import { AbilityConstant, UIAbility, Want } from '@kit.AbilityKit';

import { hilog } from '@kit.PerformanceAnalysisKit';

import { window } from '@kit.ArkUI';

import { BreakpointSystem } from '@jellyfin-harmony/core';

import { AvoidAreaSystem } from '@jellyfin-harmony/core';

export default class EntryAbility extends UIAbility {

private breakpointSystem: BreakpointSystem = new BreakpointSystem();

private avoidAreaSystem: AvoidAreaSystem = new AvoidAreaSystem();

onCreate(want: Want, launchParam: AbilityConstant.LaunchParam): void {

hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onCreate');

}

onDestroy(): void {

hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onDestroy');

}

onWindowStageCreate(windowStage: window.WindowStage): void {

// Main window is created, set main page for this ability

hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onWindowStageCreate');

windowStage.getMainWindowSync().setImmersiveModeEnabledState(true)

this.breakpointSystem.register();

this.avoidAreaSystem.register(windowStage.getMainWindowSync())

windowStage.loadContent('pages/splash/SplashPage', (err) => {

if (err.code) {

hilog.error(0x0000, 'testTag', 'Failed to load the content. Cause: %{public}s', JSON.stringify(err) ?? '');

return;

}

hilog.info(0x0000, 'testTag', 'Succeeded in loading the content.');

});

}

onWindowStageDestroy(): void {

this.avoidAreaSystem.unregister()

this.breakpointSystem.unregister()

// Main window is destroyed, release UI related resources

hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onWindowStageDestroy');

}

onForeground(): void {

// Ability has brought to foreground

hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onForeground');

}

onBackground(): void {

// Ability has back to background

hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onBackground');

}

}

import { AbilityStage, Configuration, ConfigurationConstant } from "@kit.AbilityKit";

import { AppPrefer } from "../prefer/AppPrefer";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/7 21:01

\* @Version V1.0

\* @Description

\*/

export class App extends AbilityStage {

onCreate(): void {

// https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/arkts-light-dark-color-adaptation-V5#section1421172621111

this.context.getApplicationContext().setColorMode(ConfigurationConstant.ColorMode.COLOR\_MODE\_NOT\_SET);

AppStorage.setOrCreate(AppPrefer.PREFER, new AppPrefer(this.context.getApplicationContext()))

}

onConfigurationUpdate(newConfig: Configuration): void {

}

}

/\*\*

\* Common constants for common component.

\*/

export class CommonConstants {

// Font family

static readonly HARMONY\_HEI\_TI\_FONT\_FAMILY = 'HarmonyHeiTi';

static readonly HARMONY\_HEITI\_MEDIUM\_FONT\_FAMILY = 'HarmonyHeiTi-Medium';

static readonly HARMONY\_HEITI\_BOLD\_FONT\_FAMILY = 'HarmonyHeiTi-Bold';

// Font weight

static readonly DIALOG\_TITLE\_FONT\_WEIGHT: number = 700;

static readonly DIALOG\_BUTTON\_FONT\_WEIGHT: number = 500;

static readonly NORMAL\_FONT\_WEIGHT: number = 400;

// Opacity

static readonly FIRST\_LEVEL\_OPACITY: number = 0.9;

static readonly SECOND\_LEVEL\_OPACITY: number = 0.6;

static readonly HALF\_OPACITY: number = 0.5;

static readonly THIRD\_LEVEL\_OPACITY: number = 0.3;

static readonly Divider\_OPACITY: number = 0.05;

// Blur

static readonly REGULAR\_BLUR: number = 250;

// Space

static readonly SPACE\_4: number = 4;

static readonly SPACE\_8: number = 8;

static readonly SPACE\_12: number = 12;

static readonly SPACE\_16: number = 16;

// MaxLines

static readonly MAX\_LINE\_TWO: number = 2;

// Column count

static readonly SM\_COLUMN\_COUNT: number = 1;

static readonly MD\_COLUMN\_COUNT: number = 2;

static readonly LG\_COLUMN\_COUNT: number = 3;

// Swiper duration

static readonly SWIPER\_DURATION: number = 1000;

// Percent

static readonly FULL\_PERCENT: string = '100%';

static readonly HALF\_PERCENT: string = '50%';

static readonly NAVI\_BAR\_WIDTH: string = '40%';

// Skeleton animation config

static readonly SKELETON\_ANIMATION: AnimateParam = {

duration: 400,

tempo: 0.6,

curve: Curve.EaseInOut,

delay: 200,

iterations: -1,

playMode: PlayMode.Alternate

}

// item

static readonly ITEM\_WIDTH = 140

static readonly ITEM\_RATIO = 0.67

static readonly ITEM\_HEIGHT = 240

}

import { ViewModel} from '@jellyfin-harmony/core'

import { Repository } from '../data/Repository'

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/6 23:28

\* @Version V1.0

\* @Description

\*/

export class AppViewModel extends ViewModel {

public readonly repository: Repository = AppStorage.get(Repository.REPOSITORY)!

constructor(context: Context) {

super(context)

}

}

import { RefreshDataSource } from "@abner/refresh";

import { LoadType } from "@jellyfin-harmony/core";

import { RefreshRepository } from "@jellyfin-harmony/core";

import { AppViewModel } from './AppViewModel'

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/27 21:36

\* @Version V1.0

\* @Description

\*/

export abstract class ListViewModel extends AppViewModel implements RefreshRepository {

protected readonly dataSource = new RefreshDataSource();

getDataSource(): RefreshDataSource {

return this.dataSource

}

abstract loadData(type: LoadType): Promise<void>

}

import { ToolBar, WebTool } from '@jellyfin-harmony/core'

import { CommonConstants } from '../../common/CommonConstants'

import { Repository } from '../../data/Repository'

@Entry

@Component

struct AboutPage {

private repository: Repository = AppStorage.get(Repository.REPOSITORY)!

