfo(Output pInfo As %List, Output pSubtypeList As %List) As %Status  
{  
 Kill pInfo,pSubtypeList  
 // $LB(name,caption,class,icon)  
 Set pInfo = $LB(..%ClassName(1), "Candle widget",..%ClassName(1), "deepsee/world\_48.gif")  
 Quit $$$OK  
}

Следующий метод будет определять интерфейс «параметров виджета» подтипа pSubtype в проектировщике виджетов. В нём определим поля pInfo и значения из KPI, по которым будет проводиться отрисовка виджетов:

/// Return information on additional "properties" supported by this widget for the given subtype.  
ClassMethod %GetWidgetPropertyInfo(pSubtype As %String, Output pInfo As %List) As %Status  
{  
 Kill pInfo  
  
 Set tTypeList = "ENUM^"\_$$$Text("Candle","%DeepSee")\_":CANDLE"  
  
 Set pInfo($I(pInfo)) = $LB("chartType","",tTypeList,$$$Text("Chart Type","%DeepSee"),$$$Text("Initial display type for the chart","%DeepSee"))  
 Set pInfo($I(pInfo)) = $LB("labelProp","Candle","%ZEN.Datatype.string","Chart title","Chart title")  
 Set pInfo($I(pInfo)) = $LB("openN","open","%ZEN.Datatype.string","Measure name : open value","Open value title of data")  
 Set pInfo($I(pInfo)) = $LB("closeN","close","%ZEN.Datatype.string","Measure name : close value","Close value title of data")  
 Set pInfo($I(pInfo)) = $LB("highN","high","%ZEN.Datatype.string","Measure name : high value","High value title of data")  
 Set pInfo($I(pInfo)) = $LB("lowN","low","%ZEN.Datatype.string","Measure name : low value","Low value title of data")  
 Set pInfo($I(pInfo)) = $LB("isDate",true, "%ZEN.Datatype.boolean", "Is X-axis date", "")  
   
 Quit $$$OK  
}

Следующий метод будет вызывать отрисовку самого виджета. Контроллер с данными применяется как наследник pGroup. Полный листинг метода смотрите в приложении:

/// This callback is responsible for creating the contents of the widget.  
Method %OnCreateWidget(pGroup As %ZEN.Component.group) As %Status  
{  
 …  
}

Следующий метод будет вызываться после загрузки страницы или перерисовки виджета. В него передаются новые параметры высоты и ширины окна. Он будет устанавливать эти новые параметры, а также вызывать повторную инициализацию виджета:

/// Called when page is loaded or widget is resized.  
ClientMethod adjustContentSize(load, width, height) [ Language = javascript ]  
{  
 var chart = this.findComponent('chart');  
 if (chart) {  
 var chartDiv = chart.getEnclosingDiv();  
 width = width > 10 ? width : 0;  
 chartDiv.style.width = width-10 + 'px';  
 chartDiv.style.height = height + 'px';  
 }  
}

Следующий метод будет возвращать контроллер с данными самого виджета, если такой есть:

/// If this widget contains a data controller, return it.  
/// This allows for generic capabilities in this base class.  
ClientMethod getDataController() [ Language = javascript ]  
{  
 return this.findComponent('controller');  
}

Следующий метод выполняет ту же роль что и предыдущий, но в COS:

/// If this widget contains a data controller, return it.  
Method %GetDataController() As %ZEN.Auxiliary.abstractController  
{  
 Quit ..%FindComponent("controller")  
}

Следующий метод будет вызываться перед перерисовкой виджета:

/// Test if this widget is ready to be resized.  
ClientMethod isReady() [ Internal, Language = javascript ]  
{  
 var controller = this.getDataController();  
 if (controller && controller.hasData) {  
 return controller.hasData();  
 }  
 return true;  
}

Ещё один метод будет вызываться при выборе любого фильтра в ui. Этот метод будет вызывать соответствующий метод из класса JQueryWidgets.CandleView:

ClientMethod onApplyFilters(refresh) [ Internal, Language = javascript ]  
{  
 var chart = this.findComponent('chart');  
 if (chart) chart.onApplyFilters(refresh);  
}

И последний метод будет вызывать обновление виджета при событиях перетаскивания на мобильных устройствах:

ClientMethod endDrag() [ Language = javascript ]  
{  
 var chart = this.findComponent('chart');  
 chart.refreshChart();  
}

И не забудем поставить завершающую скобку класса:

}

### JQueryWidgets.HorizontalBarView

Следующий класс будет включать методы отрисовки виджета горизонтальных столбиковых диаграмм. Содержание этого класса аналогично содержанию класса *JQueryWidgets.CandleView*, поэтому приводить мы его здесь не будем. Листинг класса можно посмотреть в приложении.

### JQueryWidgets.HorizontalBarWidget

Следующий класс – класс -интерфейс для того, чтобы Cache могла обращаться к методам класса *JQueryWidgets.HorizontalBarView*: Содержание этого класса аналогично содержанию класса *JQueryWidgets.CandleWidget*, поэтому приводить мы его здесь не будем. Листинг класса можно посмотреть в приложении.

### JQueryWidgets.VerticalBarView

Следующий класс будет включать методы отрисовки виджета вертикальных столбиковых диаграмм. Содержание этого класса аналогично содержанию класса *JQueryWidgets.HorizontalBarView*, поэтому приводить мы его здесь не будем. Листинг класса можно посмотреть в приложении.

### JQueryWidgets.VerticalBarWidget

Следующий класс – класс -интерфейс для того, чтобы Cache могла обращаться к методам класса *JQueryWidgets.VerticalBarView*: Содержание этого класса аналогично содержанию класса *JQueryWidgets. HorizontalBarWidget*, поэтому приводить мы его здесь не будем. Листинг класса можно посмотреть в приложении.

## Добавление виджетов в портал

* + 1. Скомпилируем проект;
    2. Откроем портал управления Cache;
    3. Выберем в меню слева «DeepSee»;
    4. Выберите область, которую Вы создали в пункте 3.с (например, «FINANCE»);
    5. Выберем «Портал пользователя»;
    6. Выберем «Меню» -> «Новая панель»;
    7. Укажем папку и название панели и нажмём «ОК»;
    8. Выберем «Меню» -> «Сохранить»;
    9. Далее выберем «Меню» -> «Добавить виджет»;
    10. Теперь выберем виджет, который хотим добавить на панель;
    11. Теперь в свойствах виджетов необходимо указать наименование полей, которые должны отображаться в виджете;
    12. Далее в меню «Источники данных» нажимаем на лупу рядом с полем «Источники данных» и выбираем интересующий KPI;
    13. А затем в меню «Свойства виджета», в области «Настройки виджета» указываются наименования полей. По умолчанию для области FINANSE ничего указывать не нужно. Далее нажимаем ОК.
    14. Всё, наш виджет создан.
    15. Напоследок добавим, что возможно, из-за зависимости классов или, например, потому что классы собирались в разной последовательности, придется несколько раз откомпилировать наш виджет, чтобы собрались все зависимые классы. Для этого, зависимости, можно указать в инспекторе класса параметр DependsOn