

JMIX + KOTLIN = ♡

Введение

- **Обзор на Kotlin**
 - История
 - Роль и популярность
 - Краткое сравнение с Java синтаксисом
- **Kotlin особенности**
- **Преимущества и недостатки**
- **Demo**
- **Заключение**



Берик Каж

Java Backend Dev



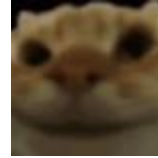
Freedom Bank Казахстан



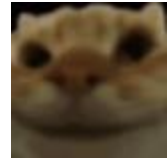
t.me/jojovka



Так ли важна скорость разработки?



Может ли Kotlin ускорить разработку с Jmix?



Стоит ли выбрать Kotlin вместо Java?



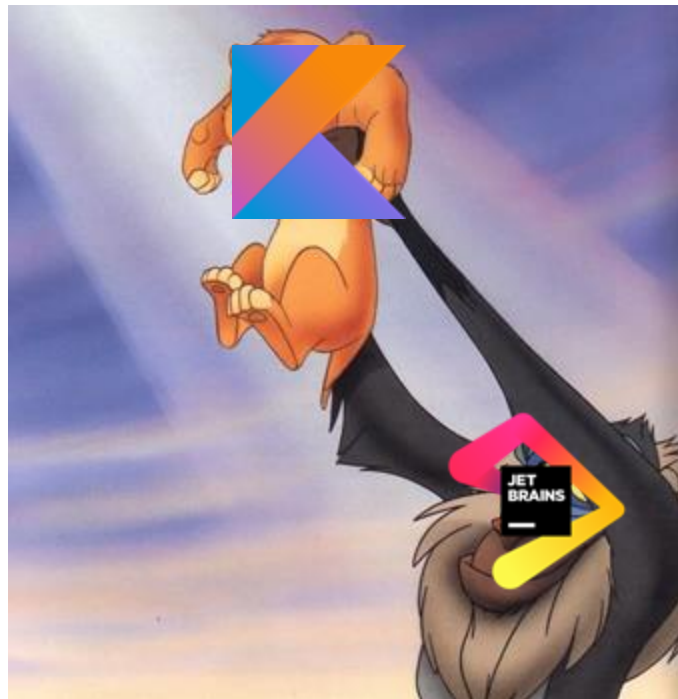
Обзор на Kotlin

История

- Современный
- Статически типизированный
- Кроссплатформенный
- Объектно-ориентированный
- Мультипарадигменный
- Функциональный
- Конвенциональный
- Авральный
- Ментальный
- Транснациональный
- Стиральный



Обзор на Kotlin. История



Обзор на Kotlin

Роль и популярность

- ❑ Stack Overflow Developer Survey 2022: 13 🏆
- ❑ JetBrains Developer Ecosystem Survey 2021: 52% 🧑
- ❑ GitHub Octoverse Report 2022: Top 10 📈
- ❑ Stack Overflow Developer Survey 2021: Top 8 🏆



Обзор на Kotlin.

Сравнение с Java синтаксисом

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println("Hello world!");
4     }
5 }
```

Java синтаксис main метода

```
Main.kt x
1 fun main(args: Array<String>) {
2     println("Hello World!")
3 }
```

Kotlin синтаксис main метода

```
Main.java x
1 import java.util.Arrays;
2 import java.util.List;
3
4 public class Main {
5     public static void main(String[] args) {
6         List<String> names = Arrays.asList("Alice", "Bob", "Charlie");
7         List<String> filtered = names.stream()
8             .filter(name -> name.startsWith("A"))
9             .toList();
10    }
11 }
```

Java синтаксис Лямбда выражений

