

QtQuick Training Course



Module Three

Objectives

1 All about lists

What is a model?

Models in QtQuick

ListView

GridView

Creating a model dynamically

2 Loading external data

XML model from network

SQLite database

Topics

- 1 All about lists
- 2 Loading external data
- 3 Questions
- 4 Lab

All about lists

Models and Lists

Models in QtQuick

ListView

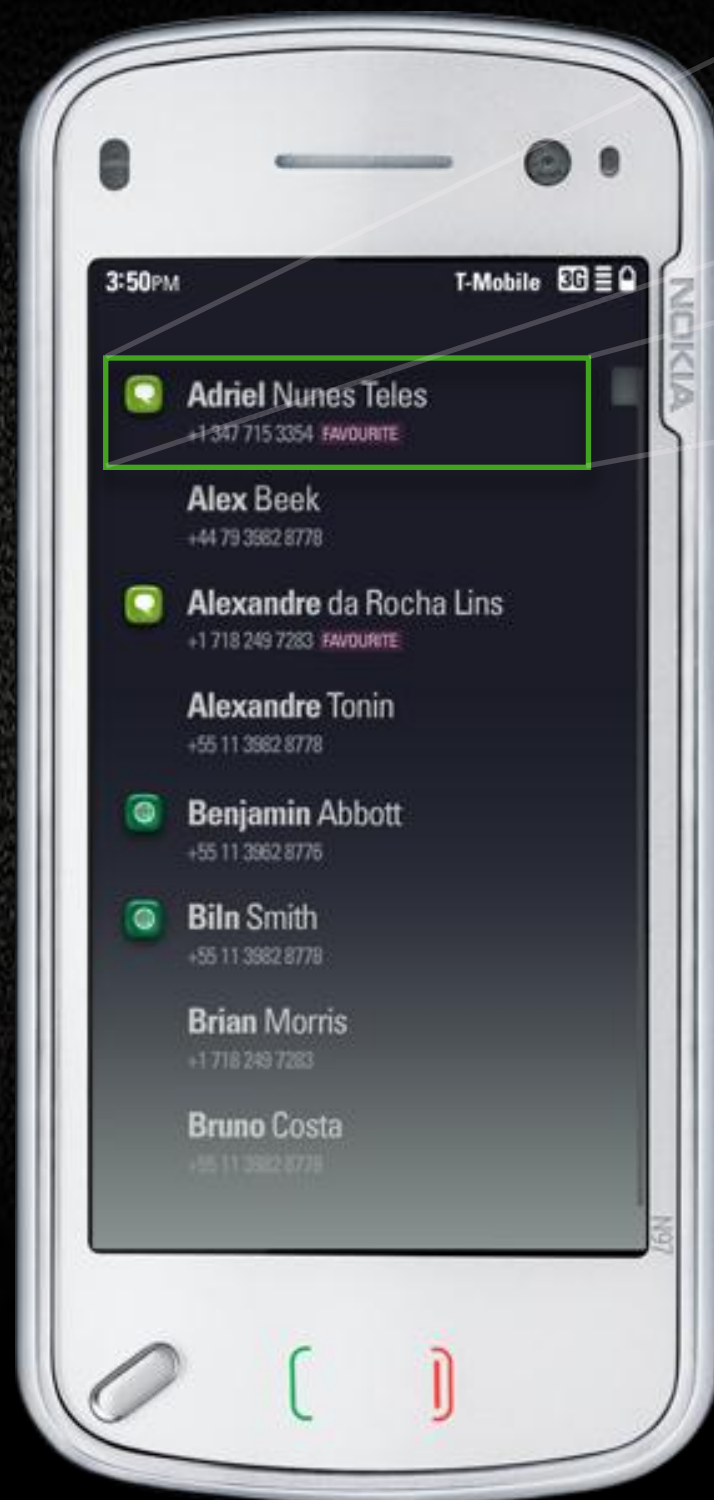
GridView


Javascript and ListModel

Animating list items

All about lists

What is a model?



