需要使用ChartView的话，需要注意：

## 1.必须在.pro文件中添加如下类容

|  |
| --- |
| QT += quick widgets charts |

## 2.在main.cpp中，

将QGuiApplication app(argc, argv);修改为QApplication **app**(*argc*, *argv*); 需要包含头文件：#include <QApplication>

## 3.在qml文件中

需要 import QtCharts 2.5

#### 1.PieSeries

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  ChartView{  width: 400  height: 300  theme:ChartView.ChartThemeBrownSand  PieSeries{  PieSlice{label: "success";value:69.9}  PieSlice{label: "failure";value:30.1}  }  }  } |  |

**注意：有一些属性的作用：angleSpan，是只读属性，表示每一个饼图所占的角度，borderColor，饼图的包括颜色，borderWidth，饼图的边框宽**

|  |
| --- |
|  |

默认的borderWidth是1，可以修改为0，borderColor是黑色，可以修改为“transparent”，这样子饼图就没有空隙了

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  //PieSeries demo  ChartView{  width: 400  height: 300  theme:ChartView.ChartThemeBrownSand  PieSeries{  PieSlice{  label: "success";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  value:69.9} //注意：value可以是任意数据，qml会根据数据自动计算所占的百分比  PieSlice{  label: "in between";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  value:20  }  PieSlice{  label: "failure";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  value:30.1  }  }  }    } |  |

exploded属性为true会将这一块饼图和整个饼图分开一个距离，具体距离由**explodeDistanceFactor指定当这个参数为0时，不会分开。**

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  //PieSeries demo  ChartView{  width: 400  height: 300  theme:ChartView.ChartThemeBrownSand  PieSeries{  PieSlice{  label: "success";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  exploded: true //设置分离  explodeDistanceFactor: 0.1 //控制分离的距离  value:69.9} //注意：value可以是任意数据，qml会根据数据自动计算所占的百分比  PieSlice{  label: "in between";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  value:20  }  PieSlice{  label: "failure";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  value:30.1  }  }  }    } |  |

**labelArmLengthFactor属性可以控制标签到饼图的距离，默认值是0.15**

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  //PieSeries demo  ChartView{  width: 400  height: 300  theme:ChartView.ChartThemeBrownSand  PieSeries{  PieSlice{  label: "success";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  exploded: true  explodeDistanceFactor: 0.1  labelArmLengthFactor: 0.1  value:69.9} //注意：value可以是任意数据，qml会根据数据自动计算所占的百分比  PieSlice{  label: "in between";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  value:20  }  PieSlice{  label: "failure";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  value:30.1  }  }  }    } |  |

labelPosition属性，默认值PieSlice.LabelOutside，可以修改，还可以修改startAngle和endAngle的值，不过注意一个圆周是360°的修改了一个必须修改另外一个洗的两个角度相差360°

还可以动态添加饼图slice

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  Button{ //实现动态添加的按钮  x:500  text:添加  width: 100  height: 50  Text {  id: *name*  text: *qsTr*("添加")  anchors.centerIn: *parent*  }  onClicked: {  var *psl* =*pseries*.append("workout",23.3)  *psl*.borderColor = "transparent"  *psl*.bordeoWidth = 0  *psl*.labelPosition = PieSlice.LabelInsideNormal  *psl*.labelVisible = true  }  }  //PieSeries demo  ChartView{  width: 400  height: 300  theme:ChartView.ChartThemeBrownSand  PieSeries{  id:*pseries*  size:1.2  PieSlice{  label: "success";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  exploded: true  explodeDistanceFactor: 0.05  labelArmLengthFactor: 0.1  labelPosition:PieSlice.LabelInsideNormal  value:69.9} //注意：value可以是任意数据，qml会根据数据自动计算所占的百分比  PieSlice{  label: "in between";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  labelPosition:PieSlice.LabelInsideNormal  value:20  }  PieSlice{  label: "failure";  labelVisible: true  borderColor: "transparent"  borderWidth: 0  labelPosition:PieSlice.LabelInsideNormal  value:30.1  }  }  }    } | 点击前    点击按钮后 |