```
Main.kt x
1 fun main(args: Array<String>) {
2     val names = listOf("Alice", "Bob", "Charlie")
3     val filtered = names.filter { it.startsWith(prefix: "A") }
4 }
```

Kotlin синтаксис Лямбда выражений

Kotlin особенности



Скорость разработки
Kotlin + Jmix



Null safety



Data классы



Sealed классы



Smart cast



Extension functions



Kotlin корутины



DSL

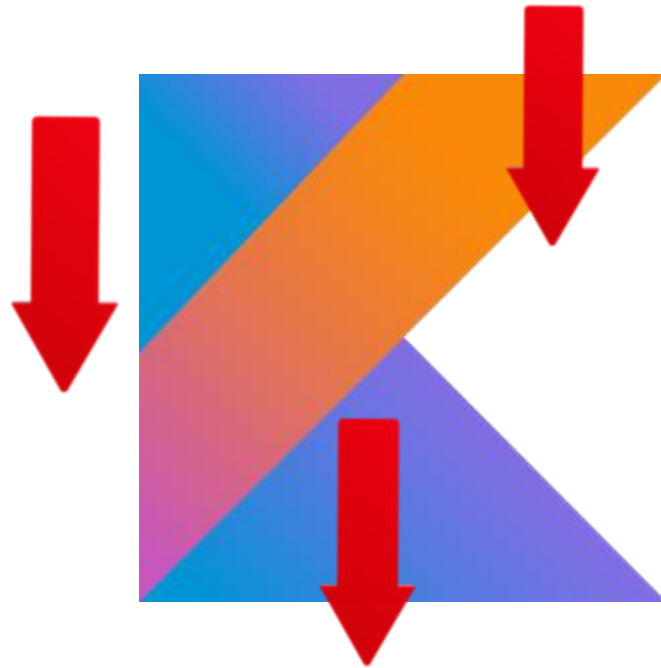
Преимущества Kotlin

- ☐ Значительное ускорение разработки
- ☐ Интеграция с существующей инфраструктурой



Недостатки Kotlin

- ❑ Ограничения в документации, сообществе и в ресурсах
- ❑ Могут возникать интеграционные проблемы



DEMO PROJECT

KotlinJmixExample

Application

Users

Expenses

User expenses

Security

Data Tools

[admin]

User expenses

Filter

Refresh

Add search condition

+ Create

Edit

Remove

4 rows

Expense	Amount	Date	Details
Fixed to...	2.5	29/10/2...	
Pasta	1	10/11/2...	Bought ...
Onay ca...	5,000	31/10/2...	Bus/Met...
Bribe to ...	3	03/11/2...	Now I h...

User

John [John]

Expense

Onay card replenishment

Amount

5,000

Date

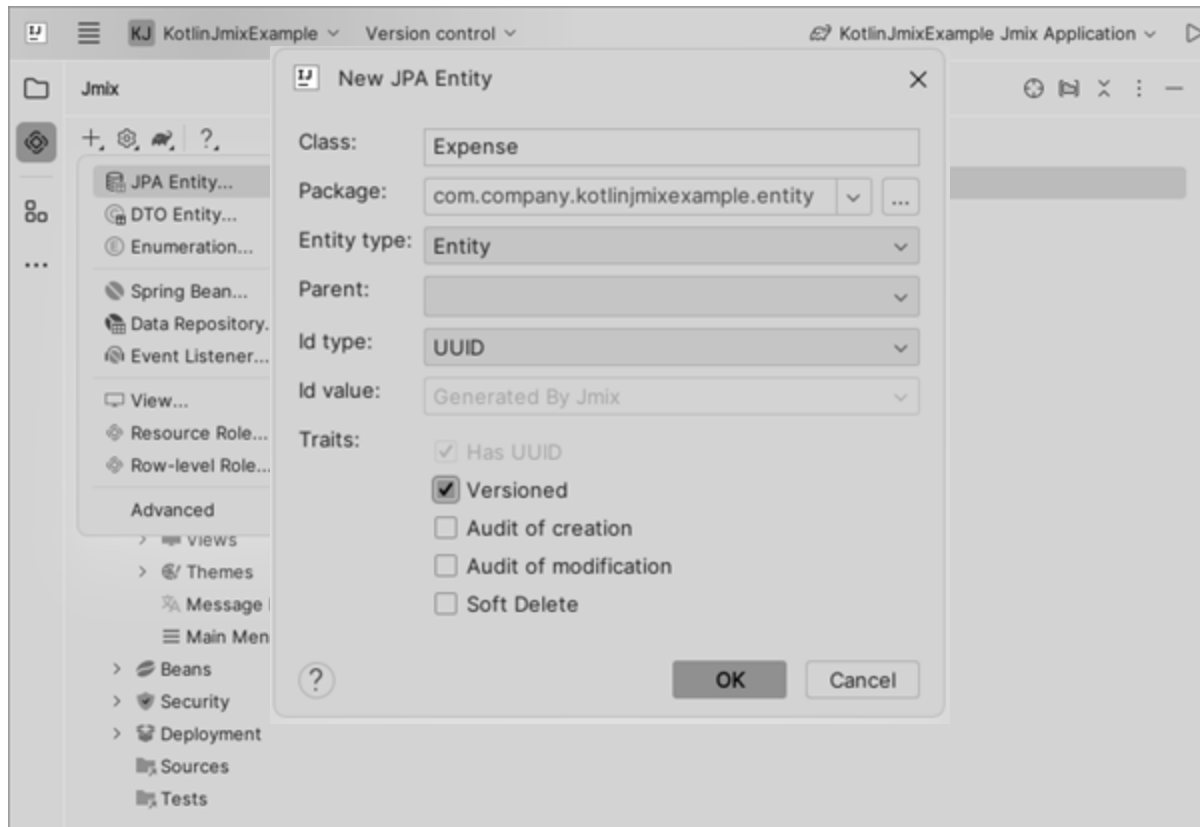
31/10/2024

Details

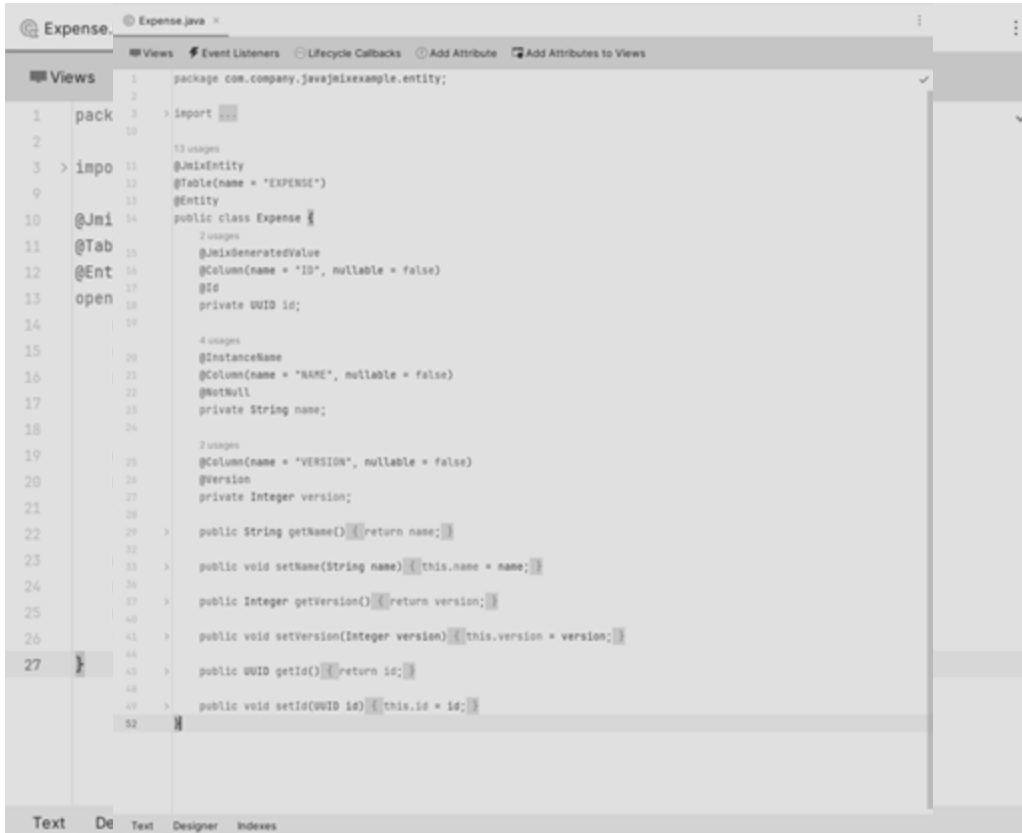
Bus/Metro time!!



Demo



Demo



```
Expense.java
package com.company.java.jmxexample.entity;

import javax.jmx.annotation.*;

@JmxEntity
@Table(name = "EXPENSE")
@Entity
public class Expense {

    @JmxGeneratedValue
    @Column(name = "ID", nullable = false)
    @Id
    private UUID id;

    @InstanceName
    @Column(name = "NAME", nullable = false)
    @NotNull
    private String name;

    @Column(name = "VERSION", nullable = false)
    @Version
    private Integer version;

