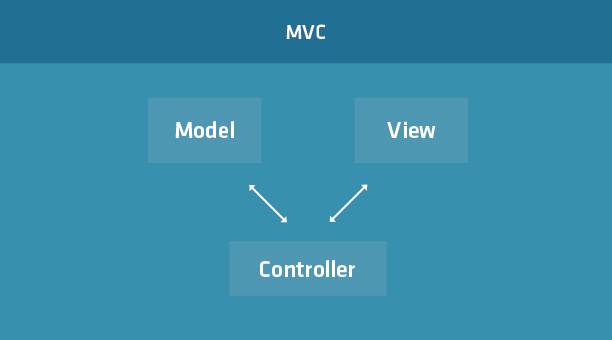
# **Основы ExtJS (продолжение)**

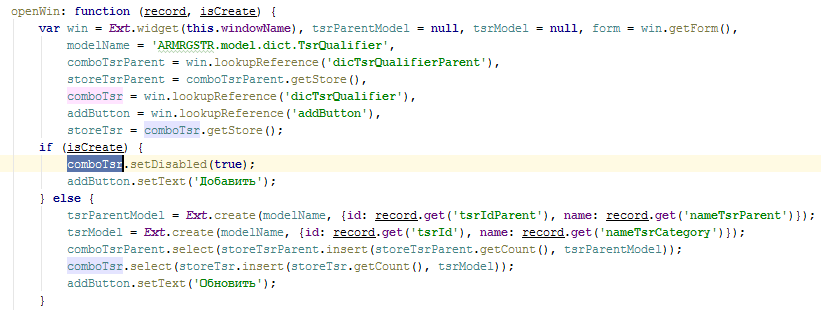
Науменко Т.Ю.

Парадигма MVC в ExtJS

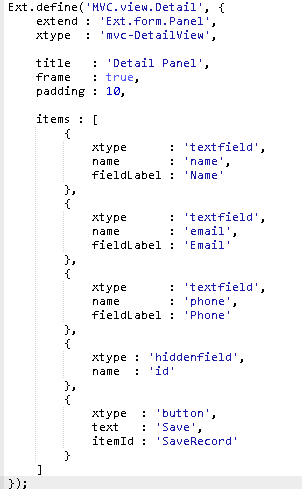


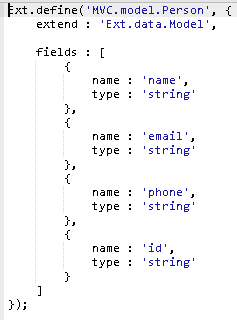
* *Model (модель)* — описывает используемый формат данных, используемых в приложении. Может содержать бизнес-правила, правила валидации и некоторые другие функции.
* *View (представление)* — отображает данные пользователю. Разные представления могут отображать те же данные разными путями (например, таблицы и графики).
* *Controller (контроллер)* — центральная часть MVC-приложения. Он следит за событиями
* внутри приложения и распределяет команды между моделью и представлением.

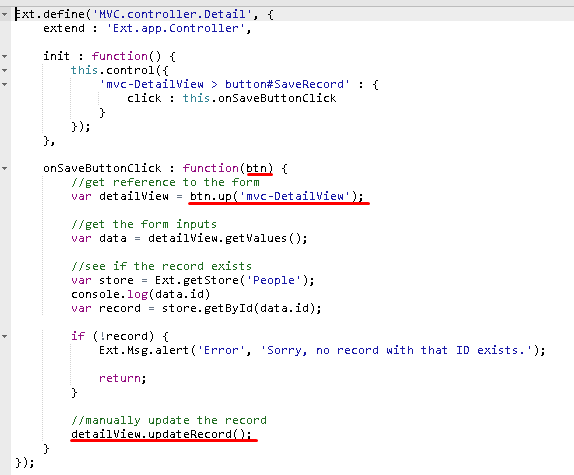
# Пример I MVC :



