

ExtJs 4

Создание
простого реестра

Часть 2

Создание контроллера

В предыдущей серии

- Создали представление грида
- Создали представление окна редактирования

| Наименование | Поставщик |
|--------------------------------------|---|
| уборка снега | ГУП ЖИЛИЩНО-КОММУНАЛЬНОГО ХОЗЯЙСТВА РЕСПУБЛИКИ САХА (ЯКУТИЯ)(Бул) |
| Взнос за монтаж | ООО УК АЛЬТАИР |
| Электроэнергия | МУП УК ПКСХАН |
| ХОЛОДНОЕ ВОДОСНАБЖЕНИЕ НСУ | ВОДОКАНАЛ |
| ЦЕНТРАЛЬНОЕ ОТОПЛЕНИЕ долги перед УК | ЭНЕРГОСБЫТ ТЕПЛО |
| холодное водоснабжение | МУП ЖАТАЙТЕПЛОСЕТЬ |
| горячее водоснабжение | МУП ЖАТАЙТЕПЛОСЕТЬ |
| центральное отопление | МУП ЖАТАЙТЕПЛОСЕТЬ |
| водоотведение | МУП ЖАТАЙТЕПЛОСЕТЬ |
| уборка подъезда | 065 ТЕСТИРОВАНИЕ |

Дополнительная услуга

Сохранить Закрыть

Наименование услуги: Взнос за монтаж

Поставщик: ООО УК АЛЬТАИР

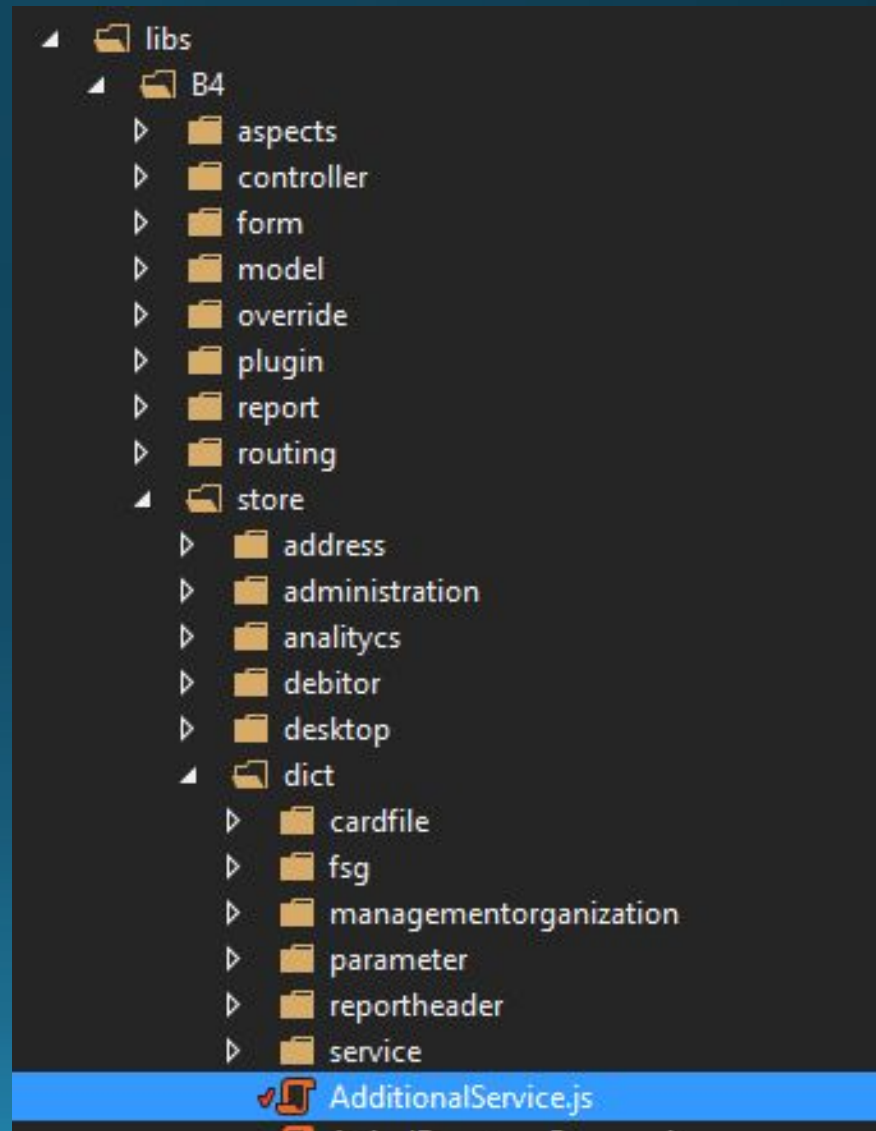
План

- Создание сторов
- Создание моделей
- Создание контроллеров
- Создание маршрута контроллера
- Создание прав для просмотра реестра
- Создание элемента главного меню для перехода в реестр