@Builder

preferItemBuilder(icon: Resource, title: ResourceStr, callback: Callback<void>) {

Row({space: CommonConstants.SPACE\_12}) {

SymbolGlyph(icon)

.fontSize(24)

.fontColor([$r('sys.color.ohos\_id\_color\_text\_primary')])

Text(title)

.fontSize(18)

.fontColor($r('sys.color.ohos\_id\_color\_text\_primary'))

Blank()

}

.width(CommonConstants.FULL\_PERCENT)

.height(48)

.padding({

left: CommonConstants.SPACE\_16,

right: CommonConstants.SPACE\_16

})

.onClick(() => {

callback()

})

}

build() {

Column({space: CommonConstants.SPACE\_8}) {

ToolBar({

title: $r('app.string.about')

})

Column({ space: CommonConstants.SPACE\_12}) {

Row() {

Text($r('app.string.prefer\_current\_server'))

Text(this.repository?.getActiveInfo()?.serverInfo.serverName)

}

Row() {

Text($r('app.string.prefer\_current\_address'))

Text(this.repository?.getActiveInfo()?.addressInfo.address)

}

Row() {

Text($r('app.string.prefer\_current\_user'))

Text(this.repository?.getActiveInfo()?.userInfo.userName)

}

}

.alignItems(HorizontalAlign.Start)

.padding({

left: CommonConstants.SPACE\_16,

right: CommonConstants.SPACE\_16

})

this.preferItemBuilder($r('sys.symbol.doc\_plaintext'), $r('app.string.prefer\_privacy'), () => {

WebTool.openBrowser(getContext(), "https://agreement-drcn.hispace.dbankcloud.cn/index.html?lang=zh&agreementId=1574895840755995584")

})

}

.alignItems(HorizontalAlign.Start)

.width(CommonConstants.FULL\_PERCENT)

.height(CommonConstants.FULL\_PERCENT)

}

}

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/19 22:32

\* @Version V1.0

\* @Description Media args

\*/

export interface MediaArgs {

id: string,

name: string,

}

import { ListRefreshView, TimeTool } from "@jellyfin-harmony/core"

import { ActivityLogEntry } from "@jellyfin-harmony/sdk"

import { CommonConstants } from "../../../common/CommonConstants"

import { Repository } from "../../../data/Repository"

import { HomeToolBar } from "../../../widget/bar/HomeToolBar"

import { ActiveViewModel } from "./ActiveViewModel"

@Component

export struct ActivePage {

private viewModel = new ActiveViewModel(getContext())

@Builder

mediaListView(info: ActivityLogEntry, index: number) {

DeviceItem({item: info})

}

build() {

Column() {

HomeToolBar({

title: $r('app.string.home\_active')

})

ListRefreshView({

source: this.viewModel,

listAttribute: (attr) => {

attr.padding = {

left: CommonConstants.SPACE\_16,

right: CommonConstants.SPACE\_16,

}

attr.width = CommonConstants.FULL\_PERCENT

attr.height = CommonConstants.FULL\_PERCENT

},

itemLayout: this.mediaListView

})

.width(CommonConstants.FULL\_PERCENT)

.layoutWeight(1)

}

}

}

@Reusable

@Component

struct DeviceItem {

@StorageProp("Repository") repository?: Repository = undefined

@Require @State item?: ActivityLogEntry = undefined

aboutToReuse(params: Record<string, ActivityLogEntry>): void {

this.item = params['item']

}

build() {

Column({space: CommonConstants.SPACE\_4}) {

Text(new Date(this.item?.Date!).toLocaleString(undefined, TimeTool.options))

.width(CommonConstants.FULL\_PERCENT)

.maxLines(1)

Text(this.item?.Name)

.width(CommonConstants.FULL\_PERCENT)

.maxLines(1)

}

.width(CommonConstants.FULL\_PERCENT)

.height(60)

}

aboutToRecycle(): void {

console.log("ActiveList item recycle: " + this.item?.Name)

}

}

import { LoadType } from "@jellyfin-harmony/core";

import { ListViewModel } from "../../../lifecycle/ListViewModel";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/16 21:35

\* @Version V1.0

\* @Description

\*/

export class ActiveViewModel extends ListViewModel {

async loadData(type: LoadType): Promise<void> {

let libraryArray = await this.repository.getAllLogs()

this.dataSource.initData(libraryArray)

}

}

import { GridRefreshView, TimeTool } from "@jellyfin-harmony/core"

import { DeviceInfos } from "@jellyfin-harmony/sdk"

import { CommonConstants } from "../../../common/CommonConstants"

import { Repository } from "../../../data/Repository"

import { HomeToolBar } from "../../../widget/bar/HomeToolBar"

import { DeviceViewModel } from "./DeviceViewModel"

@Component

export struct DevicePage {

private viewModel = new DeviceViewModel(getContext())

@Builder

mediaListView(info: DeviceInfos, index: number) {

DeviceItem({item: info})

}

build() {

Column() {

HomeToolBar({

title: $r('app.string.home\_device')

})

GridRefreshView({

source: this.viewModel,

gridAttribute: (attr) => {

attr.padding = {

left: CommonConstants.SPACE\_16,

right: CommonConstants.SPACE\_16,

}

attr.rowsGap = CommonConstants.SPACE\_12

attr.columnsGap = CommonConstants.SPACE\_12

attr.columnsTemplate = ('1fr 1fr')

attr.width = CommonConstants.FULL\_PERCENT

attr.height = CommonConstants.FULL\_PERCENT

},

itemLayout: this.mediaListView

})

.width(CommonConstants.FULL\_PERCENT)

.layoutWeight(1)

}

}

}

@Reusable

@Component

struct DeviceItem {

@StorageProp("Repository") repository?: Repository = undefined

@Require @State item?: DeviceInfos = undefined

aboutToReuse(params: Record<string, object>): void {

this.item = params['item']

}

build() {

Column({space: CommonConstants.SPACE\_8}) {

Image("")

.alt($r('app.media.alt'))

.objectFit(ImageFit.Cover)

.autoResize(true)

.borderRadius($r('app.float.lg\_border\_radius'))

.width(CommonConstants.FULL\_PERCENT)

.aspectRatio(1.5)

Text(this.item?.Name)

.width(CommonConstants.FULL\_PERCENT)

.maxLines(1)

Row({space: CommonConstants.SPACE\_8}) {

Text(this.item?.AppName)

Text(this.item?.AppVersion)

}

.width(CommonConstants.FULL\_PERCENT)

Text(this.item?.LastUserName)

.width(CommonConstants.FULL\_PERCENT)

}

}

aboutToRecycle(): void {

}

}

import { LoadType } from "@jellyfin-harmony/core";

import { ListViewModel } from "../../../lifecycle/ListViewModel";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/16 21:19

\* @Version V1.0

\* @Description

\*/

export class DeviceViewModel extends ListViewModel {

async loadData(type: LoadType): Promise<void> {

let libraryArray = await this.repository.getAllDevices()

this.dataSource.initData(libraryArray)

}

}

import { router, window } from '@kit.ArkUI';

import { BaseItemDto, ImageType } from '@jellyfin-harmony/sdk';

import { BreakpointTypeEnum, GridRefreshView } from '@jellyfin-harmony/core';

import { CommonConstants } from '../../../common/CommonConstants';

import { HomeToolBar } from '../../../widget/bar/HomeToolBar';

import { Repository } from '../../../data/Repository';

import { MediaArgs } from '../../detail/MediaArgs';

import { LibraryViewModel } from './LibraryViewModel';