    public String getName() {return name;}

    public void setName(String name) {this.name = name;}

    public Integer getVersion() {return version;}

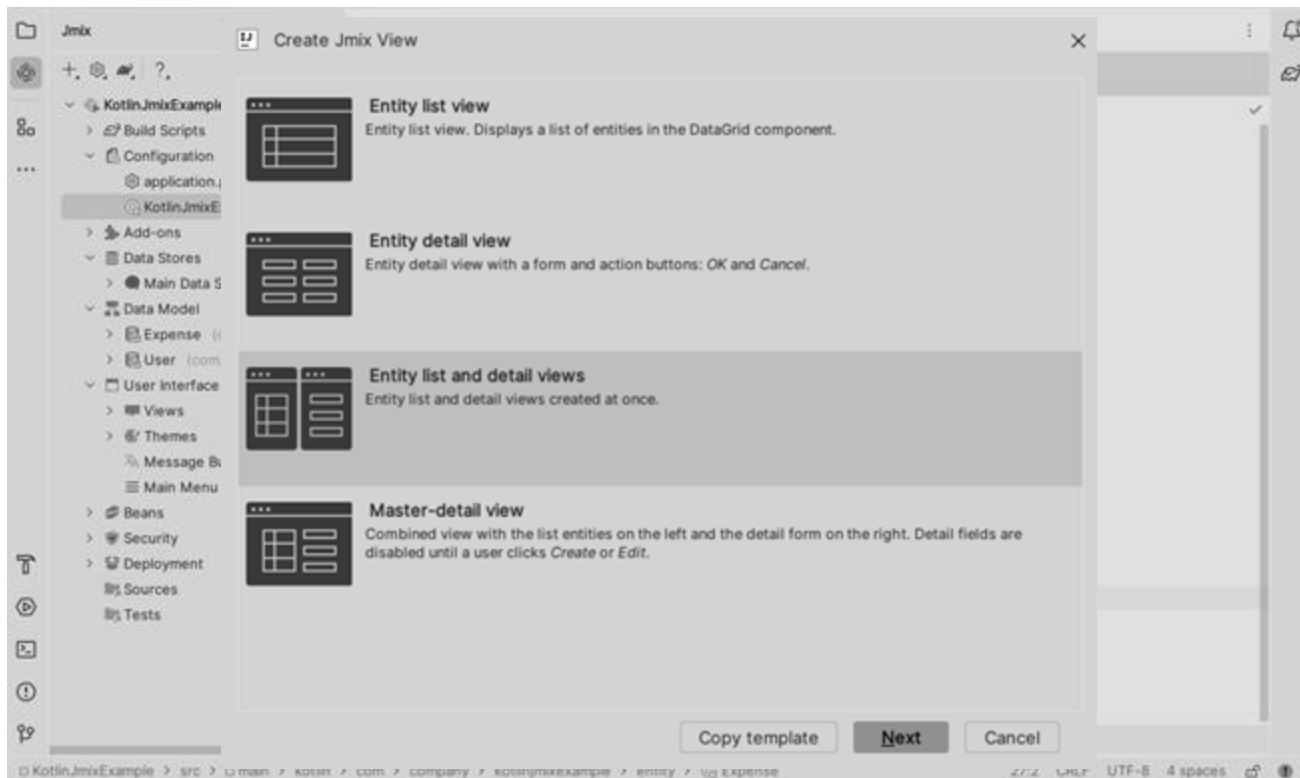
    public void setVersion(Integer version) {this.version = version;}

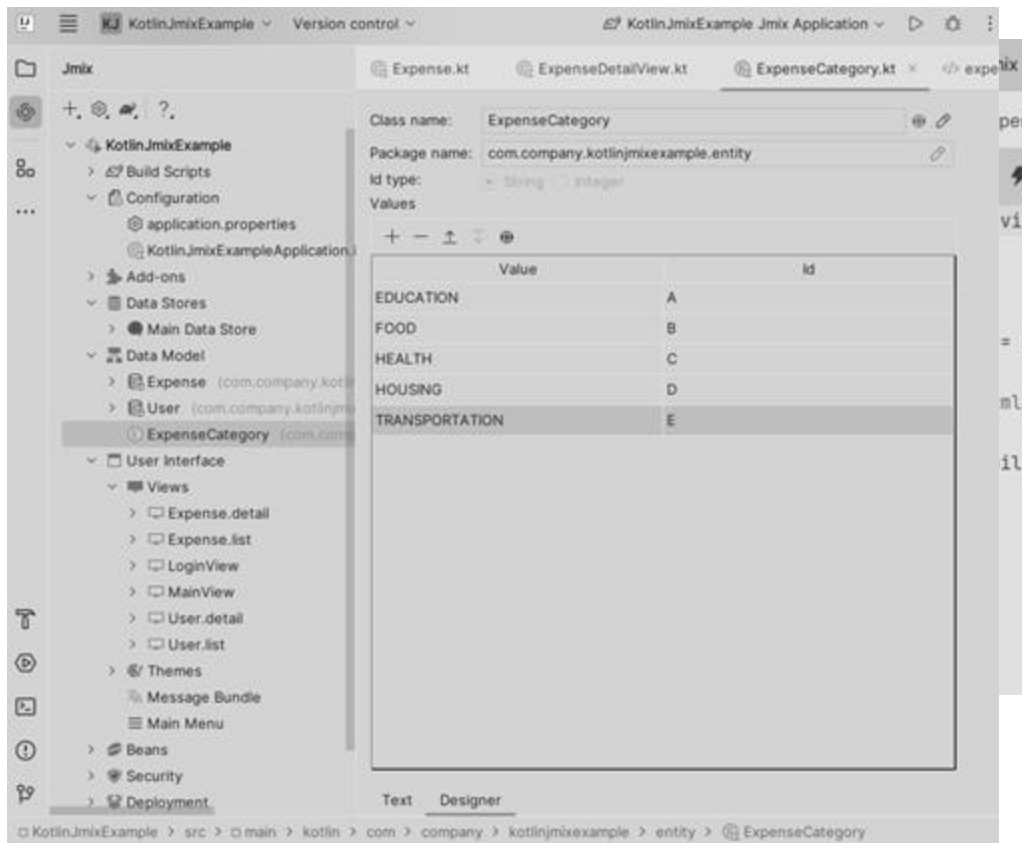
    public UUID getId() {return id;}

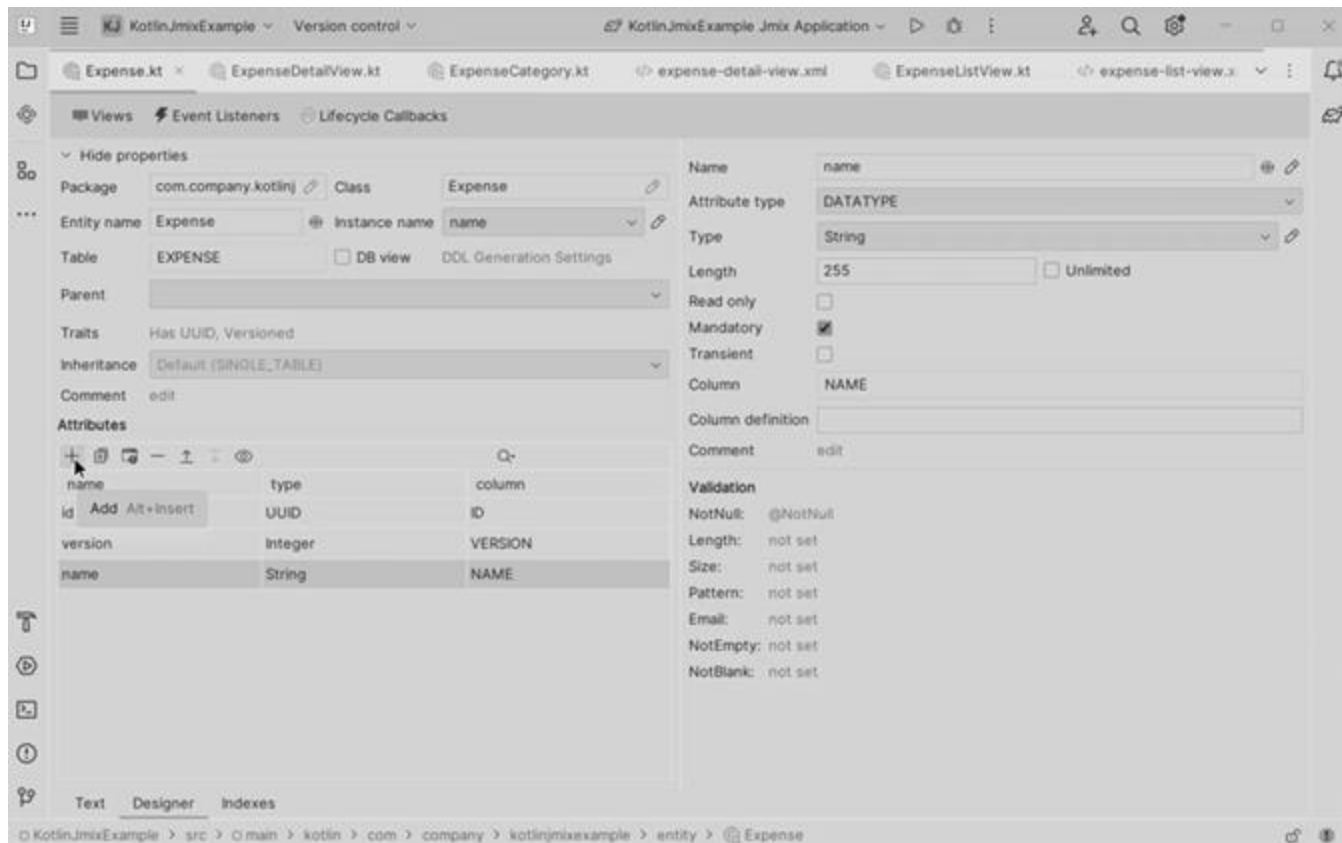
    public void setId(UUID id) {this.id = id;}
}
```