 **Adriel Nunes Teles**
+1 347 715 3354 FAVOURITE

All information about one single item

Name

Surname

Telephone Number

Favorite

Icon

Models in QtQuick

This is how a list item is organized in a QtQuick model. The ListModel element is an array of ListElements



```
ListModel {  
    ListElement {  
        picture: "images/user_1.png"  
        name: "Adriel"  
        surname: "Nunes Teles"  
        telephone: "+1 347 715 3354"  
        favorite: true  
        icon: "sms"  
    }  
}
```

See example: [addon/module-003/examples/ListModelExample.qml](#)

All about lists

Models and Lists

Models in QtQuick

ListView

GridView

Javascript and ListModel

Animating list items

ListView

To start creating lists in QtQuick, you must organize your files first. You need, at least, three files

Single item view

All the visual elements from each list item and how they appear are in this file

ListModel

This is the file presented before, an array of ListElements containing all information from each item

Main file

This file will contain the ListView call, the ListModel and the Component for each single item view that will be repeated

ListView

The first thing to do is to create the list item layout file ...



```
...  
Text {  
    id: nameTxt  
    font.family: "Univers LT Std"  
    color: "#c8c8c8"  
    font.pixelSize: 22  
    text: "<b>Adriel</b> Nunes Teles"  
    anchors.left: icon.right  
    anchors.top: icon.top  
    anchors.topMargin: 8  
}  
...
```

See example: [addon/module-003/examples/ItemRenderDelegate.qml](#)

ListView

... then you need to make the item flexible enough to accept the variables you need

```
...  
property string intern_name  
property string intern_surname  
property string intern_telephone  
  
Text {  
    id: nameTxt  
    font.family: "Univers LT Std"  
    color: "#c8c8c8"  
    font.pixelSize: 22  
    text: "<b>" + intern_name " </b> " + intern_surname  
    anchors.left: icon.right  
    anchors.top: icon.top  
    anchors.topMargin: 8  
}  
...
```

See example: [addon/module-003/examples/ListItemDelegate.qml](#)

ListView

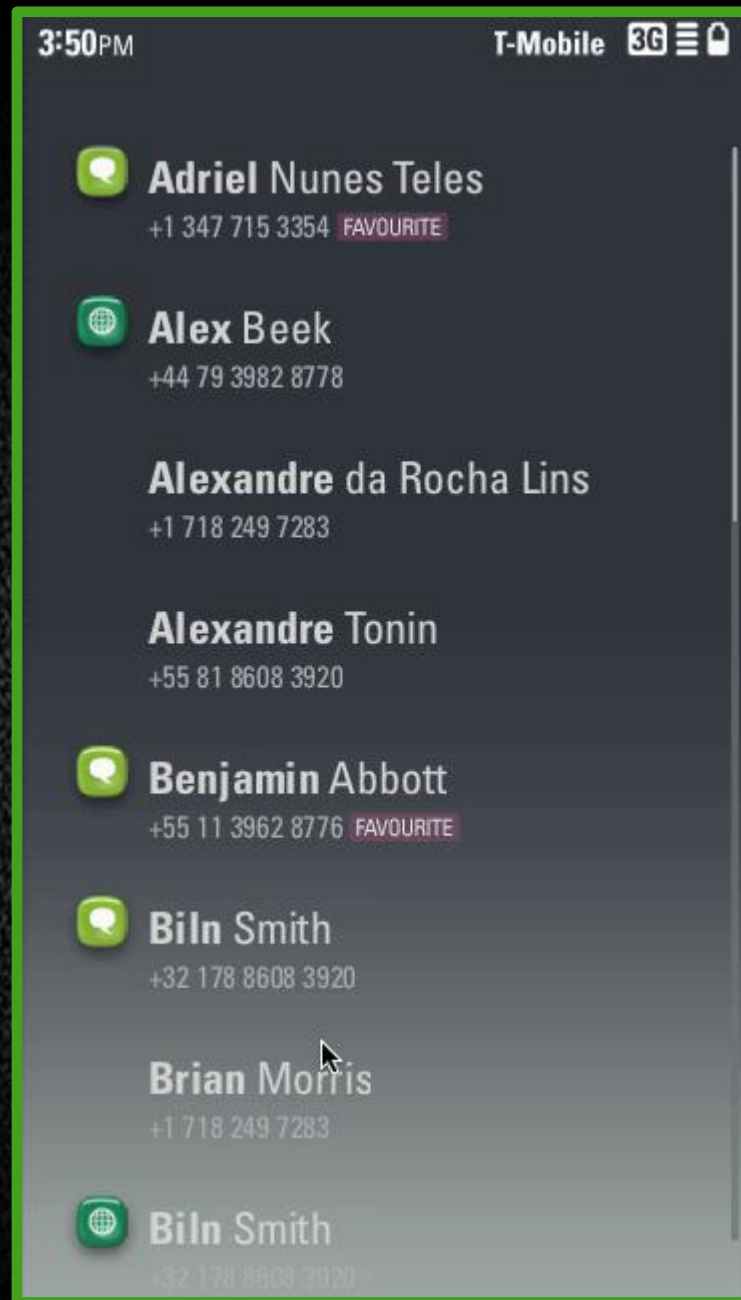
Once you've done all of this, now you just need to set the ListView inside the main file

```
ListView {  
    id: listExample  
    orientation: "Vertical"  
    model: ListModelExample  
    delegate: itemComponent  
}  
Component {  
    id: itemComponent  
    ListItemExample {  
        intern_name: model.name  
        intern_surname: model.surname  
        intern_telephone: model.telephone  
        intern_icon: model.icon  
    }  
}  
...
```