可以利用ChartView的createSeries方法来创建饼图

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  //PieSeries demo  // ChartView{  // width: 400  // height: 300  // theme:ChartView.ChartThemeBrownSand  // PieSeries{  // id:pseries  // size:1.2  // PieSlice{  // label: "success";  // labelVisible: true  // borderColor: "transparent"  // borderWidth: 0  // exploded: true  // explodeDistanceFactor: 0.05  // labelArmLengthFactor: 0.1  // labelPosition:PieSlice.LabelInsideNormal  // value:69.9} //注意：value可以是任意数据，qml会根据数据自动计算所占的百分比  // PieSlice{  // label: "in between";  // labelVisible: true  // borderColor: "transparent"  // borderWidth: 0  // labelPosition:PieSlice.LabelInsideNormal  // value:20  // }  // PieSlice{  // label: "failure";  // labelVisible: true  // borderColor: "transparent"  // borderWidth: 0  // labelPosition:PieSlice.LabelInsideNormal  // value:30.1  // }  // }  // }    //点击按钮动态创建饼图  ChartView{  id:*cv*  title:"Series sample"  width: 400  height: 400  anchors.fill: *parent*  }  Column{  spacing: 5  width:100  height:300  Button{  x:520  width: 100  height: 50  Text {  id: *name*  text: *qsTr*("Series")  anchors.centerIn: *parent*  }  onClicked: {  //PieSeries  *cv*.title = "Pie Series sample"  var *pie* = *cv*.createSeries(ChartView.SeriesTypePie,"PieSeries")  *pie*.append("success",200)  *pie*.append("not good",100)  *pie*.labelVisible = true  *pie*.labelPosition = PieSlice.LabelInsideNormal    }  }    }    } | 点击按钮前    点击按钮后 |

#### 2.LineSeries

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")    //LineSeries  ChartView{  title:"LineSeries sample"  width: 400  height: 400  anchors.fill: *parent*  LineSeries{  name:"LineSeries"  XYPoint{x:1.0;y:0.0}  XYPoint{x:1.2;y:0.8}  XYPoint{x:1.7;y:1.0}  XYPoint{x:2.0;y:1.5}  XYPoint{x:2.5;y:1.7}  XYPoint{x:3.5;y:1.8}  XYPoint{x:5.0;y:2.9}  }  }  } |  |

注意在实际开发在很少这样子绘制曲线图，可以使用createSeries来绘制

这个类型提供很多series：

|  |  |
| --- | --- |
| ChartView.SeriesTypeLine | A line series. 折线图 |
| ChartView.SeriesTypeArea | An area series. 区域图 |
| ChartView.SeriesTypeBar | A bar series. 柱形图 |
| ChartView.SeriesTypeStackedBar | A stacked bar series. 往上堆叠的柱形图 |
| ChartView.SeriesTypePercentBar | A percent bar series. |
| ChartView.SeriesTypeBoxPlot | A box plot series. |
| ChartView.SeriesTypeCandlestick | A candlestick series. |
| ChartView.SeriesTypePie | A pie series. 饼图 |
| ChartView.SeriesTypeScatter | A scatter series. |
| ChartView.SeriesTypeSpline | A spline series.画曲线 |
| ChartView.SeriesTypeHorizontalBar | A horizontal bar series. |
| ChartView.SeriesTypeHorizontalStackedBar | A horizontal stacked bar series. |
| ChartView.SeriesTypeHorizontalPercentBar | A horizontal percent bar series. |