# Пример II MVC :



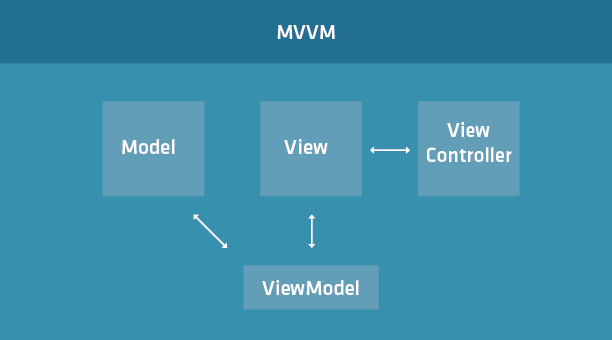




# Минусы:

* Изменения в view требует изменений в controller, не всегда очевидных, не всегда точечных 🡺
* Сложность в тестировании.
* Появляются действия связанные с навигацией в контроллере по view («лишний» код), возрастает вероятность появления ошибки.
* Код контроллера не всегда линеен, асинхронность требует объявления callback функций, это еще одни источник «лишнего» кода.

Парадигма MVVM (Model-View-ViewModel) в ExtJS (с версии 5)

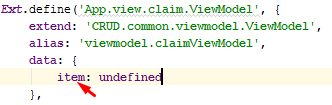


**Представление является подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления.**

* В случае, если в ViewModel изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и View, в свою очередь, запрашивает обновленное значение свойства из ViewModel. В случае, если пользователь воздействует на какой-либо элемент интерфейса, View вызывает соответствующую команду, предоставленную ViewModel
* главный выигрыш - абстрагирование model от view слоя.

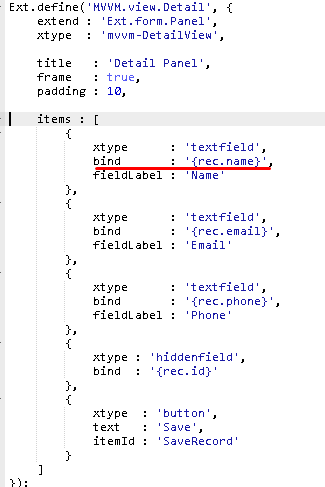
*ViewModel* является, с одной стороны, абстракцией View, а с другой, предоставляет обёртку данных из Model, которые подлежат связыванию (bind). То есть, она содержит model, которая преобразована к Представлению, а также содержит в себе команды, которыми может пользоваться View, чтобы влиять на Model.

# Пример I MVVM





# Пример II MVVM



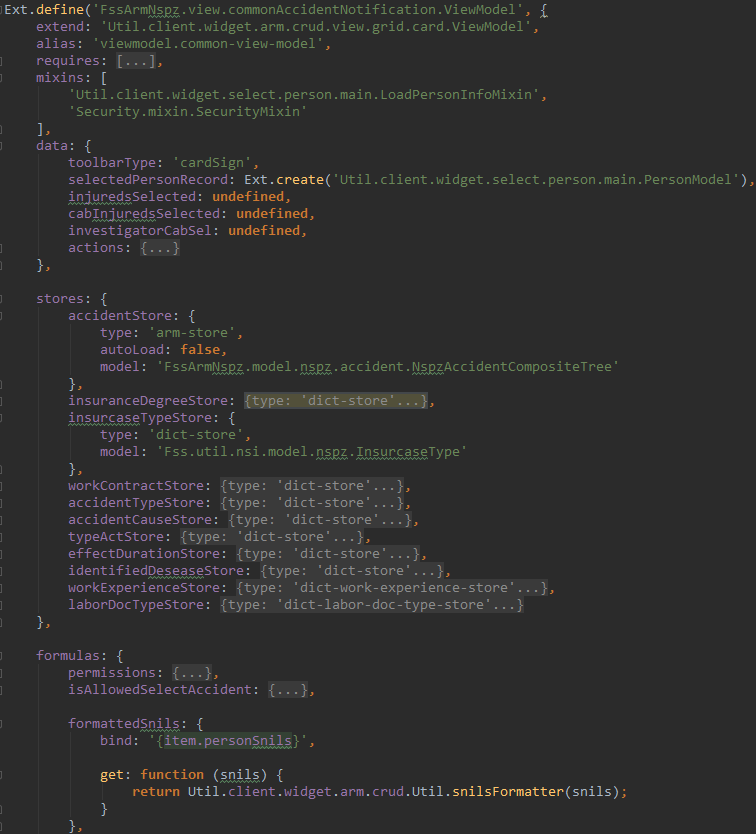
# [Ext.app.ViewModel](https://docs.sencha.com/extjs/6.2.1/classic/src/ViewModel.js.html)

