МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Программная инженерия»

**ОТЧЁТ**

по лабораторной работе №6

**ИСПОЛЬЗОВАНИЕ JAVASCRIPT**

**Выполнил:** студент группы

382008-1

\_\_\_А.Д. Кандрин \_\_\_\_\_\_\_

Подпись

**Проверил:** младший научный сотрудник

\_\_\_\_Д.А. Карчков\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород  
2023 г.

## Содержание

[1. Постановка задачи 3](#_Toc27873363)

[2. ХОД РАБОТЫ](#_Toc27873364) 4

[3. ЗАКЛЮЧЕНИЕ](#_Toc27873365) 9

[4. ЛИТЕРАТУРА](#_Toc27873374) 10

## Постановка задачи.

Цель: научиться использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения.

## Ход работы.

1. Создадим приложение, которое позволяет отображать список из прямоугольников с использованием ListModel. В модели будут настраиваться цвет фона и текста внутри прямоугольника. Текст будет содержать название цвета фона прямоугольника.

ListModel {

id: dataModel

ListElement { colorFont: "black"; colorBackground: "white"; text: "Белый"; }

ListElement { colorFont: "white"; colorBackground: "black"; text: "Черный"; }

ListElement { colorFont: "red"; colorBackground: "blue"; text: "Синий"; }

}

SilicaListView {

anchors.fill: parent

model: dataModel

delegate: Rectangle {

width: parent.width

height: 100

color: model.colorBackground

Text {

anchors.centerIn: parent

text: model.text

color: model.colorFont

}

}

}

1. Создадим приложение, которое позволяет отображать список из прямоугольников. Нажатие на кнопку над списком добавит новый элемент. Нажатие на элемент в списке удалит его из списка. В прямоугольниках должен отображаться порядковый номер, присваиваемый при добавлении в список. При удалении элементов порядковые номера у добавленных прямоугольников остаются неизменными.

id: page

ListModel { id: dataModel }

Column {

width: page.width

Button {

id: addButton

anchors.horizontalCenter: parent.horizontalCenter

text: "Добавить"

property int counter: 1

onClicked: dataModel.append({ text: "Элемент " + counter++ })

}

SilicaListView {

VerticalScrollDecorator{}

width: parent.width;

height: 800

header: PageHeader { title: "Список" }

model: dataModel

delegate: Rectangle {

width: parent.width; height: 70; color: "skyblue";

Text {

anchors.centerIn: parent

text: model.text

}

MouseArea

{

anchors.fill: parent

onClicked: dataModel.remove(model.index)

}

}

}

}

1. Выполним задание 1 с использованием javascript-модели.

property var dataModel: [

{ colorFont: "black", colorBackground: "white", text: "Белый" },

{ colorFont: "white", colorBackground: "black", text: "Черный" },

{ colorFont: "red", colorBackground: "blue", text: "Синий" }

]

SilicaListView {

anchors.fill: parent

model: dataModel

delegate: Rectangle {

width: parent.width

height: 100

color: modelData.colorBackground

Text {

anchors.centerIn: parent

text: modelData.text

color: modelData.colorFont

}

}

}

1. Получим и отобразим курсы валют из ресурса ЦБ РФ по адресу <http://www.cbr.ru/scripts/XML_daily.asp> .

XmlListModel {

id: xmlListModel

source: "http://www.cbr.ru/scripts/XML\_daily.asp"

query: "/ValCurs/Valute"

XmlRole { name: "name"; query: "Name/string()" }

XmlRole { name: "value"; query: "Value/string()" }

}

SilicaListView {

anchors.fill: parent

header: PageHeader { title: "Валюты" }

model: xmlListModel

delegate: Column {

x: Theme.horizontalPageMargin

Label { text: name }

Label { text: value }

}

}

1. Выполним задание 4 с использованием XMLHttpRequest.

function loadNews() {

var xhr = new XMLHttpRequest();

xhr.open('GET', 'https://www.cbr-xml-daily.ru/daily\_utf8.xml', true);

xhr.onreadystatechange = function() {

if (xhr.readyState === XMLHttpRequest.DONE) {

xmlListModel.xml = xhr.responseText;