See example: [addon/module-003/examples/ListExample.qml](#)

All about lists

ListView



See video: [addon/module-003/videos/list-view.mov](#)

See example: [addon/module-003/examples/ListExample.qml](#)

All about lists

Models and Lists

Models in QtQuick

ListView

GridView

Javascript and ListModel

Animating list items

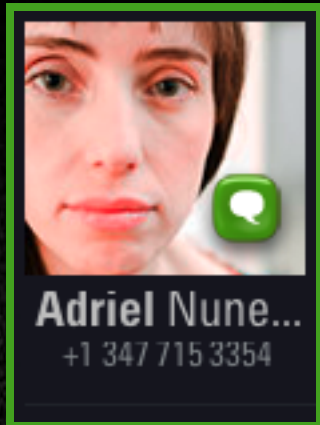
All about lists

GridView



Since you've created the model, now you can use the same information to create different views

GridView



You will need to realign the elements and use the picture element that was not used in the other view

```
...  
property string intern_name  
property string intern_surname  
property string intern_picture  
property string intern_icon  
property string intern_telephone  
  
Image {  
    id: picture  
    anchors.top: parent.top  
    anchors.topMargin: 15  
    source: intern_picture  
}  
...
```

See example: [addon/module-003/examples/GridItemDelegate.qml](#)

GridView

To finalize the new view, you will need to call the GridView element and set its specific parameters

```
...
GridView {
    id: listExample
    cellWidth: 110
    cellHeight: 160
    anchors.fill: parent
    model: ListModelExample
    delegate: itemComponent
}
Component {
    id: itemComponent
    GridItemDelegate {
        intern_name: model.name
        intern_surname: model.surname
        intern_picture: model.picture
    }
}
...
```

See example: [addon/module-003/examples/GridExample.qml](#)

All about lists

GridView



See video: [addon/module-003/videos/grid-view.mov](#)

See example: [addon/module-003/examples/GridExample.qml](#)

All about lists

Models and Lists

Models in QtQuick

ListView

GridView

Javascript and ListModel

Animating list items

Using Javascript to modify list elements

It is easy to add and modify item elements using ListModel properties and javascript. These are the main ListModel elements:

append - Adds new item to the end of the list

get - Returns an item property value at a specific index

set or **setProperty** - Changes the item value at specify index

insert - Adds new item at a specific index position

remove - Removes an item at a specific index

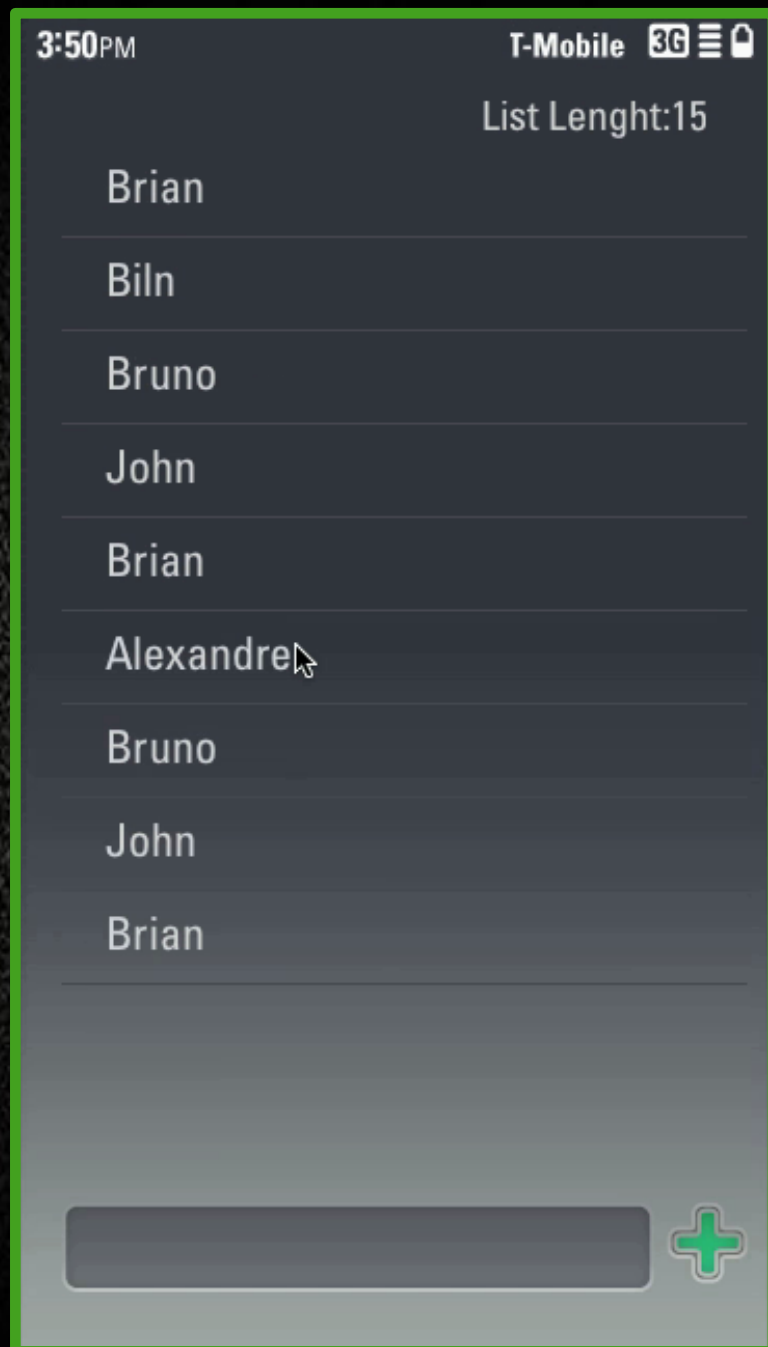
clear - Deletes all content from the model

In the next example, you will see how some of these elements work

To know more about ListModel properties:

<http://doc.qt.nokia.com/4.7-snapshot/qml-listmodel.html>

Using Javascript to add list elements



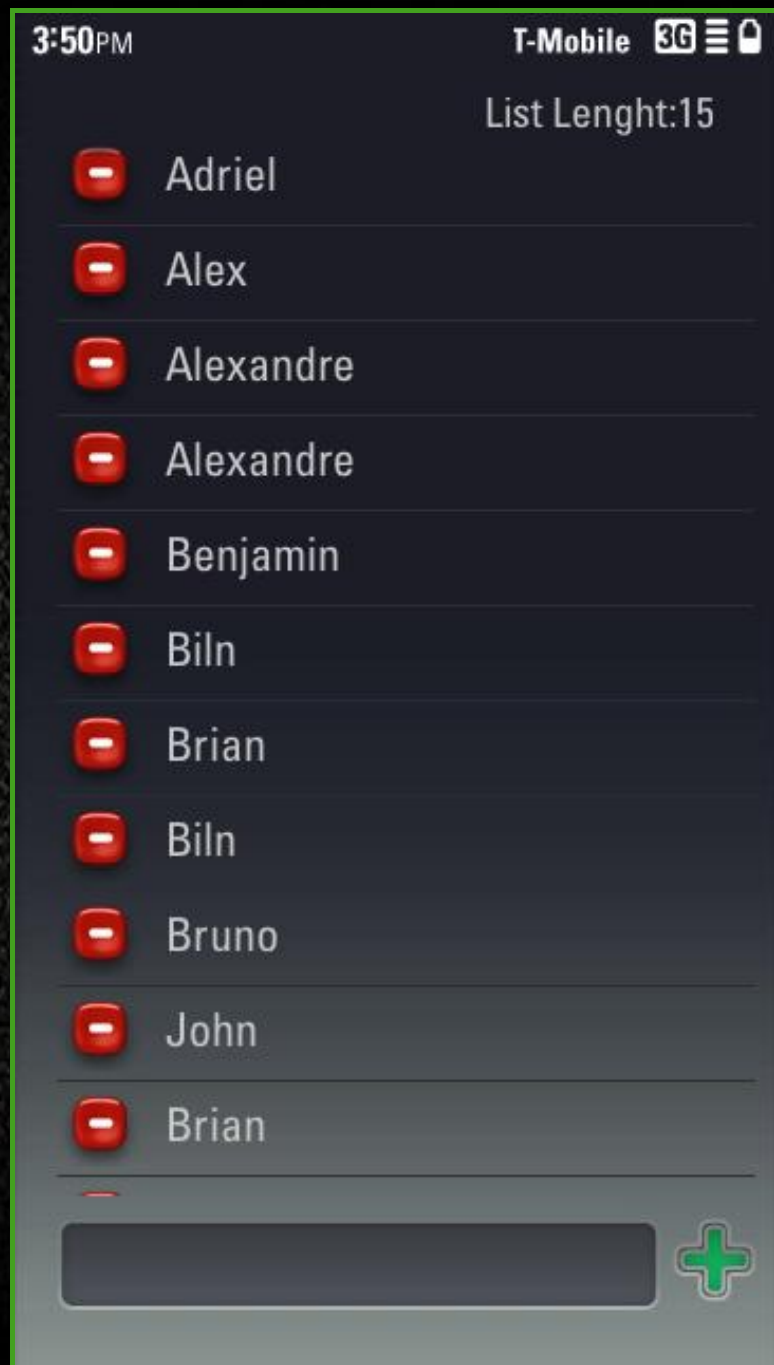
See video: [addon/module-003/videos/add.mov](#)

Changing the list a little bit to use just the contact name and adding a `TextInput` element can demonstrate how it is simple to add elements on the top of the list

```
...
MouseArea{
    anchors.fill: parent
    onClicked: {
        if(inputText.text != "") {
            listModel.insert(0, {"name": inputText.text});
            inputText.text = "";
        }
    }
}
...
insert
```

See example: [addon/module-003/examples/ListProperties.qml](#)

Using Javascript to remove list elements



See video: [addon/module-003/videos/remove.mov](#)

Adding a button in a list item and a simple action in it you can see how the remove property works.

```
Image {  
    id: deleteButton  
    source: "images/delete.png"  
    anchors.left: parent.left  
  
    MouseArea {  
        anchors.fill: parent  
        onClicked: listModel.remove(index)  
    }  
}  
...
```

remove

See example: [addon/module-003/examples/ListPropertiesDelete.qml](#)

All about lists

Models and Lists

Models in QtQuick

ListView

GridView

Javascript and ListModel

Animating list items

And what about animations?

It is easy to create movement when you insert and remove items. The ListView element has its own properties that can deal with this.

ListView.onAdd / ListView.onRemove

```
ListView.onAdd: SequentialAnimation {
  PropertyAction { target: listItem; property: "height"; value: 0; }
  PropertyAction { target: listItem; property: "opacity"; value: 0.0; }
  NumberAnimation { target: listItem; property: "height"; to: 45; duration: 300; }
  NumberAnimation { target: listItem; property: "opacity" to: 1.0; duration: 300; }
}

ListView.onRemove: SequentialAnimation {
  PropertyAction { target: listItem; property: "ListView.delayRemove"; value: true; }
  NumberAnimation { target: listItem; property: "opacity"; to: 0.0; duration: 300; }
  NumberAnimation { target: listItem; property: "height"; to: 0; duration: 300; }
  PropertyAction { target: listItem; property: "ListView.delayRemove"; value: false; }
}
...
```

See example: [addon/module-003/examples/ListPropertiesMotion.qml](#)

All about lists

And what about animations?