**Data – хранилище переменных, доступных для связывания**

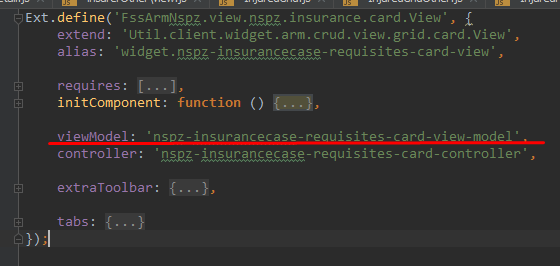
**Formulas – свои функции внутри viewModel**

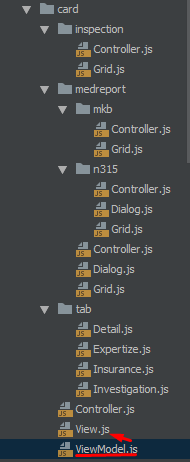
**Stores – коллекции внутри viewModel**

# Пример ViewModel

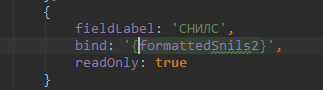


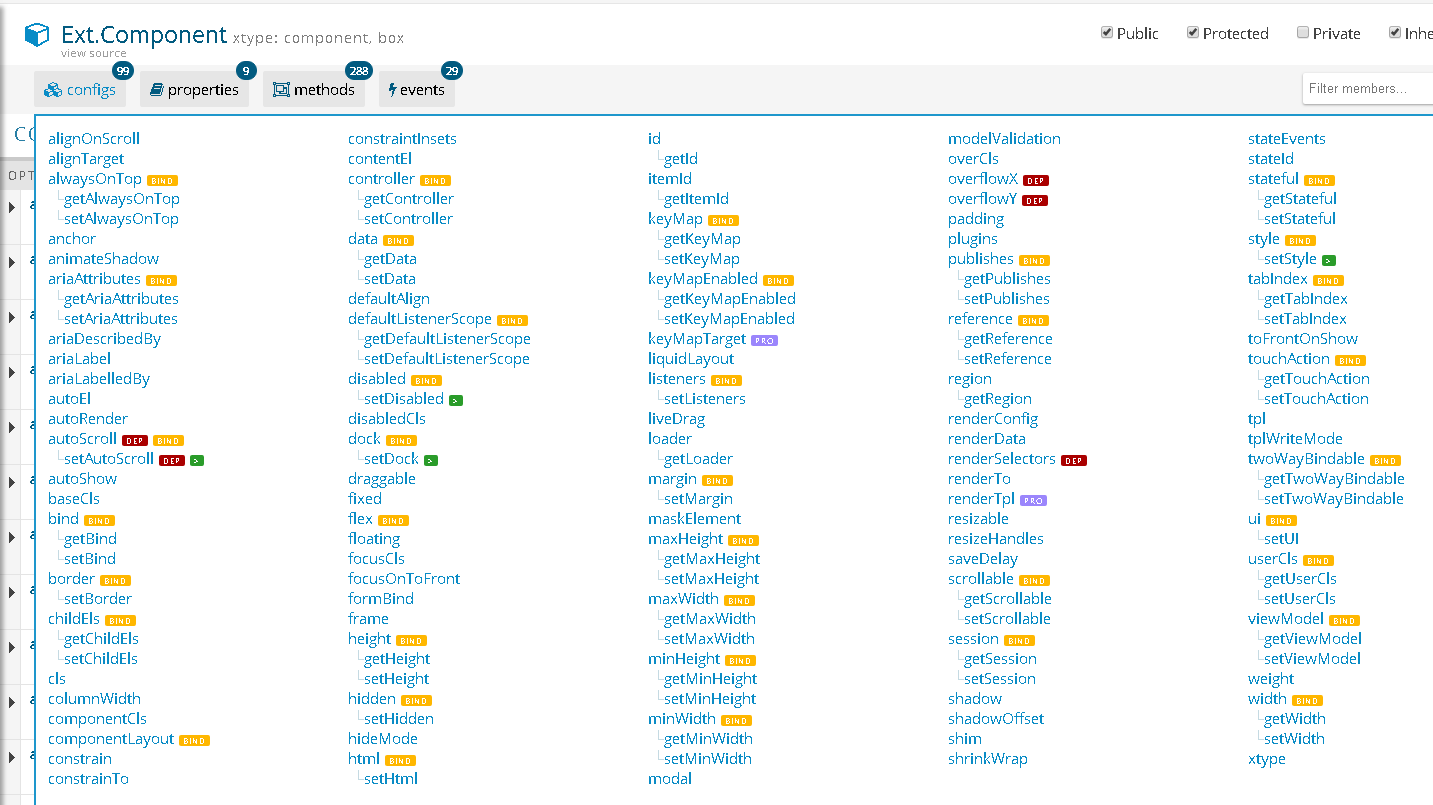
# Подключение viewModel



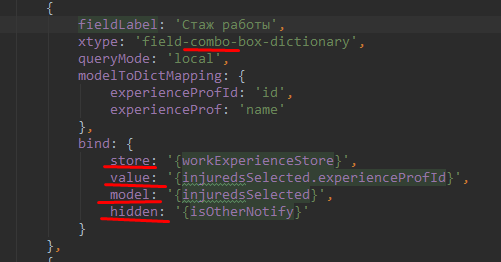


# На любом view





# Еще пример связывания

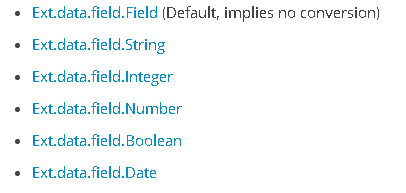


примеры: <http://examples.sencha.com/extjs/6.6.0/examples/kitchensink/?classic#data-binding>

Model Associations.

# [Ext.data.Model](https://docs.sencha.com/extjs/6.2.1/classic/src/Model.js.html)

# [fields](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#cfg-fields) : [Ext.data.field.Field](https://docs.sencha.com/extjs/6.2.1/classic/src/Field.js-3.html)



## [idProperty](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#cfg-idProperty)  - имя уникального поля

# [proxy](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#cfg-proxy) - [Ext.data.proxy.Proxy](https://docs.sencha.com/extjs/6.2.1/classic/src/Proxy.js-1.html) – взаимодействие с источниками данными

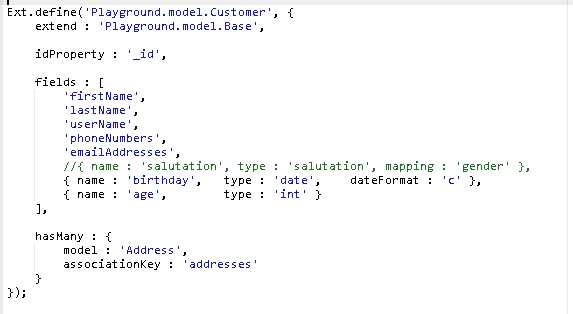
# [validators](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#cfg-validators) массив [Ext.data.validator.Validator](https://docs.sencha.com/extjs/6.2.1/classic/src/Validator.js.html)(валидаторов)

# [hasMany](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#cfg-hasMany) - [Ext.data.schema.HasMany](https://docs.sencha.com/extjs/6.2.1/classic/src/HasMany.js.html)