```
Expense.kt x
Views Event Listeners Lifecycle Callbacks Add Attribute Add Attributes to Views
1 package com.company.kotlinjmixexample.entity
2
3 > import ...
4
5
6
7
8 @JmixEntity
9 @Table(name = "EXPENSE")
10 @Entity
11 open class Expense {
12     @JmixGeneratedValue
13     @Column(name = "ID", nullable = false)
14     @Id
15     var id: UUID? = null
16
17     @Column(name = "VERSION", nullable = false)
18     @Version
19     var version: Int? = null
20
21     @InstanceName
22     @Column(name = "NAME", nullable = false)
23     @NotNull
24     var name: String? = null
25
26 }
27
Text Designer Indexes
```

```
Expense.java x
Views Event Listeners Lifecycle Callbacks Add Attribute Add Attributes to Views
1 package com.company.javaexample.entity;
2
3 > import ...
4
5
6
7
8
9
10
11 @JmixEntity
12 @Table(name = "EXPENSE")
13 @Entity
14 public class Expense {
15     @JmixGeneratedValue
16     @Column(name = "ID", nullable = false)
17     @Id
18     private UUID id;
19
20     @JmixGeneratedValue
21     @Column(name = "VERSION", nullable = false)
22     @Version
23     private Integer version;
24
25     @InstanceName
26     @Column(name = "NAME", nullable = false)
27     @NotNull
28     private String name;
29
30     public String getName() { return name; }
31
32     public void setName(String name) { this.name = name; }
33
34     public Integer getVersion() { return version; }
35
36     public void setVersion(Integer version) { this.version = version; }
37
38     public UUID getId() { return id; }
39
40     public void setId(UUID id) { this.id = id; }
41
42 }
43
Text Designer Indexes
```





The screenshot shows the JetBrains IDE interface with the database design tool open. The main window displays the properties of the 'Expense' entity. The left sidebar shows the project structure with 'Expense.kt', 'ExpenseDetailView.kt', 'ExpenseCategory.kt', 'expense-detail-view.xml', 'ExpenseListView.kt', and 'expense-list-view.x'. The main panel is divided into two sections: 'Attributes' and 'Properties'.

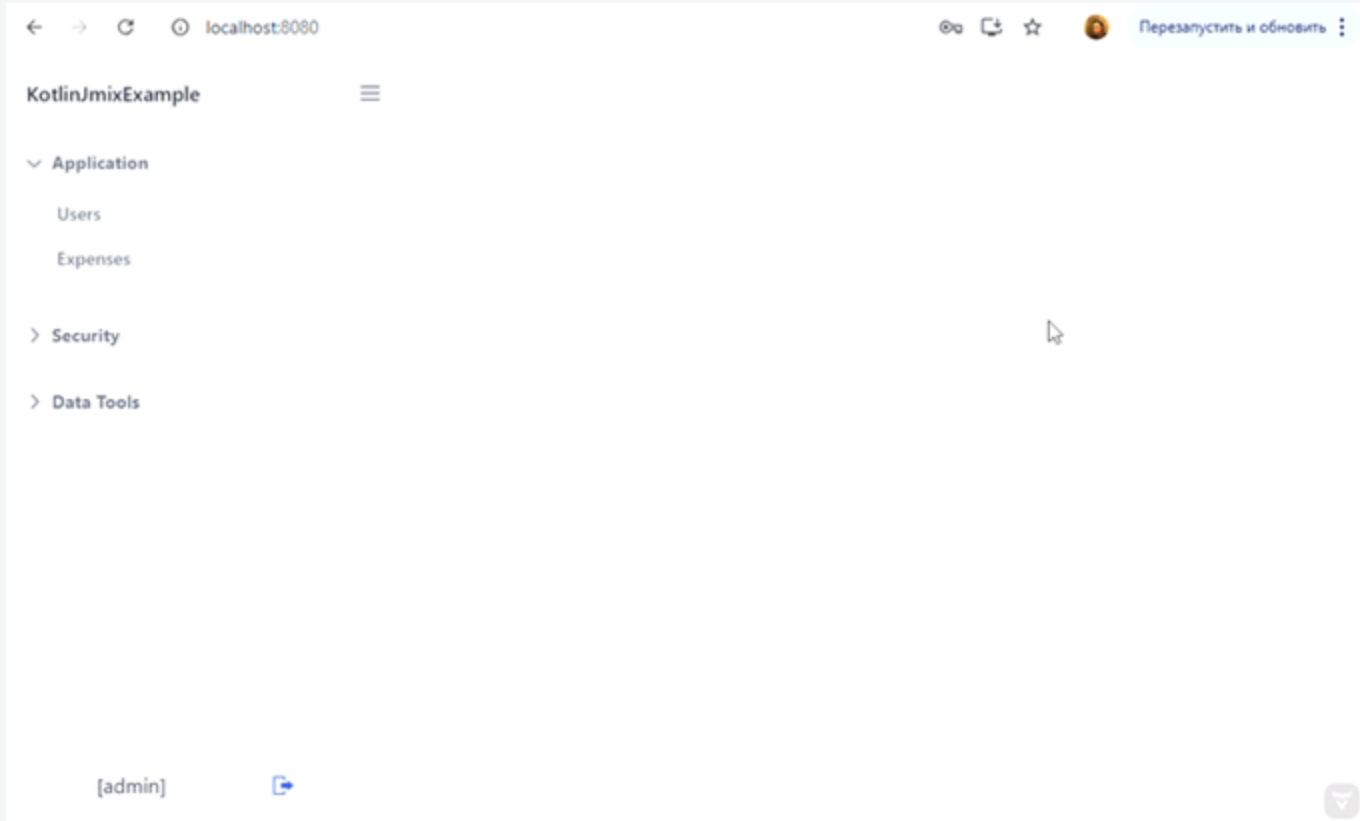
Attributes:

name	type	column
id	UUID	ID
version	Integer	VERSION
name	String	NAME

Properties:

Name	name
Attribute type	DATATYPE
Type	String
Length	255
Read only	<input type="checkbox"/>
Mandatory	<input checked="" type="checkbox"/>
Transient	<input type="checkbox"/>
Column	NAME
Column definition	
Comment	edit
Validation	
NotNull	@NotNull
Length	not set
Size	not set
Pattern	not set
Email	not set
NotEmpty	not set
NotBlank	not set

The bottom status bar shows the path: KotlinJmixExample > src > main > kotlin > com > company > kotlinjmixexample > entity > Expense.



Demo