There are many more ListView features that will be covered on the next topics, but the previous ones are the basics for you to start developing like a pro

To know more about ListView properties:

<http://doc.qt.nokia.com/4.7-snapshot/qml-listview.html>

See video: `addon/module-003/videos/list-motion.mov`

See example: `addon/module-003/examples/ListPropertiesMotion.qml`

Topics

- 1 All about lists
- 2 Loading external data
- 3 Questions
- 4 Lab

Loading external data

XmlListModel



Using the same view from the previous examples and just changing how the models are acquired, you can create a list using the Twitter public timeline feed

```
...
XmlListModel {
    id: xmlModel
    source: "http://twitter.com/favorites/113425681.rss"
    query: "/rss/channel/item"

    XmlRole { name: "pubDate"; query: "pubDate/string()" }
    XmlRole { name: "desc"; query: "description/string()" }
}
...
```

See video: [addon/module-003/videos/xmlmodel.mov](#)

See example: [addon/module-003/examples/ListXMLExample.qml](#)

XmlListModel Properties

QtQuick parses the XML in an easy way into a ListModel element. This is a RSS file example, the same one used on the previous example

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:atom="http://www.w3.org/2005/Atom" version="2.0" >
  <channel>
    <title>Twitter / Favorites from toptweets</title>
    <link>http://twitter.com/toptweets/favorites</link>

    <item>
      ...
      <description>planetjedward: Edward has torn his knee ligaments and will have to get
surgery. He is going to make a quick recovery and we are (cont)</description>
      <pubDate>Mon, 05 Jul 2010 13:03:11 +0000</pubDate>
      ...
    </item>
    ...
  </channel>
</rss>
```

The information that is necessary to make the previous example is in “<description>” and “<pubDate>”. They are “<item>” children

XmlListModel Properties

Just to clarify what is needed to build a ListModel, here is a simplified version from the previous RSS file and how a XmlListModel works

```
<rss>
  <channel>
    <item>
      <description></description>
      <pubDate></pubDate>
    </item>
  </channel>
</rss>
```

RSS

```
XmlListModel {
  id: xmlModel
  source: "http://twitter.com/favorites/113425681.rss"
  query: "/rss/channel/item"

  XmlRole { name: "pubDate"; query: "pubDate/string()" }
  XmlRole { name: "desc"; query: "description/string()" }
}
```

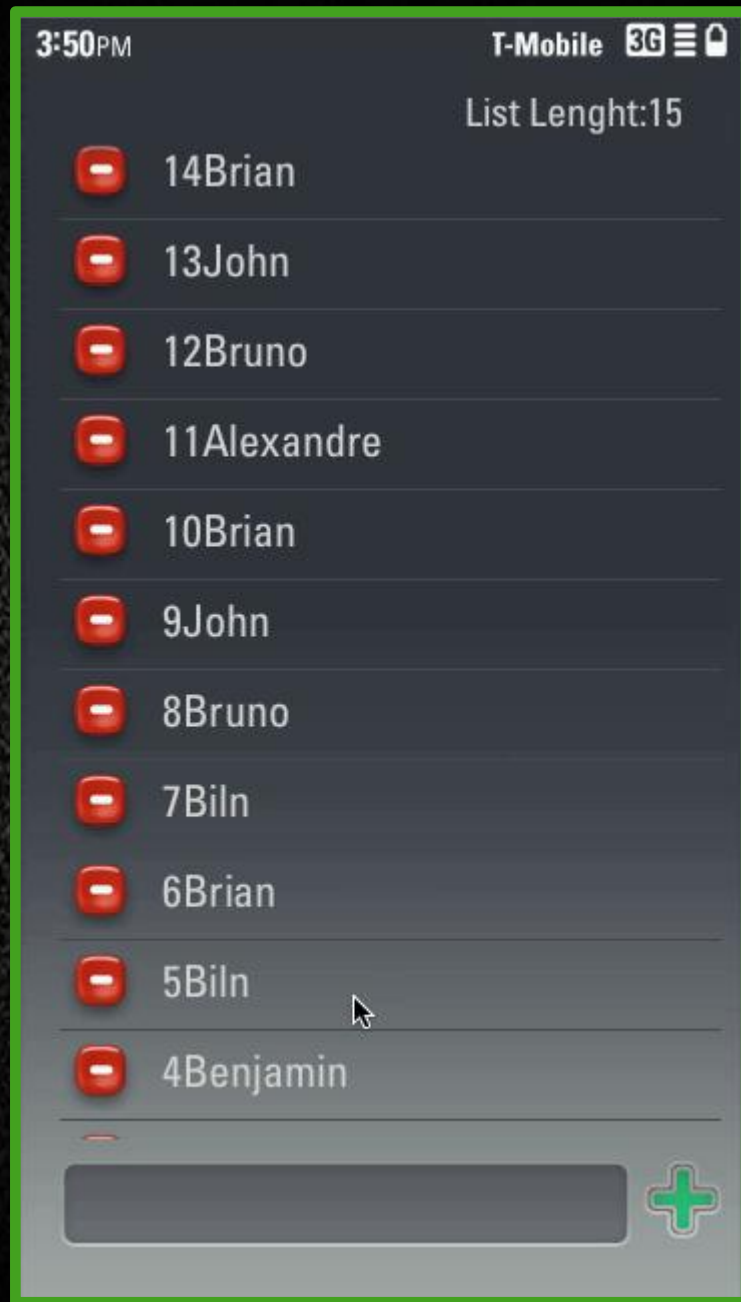
XmlListModel

query: "/rss/channel/item" - Sets the scope for the model creation. This means that only the "<item>" children will be watched

XmlRole { name: "pubDate"; query: "pubDate/string()" } - The XmlRole acts like a value inside a ListElement. All XmlRoles combined that create a ListElement

Loading external data

SQLite database



This is a simple example of how to create persistent data in a QtQuick app. If you are used to create databases in SQLite, there are almost no new things for you...