Создание стора

- Создать скрипт в поддиректории store, название класса создаваемого стора обязательно должно совпадать с физическим расположением скрипта

```
Ext.define('B4.store.dict.AdditionalService', {
```



Создание стора

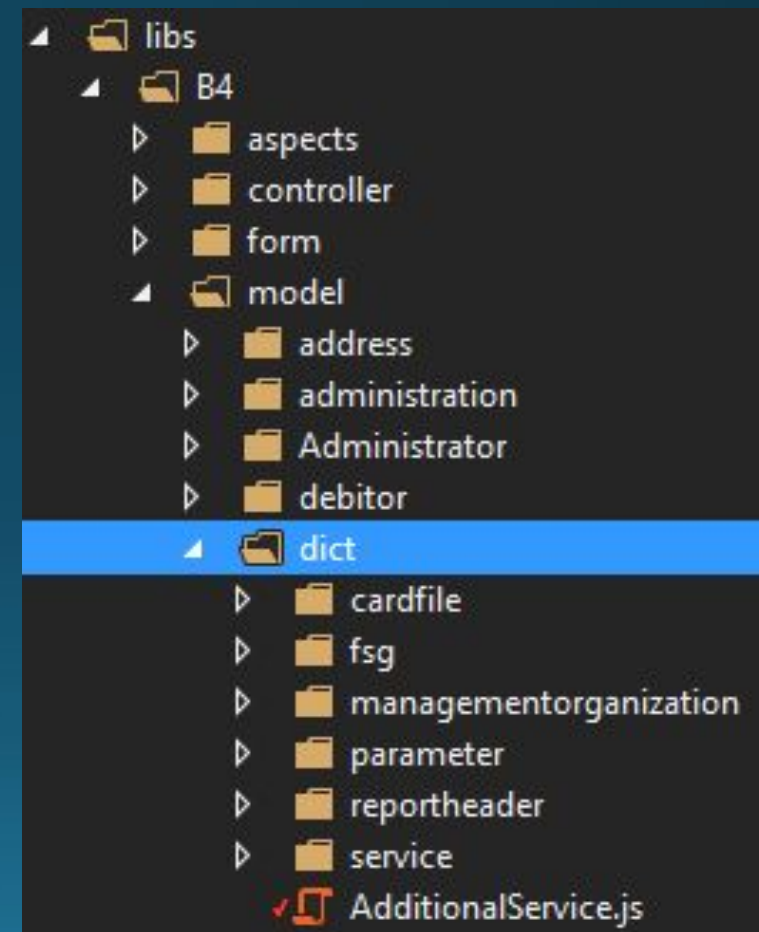
- Выбор базового класса делаем на основании предназначения стора – `B4.base.Store/Ext.data.TreeStore`
- Указать выбранный базовый класс в свойстве `extend`
- Указать в `requires` зависимость от модели, с которой будет связан создаваемый стор
- Если не требуется автоматическая загрузка стора, можно указать `autoLoad: false`
- В свойстве `model` указываем модель

```
Ext.define('B4.store.dict.AdditionalService', {  
    extend: 'B4.base.Store',  
    requires: ['B4.model.dict.AdditionalService'],  
    autoLoad: false,  
    model: 'B4.model.dict.AdditionalService'  
});
```

Создание модели

- Создать скрипт в поддиректории model, название класса создаваемой модели обязательно должно совпадать с физическим расположением скрипта

```
Ext.define('B4.model.dict.AdditionalService', {
```