**使用createSeries方法来创建折线图需要设置坐标轴。需要用到ValueAxis类型**

**使用ValueAxis类型分别设置x轴和y轴的最大值和最小值，然后传入createSeries方法中**

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  //LineSeries  // ChartView{  // title:"LineSeries sample"  // width: 400  // height: 400  // anchors.fill: parent  // LineSeries{  // name:"LineSeries"  // XYPoint{x:1.0;y:0.0} //这样子绘制不好  // XYPoint{x:1.2;y:0.8}  // XYPoint{x:1.7;y:1.0}  // XYPoint{x:2.0;y:1.5}  // XYPoint{x:2.5;y:1.7}  // XYPoint{x:3.5;y:1.8}  // XYPoint{x:5.0;y:2.9}  // }  // }  ValueAxis{ //设置x轴的范围  id:*xAxis*  min:0  max:50  }  ValueAxis{ //设置y轴的范围  id:*yAxis*  min:0  max:50  minorTickCount: 3 //设置每一个格分成多少个小格  tickCount: 3//设置将坐标轴分成多少格  }  //折线图的比较好的绘制方法，使用createSeries,利用点击按钮的方式触发  ChartView{  id:*cv*  title:"Series sample"  width: 400  height: 400  anchors.fill: *parent*  }  Button{  x:520  width: 100  height: 50  Text {  id: *name*  text: *qsTr*("Series")  anchors.centerIn: *parent*  }  onClicked: {    *cv*.title = "Line Series sample"  var *line* = *cv*.createSeries(ChartView.SeriesTypeLine,"LineSeries",*xAxis*,*yAxis*)  *line*.append(10,10)  *line*.append(20,10)  *line*.append(30,20)  *line*.append(40,15)  *line*.append(50,60)  //利用循环来创建折线  // for(var i=0;i<20;i++){  // line.append(i,i\*2)  // }  }  }  } | 点击按钮前    点击按钮后 |

在实际的开发中，我们更多是利用property设置linePoints数组设置需要的点，然后利用循环来创建

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  property var linePoints:[*Qt*.point(10,10),*Qt*.point(20,10),*Qt*.point(30,20),*Qt*.point(40,15),*Qt*.point(50,60)]  ValueAxis{ //设置x轴的范围  id:*xAxis*  min:0  max:50  }  ValueAxis{ //设置y轴的范围  id:*yAxis*  min:0  max:50  minorTickCount: 3 //设置每一个格分成多少个小格  tickCount: 3//设置将坐标轴分成多少格  }  //折线图的比较好的绘制方法，使用createSeries,利用点击按钮的方式触发  ChartView{  id:*cv*  title:"Series sample"  width: 400  height: 400  anchors.fill: *parent*  }  Button{  x:520  width: 100  height: 50  Text {  id: *name*  text: *qsTr*("Series")  anchors.centerIn: *parent*  }  onClicked: {    //LineSeries  *cv*.title = "Line Series sample"  var *line* = *cv*.createSeries(ChartView.SeriesTypeLine,"LineSeries",*xAxis*,*yAxis*)  *line*.style = *Qt*.DashLine //修改线条的样式  //利用循环来创建折线  for(var *i*=0;*i*<*linePoints*.length;*i*++){  *line*.append(*linePoints*[*i*].x,*linePoints*[*i*].y)  }  }  }  } | 点击按钮前    点击按钮后 |

#### 3.BarSeries

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")    //BarSeries  ChartView{  title:"BarSeries sample"  width: 400  height: 400  legend.alignment: *Qt*.AlignBottom  anchors.fill:*parent*  BarSeries{  name:"Bar Series"  axisX: BarCategoryAxis{categories: ["2010","2011","2013","2014","2015","2016"]}  BarSet{label: "Jack";values:[2,2,3,4,5,6]}  BarSet{label: "Mary";values:[10,8,7,9,5,6]}  BarSet{label: "Jesse";values:[5,7,6,8,9,10]}  }  }  } |  |

可以使用ChartView的createSeries方法来创建柱形图，注意一定要设置x轴和y轴，而且必须设置正确的类型，例如y轴很多时候都是表示数值的，需要使用ValueAxis，使用错误的类型会导致结果错误

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("ChartView Demos")  BarCategoryAxis {//x轴  id:*xAxis*  categories: ["2007", "2008", "2009", "2010", "2011", "2012" ]  }  ValueAxis{ //y轴这个才是表示数值的轴，这里如果使用分类轴，结果会不正确  id:*yAxis*  min:0  max:60  }  //折线图的比较好的绘制方法，使用createSeries,利用点击按钮的方式触发  ChartView{  id:*cv*  title:"Series sample"  width: 400  height: 400  legend.alignment: *Qt*.AlignBottom  anchors.fill: *parent*  Component.onCompleted: {  var *bars* = *cv*.createSeries(ChartView.SeriesTypeBar,"bar series",*xAxis*,*yAxis*)  *bars*.append("cpp",[5,10,15,20,30,40])  *bars*.append("java",[2,3,4,5,6,7])  }  }  } |  |

