```
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-ad
aptation-V5#section1421172621111

this.context.getApplicationContext().setColorMode(ConfigurationConstant.ColorMode.COLOR_M
ODE_NOT_SET);
        AppStorage.setOrCreate(AppPrefer.PREFER,                                    new
AppPrefer(this.context.getApplicationContext()))
    }

    onConfigurationUpdate(newConfig: Configuration): void {
    }

}
import { deviceInfo } from "@kit.BasicServicesKit";
import { BaseItemDto, BaseItemKind,
    BaseItemPerson,
    ClientInfo,
    CollectionType,
    DeviceInfo, ImageType,
    ItemFields,
```

```
    ItemFilter,
    ItemSortBy,
    Jellyfin,
    PlaybackInfoResponse,
    SortOrder,
    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 BuildProfile from "BuildProfile";
import { uri } from "@kit.ArkTS";

/**
 * @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: "FinVideo",
        version: BuildProfile.VERSION_NAME
    }

    /**
     * Device info
     */
    private readonly deviceInfo: DeviceInfo = {
        id: deviceInfo.ODID + '-video',
        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.Tvshows
                || type === CollectionType.Movies) {
                if (newItems == undefined) {
                    newItems = new Array()
                }
                newItems.push(item)
            }
        }
        )
        return newItems
    }

    public buildImage(dto: BaseItemDto, type: ImageType): string | undefined {
        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 | undefined {
        let imageUri = new uri.URI(this.jellyfin.apiClient.getServerInfo()!.address)
        imageUri = imageUri.addEncodedSegment(`items/${person.Id!}/Images/${type}`)
        return imageUri.toString()
```

```
    }

    public async getMediaList(): Promise<Array<BaseItemDto>> {
        let result = await this.jellyfin.getUserViewsApi().getUserViews({
            userId: this.activeInfo!.userInfo.userId
        })
        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 []
    }

    /**
     * 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.getVideosApi().getVideoStreamUrl({
        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
        })
    }

}
import { BaseItemDto } from "@jellyfin-harmony/sdk";

/**
  * @Author peerless2012
  * @Email peerless2012@126.com
  * @DateTime 2024/11/11 22:28
  * @Version V1.0
  * @Description
  */
export interface GroupInfo {

    group?: BaseItemDto

    items?: Array<BaseItemDto>

}
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)
    }

}
/**
 * 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 Logger from '@jellyfin-harmony/core/src/main/ets/system/Logger';
import { ObservedArray } from './ObservedArray';

const TAG = '[BasicDataSource]';

class BasicDataSource <T> implements IDataSource {
    private listeners: DataChangeListener[] = [];

    public totalCount(): number {
        return 0;
    }

    public getData(index: number): T | undefined {
        return undefined;
    }

    registerDataChangeListener(listener: DataChangeListener): void {
        if (this.listeners.indexOf(listener) < 0) {
```

```
            Logger.info(TAG, 'add listener');
            this.listeners.push(listener);
        }
    }

    unregisterDataChangeListener(listener: DataChangeListener): void {
        const pos = this.listeners.indexOf(listener);
        if (pos >= 0) {
            Logger.info(TAG, 'remove listener');
            this.listeners.splice(pos, 1);
        }
    }

    notifyDataReload(): void {
        this.listeners.forEach(listener => {
            listener.onDataReloaded();
        })
    }

    notifyDataAdd(index: number): void {
        this.listeners.forEach(listener => {
            listener.onDataAdd(index);
        })
    }

    notifyDataChange(index: number): void {
        this.listeners.forEach(listener => {
            listener.onDataChange(index);
        })
    }

    notifyDataDelete(index: number): void {
        this.listeners.forEach(listener => {
            listener.onDataDelete(index);
        })
    }

    notifyDataMove(from: number, to: number): void {
        this.listeners.forEach(listener => {
            listener.onDataMove(from, to);
        })
    }
}
```

```
@Observed
export default class LazyDataSource<T> extends BasicDataSource<T> {
  dataArray: T[] = [];

  public totalCount(): number {
    return this.dataArray.length;
  }

  public getData(index: number): T {
    return this.dataArray[index];
  }

  public addData(index: number, data: T): void {
    this.dataArray.splice(index, 0, data);
    this.notifyDataAdd(index);
  }

  public pushData(data: T): void {
    this.dataArray.push(data);
    this.notifyDataAdd(this.dataArray.length - 1);
  }

  public pushArrayData(newData: ObservedArray<T>): void {