Создание модели

- Указываем родительский класс 'B4.base.Model' в свойстве extend
- Описываем прокси – слой, который будет перенаправлять CRUD запросы, посылаемые нашей модели
- Т.к. реализация CRUD операций для нашего реестра будет находиться на сервере, выбираем тип прокси 'b4proxy' и указываем в свойстве controllerName имя MVC контроллера, ответственного за взаимодействие с данными нашего реестра
- Описываем поля нашей модели в массиве fields

```
Ext.define('B4.model.dict.AdditionalService', {
    extend: 'B4.base.Model',
    proxy: {
        type: 'b4proxy',
        controllerName: 'AdditionalService'
    },
    fields: [
        { name: 'Name' },
        { name: 'ContragentName' },
        { name: 'Contragent' },
        { name: 'IsActual' },
        { name: 'IsActualNative' },
        { name: 'DisplayName' }
    ]
});
```

Создание модели

Настройка прокси

Метод действия

- Create
- Read
- Update
- Delete
- List

Имя свойства проху

- createAction
- readAction
- updateAction
- destroyAction
- listAction

Значение по-умолчанию

- create
- get
- update
- delete
- list

Создание модели

Настройка полей

```
fields: [  
  { name: 'Name' },  
  { name: 'ContragentName' },  
  { name: 'Contragent' },  
  { name: 'IsActual' },  
  { name: 'IsActualNative' },  
  { name: 'DisplayName' }  
]
```

Модель

```
columns: [  
  {  
    xtype: 'b4editcolumn'  
  },  
  {  
    xtype: 'gridcolumn',  
    dataIndex: 'Name',  
    flex: 1,  
    text: 'Наименование',  
    filter: {  
      xtype: 'textfield'  
    }  
  },  
  {  
    xtype: 'gridcolumn',  
    dataIndex: 'ContragentName',  
    flex: 1,  
    text: 'Поставщик',  
    filter: {  
      xtype: 'textfield'  
    }  
  },  
  {  
    xtype: 'b4deletecolumn'  
  }  
]
```

Грид

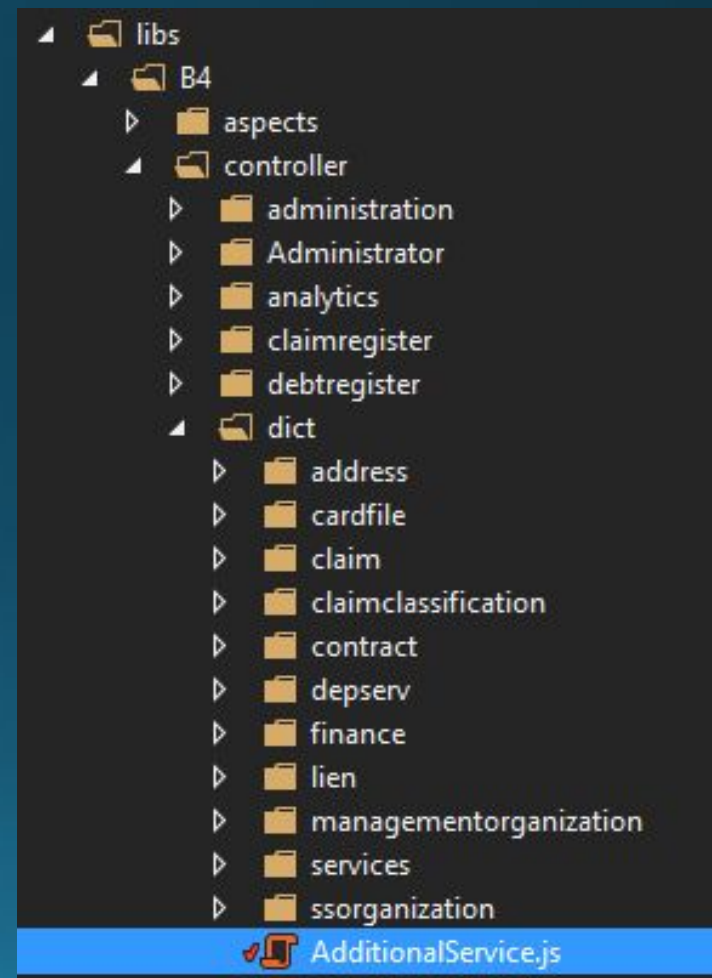
```
items: [  
  {  
    xtype: 'textfield',  
    fieldLabel: 'Наименование услуги',  
    allowBlank: false,  
    name: 'Name',  
    maxLength: 100,  
    anchor: '100%'  
  },  
  {  
    xtype: 'b4selectfield',  
    store: contragentStore,  
    name: 'Contragent',  
    modal: true,  
    anchor: '100%',  
    fieldLabel: 'Поставщик',  
    displayField: 'Name',  
    valueField: 'Id',  
    editable: false,  
    allowBlank: false,  
    columns: [  
      {  
        text: 'Наименование',  
        dataIndex: 'Name',  
        flex: 1,  
        filter: {  
          xtype: 'textfield'  
        }  
      }  
    ]  
  }  
]
```