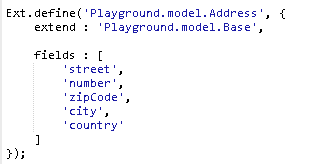
имеем json



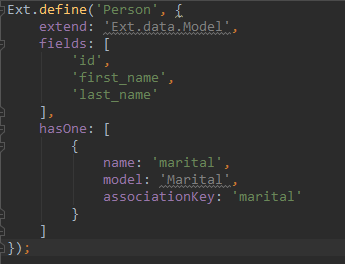
# Основная model

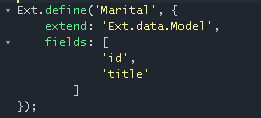


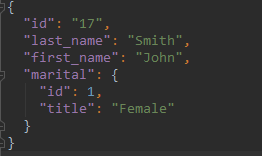
# Вложенная model



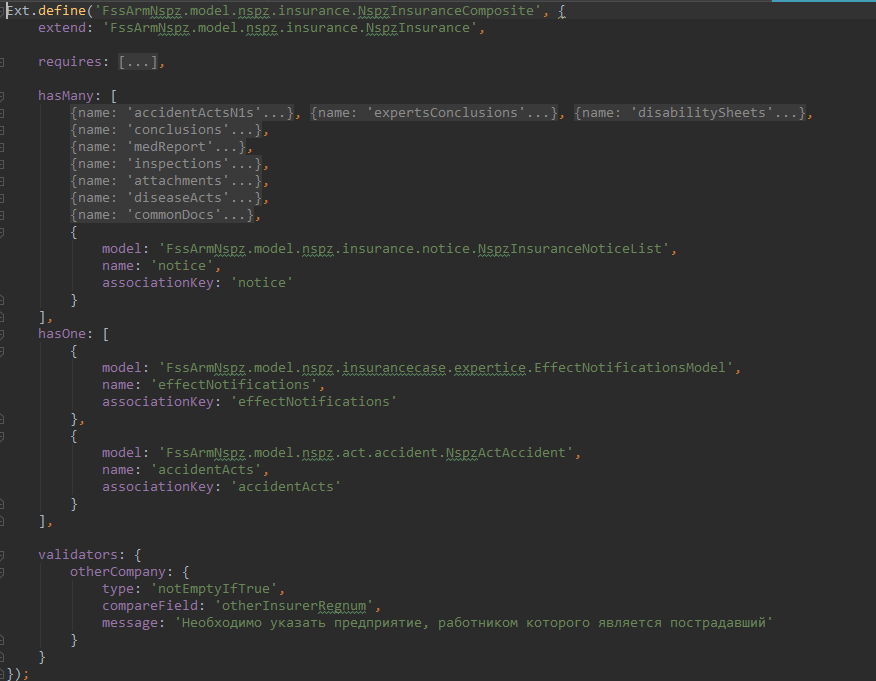
# [hasOne](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#cfg-hasOne) - [Ext.data.schema.HasOne](https://docs.sencha.com/extjs/6.2.1/classic/src/HasOne.js.html)





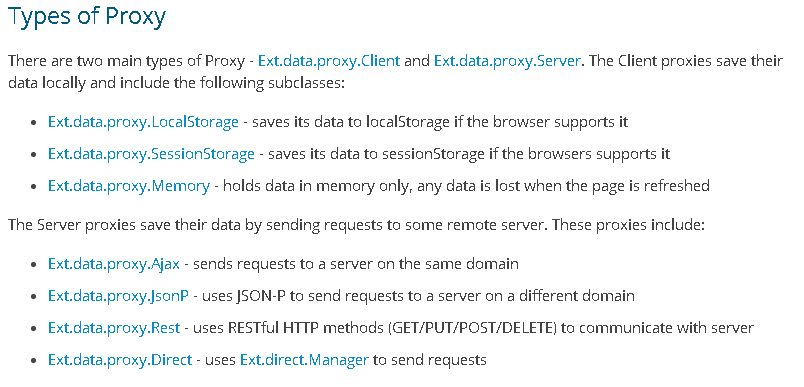


# Пример сложной комплексной модели



Model - Взаимодействие с внешними источниками данными

# [Ext.data.Model](https://docs.sencha.com/extjs/6.2.1/classic/src/Model.js.html).[proxy](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html" \l "cfg-proxy) - [Ext.data.proxy.Proxy](https://docs.sencha.com/extjs/6.2.1/classic/src/Proxy.js-1.html) – взаимодействие с внешними источниками данными



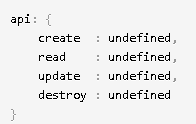
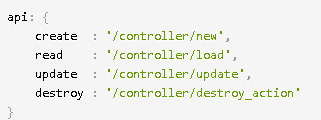
# [Ext.data.proxy.Rest](https://docs.sencha.com/extjs/6.2.1/classic/src/Rest.js.html) alias: proxy.rest

## [actionMethods](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.proxy.Rest.html#cfg-actionMethods)



## [url](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.proxy.Rest.html#cfg-url) («/controller/»)

## [api](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.proxy.Rest.html#cfg-api)

## [directionParam](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.proxy.Rest.html#cfg-directionParam), [extraParams](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.proxy.Rest.html#cfg-extraParams), [filterParam](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.proxy.Rest.html#cfg-filterParam)