@Component

export struct LibraryPage {

private viewModel = new LibraryViewModel(getContext())

@StorageProp('currentBreakpoint') currentBreakpoint: string = BreakpointTypeEnum.MD;

@StorageProp('currentAvoidArea') currentAvoidArea?: window.AvoidArea = undefined

@Builder

mediaListView(info: BaseItemDto, index: number) {

MediaItem({item: info})

}

build() {

Column() {

HomeToolBar({

title: $r('app.string.home\_library')

})

GridRefreshView({

source: this.viewModel,

gridAttribute: (attr) => {

attr.padding = {

left: CommonConstants.SPACE\_16,

right: CommonConstants.SPACE\_16,

bottom: px2vp(this.currentAvoidArea?.bottomRect.height)

}

attr.rowsGap = CommonConstants.SPACE\_12

attr.columnsGap = CommonConstants.SPACE\_12

attr.columnsTemplate = ('1fr 1fr')

attr.width = CommonConstants.FULL\_PERCENT

attr.height = CommonConstants.FULL\_PERCENT

},

itemLayout: this.mediaListView

})

.width(CommonConstants.FULL\_PERCENT)

.layoutWeight(1)

}

}

}

@Reusable

@Component

struct MediaItem {

@StorageProp("Repository") repository?: Repository = undefined

@Require item?: BaseItemDto

aboutToReuse(params: Record<string, object>): void {

this.item = params['item']

}

build() {

Column({space: CommonConstants.SPACE\_8}) {

Image(this.repository?.buildImage(this.item!, ImageType.Primary))

.alt($r('app.media.alt'))

.objectFit(ImageFit.Cover)

.autoResize(true)

.borderRadius($r('app.float.lg\_border\_radius'))

.width('100%')

.aspectRatio(1.5)

Image($r('sys.symbol.dot\_grid\_1x2'))

Text(this.item?.Name)

.width('100%')

.maxLines(1)

}

.onClick(() => {

let args: MediaArgs = {

id: this.item?.Id!,

name: this.item?.Name!

}

router.pushUrl({url: 'pages/detail/DetailPage', params: args})

})

}

aboutToRecycle(): void {

}

}

import { LoadType } from "@jellyfin-harmony/core"

import { ListViewModel } from "../../../lifecycle/ListViewModel"

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/16 20:37

\* @Version V1.0

\* @Description

\*/

export class LibraryViewModel extends ListViewModel {

async loadData(type: LoadType): Promise<void> {

let libraryArray = await this.repository.loadMedia()

this.dataSource.initData(libraryArray)

}

}

import { ActivityLogEntry, SessionInfo, SystemInfo } from "@jellyfin-harmony/sdk";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/17 22:42

\* @Version V1.0

\* @Description 总览信息

\*/

export interface OverViewInfo {

systemInfo: SystemInfo

sessionInfos?: Array<SessionInfo>

loginInfos?: Array<ActivityLogEntry>

}

import { ScrollRefreshView, TimeTool } from "@jellyfin-harmony/core"

import { ActivityLogEntry, SessionInfo } from "@jellyfin-harmony/sdk"

import { CommonConstants } from "../../../common/CommonConstants"

import { HomeToolBar } from "../../../widget/bar/HomeToolBar"

import { OverViewInfo } from "./OverViewInfo"

import { OverviewViewModel } from "./OverviewViewModel"

@Component

export struct OverviewPage {

private viewModel = new OverviewViewModel(getContext())

@Builder

overviewBuilder(info: OverViewInfo) {

overviewView({

info: info

})

}

build() {

Column() {

HomeToolBar({

title: $r('app.string.home\_overview')

})

ScrollRefreshView({

source: this.viewModel,

itemLayout: (info: ESObject) => {

this.overviewBuilder(info)

}

})

.width(CommonConstants.FULL\_PERCENT)

.layoutWeight(1)

}

}

}

@Component

struct overviewView {

@Require @State info?: OverViewInfo = undefined

build() {

// Server info

Column({space: CommonConstants.SPACE\_16}) {

Column({space: CommonConstants.SPACE\_12}) {

Text($r('app.string.home\_overview\_server'))

.fontSize($r('sys.float.Title\_M'))

Row() {

Text($r('app.string.home\_overview\_server\_address'))

Text(this.info?.systemInfo.LocalAddress)

}

Row() {

Text($r('app.string.home\_overview\_product\_ark'))

Text(this.info?.systemInfo.SystemArchitecture)

}

Row() {

Text($r('app.string.home\_overview\_server\_name'))

Text(this.info?.systemInfo.ServerName)

}

Row() {

Text($r('app.string.home\_overview\_server\_version'))

Text(this.info?.systemInfo.Version)

}

}

.backgroundColor($r('sys.color.comp\_background\_list\_card'))

.shadow({ radius: CommonConstants.SPACE\_16, color: Color.Gray })

.padding(CommonConstants.SPACE\_16)

.border({radius: CommonConstants.SPACE\_16})

.width(CommonConstants.FULL\_PERCENT)

.alignItems(HorizontalAlign.Start)

// Active Session

if (this.info?.sessionInfos) {

Column({space: CommonConstants.SPACE\_12}) {

Text($r('app.string.home\_overview\_session'))

.fontSize($r('sys.float.Title\_M'))

List({space: CommonConstants.SPACE\_12}) {

ForEach(this.info?.sessionInfos, (item: SessionInfo, index) => {

ListItem() {

Column() {

Text(item.DeviceName)

Text(`${item.Client} ${item.ApplicationVersion}`)

}

.width(CommonConstants.FULL\_PERCENT)

.alignItems(HorizontalAlign.Start)

}

})

}

}

.backgroundColor($r('sys.color.comp\_background\_list\_card'))

.shadow({ radius: CommonConstants.SPACE\_16, color: Color.Gray })

.padding(CommonConstants.SPACE\_16)

.border({radius: CommonConstants.SPACE\_16})

.width(CommonConstants.FULL\_PERCENT)

.alignItems(HorizontalAlign.Start)

}

// Active Log

if (this.info?.loginInfos) {

Column({space: CommonConstants.SPACE\_12}) {

Text($r('app.string.home\_overview\_log'))

.fontSize($r('sys.float.Title\_M'))

List({space: CommonConstants.SPACE\_12}) {

ForEach(this.info?.loginInfos, (item: ActivityLogEntry, index) => {

ListItem() {

Column() {

Text(new Date(item.Date!).toLocaleString(undefined, TimeTool.options))