```
...
db = openDatabaseSync("Contacts" "1.0" "Contacts SQL",
1000000);
db.transaction(function(tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS
ContactsData(id INT, name STRING)');
    resultSet = tx.executeSql('SELECT * FROM ContactsData
ORDER BY id DESC');
});
...
```

See video: [addon/module-003/videos/sqlite-example.mov](#)

See example: [addon/module-003/examples/ListDBExample.qml](#)

Loading external data

SQLite database

...but now you will know how to populate a ListModel using data from a SQLite database

```
function populateModel(){
  for(var i = 0; i<=resultSet.rows.length-1; i++){
    var nameTxt = resultSet.rows.item(i).name;
    var indexNum = resultSet.rows.item(i).id;
    listModel.append({id:indexNum, name: nameTxt } ;
  }
}
...
```

You just need to use “append” that a new ListElement will be added to the end of the ListModel

Topics

- 1 All about lists
- 2 Loading external data
- 3 Questions
- 4 Lab

Questions

What is a Model?

What is a ListModel?

What do you need to create a ListView?

How do you add and remove list items?

How do you animate items that are inserted on a list?

What is a XmlRole?

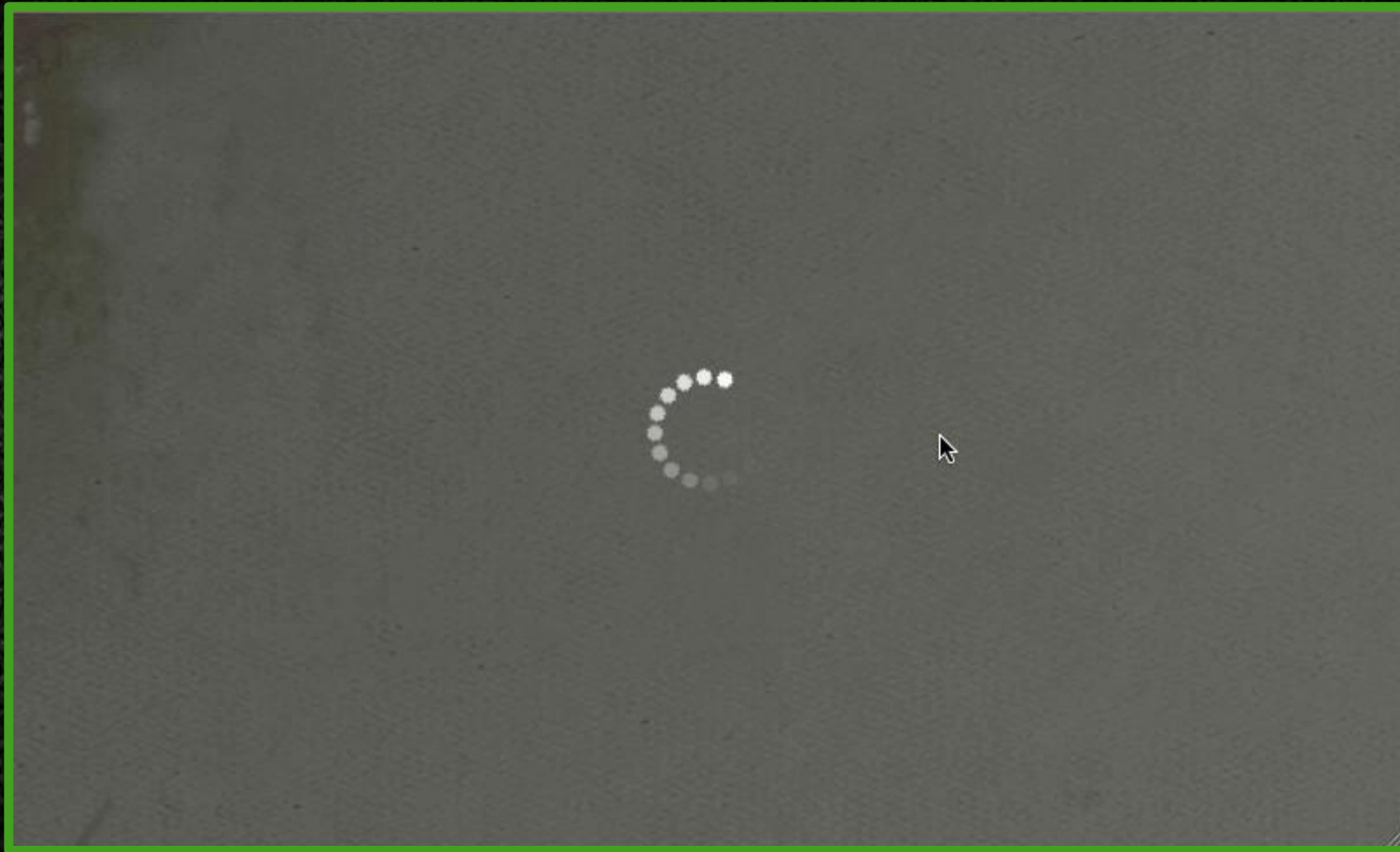
Which ListModel method adds an item to the end of the list?

Topics

- 1 All about lists
- 2 Loading external data
- 3 Questions
- 4 Lab

Lab

Create a horizontal list using XmlModelList and the twitter public timeline XML. Try to use snapMode's ListView property



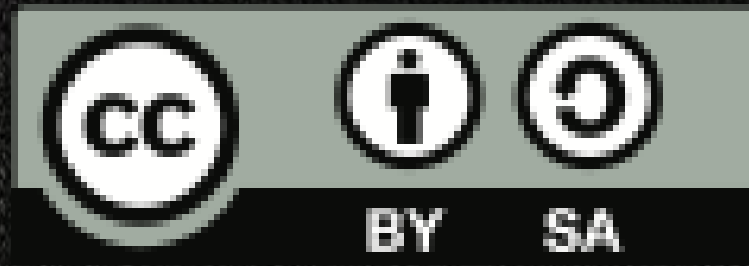
See video: [addon/module-003/videos/twitter_app.mov](#)

Optional: Create a loading

See lab: [labs/lab-three/lab-twitter/labThree.qmlproject](#)

(c) 2011 Nokia Corporation and its Subsidiary(-ies).

The enclosed Qt Training Materials are provided under the Creative Commons Attribution ShareAlike 2.5 License Agreement.



The full license text is available here: <http://creativecommons.org/licenses/by-sa/2.5/legalcode>

Nokia, Qt and the Nokia and Qt logos are the registered trademarks of Nokia Corporation in Finland and other countries worldwide.