Describe common structure/architecture

Some URLs in backend start with “admin/admin/…” - 2 times admin, some only 1 - “admin/…”. How?

Sample router:

```
<router id="admin">
    <route id="adminhtml"> <!-- frontName="admin" --
>
        <module name="Magento_Theme" />
    </route>
    <route id="theme" frontName="theme">
        <module name="Magento_Theme" />
    </route>
</router>
```

Sample menu items:

- menu add action=“adminhtml/system_design_theme” http://example.com/admin/admin/system_design_theme/index/key/$secret/ \Magento\Theme\Controller\Adminhtml\System\Design\Theme\Index
  - /admin/admin/system_design_theme/index
  - App\AreaList.getCodeByFrontName(‘admin’)
  - \Magento\Backend\App\Area\FrontNameResolver::getFrontName
  - default front name = deployment config `backend/frontName` - from env.php
  - if config `admin/url/use_custom_path`, admin front name = config `admin/url/custom_path`
- menu add action=“theme/design_config” http://example.com/admin/theme/design_config/index/key/$secret/ \Magento\Theme\Controller\Adminhtml\Design\Config\Index

Fun fact: same “Content > Design > Themes” link can open by 2 different URLs:

- http://example.com/admin/admin/system_design_theme/index/key/82c8...6bfa/
- http://example.com/admin/theme/system_design_theme/index/key/0ed6...6c75/

First link “hides” behind Magento_Backend original route “adminhtml”=“admin” frontName. Second link has own frontName=“theme” and is more beautiful.

Admin app flow:

- bootstrap.run(application)
- application.launch
- front controller.dispatch
- admin routerList:
  - `admin`
  - `default`
- Magento\Backend\App\Router.match - same as frontend, but admin parses*4* sections:
  - parseRequest
    - `_requiredParams = ['areaFrontName', 'moduleFrontName', 'actionPath', 'actionName']`
    - “admin/admin/system_design_theme/index”

`router id="admin"`

Admin router - Base, same as frontend, BUT:

- `_requiredParams = ['areaFrontName', 'moduleFrontName', 'actionPath', 'actionName']` - *4* sections, rest is params. Example: “kontrollpanel/theme/design_config/index/key/3dd89…7f6e/”:
    - moduleFrontName = “theme”, actionPath = “design_config”, actionName = “index”,
    - parameters = [key: 3dd89…7f6e]
    - fromt name “theme” - search adminhtml/routers.xml for …? Example 2 "/kontrollpanel/admin/system_design_theme/index/key/4bd2…13/:
    - moduleFrontName = “admin”, actionPath = “system_design_theme”, actionName = “index”
    - parameters = [key: 4bd2…13]
- `pathPrefix = 'adminhtml'` – all controller classes are searched in `Module\Controller\Adminhtml\...`

When in *adminhtml area*, ALL controllers will reside in Controller/Adminhtml/ modules path.

Module_Backend registers router `<route name="adminhtml" frontName="admin">` => `<module name="Magento_Backend"/>`. This means that `getUrl("adminhtml/controller/action")`:

- Magento\Backend\Model\Url adds prefix `$areaFrontName/` in `_getActionPath`
- *route name* “adminhtml”, frontName will always be “admin/”
- default controllers search module is *Magento_Backend*
- declaring other modules before “Magento_Backend” you risk overriding some system backend controllers:
    - ajax/translate, auth/login, dashboard/index, index/index, system/store etc.

You can in turn *register own router* `<route name="theme" frontName="theme">` => `<module name="Module_Alias"/>`. You will generate links with `getUrl("theme/controller/action")`:

- backend URL model._getActionPath adds prefix areaFrontName
- *route name* “theme”, frontName “theme”
- controlles are searched *only in your module* Module_Alias, no interference with Magento_Backend controllers.

What does `router id="admin"`, `router id="standard"` mean in routes.xml?

```
<config>
    <router id="admin">
        <routes>
            <route name="adminhtml" frontName="admin">...</route>
        </routes>
    </router>
    <router id="standard">
        <routes>
            <route name="checkout" frontName="checkout">...
</route>
        </routes>
    </router>
</config>
```

We know that frontend and admin areas main *router* is almost the same -

\Magento\Framework\App\Router\Base. It simply finds *routes* by matching `frontName`.

Key difference is matched *area* in *area list* defined `router` name:

- `adminhtml` area - `admin` *router*.
- `frontend` area - `standard` *router*. All routes and frontNames are read only from router by ID.

This means that admin routes always lie in etc/adminhtml/routes.xml and have `router id="admin"`. Frontend routes - etc/frontend/routes.xml and always have `router id="standard"`. This looks like redundancy to me.

Admin controller load summary (one more time):

- match backend route name
- match adminhtml area, router = 'admin'
- load adminhtml configuration
- adminhtml/di.xml - routerList = [admin, default]
- admin router = extends frontend Base router
- parse params - 4 parts: [front area from env.php, frontName, controllerPath, actionName]
- search routes.xml/router id="admin"/route by frontName=$frontName - second parameter
- matched route has 1+ ordered module names, e.g. [Module_Theme, Module_Backend]
- search controller class in each Module_Name/Controller/Adminhtml/[Controller/Path]/[ActionName]

URL generation:

In *frontend* URLs are generated with `\Magento\Framework\Url.getUrl`:

- getUrl() -> createUrl() = getRouterUrl + ?query + #fragment
- getRouterUrl() = getBaseUrl() + `_getRoutePath`
- getBaseUrl() example 'http://www.example.com/'
- `_getRoutePath` = `_getActionPath` + params pairs. For example: 'catalog/product/view/id/47'
- `_getActionPath` example: 'catalog/product/view'

In *backend* use customized version `\Magento\Backend\Model\Url`:

- `_getActionPath` *prepends* area front name, e.g. 'kontrollpanel/catalog/product/view'. Now all backend links have this prefix.
- getUrl() adds secret key /key/hash
- can disable secret key generation with turnOffSecretKey() and revert with turnOnSecretKey(). Global admin config `admin/security/use_form_key`.

Secret key = `hash($routeName . $controllerName . $actionName . $sessionFormKey)`

Describe the difference between Adminhtml and frontend. What additional tools and requirements exist in the admin?

- ACL permissions - `_isAllowed`, change static const `ADMIN_RESOURCE`
- base controller Magento\Backend\App\Action = Magento\Backend\App\AbstractAction
- custom *URL model* - secret key `key`
- layout `acl` block attribute
- ...