Окно редактирования

Создание контроллера

- Создать скрипт в поддиректории controller, название класса создаваемого контроллера обязательно должно совпадать с физическим расположением скрипта

```
Ext.define('B4.controller.dict.AdditionalService', {  
    extends: 'B4.base.Controller',
```



Создание контроллера

- Наследуемся от `B4.base.Controller`
- Указываем зависимости в `requires`
- Перечисляем используемые представления в массиве `views`
- Расширяем функциональность контроллера миксинами в объекте `mixins`
- Перечисляем ссылки на компоненты в массиве `refs` для унифицированного доступа к ним, в свойстве `ref` указывается имя ссылки, а в свойстве `selector` – селектор компонента, в последствии к компоненту можно будет обращаться `controller.get[Имя ссылки]()`

```
Ext.define('B4.controller.dict.AdditionalService',
    extend: 'B4.base.Controller',
    requires: [
        'B4.aspects.GridEditWindow'
    ],
    views: [
        'dict.additionalservice.Grid',
        'dict.additionalservice.Window'
    ],
    mixins: {
        context: 'B4.mixins.Context'
    },
    refs: [
        {
            ref: 'mainView',
            selector: 'additionalservicegrid'
        },
        {
            ref: 'mainWindow',
            selector: 'additional servicewindow'
        }
    ],
    ],
```

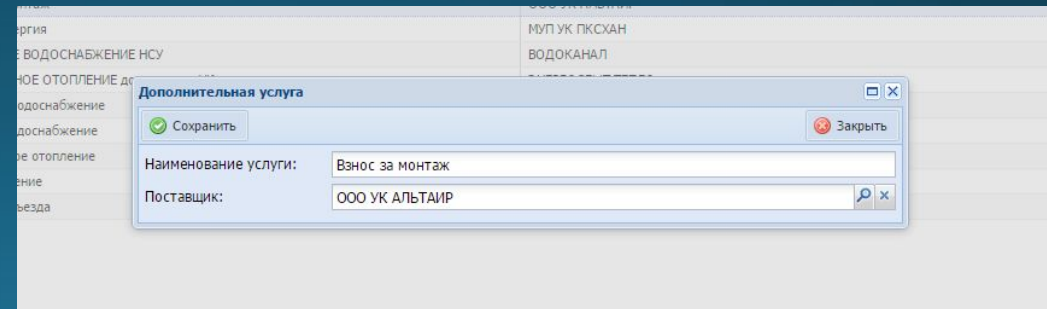
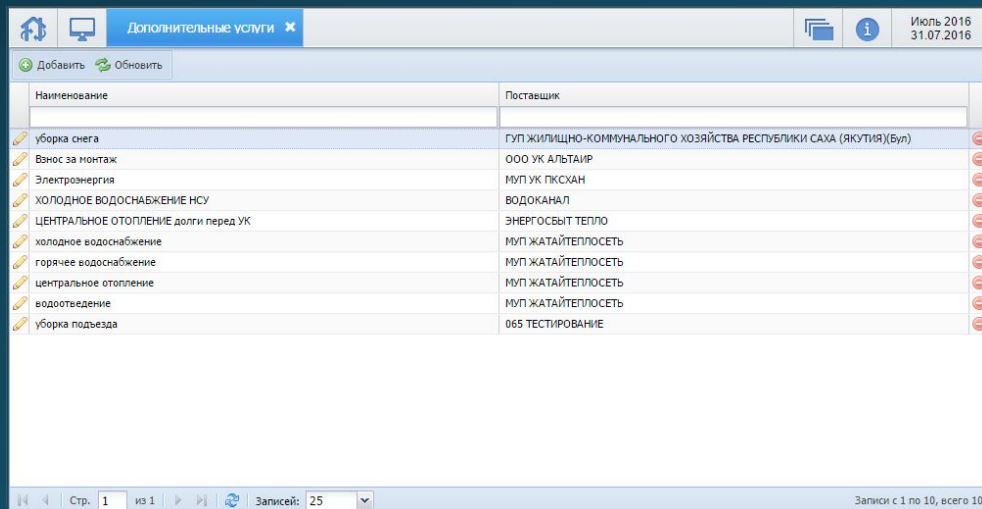
Создание контроллера Аспект