Text(item.Name)

}

.width(CommonConstants.FULL\_PERCENT)

.alignItems(HorizontalAlign.Start)

}

})

}

}

.backgroundColor($r('sys.color.comp\_background\_list\_card'))

.shadow({ radius: CommonConstants.SPACE\_16, color: Color.Gray })

.padding(CommonConstants.SPACE\_16)

.border({radius: CommonConstants.SPACE\_16})

.width(CommonConstants.FULL\_PERCENT)

.alignItems(HorizontalAlign.Start)

}

// path

Column({space: CommonConstants.SPACE\_12}) {

Text($r('app.string.home\_overview\_path'))

.fontSize($r('sys.float.Title\_M'))

Row() {

Text($r('app.string.home\_overview\_path\_catch'))

Text(this.info?.systemInfo.CachePath)

.layoutWeight(1)

.maxLines(1)

}

.width(CommonConstants.FULL\_PERCENT)

Row() {

Text($r('app.string.home\_overview\_path\_logo'))

Text(this.info?.systemInfo.LogPath)

.layoutWeight(1)

.maxLines(1)

}

.width(CommonConstants.FULL\_PERCENT)

Row() {

Text($r('app.string.home\_overview\_path\_meta'))

Text(this.info?.systemInfo.InternalMetadataPath)

.layoutWeight(1)

.maxLines(1)

}

.width(CommonConstants.FULL\_PERCENT)

Row() {

Text($r('app.string.home\_overview\_path\_transcode'))

Text(this.info?.systemInfo.TranscodingTempPath)

.layoutWeight(1)

.maxLines(1)

}

.width(CommonConstants.FULL\_PERCENT)

Row() {

Text($r('app.string.home\_overview\_path\_web'))

Text(this.info?.systemInfo.WebPath)

.layoutWeight(1)

.maxLines(1)

}

.width(CommonConstants.FULL\_PERCENT)

}

.backgroundColor($r('sys.color.comp\_background\_list\_card'))

.shadow({ radius: CommonConstants.SPACE\_16, color: Color.Gray })

.padding(CommonConstants.SPACE\_16)

.border({radius: CommonConstants.SPACE\_16})

.width(CommonConstants.FULL\_PERCENT)

.alignItems(HorizontalAlign.Start)

}

.padding(CommonConstants.SPACE\_16)

.width(CommonConstants.FULL\_PERCENT)

.align(Alignment.Top)

.alignItems(HorizontalAlign.Start)

}

}

import { LoadType, SimpleRepository } from "@jellyfin-harmony/core";

import { AppViewModel } from "../../../lifecycle/AppViewModel";

import { OverViewInfo } from "./OverViewInfo";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/16 20:24

\* @Version V1.0

\* @Description

\*/

export class OverviewViewModel extends AppViewModel implements SimpleRepository {

async loadData(type: LoadType): Promise<OverViewInfo> {

let systemInfo = await this.repository.getServiceInfo()

let sessionInfos = await this.repository.getSessionInfos()

let logInfos = await this.repository.getLogInfos()

return {

systemInfo: systemInfo,

sessionInfos: sessionInfos,

loginInfos: logInfos

}

}

}

import { GridRefreshView, TimeTool } from "@jellyfin-harmony/core"

import { UserDto } from "@jellyfin-harmony/sdk"

import { CommonConstants } from "../../../common/CommonConstants"

import { Repository } from "../../../data/Repository"

import { HomeToolBar } from "../../../widget/bar/HomeToolBar"

import { MediaArgs } from "../../detail/MediaArgs"

import { UserViewModel } from "./UserViewModel"

import { router } from "@kit.ArkUI"

@Component

export struct UserPage {

private viewModel = new UserViewModel(getContext())

@Builder

mediaListView(info: UserDto, index: number) {

UserItem({item: info})

}

build() {

Column() {

HomeToolBar({

title: $r('app.string.home\_user')

})

GridRefreshView({

source: this.viewModel,

gridAttribute: (attr) => {

attr.padding = {

left: CommonConstants.SPACE\_16,

right: CommonConstants.SPACE\_16,

}

attr.rowsGap = CommonConstants.SPACE\_12

attr.columnsGap = CommonConstants.SPACE\_12

attr.columnsTemplate = ('1fr 1fr')

attr.width = CommonConstants.FULL\_PERCENT

attr.height = CommonConstants.FULL\_PERCENT

},

itemLayout: this.mediaListView

})

.width(CommonConstants.FULL\_PERCENT)

.layoutWeight(1)

}

}

}

@Reusable

@Component

struct UserItem {

@StorageProp("Repository") repository?: Repository = undefined

@Require item?: UserDto

aboutToReuse(params: Record<string, object>): void {

this.item = params['item']

}

build() {

Column({space: CommonConstants.SPACE\_8}) {

Image("")

.alt($r('app.media.alt'))

.objectFit(ImageFit.Cover)

.autoResize(true)

.borderRadius($r('app.float.lg\_border\_radius'))

.width('100%')

.aspectRatio(1.5)

Text(this.item?.Name)

.width('100%')

.maxLines(1)

Text(new Date(this.item?.LastActivityDate).toLocaleString(undefined, TimeTool.options))

.width('100%')

.maxLines(1)

}

.onClick(() => {

let args: MediaArgs = {

id: this.item?.Id!,

name: this.item?.Name!

}

router.pushUrl({url: 'pages/detail/DetailPage', params: args})

})

}

aboutToRecycle(): void {

}

}

import { LoadType } from "@jellyfin-harmony/core";

import { ListViewModel } from "../../../lifecycle/ListViewModel";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/16 21:05

\* @Version V1.0

\* @Description

\*/

export class UserViewModel extends ListViewModel {

async loadData(type: LoadType): Promise<void> {

let libraryArray = await this.repository.getAllUsers()

this.dataSource.initData(libraryArray)

}

}

import { window } from '@kit.ArkUI';

import { CommonConstants } from '../../common/CommonConstants';

import { ActivePage } from './active/ActivePage';

import { DevicePage } from './device/DevicePage';

import { LibraryPage } from './library/LibraryPage';

import { OverviewPage } from './overview/OverviewPage';

import { UserPage } from './user/UserPage';

@Entry

@Component

struct HomePage {

@State currentIndex: number = 0;

@StorageProp('currentAvoidArea') currentAvoidArea?: window.AvoidArea = undefined

private tabsController: TabsController = new TabsController();

aboutToAppear(): void {

}

@Builder TabBuilder(title: ResourceStr, targetIndex: number, img: Resource) {

Column({

space: CommonConstants.SPACE\_4

}) {

SymbolGlyph(img)