}

}

xhr.send();

}

XmlListModel {

id: xmlListModel

query: "/ValCurs/Valute"

XmlRole { name: "name"; query: "Name/string()" }

XmlRole { name: "value"; query: "Value/string()" }

}

SilicaListView {

anchors.fill: parent

header: PageHeader { title: "Валюты" }

model: xmlListModel

delegate: Column {

x: Theme.horizontalPageMargin

Label { text: name }

Label { text: value }

}

}

Component.onCompleted: loadNews()

1. Создадим приложение, позволяющее добавлять и удалять заметки с использованием базы данных и отображать их в списке. Текстовое поле служит для ввода текста, кнопка для добавления заметки, нажатие на заметку удалит её.

DAO { id: dao }

Column

{

width: parent.width

TextField

{

id: textField

width: parent.width

placeholderText: "Введите заметку"

}

Button

{

text: "Добавить"

onClicked:

{

dao.insertNote(textField.text);

selectBooks();

}

}

SilicaListView {

width: parent.width;

height: 800

model: ListModel { id: notesListModel }

delegate: ListItem {

Label {

text: (index + 1) + ". " + description

MouseArea

{

anchors.fill: parent

onClicked:

{

dao.deleteNote(id);

selectBooks();

}

}

}

}

}

}

function selectBooks() {

notesListModel.clear();

dao.retrieveNotes(function(notes) {

for (var i = 0; i < notes.length; i++) {

var note = notes.item(i);

notesListModel.append({id: note.id, description: note.description});

}

});

}

Component.onCompleted: selectBooks()

// Код DataAccessObject:

Item {

property var db;

Component.onCompleted: {

db = LocalStorage.openDatabaseSync("notes", "1.0");

createTable();

}

function createTable() {

db.transaction(function(tx) {

tx.executeSql("CREATE TABLE IF NOT EXISTS notes (

id INTEGER PRIMARY KEY AUTOINCREMENT,"

+ "description TEXT NOT NULL);");

});

}

function retrieveNotes(callback) {

db.readTransaction(function(tx) {

var result = tx.executeSql("SELECT \* FROM notes;");

callback(result.rows);

});

}

function insertNote(description) {

db.transaction(function(tx) {

tx.executeSql("INSERT INTO notes (description) VALUES(?);", [description]);

});

}

function updateNote(id, description) {

db.transaction(function(tx) {

tx.executeSql("UPDATE notes SET description = ? WHERE id = ?;", [description, id]);

});

}

function deleteNote(id) {

db.transaction(function(tx) {

tx.executeSql("DELETE FROM notes WHERE id = ?;", [id]);

});

}

}

1. Создадим приложение с текстовым полем и полем с флажком, значение которых сохраняется в настройках приложения с помощью ConfigurationValue.

ConfigurationValue {

id: textFieldSetting

key: "/apps/Lab6/textField"

defaultValue: ""

}

ConfigurationValue {

id: switchSetting

key: "/apps/Lab6/textSwitch"

defaultValue: false

}

Column {

width: parent.width

PageHeader { title: "Settings" }

TextField {

width: parent.width

text: textFieldSetting.value

onTextChanged: textFieldSetting.value = text

}

TextSwitch

{

width: parent.width

text: checked ? "Включен" : "Выключен"

checked: switchSetting.value

onCheckedChanged: switchSetting.value = checked

}

}

1. Выполним задание 7 с помощью ConfigurationGroup.

ConfigurationGroup {

id: settings

path: "/apps/Lab6"

property string textField: ""

property bool textSwitch: false

}

Column {

width: parent.width

PageHeader { title: "Settings" }

TextField {

width: parent.width

text: settings.textField

onTextChanged: settings.textField = text

}

TextSwitch

{

width: parent.width

text: checked ? "Включен" : "Выключен"

checked: settings.textSwitch

onCheckedChanged: settings.textSwitch = checked

}

}

1. **Заключение.**

Мы научились использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения.

**4. Литература.**

1. Документация Qt: сайт. — URL: https://doc.qt.io (дата обращения: 01.03.2022). — Текст: электронный.