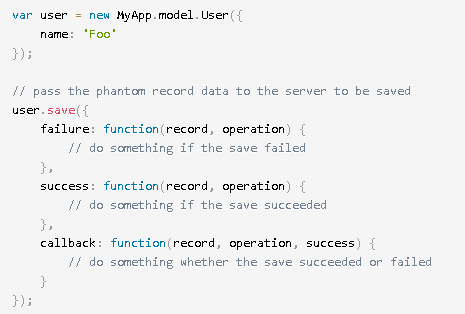
## [headers](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.proxy.Rest.html#cfg-headers)



ExtJS предоставляет реализацию из «коробки» действий: Create, Read, Update, Delete (CRUD)

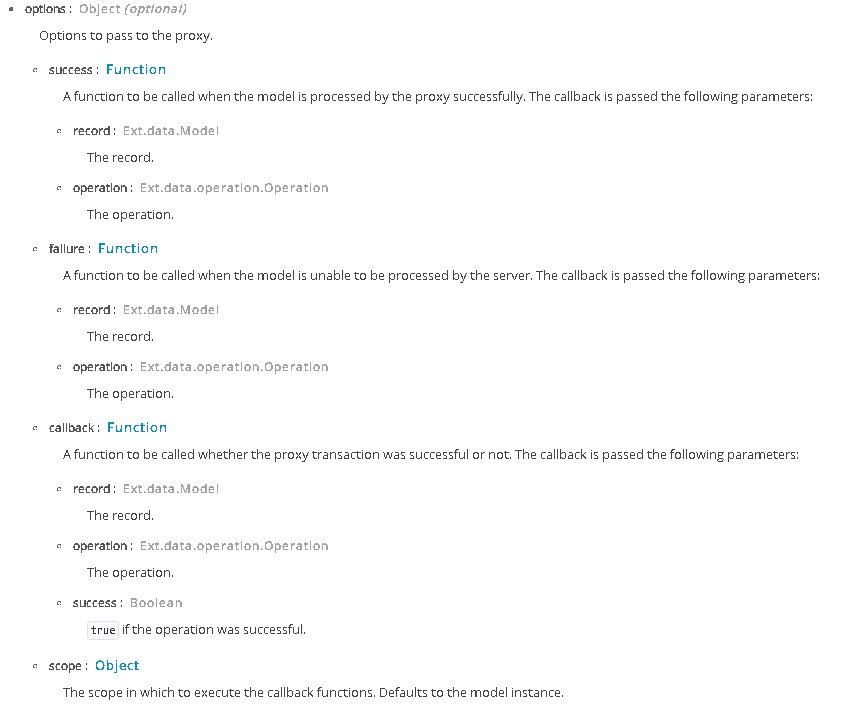
# Create, Update:

## [phantom](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#property-phantom): false/true

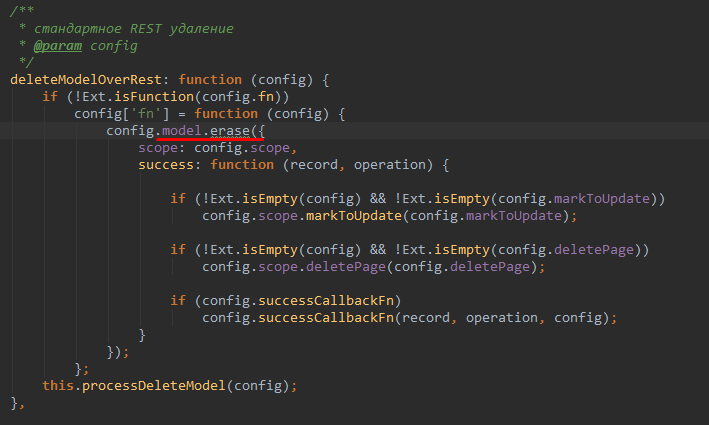


# Read





# Delete



Все ответы != 200 - failure

Store ([Ext.data.Store](https://docs.sencha.com/extjs/6.2.1/classic/src/Store.js-1.html)) – хранилище коллекций

# Model. [hasMany](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Model.html#cfg-hasMany): [] - Store

[model](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Store.html#cfg-model)