```
1 package com.company.kotlinjmixexample.entity
2
3 > import ...
4
5
6
7
8
9
10 @JmixEntity
11 @Table(name = "EXPENSE", uniqueConstraints = [
12     UniqueConstraint(name = "IDX_EXPENSE_UNQ", columnNames = ["NAME"])
13 ])
14 @Entity
15 open class Expense {
16     @JmixGeneratedValue
17     @Column(name = "ID", nullable = false)
18     @Id
19     var id: UUID? = null
20
21     @Column(name = "VERSION", nullable = false)
22     @Version
23     var version: Int? = null
24
25     @InstanceName
26     @Column(name = "NAME", nullable = false)
27     @NotNull
28     var name: String? = null
29
30     @Column(name = "CATEGORY", nullable = false)
31     @NotNull
32     private var category: String? = null
33
34     fun getCategory(): ExpenseCategory? = category?.let { ExpenseCategory.fromId(it) }
35
36     fun setCategory(category: ExpenseCategory?) {
37         this.category = category?.id
38     }
39 }
```

KotlinJmixExample

Application

Users

Expenses

Security

Data Tools

[admin]

Expense

Name *

Pasta

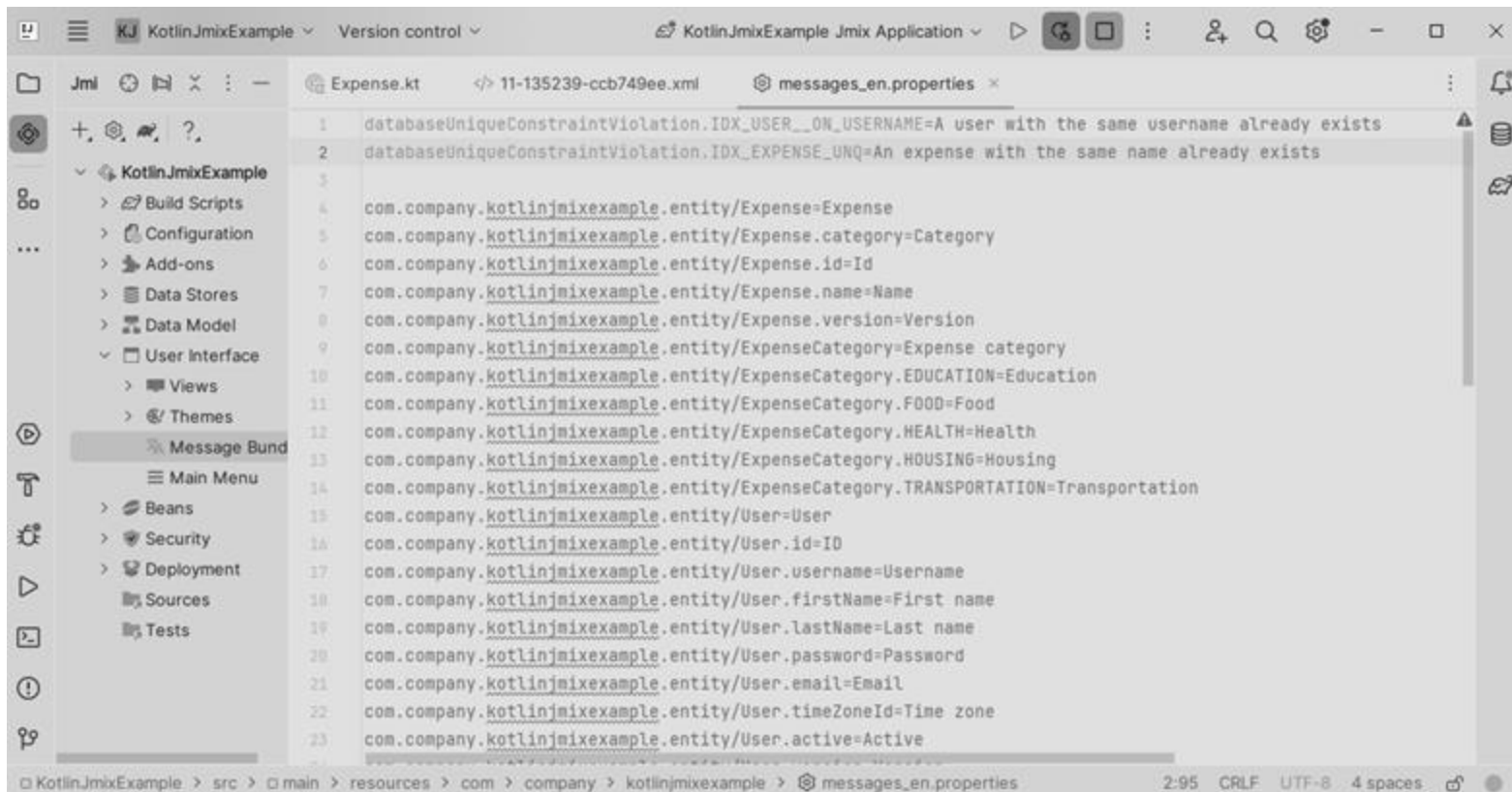
Category *

Food

✓ OK

⌕ Cancel

Unique constraint violation occurred (IDX_EXPENSE_UNQ) ✕



```
1 databaseUniqueConstraintViolation.IDX_USER__ON_USERNAME=A user with the same username already exists
2 databaseUniqueConstraintViolation.IDX_EXPENSE_UNQ=An expense with the same name already exists
3
4 com.company.kotlinjmixexample.entity/Expense=Expense
5 com.company.kotlinjmixexample.entity/Expense.category=Category
6 com.company.kotlinjmixexample.entity/Expense.id=Id
7 com.company.kotlinjmixexample.entity/Expense.name=Name
8 com.company.kotlinjmixexample.entity/Expense.version=Version
9 com.company.kotlinjmixexample.entity/ExpenseCategory=Expense category
10 com.company.kotlinjmixexample.entity/ExpenseCategory.EDUCATION=Education
11 com.company.kotlinjmixexample.entity/ExpenseCategory.FOOD=Food
12 com.company.kotlinjmixexample.entity/ExpenseCategory.HEALTH=Health
13 com.company.kotlinjmixexample.entity/ExpenseCategory.HOUSING=Housing
14 com.company.kotlinjmixexample.entity/ExpenseCategory.TRANSPORTATION=Transportation
15 com.company.kotlinjmixexample.entity/User=User
16 com.company.kotlinjmixexample.entity/User.id=ID
17 com.company.kotlinjmixexample.entity/User.username=Username
18 com.company.kotlinjmixexample.entity/User.firstName=First name
19 com.company.kotlinjmixexample.entity/User.lastName=Last name
20 com.company.kotlinjmixexample.entity/User.password=Password
21 com.company.kotlinjmixexample.entity/User.email=Email
22 com.company.kotlinjmixexample.entity/User.timeZoneId=Time zone
23 com.company.kotlinjmixexample.entity/User.active=Active
```

KotlinJmixExample

Application

- Users
- Expenses

Security

Data Tools

Expense


Name *
Pasta


Category *
Food

✓ OK

⌛ Cancel

An expense with the same name already exists ✕

[admin] 



Demo