- Универсальная настраиваемая функциональность
- Перечисляем используемые аспекты в массиве `aspects`
- У каждого аспекта набор настраиваемых свойств уникален, общие у них свойства `xtype` – алиас аспекта и `name` – имя, по которому к аспекту можно будет обратиться из контроллера

```
aspects: [  
  {  
    xtype: 'grideditwindowaspect',  
    name: 'additionalServiceGridWindowAspect',  
    gridSelector: 'additionalServicegrid',  
    editFormSelector: 'additionalServiceWindow',  
    modelName: 'dict.AdditionalService',  
    editWindowView: 'dict.additionalService.Window'  
  }  
],
```

Создание контроллера GridEditWindowAspect

- Загрузка и обновление списка в гриде
- Удаление записей из грида
- Открытие записи на редактирование
- Сохранение редактируемой записи
- Создание новой записи



Создание контроллера GridEditWindowAspect

- Указываем xtype 'grideditwindowaspect'
- Указываем в свойстве name уникальное в текущем контроллере имя для аспекта
- Указываем в свойстве gridSelector селектор грида
- Указываем в свойстве editFormSelector селектор окна редактирования
- Указываем используемую модель в свойстве modelName
- Указываем класс представления окна редактирования в свойстве editWindowView

```
xtype: 'grideditwindowaspect',  
name: 'additionalServiceGridWindowAspect',  
gridSelector: 'additionalServiceGrid',  
editFormSelector: 'additionalServiceWindow',  
modelName: 'dict.AdditionalService',  
editWindowView: 'dict.additionalService.Window'
```

Создание контроллера События GridEditWindowAspect

- Подписка производится перечислением в массиве listeners объектов, где имена свойств – имена событий, а значения – обработчики событий
- Beforewindowcreated – перед созданием окна редактирования
- Windowcreated – после создания окна редактирования
- Beforerowaction – перед выполнением действия со строкой грида
- Beforegridaction – перед выполнением действия с гридом
- Beforesetformdata – перед заполнением окна редактирования данными
- Aftersetformdata – после заполнения окна редактирования данными
- Beforesaverequest – перед выполнением запроса на сохранение
- Getdata – при получении данных для сохранения
- Validate – при валидации формы
- Beforesave – перед сохранением
- Savesuccess – после успешного сохранения
- Deletesuccess – после успешного удаления
- Savefailure – при неудачном сохранении
- Beforedelete – перед удалением

Создание контроллера

Инициализация

- Описываем метод инициализации контроллера
- Перечисляем события, на которые подписывается контроллер, передавая их в качестве аргумента методу `control()`
- При подписке на событие указываем в качестве имени свойства селектор компонента, который собираемся прослушивать, а в качестве значения – вложенный объект, у которого имена свойств – имена событий, а значения – методы обработки событий или еще вложенный объект, у которого указывается метод-обработчик в свойстве `fn` и контекст вызова обработчика в свойстве `scope`
- Обязательно вызываем родительский инициализатор через `callParent()`

```
init: function () {
  var me = this,
      actions = {
        'additionalServiceGrid': {
          afterrender: { fn: me.onRenderGrid, scope: me }
        }
      };

  me.callParent(arguments);
  me.control(actions);
},
```


Создание контроллера

Метод index

- Описываем метод, выполняющийся каждый раз при переходе по маршруту, соответствующему контроллеру, если в маршруте были переданы параметры, они придут в качестве аргументов метода index
- Необходимо отобразить представление, соответствующее маршруту – для этого мы или получаем уже созданное представление, вызывая геттер `getMainView()`, или, если представление еще не было создано, создаем его, используя метод `Ext.widget()`, который принимает на вход алиас представления
- Необходимо выполнить привязку контекста, вызвав метод `bindContext()`, для корректной работы при одновременном открытии несколько экземпляров представлений
- Отобразить представление, вызвав метод `application.deployView()`

```
index: function () {  
    var view = this.getMainView() || Ext.widget('additionalervicegrid');  
    this.bindContext(view);  
    this.application.deployView(view);  
},
```