[proxy](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Store.html#cfg-proxy)

[autoLoad](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Store.html#cfg-autoLoad)

[storeId](https://docs.sencha.com/extjs/6.2.1/classic/Ext.data.Store.html#cfg-storeId) – уникальный идентификатор Store

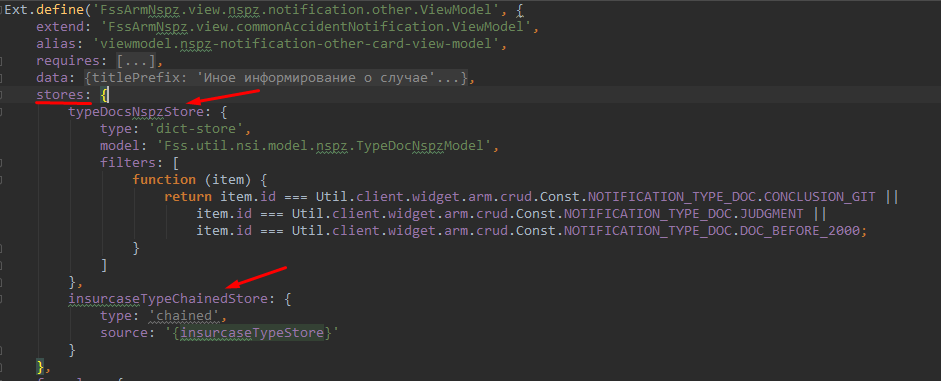
# Глобальное объявление



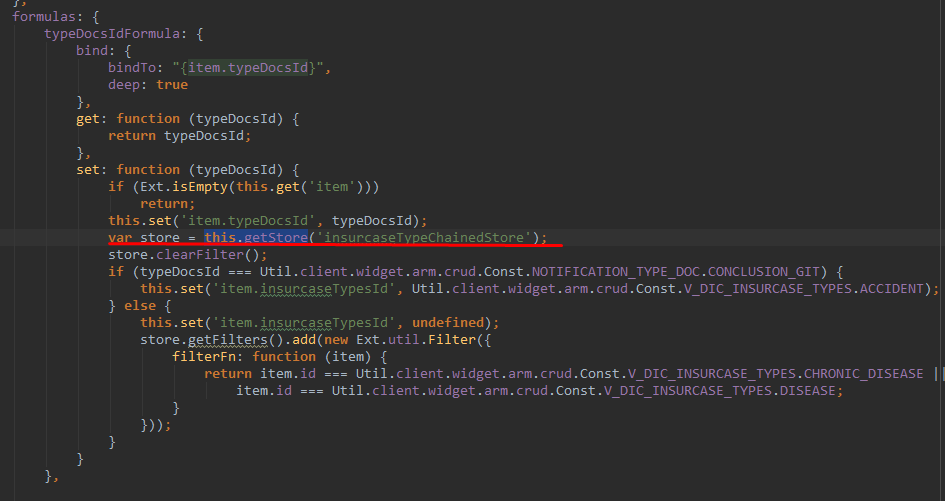
Обращение через storeId



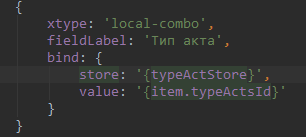
# Локальное объявление



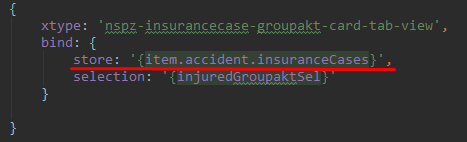
Локальное обращение (в контексте текущей ViewModel)



Связывание с комбо

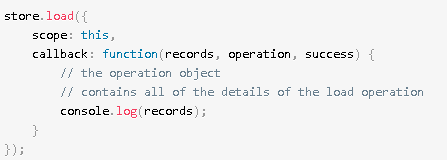


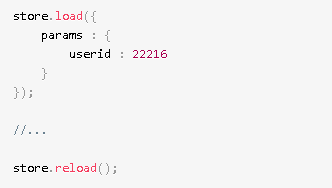
Связывание с гридом



Store - Взаимодействие с внешними источниками данными

Proxy **модели** определяет взаимодействие с данными, если не определен Proxy **стора**.





**Store**.sync(options) – синхронизация Store