```
10 @JmixEntity
11 @Table(name = "USER_EXPENSE", indexes = [
12     Index(name = "IDX_USER_EXPENSE_USER", columnList = "USER_ID"),
13     Index(name = "IDX_USER_EXPENSE_EXPENSE", columnList = "EXPENSE_ID")
14 ])
15 @Entity
16 open class UserExpense {
17     @JmixGeneratedValue
18     @Column(name = "ID", nullable = false)
19     @Id
20     var id: UUID? = null
21
22     @Column(name = "VERSION", nullable = false)
23     @Version
24     var version: Int? = null
25
26     @JoinColumn(name = "USER_ID", nullable = false)
27     @NotNull
28     @ManyToOne(fetch = FetchType.LAZY, optional = false)
29     var user: User? = null
30
31     @JoinColumn(name = "EXPENSE_ID", nullable = false)
32     @NotNull
33     @ManyToOne(fetch = FetchType.LAZY, optional = false)
34     var expense: Expense? = null
35
36     @Column(name = "AMOUNT", nullable = false)
37     @NotNull
38     var amount: Double? = null
39
40     @Column(name = "DATE_", nullable = false)
41     @NotNull
42     var date: LocalDate? = null
43
44     @Column(name = "DETAILS")
45     var details: String? = null
46 }
```

Text Designer Indexes

Demo



KotlinJmixExample

Application

Users

Expenses

User expenses

Security

Data Tools

(admin)

User expenses

Filter

Refresh

Add search condition

+ Create

Edit

Remove

4 rows

Expense	Amount	Date	Details
Pasta	1	10/11/2024	Bought some pasta, ...
Bribe to the teacher	3	03/11/2024	Now I have a master's...
Fixed tooth	2.5	29/10/2024	
Onay card replenish...	5,000	31/10/2024	Bus/Metro time!!

User

David [David]

Expense

Bribe to the teacher

Amount

3

Date

03/11/2024

Details

Now I have a master's degree!

Demo

Hide properties

Package: Class:

Entity name: Instance name:

Attributes

name	type	column
id	UUID	ID
version		VERSION
username		
password		
firstName		
lastName		
email		
active		
timeZoneId		
expenses	UserExpense	

Attributes Collection Modified

Select attributes that should be added to views and their fetch plans:

View	expenses
User.detail : layout	<input checked="" type="checkbox"/>

name	type	column
active	Boolean	ACTIVE
timeZoneId	String	TIME_ZONE_ID
expenses	UserExpense	

KotlinJmixExample

Application

Users

Expenses

User expenses

Security

Data Tools

User

Username *

David

First name

David

Last name

Email

Time zone

Active

☒

+ Create

Edit

Remove

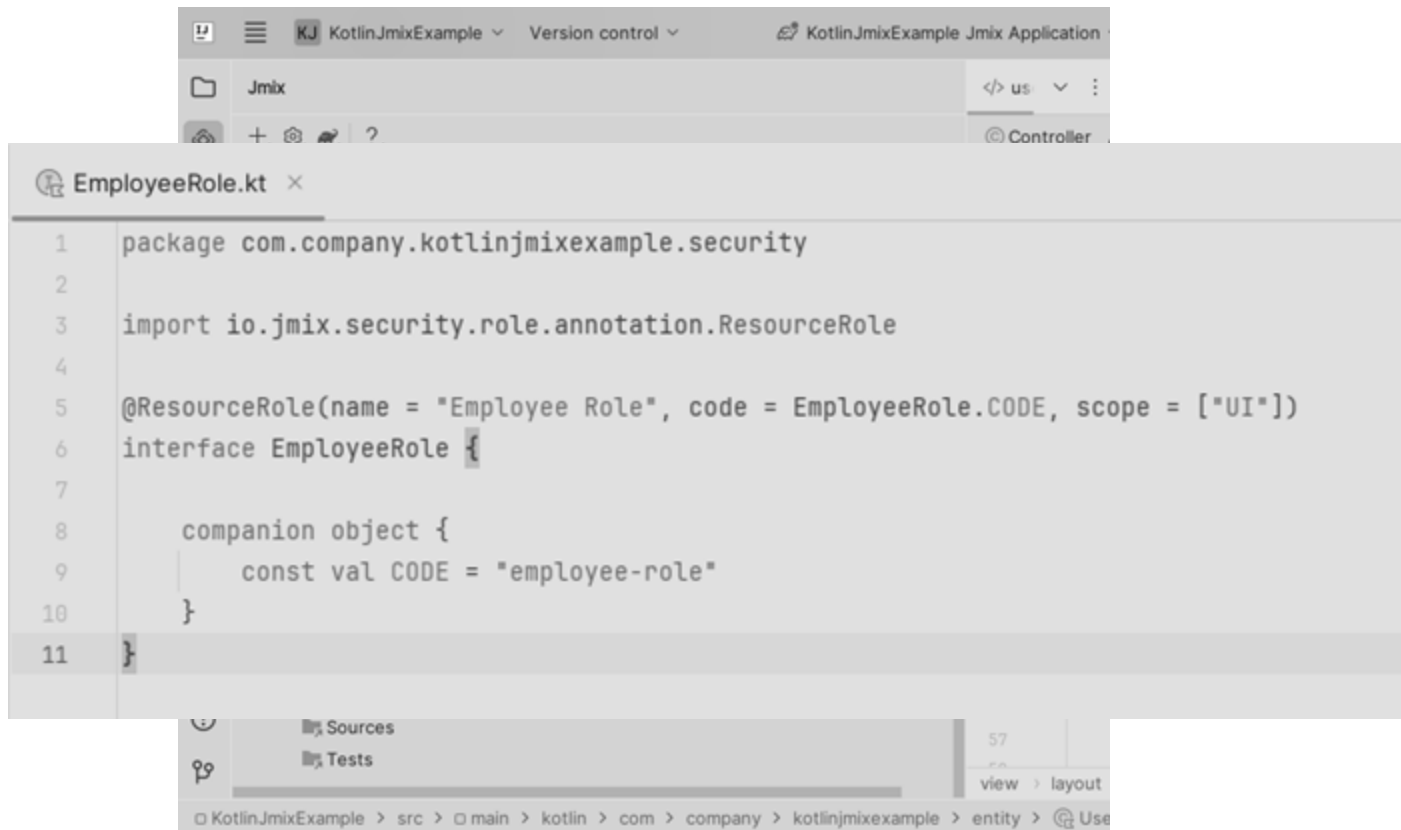
Version	Amount	Date	Details	Name	Category
1	3	03/11/2024	Now I have...	Bribe to th...	Education

[admin]

OK

Cancel

Demo