    this.clear();
    this.dataArray.push(...newData);
    this.notifyDataReload();
  }

  public appendArrayData(addData: ObservedArray<T>): void {
    this.dataArray.push(...addData);
    this.notifyDataReload();
  }

  public deleteData(index: number): void {
    this.dataArray.splice(index, 1);
    this.notifyDataDelete(index);
  }

  public getDataList(): ObservedArray<T> {
    return this.dataArray;
  }

  public clear(): void {
    this.dataArray.splice(0, this.dataArray?.length);
```

```
    }

    public isEmpty(): boolean {
        return this.dataArray.length === 0;
    }
}

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 { BaseItemKind } from "@jellyfin-harmony/sdk";
import { DetailArgs } from "../detail/DetailArgs";
import { router } from "@kit.ArkUI";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/19 23:18
 * @Version V1.0
 * @Description
 */
export class MediaRouter {

    public static openDetail(type: BaseItemKind, args: DetailArgs) {
        if (type === BaseItemKind.Movie) {
```

```
            router.push({url: "pages/detail/movie/MoviePage", params: args})
        } else if (type === BaseItemKind.Series) {
            router.push({url: "pages/detail/show/ShowPage", params: args})
        } else {

        }
    }

}
/**
 * Wraps an array as an observed object
 */
@Observed
export class ObservedArray<T> extends Array<T> {
    constructor(args?: T[]) {
        if (args instanceof Array) {
            super(...args);
        } else {
            super();
        }
    }
}

import { window } from '@kit.ArkUI'
import { DetailArgs } from '../DetailArgs'
import { MovieViewModel } from './MovieViewModel'
import { CommonConstants } from '../../common/CommonConstants'
import { DetailFooter, DetailHeader } from '../DetailComponent'
import { ScrollRefreshView } from '@jellyfin-harmony/core'
import { BaseItemDto } from '@jellyfin-harmony/sdk'
import { ToolBar } from '@jellyfin-harmony/core'

let movieStorage: LocalStorage = new LocalStorage();
@Entry(movieStorage)
@Component
struct MoviePage {

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

    private scroller = new Scroller()

    private args = this.getUIContext().getRouter().getParams() as DetailArgs

    private viewModel = new MovieViewModel(getContext(), this.args)
```

```
  aboutToAppear(): void {
    movieStorage.setOrCreate("viewModel", this.viewModel)
  }

  @Builder
  movieScrollView(item?: BaseItemDto) {
    Scroll(this.scroller) {
      Column({space: CommonConstants.SPACE_12}) {
        DetailHeader({itemDto: item})
        DetailFooter({itemDto: item})
      }
      .padding({
        bottom: px2vp(this.avoidArea?.bottomRect.height)
      })
      .alignItems(HorizontalAlign.Start)
    }
    .align(Alignment.Top)
    .width(CommonConstants.FULL_PERCENT)
    .height(CommonConstants.FULL_PERCENT)
  }

  build() {
    Stack() {
      ScrollRefreshView({
        source: this.viewModel,
        itemLayout: (info: ESObject) => {
          this.movieScrollView(info)
        }
      })

      ToolBar({
        color: $r('sys.color.font_on_primary')
      })
    }
    .alignContent(Alignment.Top)
    .width(CommonConstants.FULL_PERCENT)
    .height(CommonConstants.FULL_PERCENT)
  }

}
import { LoadType } from "@jellyfin-harmony/core";
import { SimpleRepository } from "@jellyfin-harmony/core";
import { DetailViewModel } from "../DetailViewModel";
```

```
/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/19 22:26
 * @Version V1.0
 * @Description Movie
 */
export class MovieViewModel extends DetailViewModel implements SimpleRepository {

  async loadData(type: LoadType): Promise<Object> {
    await this.getUserItem()
    return this.repository.loadMovie(this.args.id)
  }

}
import { BaseItemDto, ImageType } from "@jellyfin-harmony/sdk"
import { Repository } from "../../../../data/Repository"
import { CommonConstants } from "../../../common/CommonConstants"
import { DetailArgs } from "../../DetailArgs"
import { DetailOperate } from "../../DetailComponent"
import { ShowEpisodeViewModel } from "./ShowEpisodeViewModel"

@Component
export struct ShowEpisodePage {

  @Prop args: DetailArgs

  @Prop storage: LocalStorage

  @State episodeDto?: BaseItemDto = undefined

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

  private viewModel = new ShowEpisodeViewModel(getContext(), this.args)

  aboutToAppear(): void {
    this.storage.setOrCreate('viewModel', this.viewModel)
    this.viewModel.getEpisode()
      .then((dto) => {
        this.episodeDto = dto
      })
      .catch((error: Error) => {
```

```
        })
    }

    build() {
        Column({space: CommonConstants.SPACE_12}) {
            Row({space: CommonConstants.SPACE_8}) {
                Image(this.episodeDto ? this.repository?.buildImage(this.episodeDto!,
ImageType.Primary) : undefined)