Создание контроллера

Передача параметров гриду

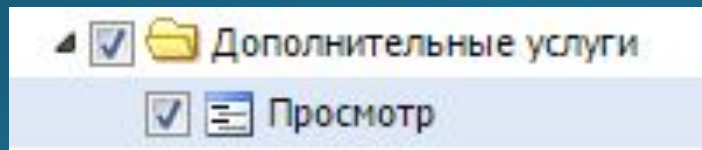
- Обработчик описанного выше события `afterrender` грида
- Получаем стор грида, вызывая метод `grid.getStore()`
- Подписываемся на его событие загрузки `beforeload`, вызываемое перед каждой загрузкой стора, где первым аргументом приходит сам стор, а вторым – объект, параметризующий запрос на получение данных
- Конфигурируем свойство `params` у объекта `operation`, указывая у него параметры, которые мы хотим передать гриду
- Подписка на событие загрузки стора нельзя выполнять в методе `index`

```
onRenderGrid: function (grid) {  
    var store = grid.getStore();  
  
    store.on({  
        beforeload: function(curStore, operation) {  
            operation.params = operation.params || {};  
            operation.params.someMyParameter = 'someMyValue';  
        }  
    });  
  
    store.load();  
}
```

Создание права на просмотр реестра

- В реализации класса `PermissionMap` (в проекте КР60 - `Кр60PermissionMap`) добавить регистрацию права
- Вызвать метод `Permission`, передав первым аргументом внутреннее имя права, а вторым – его наименование на понятном юзеру языке

```
Namespace("Dictionaries.Register.AdditionalService", "Дополнительные услуги");  
Permission("Dictionaries.Register.AdditionalService.View", "Просмотр");
```



Создание маршрута

- В реализации интерфейса `IClientRouteMapRegistrar` (в КП60 - `ClientRouteMapRegistrar`) зарегистрировать новый маршрут
- Вызвать метод `AddRoute` на объекте `map`, передав ему экземпляр класса `ClientRoute`, у которого в конструкторе первым параметром передается имя маршрута, а вторым – имя `ExtJs` класса контроллера, соответствующего указанному маршруту

```
map.AddRoute(new ClientRoute("additionalservice", "B4.controller.dict.AdditionalService"));
```

 kp-test.bars-open.ru/test-saha/#additionalservice/

Создание элемента главного меню для перехода в реестр

- В реализации интерфейса `INavigationProvider` (в КП6о - `NavigationProvider`) зарегистрировать элемент меню
- Вызвать метод `Add` на объекте `dictRegister`, передав первым параметром наименование элемента меню, а вторым параметром – имя маршрута, на который будет перенаправлять нажатие на элемент меню
- При желании можно последовательно вызвать метод `AddRequiredPermission` для указания имени права, которое будет требоваться для отображения элемента меню

```
dictRegister.Add("Доступные услуги", "calcmonthservices")  
    .AddRequiredPermission("Dictionaries.Register.CalcMonthServices.View");
```



Дополнительные
услуги

Создание стора

- Создаем скрипт в поддиректории store
- Указываем класс, от которого наследуемся
- Указываем зависимости
- Указываем модель

Создание модели

- Создаем скрипт в поддиректории model
- Указываем класс, от которого наследуемся
- Указываем прокси
- Указываем поля модели

Создание контроллера

- Создаем скрипт в поддиректории controller
- Указываем класс, от которого наследуемся
- Указываем зависимости
- Указываем представления
- Указываем ссылки
- При необходимости указываем аспекты
- Описываем метод `init` с подписками на события компонентов
- Описываем метод `index` с отрисовкой представления

Завершение создания контроллера

- Создаем право в реализации класса `PermissionMap`
- Регистрируем маршрут в реализации интерфейса `IClientRouteMapRegistrar`
- Регистрируем элемент меню в реализации интерфейса `INavigationProvider`
- У скриптов должен быть тип `Embedded resource`
- После добавления скриптов необходимо выполнять построение шаблона `ResourceManifest.tt`

Ресурсы с документацией

- Официальная документация ExtJs - <http://docs.sencha.com/extjs/4.2.1/>
- Документация Б4 - <http://docs.bars-open.ru/b4docsjs/>