```
1 package com.company.kotlinjmixexample.security
2
3 import io.jmix.security.role.annotation.ResourceRole
4
5 @ResourceRole(name = "Employee Role", code = EmployeeRole.CODE, scope = ["UI"])
6 interface EmployeeRole {
7
8     companion object {
9         const val CODE = "employee-role"
10     }
11 }
```

The screenshot shows an IDE window titled "EmployeeRole.kt". The code defines a package, imports a Jmix annotation, and creates an annotated interface with a companion object containing a constant. The IDE interface includes a top toolbar, a file explorer on the left, and a bottom toolbar with tabs for Sources and Tests.

KotlinJmixExample Version control KotlinJmixExample Jmix Application

EmployeeRole.kt

Q- ☒ Current project only ☐ Assigned only

Entity	Allow all	Create	Read	Update	Delete	Attributes
Expense	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	*
User	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	*
UserExpense...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	*

Attributes Permissions

Entity: UserExpense

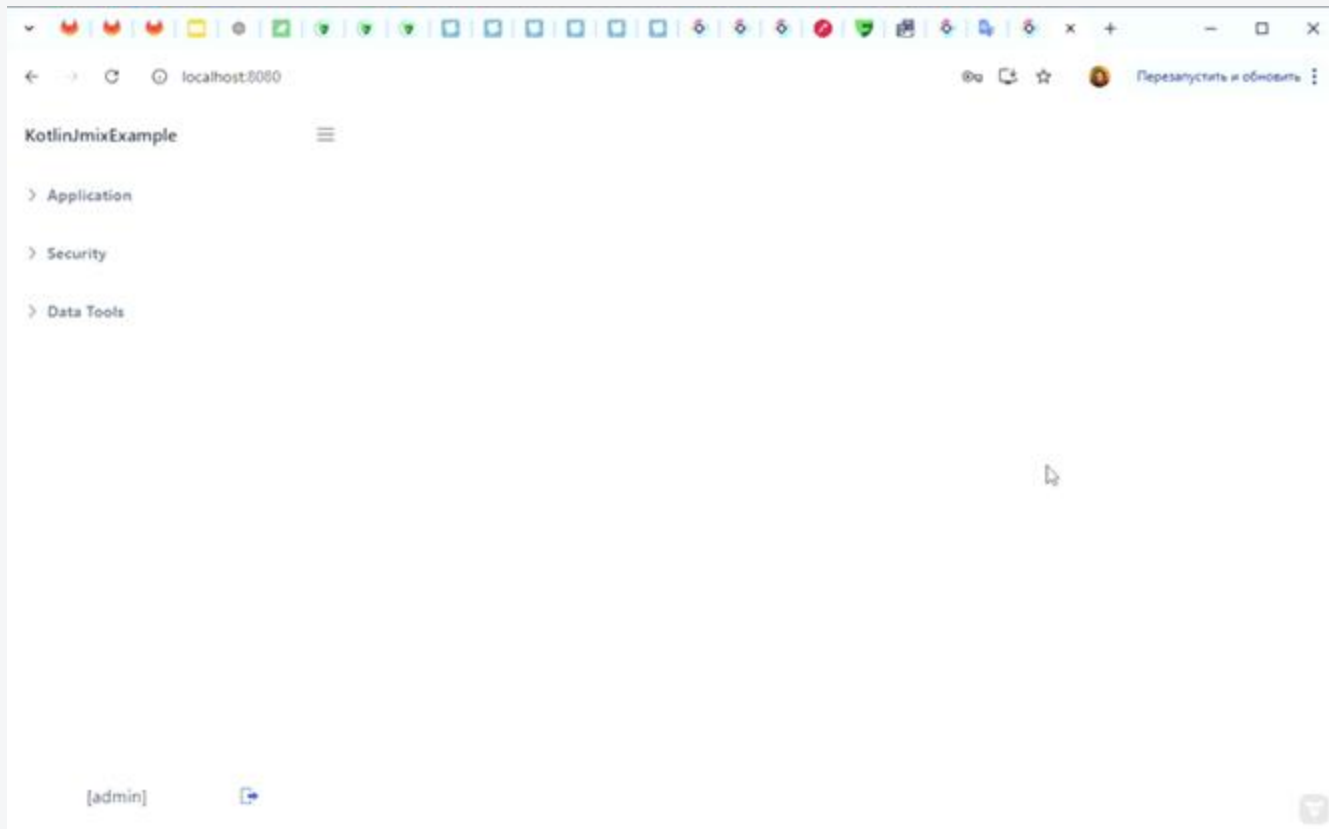
Selecting [*] option enables **View** or **Modify** permissions for all attributes including those attributes that will be added to this entity in the future.

Q-

Attribute	View	Modify
[*]	<input checked="" type="checkbox"/>	<input type="checkbox"/>
id	<input checked="" type="checkbox"/>	<input type="checkbox"/>
version	<input checked="" type="checkbox"/>	<input type="checkbox"/>
user	<input checked="" type="checkbox"/>	<input type="checkbox"/>
expense	<input checked="" type="checkbox"/>	<input type="checkbox"/>
amount	<input checked="" type="checkbox"/>	<input type="checkbox"/>
date	<input checked="" type="checkbox"/>	<input type="checkbox"/>
details	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Text Definition User Interface **Entities** Specific

KotlinJmixExample > src > main > kotlin > com > company > kotlinjmixexample > security > EmployeeRole



DSL

```
1 import com.vaadin.flow.component.button.Button
2 import com.vaadin.flow.component.button.ButtonVariant
3 import com.vaadin.flow.component.dialog.Dialog
4 import com.vaadin.flow.component.html.Div
5 import com.vaadin.flow.component.html.H3
6 import com.vaadin.flow.shared.Registration
7
8 class ConfirmationDialog : Dialog() {
9
10     private lateinit var titleField: H3
11     private lateinit var messageLabel: Div
12     private lateinit var extraMessageLabel: Div
13     private lateinit var confirmButton: Button
14     private lateinit var cancelButton: Button
15     private var registrationForConfirm: Registration? = null
16
17     init {
18         addClassNames("confirm-dialog")
19         isCloseOnEsc = true
20         isCloseOnOutsideClick = false
21
22         header {
23             titleField = h3()
24         }
25         div {
26             className = "confirm-text"
27             messageLabel = div()
28             extraMessageLabel = div()
29         }
30         footer {
31             confirmButton = button {
32                 addClickListener { close() }
33                 isAutofocus = true
34             }
35             cancelButton = button("Cancel") {
36                 addClickListener { close() }
37                 addThemeVariants(ButtonVariant.LUMO_TERTIARY)
38             }
39         }
40     }
41
42     fun open(title: String, message: String = "", additionalMessage: String = "",
43             actionName: String, isDisruptive: Boolean, confirmHandler: () -> Unit) {
44         titleField.text = title
45         messageLabel.text = message
46         extraMessageLabel.text = additionalMessage
47         confirmButton.text = actionName
48
49         registrationForConfirm?.remove()
50         registrationForConfirm = confirmButton.addClickListener { confirmHandler() }
51         if (isDisruptive) {
52             confirmButton.addThemeVariants(ButtonVariant.LUMO_ERROR)
53         }
54         open()
55     }
56 }
```

EXTENSION FUNCTIONS