#### 4、AreaSeries

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  //AreaSeries  ChartView {  title: "NHL All-Star Team Players"  anchors.fill: *parent*  antialiasing: true  // Define x-axis to be used with the series instead of default one  ValueAxis {  id: *valueAxis*  min: 2000  max: 2011  tickCount: 12  labelFormat: "%.0f"  }  AreaSeries {  name: "Russian"  axisX: *valueAxis*  upperSeries: LineSeries {  XYPoint { x: 2000; y: 1 }  XYPoint { x: 2001; y: 1 }  XYPoint { x: 2002; y: 1 }  XYPoint { x: 2003; y: 1 }  XYPoint { x: 2004; y: 1 }  XYPoint { x: 2005; y: 0 }  XYPoint { x: 2006; y: 1 }  XYPoint { x: 2007; y: 1 }  XYPoint { x: 2008; y: 4 }  XYPoint { x: 2009; y: 3 }  XYPoint { x: 2010; y: 2 }  XYPoint { x: 2011; y: 1 }  }  }  AreaSeries {  name: "Swidesh"  axisX: *valueAxis*  upperSeries: LineSeries {  XYPoint { x: 2000; y: 1 }  XYPoint { x: 2001; y: 1 }  XYPoint { x: 2002; y: 3 }  XYPoint { x: 2003; y: 3 }  XYPoint { x: 2004; y: 2 }  XYPoint { x: 2005; y: 0 }  XYPoint { x: 2006; y: 2 }  XYPoint { x: 2007; y: 1 }  XYPoint { x: 2008; y: 2 }  XYPoint { x: 2009; y: 1 }  XYPoint { x: 2010; y: 3 }  XYPoint { x: 2011; y: 3 }  }  }  AreaSeries {  name: "Finnish"  axisX: *valueAxis*  upperSeries: LineSeries {  XYPoint { x: 2005; y: 0 }  XYPoint { x: 2006; y: 1 }  XYPoint { x: 2007; y: 0 }  XYPoint { x: 2010; y: 0 }  XYPoint { x: 2011; y: 1 }  }  }  }  } |  |

也可以使用createSeries方法来动态创建AreaSeries

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("ChartView Demos")  //折线图的比较好的绘制方法，使用createSeries,利用点击按钮的方式触发  ChartView{  id:*cv*  title:"Series sample"  width: 400  height: 400  anchors.fill: *parent*  LineSeries {  id:*l1*  name: "LineSeries1"  color:"orange"  XYPoint { x: 0; y: 0 }  XYPoint { x: 1.1; y: 2.1 }  XYPoint { x: 1.9; y: 3.3 }  XYPoint { x: 2.1; y: 2.1 }  XYPoint { x: 2.9; y: 4.9 }  XYPoint { x: 3.4; y: 3.0 }  XYPoint { x: 4.1; y: 3.3 }  }  LineSeries {  id:*l2*  color:"lime"  name: "LineSeries2"  XYPoint { x: 0; y: 0 }  XYPoint { x: 1.1; y: 1.1 }  XYPoint { x: 1.9; y: 2.3 }  XYPoint { x: 2.1; y: 1.1 }  XYPoint { x: 2.9; y: 3.9 }  XYPoint { x: 3.4; y: 2.0 }  XYPoint { x: 4.1; y: 2.3 }  }  Component.onCompleted: {  *cv*.title = "Area Series sample"  var *areaSeries* = *cv*.createSeries(ChartView.SeriesTypeArea, "Area series");  *areaSeries*.upperSeries =*l1*  *areaSeries*.lowerSeries =*l2* //可以不设置lowerSeries,此时默认是0,注意：上下限可以交换，效果不变  *areaSeries*.color = "lightsteelblue"  *areaSeries*.opacity = 0.3  // var areaSeries2 = cv.createSeries(ChartView.SeriesTypeArea, "Area series");  // areaSeries2.upperSeries =l2  // areaSeries2.color = "lime"  }  }  } |  |