.fontSize(24)

.fontColor([this.currentIndex === targetIndex ? '#1698CE' : '#6B6B6B' ])

Text(title)

.fontSize($r('sys.float.Body\_M'))

.fontColor(this.currentIndex === targetIndex ? '#1698CE' : '#6B6B6B')

}

.width('100%')

.height(50)

.justifyContent(FlexAlign.Center)

.onClick(() => {

this.currentIndex = targetIndex;

this.tabsController.changeIndex(this.currentIndex);

})

}

// https://developer.huawei.com/consumer/cn/design/harmonyos-symbol/

build() {

Tabs({ barPosition: BarPosition.End, controller: this.tabsController }) {

TabContent() {

OverviewPage()

}

.tabBar(this.TabBuilder($r('app.string.home\_overview'), 0, $r('sys.symbol.house\_fill')))

TabContent() {

LibraryPage()

}

.tabBar(this.TabBuilder($r('app.string.home\_library'), 1, $r('sys.symbol.externaldrive\_fill')))

TabContent() {

UserPage()

}

.tabBar(this.TabBuilder($r('app.string.home\_user'), 2, $r('sys.symbol.person\_2\_fill')))

TabContent() {

DevicePage()

}

.tabBar(this.TabBuilder($r('app.string.home\_device'), 3, $r('sys.symbol.device\_2')))

TabContent() {

ActivePage()

}

.tabBar(this.TabBuilder($r('app.string.home\_active'), 4, $r('sys.symbol.list\_bullet\_square\_fill')))

}

.expandSafeArea([SafeAreaType.SYSTEM], [SafeAreaEdge.BOTTOM])

.barWidth('100%')

.barHeight(50)

.animationDuration(0)

.backgroundColor($r('sys.color.background\_primary'))

.onChange((index: number) => {

this.currentIndex = index;

})

.padding({

bottom: px2vp(this.currentAvoidArea?.bottomRect.height)

})

}

aboutToDisappear(): void {

}

}

import { AppViewModel } from "../../lifecycle/AppViewModel";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/12/7 11:28

\* @Version V1.0

\* @Description

\*/

export class HomeViewModel extends AppViewModel {

}

import { promptAction, router } from '@kit.ArkUI'

import { Repository } from '../../data/Repository'

import { ProgressDialog } from '@jellyfin-harmony/core'

import { common } from '@kit.AbilityKit'

@Entry

@Component

struct LoginPage {

private repository: Repository = AppStorage.get(Repository.REPOSITORY)!

private progressDialog?: ProgressDialog

private url?: string

private userName?: string

private psw?: string

public aboutToAppear(): void {

this.url = (router.getParams() as object)["url"]

this.progressDialog = new ProgressDialog(this.getUIContext())

}

build() {

RelativeContainer() {

TextInput({ placeholder: $r('app.string.login\_user\_name') })

.id("login\_name")

.constraintSize({

maxWidth: 300

})

.alignRules({

middle: { anchor: "\_\_container\_\_", align: HorizontalAlign.Center },

center: { anchor: "\_\_container\_\_", align: VerticalAlign.Center },

})

.onChange((value) => {

this.userName = value

})

Image($r('app.media.foreground'))

.size({

width: 100,

height: 100

})

.alignRules({

middle: { anchor: "\_\_container\_\_", align: HorizontalAlign.Center },

top: { anchor: "\_\_container\_\_", align: VerticalAlign.Top },

bottom: { anchor: "login\_name", align: VerticalAlign.Top },

})

TextInput({ placeholder: $r('app.string.login\_psw') })

.id("id\_psw")

.type(InputType.Password)

.constraintSize({

maxWidth: 300

})

.margin({

top: 15

})

.alignRules({

middle: { anchor: "\_\_container\_\_", align: HorizontalAlign.Center },

top: { anchor: "login\_name", align: VerticalAlign.Bottom },

})

.onChange((value) => {

this.psw = value

})

Button($r('app.string.login\_auth'))

.id("login\_login")

.constraintSize({

minWidth: 100

})

.margin({

top: 15

})

.alignRules({

middle: { anchor: "\_\_container\_\_", align: HorizontalAlign.Center },

top: { anchor: "id\_psw", align: VerticalAlign.Bottom },

})

.onClick(() => {

this.auth()

})

}

}

private auth() {

if (this.url == undefined) {

promptAction.showToast({message: $r('app.string.error\_url\_not\_validate')})

return

}

if (this.userName == undefined || this.userName.length <= 0) {

promptAction.showToast({message: $r('app.string.error\_user\_name\_not\_validate')})

return

}

this.progressDialog?.show()

this.repository?.authAccount(this.url, this.userName, this.psw)

.then((activeInfo) => {

this.progressDialog?.dismiss()

this.repository!.setActiveInfo(activeInfo)

let state = router.getStateByIndex(0)

if (state?.name == "pages/home/HomePage") {

router.back(0)

} else {

(getContext() as common.UIAbilityContext).windowStage.loadContent("pages/home/HomePage")

}

})

.catch((error: Error) => {

this.progressDialog?.dismiss()

promptAction.showToast({message: error.message})

})

}

}

import { Repository } from '../../data/Repository'

import { ProgressDialog } from '@jellyfin-harmony/core'

import { promptAction, router } from '@kit.ArkUI'

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/7 21:21

\* @Version V1.0

\* @Description

\*/

@Entry

@Component

struct ServerPage {

private url: string = ""

private progressDialog?: ProgressDialog

private repository: Repository = AppStorage.get(Repository.REPOSITORY)!

aboutToAppear(): void {

this.progressDialog = new ProgressDialog(this.getUIContext())

}

build() {

RelativeContainer() {

TextInput({ placeholder: $r('app.string.server\_address') })

.id("server\_address")

.constraintSize({

maxWidth: 300

})

.alignRules({

middle: {anchor: "\_\_container\_\_", align: HorizontalAlign.Center},

center: { anchor: "\_\_container\_\_", align: VerticalAlign.Center}

})

.onChange((value) => {

this.url = value

})

Image($r('app.media.foreground'))

.width(120)

.height(120)

.alignRules({

middle: {anchor: "\_\_container\_\_", align: HorizontalAlign.Center},

top: { anchor: "\_\_container\_\_", align: VerticalAlign.Top},

bottom: { anchor: "server\_address", align: VerticalAlign.Top}

})

Button($r('app.string.server\_connect'))

.id("server\_connect")

.width(120)

.margin(15)

.constraintSize({

minWidth: 100

})