```

1 package com.company.kotlinoxexample.extension
2
3 import com.vaadin.flow.component.button.Button
4 import com.vaadin.flow.component.button.ButtonVariant
5 import com.vaadin.flow.component.notification.Notification
6
7 class ButtonExtensions {
8 }
9
10 fun Button.configureAsPrimary(text: String, message: String) {
11     this.text = text
12     this.addThemeVariants(ButtonVariant.LUMO_PRIMARY)
13     this.addClickListener { it: ClickEvent<Button>? -> {
14         Notification.show(message)
15     }
16 }

```

```

1 package com.company.kotlinoxexample.view.expense
2
3 > import ...
4
5 @Route(value = "expenses/:id", layout = MainView::class)
6 @ViewController("Expense.detail")
7 @ViewDescriptor("expense-detail-view.xml")
8 @EditedEntityContainer("expenseDc")
9
10 class ExpenseDetailView : StandardDetailView<Expense>() {
11     @ViewComponent
12     private lateinit var saveAndCloseBtn: JmixButton
13
14     @Subscribe
15     fun onInit(event: InitEvent) {
16         saveAndCloseBtn.configureAsPrimary(text = "Запись была сохранена", message = "Ваши траты записаны")
17     }
18 }

```

Ваши траты записаны



КОТЛИН КОРУТИНЫ

```
ButtonExtensions.kt ExpenseDetailView.kt
1 package com.company.kotlinfxexample.view.expense
2
3 import androidx.compose.foundation.layout.*
4
5 @Route(value = "expenses/:id", layout = MainView::class)
6 @ViewController("ExpenseDetail")
7 @ViewDescriptor("expense-detail-view.xml")
8 @EditResourceContainer("expenseDetail")
9 class ExpenseDetailView : StandardDetailView<Expense>(), CoroutineScope {
10
11     private val job = SupervisorJob()
12     override val coroutineContext = Dispatchers.Default + job
13
14     @ViewComponent
15     private lateinit var saveAndCloseBtn: JalaButton
16
17     @Autowired
18     private lateinit var notifications: Notifications
19
20     @Subscribe
21     fun onInit(event: InitEvent) {
22         saveAndCloseBtn.configurePrimary({ id: "save_and_close_btn" }) {
23             launch { thisCoroutineScope {
24                 val result = withContext(Dispatchers.IO) { thisCoroutineScope {
25                     performLongRunningOperation()
26                 }
27                 UI.getCurrent().access {
28                     notifications.create("Expense response: $result")
29                     .show()
30                 }
31             }
32         }
33     }
34
35     private suspend fun performLongRunningOperation(): String {
36         delay(5000L)
37         return "Expense response: $id"
38     }
39
40     @Subscribe
41     fun onBeforeClose(event: BeforeCloseEvent) {
42         coroutineContext.cancel()
43     }
44 }
```

SEALED КЛАССЫ


```
OperationStatus.kt × ExpenseDetailView.kt </> expense-detail-view.xml
1 package com.company.kotlinjmixexample.sealedclass
2
3 @sealed class OperationStatus {
4     object Loading : OperationStatus()
5     data class Success(val message: String) : OperationStatus()
6     data class Failure(val error: Throwable) : OperationStatus()
7 }
```

Demo

```
OperationStatus.kt ExpenseDetailView.kt expense-detail-view.xml
Descriptor Expense Main Menu Inject Generate Handler Code Snippets

21 @Subscribe
22 fun onInit(event: InitEvent) {
23     checkIn.configureAsPrimary(when: "copyFrom stepotame"){
24         val ui = UI.getCurrent()
25         launch { this.consumeScope()
26             try {
27                 ui.access {
28                     notifications.create("message: success...")
29                         .withType(Notifications.Type.SUCCESS)
30                         .show()
31                 }
32             }
33             val status = withContext(Dispatchers.IO) { this.consumeScope()
34                 performLongRunningOperation()
35             }
36             ui.access {
37                 when (status) {
38                     is OperationStatus.Success -> {
39                         notifications.create("Index: ${status.message}")
40                             .withType(Notifications.Type.SUCCESS)
41                             .show()
42                     }
43                     is OperationStatus.Failure -> {
44                         notifications.create("Error: ${status.error.message}")
45                             .withType(Notifications.Type.ERROR)
46                             .show()
47                     }
48                     OperationStatus.Loading -> {
49                     }
50                 }
51             }
52         } catch (e: Exception) {
53             ui.access {
54                 notifications.create("Error: copyFrom: ${e.message}")
55                     .withType(Notifications.Type.ERROR)
56                     .show()
57             }
58         }
59     }
60 }
```

Demo

```
72     private suspend fun performLongRunningOperation(): OperationStatus {
73         return try {
74             delay( timeMillis= 2000)
75             OperationStatus.Success("Данные успешно обработаны")
76         } catch (e: Exception) {
77             OperationStatus.Failure(e)
78         }
79     }
80 }
```

Kotlin/mixExample

Application

Users

Expenses

User expenses

Security

Data Tools

[admin]

Expense

Name +

Test

Category +

Housing

✓ OK

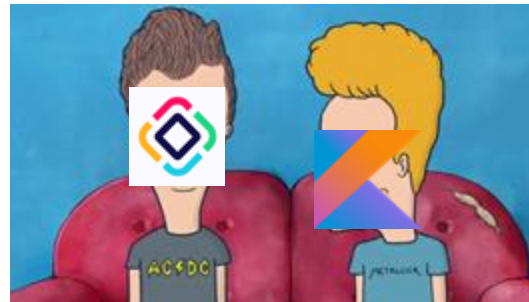
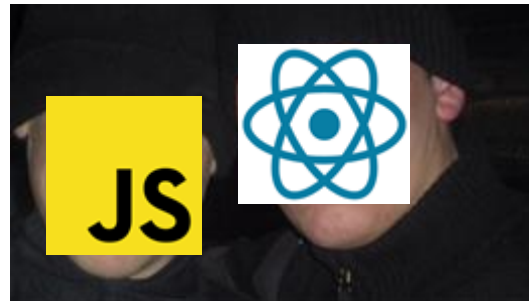
⌕ Cancel

Sealed класс отработал

Успех. Данные успешно обработаны

Заключение

- **Kotlin стремительно движется к FullStack-разработке:**
 - Kotlin/JS и React
 - Kotlin Multiplatform (KMP)
 - Ktor с DSL для HTML и CSS
- **Jmix развивается в сторону FullStack:**
 - Интеграция с Kotlin для сокращения шаблонного кода
 - Ускорение разработки веб-приложений в корпоративной среде
 - Повышение продуктивности и эффективности команд разработчиков
- **Синергия Kotlin и Jmix:**
 - Совместное использование открывает новые возможности для полного стека разработки на JVM
 - Сокращение цикла разработки и ускорение выхода продуктов на рынок



Спасибо



Google Presentation



GitHub Demo Project