.alignRules({

middle: {anchor: "\_\_container\_\_", align: HorizontalAlign.Center},

top: { anchor: "server\_address", align: VerticalAlign.Bottom}

})

.onClick(() => {

this.connectServer()

})

LoadingProgress()

.alignRules({

start: {anchor: "server\_connect", align: HorizontalAlign.Start},

top: { anchor: "server\_connect", align: VerticalAlign.Top},

bottom: { anchor: "server\_connect", align: VerticalAlign.Bottom}

})

.visibility(Visibility.Hidden)

}

}

private connectServer() {

this.progressDialog?.show()

this.repository?.connectServer(this.url)

.then((serverInfo) => {

this.progressDialog?.dismiss()

let obj: Record<string, string> = {

"url": this.url

}

router.pushUrl({url: "pages/login/LoginPage", params: obj})

})

.catch((error: Error) => {

this.progressDialog?.dismiss()

promptAction.showToast({message: error.message})

})

}

aboutToDisappear(): void {

this.progressDialog?.dismiss()

this.progressDialog = undefined

}

}

import { PrivacyDialog } from '@jellyfin-harmony/core'

import { router } from '@kit.ArkUI'

import { Repository } from '../../data/Repository'

import { AppPrefer } from '../../prefer/AppPrefer'

@Entry

@Component

struct SplashPage {

private repository: Repository = AppStorage.get(Repository.REPOSITORY)!

private appPrefer: AppPrefer = AppStorage.get(AppPrefer.PREFER)!

private privacyDialog: PrivacyDialog = new PrivacyDialog(this.getUIContext())

aboutToAppear(): void {

if (this.appPrefer.isPrivacyGrant()) {

setTimeout(() => {

this.enterHomeOrAddServer()

}, 1000)

} else {

let resourceManager = getContext().resourceManager

let privacyUrl = resourceManager.getStringSync($r('app.string.privacy'))

this.privacyDialog.setAppInfo(privacyUrl)

this.privacyDialog.setPrivacyCallback(() => {

this.appPrefer.setPrivacyGrant()

this.privacyDialog.dismiss(true)

setTimeout(() => {

this.enterHomeOrAddServer()

}, 1000)

})

this.privacyDialog.show()

}

}

build() {

Stack() {

Image($r('app.media.foreground'))

.size({

width: 100,

height: 100

})

}

.width('100%')

.height('100%')

}

onBackPress(): boolean | void {

return true

}

private enterHomeOrAddServer() {

let path = 'pages/server/ServerPage'

if (this.repository?.getActiveInfo()) {

path = 'pages/home/HomePage'

}

router.replaceUrl({url: path})

}

}

import { preferences } from "@kit.ArkData"

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/9 18:36

\* @Version V1.0

\* @Description

\*/

export class AppPrefer {

public static readonly PREFER = "prefer"

private static readonly KEY\_PRIVACY\_GRANT = "privacy\_grant"

private readonly preference: preferences.Preferences

constructor(context: Context) {

this.preference = preferences.getPreferencesSync(context, { name: "app\_prefer" } )

}

public isPrivacyGrant(): boolean {

if (this.preference.hasSync(AppPrefer.KEY\_PRIVACY\_GRANT)) {

return this.preference.getSync(AppPrefer.KEY\_PRIVACY\_GRANT, false) as boolean

}

return false

}

public setPrivacyGrant(): void {

this.preference.putSync(AppPrefer.KEY\_PRIVACY\_GRANT, true)

this.preference.flush()

}

}

import { StartupConfig, StartupConfigEntry } from '@kit.AbilityKit';

import { BusinessError } from '@ohos.base';

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/9 21:20

\* @Version V1.0

\* @Description

\*/

export class AppStartupConfig extends StartupConfigEntry {

private onCompleteCallback = (error: BusinessError<void>) => {

console.log("AppStartupConfig: onComplete.")

}

onConfig(): StartupConfig {

console.log("AppStartupConfig: onConfig.")

let config: StartupConfig = {

timeoutMs: 10000,

startupListener: {onCompleted: this.onCompleteCallback}

}

return config

}

}

import { common, StartupTask } from "@kit.AbilityKit";

import { Repository } from "../data/Repository";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/10 1:04

\* @Version V1.0

\* @Description

\*/

@Sendable

export class AppStartupTask extends StartupTask {

public async init(context: common.AbilityStageContext): Promise<void> {

console.log("AppStartupTask: init 1.")

let repository = new Repository(context.getApplicationContext())

AppStorage.setOrCreate(Repository.REPOSITORY, repository)

try {

await repository.init()

} catch (error) {

console.error("AppStartupTask: init error " + error['message'])

}

console.log("AppStartupTask: init 2.")

}

}

import { CommonConstants } from "../../common/CommonConstants"

import { router, window } from "@kit.ArkUI"

@Component

export struct HomeToolBar {

@Require @Prop title: ResourceStr

@StorageProp("currentAvoidArea") avoidArea?: window.AvoidArea = undefined

build() {

Column() {

Column()

.width('100%')

.height(px2vp(this.avoidArea?.topRect.height))

Row() {

Text(this.title)

.fontSize($r('app.float.header\_font\_size'))

.fontColor($r('sys.color.ohos\_id\_color\_text\_primary'))

.fontWeight(FontWeight.Bold)

.textAlign(TextAlign.Start)

.fontFamily(CommonConstants.HARMONY\_HEITI\_BOLD\_FONT\_FAMILY)

.margin({ right: $r('app.float.sm\_padding\_margin') })

// Search({ placeholder: $r('app.string.search') })

// .focusable(false)

// .fontColor($r('sys.color.ohos\_id\_color\_text\_primary'))

// .textFont({ size: $r('app.float.large\_text\_size') })

// .width($r('app.float.search\_width'))

// .height($r('app.float.search\_height'))

// .onClick(() => {

// this.getUIContext().getRouter().pushUrl({

// url: "pages/search/SearchPage"

// })

// })

SymbolGlyph($r('sys.symbol.more'))

.fontSize($r('app.float.header\_font\_size'))

.fontColor([$r('sys.color.ohos\_id\_color\_text\_primary')])

.onClick(() => {

router.pushUrl({url: "pages/about/AboutPage"})

})

}

.padding({

left: $r('app.float.xxl\_padding\_margin'),

right: $r('app.float.lg\_padding\_margin'),

})

.justifyContent(FlexAlign.SpaceBetween)

.width(CommonConstants.FULL\_PERCENT)

.height($r('app.float.top\_navigation\_height'))

}

}

}

import { deviceInfo } from "@kit.BasicServicesKit";

import {

ActivityLogEntry,

BaseItemDto, BaseItemKind,

BaseItemPerson,

ClientInfo,

CollectionType,

DeviceInfo,

DeviceInfos,

ImageType,

ItemFields,

ItemFilter,

ItemSortBy,

Jellyfin,

PlaybackInfoResponse,

PublicSystemInfo,

SessionInfo,

SortOrder,

SystemInfo,

UserDto,

UserItemDataDto} from "@jellyfin-harmony/sdk";

import { ServerInfo } from "@jellyfin-harmony/core";

import { UserInfo } from "@jellyfin-harmony/core";

import { DataStore } from "@jellyfin-harmony/core";

import { AddressInfo } from "@jellyfin-harmony/core";

import { ActiveInfo } from "@jellyfin-harmony/core";

import { MD5 } from "@jellyfin-harmony/core";

import { uri } from "@kit.ArkTS";

import BuildProfile from "BuildProfile";

/\*\*

\* @Author peerless2012

\* @Email peerless2012@126.com

\* @DateTime 2024/11/9 11:50

\* @Version V1.0

\* @Description Repository

\*/

export class Repository {

public static readonly REPOSITORY = "Repository"

/\*\*

\* Client info

\*/

private readonly clientInfo: ClientInfo = {

name: "FinAdmin",

version: BuildProfile.VERSION\_NAME

}

/\*\*

\* Device info

\*/

private readonly deviceInfo: DeviceInfo = {

id: deviceInfo.ODID + '-admin',

name: deviceInfo.marketName

}

/\*\*

\* Jellyfin api

\*/

private jellyfin: Jellyfin = new Jellyfin({

clientInfo: this.clientInfo,

deviceInfo: this.deviceInfo,

})

private jellyfinTmp: Jellyfin = new Jellyfin({

clientInfo: this.clientInfo,

deviceInfo: this.deviceInfo,

})

private context: Context

private dataStore: DataStore

private activeInfo?: ActiveInfo

constructor(context: Context) {

this.context = context

this.dataStore = new DataStore(context)

}

/\*\*

\* Init active server and user

\* @returns

\*/

public async init(): Promise<void> {

let activeInfo = await this.dataStore.queryActive()

if (activeInfo) {

this.initActiveInfo(activeInfo)

}

}

private initActiveInfo(activeInfo: ActiveInfo): void {

this.jellyfin.apiClient.updateServerInfo({address: activeInfo.addressInfo.address})

this.jellyfin.apiClient.updateUserInfo({id: activeInfo.userInfo.userId, token: activeInfo.userInfo.accessToken})

this.activeInfo = activeInfo

}

public getActiveInfo(): ActiveInfo | undefined {

return this.activeInfo

}

public setActiveInfo(activeInfo: ActiveInfo) {

this.initActiveInfo(activeInfo)

this.dataStore.insertActive(activeInfo)

.then(() => {

// save success

})

.catch((error: Error) => {

// save fail

})

}

/\*\*

\* get all server

\* @returns

\*/

public async getServers(): Promise<Array<ServerInfo>> {

await this.dataStore.queryActive()

return this.dataStore.queryAllServer()

}

/\*\*

\* get all address

\* @returns

\*/

public async getAddresses(): Promise<Array<AddressInfo>> {

if (this.activeInfo == undefined) {

return []

}

return this.dataStore.queryAllAddress(this.activeInfo.serverInfo.serverId)

}

/\*\*

\* get all user

\* @returns

\*/

public async getUsers(): Promise<Array<UserInfo>> {

if (this.activeInfo == undefined) {

return []

}

return this.dataStore.queryAllUser(this.activeInfo.serverInfo.serverId)

}

/\*\*

\* connect server

\* @returns

\*/

public async connectServer(url: string): Promise<ServerInfo> {

this.jellyfinTmp.apiClient.updateServerInfo({address: url})

let systemInfo = await this.jellyfinTmp.getSystemApi().getPublicSystemInfo()

// save server info

let serverInfo: ServerInfo = {

serverId: systemInfo.Id!,

serverName: systemInfo.ServerName!

}

await this.dataStore.insertServer(serverInfo)

// save address info

let addressInfo: AddressInfo = {

addressId: MD5.digestSync(url),

address: url,

serverId: systemInfo.Id!

}

await this.dataStore.insertAddress(addressInfo)

return serverInfo

}

public async addServer(url: string): Promise<ServerInfo> {

throw new Error("Not impl")

}

/\*\*

\* auth account

\* @param name

\* @param psw

\* @returns

\*/

public async authAccount(url: string, name: string, psw?: string): Promise<ActiveInfo> {

this.jellyfinTmp.apiClient.updateServerInfo({address: url})

let authResult = await this.jellyfinTmp.getUserApi().authenticateUserByName({

Username: name,

Pw: psw

})

let serverInfo = await this.dataStore.queryServer(authResult.ServerId!)

let addressInfo: AddressInfo = {

addressId: MD5.digestSync(url),

address: url,

serverId: authResult.ServerId!

}

let userInfo: UserInfo = {

userId: authResult.User!.Id!,

userName: authResult.User!.Name!,

serverId: authResult.ServerId!,

accessToken: authResult.AccessToken!

}

return { serverInfo: serverInfo!, addressInfo: addressInfo, userInfo: userInfo}

}

public async addAccount(name: string, psw?: string): Promise<UserInfo> {

throw new Error("Not impl")

}

private filterItem(items?: Array<BaseItemDto> | null): Array<BaseItemDto> | undefined {

let newItems: Array<BaseItemDto> | undefined

items?.forEach((item) => {

let type = item.CollectionType

if (type === CollectionType.Movies

|| type === CollectionType.Tvshows

|| type === CollectionType.Music

|| type === CollectionType.Books) {

if (newItems == undefined) {

newItems = new Array()

}

newItems.push(item)

}

}

)

return newItems

}

public buildImage(dto: BaseItemDto, type: ImageType): string {

let imageUri = new uri.URI(this.jellyfin.apiClient.getServerInfo()!.address)

imageUri = imageUri.addEncodedSegment(`items/${dto.Id!}/Images/${type}`)

return imageUri.toString()

}

public buildPersonImage(person: BaseItemPerson, type: ImageType): string {

let imageUri = new uri.URI(this.jellyfin.apiClient.getServerInfo()!.address)

imageUri = imageUri.addEncodedSegment(`items/${person.Id!}/Images/${type}`)

return imageUri.toString()

}

/\*\*

\* 查询媒体库

\* @returns

\*/

public async getLibraryList(): Promise<Array<BaseItemDto>> {

let result = await this.jellyfin.getUserViewsApi().getUserViews({

userId: this.activeInfo!.userInfo.userId,

presetViews: [CollectionType.Movies, CollectionType.Tvshows, CollectionType.Music, CollectionType.Books]

})

let groups = this.filterItem(result.Items)

if (!groups) {

return []

}

return groups

}

public async getLatestMedia(id?: string): Promise<Array<BaseItemDto>> {

let result = await this.jellyfin.getUserLibraryApi().getLatestMedia({

userId: this.activeInfo!.userInfo.userId,

parentId: id,

limit: 8

})

if (result) {

return result

}

return []

}

public async getServiceInfo(): Promise<SystemInfo> {

return await this.jellyfin.getSystemApi().getSystemInfo()

}

public async getSessionInfos(): Promise<Array<SessionInfo>> {

return await this.jellyfin.getSessionApi().getSessions({activeWithinSeconds: 60 \* 60})

}

public async getLogInfos(): Promise<Array<ActivityLogEntry>> {

let result = await this.jellyfin.getActivityLogApi().getLogEntries({limit: 10})

if (result.Items) {

return result.Items

}

return []

}

/\*\*

\* 查询

\* @returns

\*/

public async getAllUsers(): Promise<Array<UserDto>> {

let result = await this.jellyfin.getUserApi().getUsers()

return result

}

/\*\*

\* 查询

\* @returns

\*/

public async getAllDevices(): Promise<Array<DeviceInfos>> {

let result = await this.jellyfin.getDevicesApi().getDevices()

if (result.Items) {

return result.Items

}

return []

}

/\*\*

\* 查询

\* @returns

\*/

public async getAllLogs(): Promise<Array<ActivityLogEntry>> {

let result = await this.jellyfin.getActivityLogApi().getLogEntries({hasUserId: true})

if (result.Items) {

return result.Items

}

return []

}

/\*\*

\* 查询所有媒体库

\* @returns

\*/

public async loadMedia(): Promise<Array<BaseItemDto>> {

let result = await this.jellyfin.getItemsApi().getItems({

userId: this.activeInfo!.userInfo.userId

})

let items = this.filterItem(result.Items)

if (items) {

return items

}

return []

}

public async loadFavourite(includeItemTypes?: Array<BaseItemKind>): Promise<Array<BaseItemDto>> {

let result = await this.jellyfin.getItemsApi().getItems({

userId: this.activeInfo!.userInfo.userId,

filters: [ItemFilter.IsFavorite],

includeItemTypes: includeItemTypes,

recursive: true

})

if (result.Items) {

return result.Items

}

return []

}

public async loadMediaList(id: string, types: Array<BaseItemKind>

, recursive: boolean, sortBy?: Array<ItemSortBy>, sortOrder?: Array<SortOrder>): Promise<Array<BaseItemDto>> {

let result = await this.jellyfin.getItemsApi().getItems({

parentId: id,

includeItemTypes: types,

recursive: recursive,

sortBy: sortBy,

sortOrder: sortOrder

})

return result.Items!

}

public loadMovie(id: string): Promise<BaseItemDto> {

return this.jellyfin.getUserLibraryApi().getItem({

userId: this.activeInfo!.userInfo.userId,

itemId: id

})

}

public loadShow(showId: string): Promise<BaseItemDto> {

return this.jellyfin.getUserLibraryApi().getItem({

userId: this.activeInfo!.userInfo.userId,

itemId: showId

})

}

public getResumeItems(): Promise<Array<BaseItemDto>> {

return this.jellyfin.getItemsApi().getResumeItems({

userId: this.activeInfo!.userInfo.userId,

limit: 8,

includeItemTypes: [BaseItemKind.Movie, BaseItemKind.Episode]

}).then((result) => {

if (result.Items) {

return result.Items

}

return []

})

}

public getNextUp(seriesId?: string): Promise<Array<BaseItemDto>> {

return this.jellyfin.getTvShowsApi().getNextUp({

userId: this.activeInfo!.userInfo.userId,

seriesId: seriesId,

enableResumable: false,

limit: 8

})

.then((result) => {

if (result.Items) {

return result.Items

}

return []

})

}

public getSeasons(seriesId: string): Promise<Array<BaseItemDto>> {

return this.jellyfin.getTvShowsApi().getSeasons({

userId: this.activeInfo!.userInfo.userId,

seriesId: seriesId

}).then((result) => {

if (result.Items) {

return result.Items

}

return []

})

}

public getEpisodes(seriesId: string, seasonId: string): Promise<Array<BaseItemDto>> {

return this.jellyfin.getTvShowsApi().getEpisodes({

userId: this.activeInfo!.userInfo.userId,

seriesId: seriesId,

seasonId: seasonId,

fields: [ItemFields.Overview]

}).then((result) => {

if (result.Items) {

return result.Items

}

return []

})

}

public getEpisode(episodeId: string): Promise<BaseItemDto> {

return this.jellyfin.getUserLibraryApi().getItem({

userId: this.activeInfo!.userInfo.userId,

itemId: episodeId

})

}

public loadMediaSource(id: string): Promise<PlaybackInfoResponse> {

return this.jellyfin.getMediaInfoApi().getPlaybackInfo({

userId: this.activeInfo!.userInfo.userId,

itemId: id

})

}

public loadStreamUrl(id: string, sourceId: string): Promise<string> {

return this.jellyfin.getAudioApi().getAudioStream({

itemId: id,

static: true,

mediaSourceId: sourceId

})

}

public searchMedia(keyword: string): Promise<Array<BaseItemDto>> {

return Promise.resolve([])

}

public getUserItem(id: string): Promise<UserItemDataDto> {

return this.jellyfin.getItemsApi().getItemUserData({

itemId: id,

userId: this.activeInfo!.userInfo.userId

})

}

public markFavorite(id: string): Promise<UserItemDataDto> {

return this.jellyfin.getUserLibraryApi().markFavoriteItem({

itemId: id,

userId: this.activeInfo!.userInfo.userId

})

}

public unmarkFavorite(id: string): Promise<UserItemDataDto> {

return this.jellyfin.getUserLibraryApi().unmarkFavoriteItem({

itemId: id,

userId: this.activeInfo!.userInfo.userId

})

}

public markPlayed(id: string): Promise<UserItemDataDto> {

return this.jellyfin.getPlayStateApi().markPlayedItem({

itemId: id,

userId: this.activeInfo!.userInfo.userId

})

}

public unmarkPlayed(id: string): Promise<UserItemDataDto> {

return this.jellyfin.getPlayStateApi().markUnplayedItem({

itemId: id,

userId: this.activeInfo!.userInfo.userId

})

}

}