                    .id('episode_cover')
                    .alt($r('app.media.alt'))
                    .autoResize(true)
                    .objectFit(ImageFit.Cover)
                    .borderRadius($r('app.float.lg_border_radius'))
                    .layoutWeight(2)
                    .aspectRatio(1.5)

                Stack() {
                    Column({space: CommonConstants.SPACE_8}) {
                        Text(this.episodeDto?.SeriesName)
                            .id('episode_name')
                            .fontSize($r('sys.float.Body_L'))

                        Text((this.episodeDto?.ParentIndexNumber && this.episodeDto?.IndexNumber)
                            ? `S${this.episodeDto?.ParentIndexNumber}:E${this.episodeDto?.IndexNumber}
- ${this.episodeDto?.Name}`
                            : this.episodeDto?.Name)
                            .fontSize($r('sys.float.Body_M'))

                        Blank()

                        DetailOperate({
                            itemDto: this.episodeDto
                        }, this.storage)
                    }
                    .layoutWeight(1)
                    .alignItems(HorizontalAlign.Start)
                }
                .layoutWeight(3)
            }
            Text(this.episodeDto?.Overview)
                .fontSize($r('sys.float.Body_M'))
            Blank(CommonConstants.SPACE_16)
        }
        .padding(CommonConstants.SPACE_16)
```

```
    }

    aboutToDisappear(): void {

    }

}
import { BaseItemDto } from "@jellyfin-harmony/sdk";
import { DetailViewModel } from "../../DetailViewModel";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/23 20:23
 * @Version V1.0
 * @Description
 */
export class ShowEpisodeViewModel extends DetailViewModel {

    public async getEpisode(): Promise<BaseItemDto> {
        await this.getUserItem()
        return this.repository.getEpisode(this.args.id)
    }

}
import { DetailArgs } from "../../DetailArgs";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/23 17:21
 * @Version V1.0
 * @Description
 */
export interface ShowSeasonArgs extends DetailArgs {

    seasonId: string

}
import { window } from '@kit.ArkUI'
import { BaseItemDto, ImageType } from '@jellyfin-harmony/sdk'
import { Repository } from '../../../../data/Repository'
import { ShowEpisodeSheet } from '../../../../sheet/ShowEpisodeSheet'
import { ListRefreshView } from '@jellyfin-harmony/core'
```

```
import { CommonConstants } from '../../../common/CommonConstants'
import { DetailArgs } from '../../DetailArgs'
import { ShowSeasonArgs } from './ShowSeasonArgs'
import { ShowSeasonViewModel } from './ShowSeasonViewModel'
import { ToolBar } from '@jellyfin-harmony/core'

@Entry
@Component
struct ShowSeasonPage {

  private args = this.getUIContext().getRouter().getParams() as ShowSeasonArgs

  private viewModel = new ShowSeasonViewModel(getContext(), this.args)

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

  @Builder
  seasonItemBuilder(item: BaseItemDto, index: number) {
    SeasonItem({item: item})
  }

  build() {
    Column() {
      ToolBar({
        title: this.args.name
      })

      ListRefreshView({
        source: this.viewModel,
        itemLayout: this.seasonItemBuilder,
        listAttribute: (attr) => {
          attr.divider = {
            color: Color.Transparent,
            strokeWidth: CommonConstants.SPACE_12
          }
          attr.width = CommonConstants.FULL_PERCENT
          attr.height = CommonConstants.FULL_PERCENT
          attr.contentEndOffset = px2vp(this.avoidArea?.bottomRect.height)
        }
      })
        .width(CommonConstants.FULL_PERCENT)
        .layoutWeight(1)
    }
  }
```

```
}

@Component
@Reusable
export struct SeasonItem {

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

  @Require item?: BaseItemDto

  @State isSheetShow: boolean = false

  build() {
    RelativeContainer(){
      Image(this.repository?.buildImage(this.item!, ImageType.Primary))
        .id("episode_cover")
        .alt($r('app.media.alt'))
        .objectFit(ImageFit.Cover)
        .autoResize(true)
        .width(100)
        .height(100)
        .borderRadius($r('app.float.lg_border_radius'))
        .alignRules({
          center: {
            anchor: "__container__",
            align: VerticalAlign.Center
          },
          left: {
            anchor: "__container__",
            align: HorizontalAlign.Start
          }
        })
      Text(this.item?.Name)
        .id("episode_name")
        .fontSize($r('sys.float.Body_L'))
        .maxLines(1)
        .margin({
          left: CommonConstants.SPACE_12
        })
        .alignRules({
          left: {
            anchor: "episode_cover",
            align: HorizontalAlign.End
```

```
            },
            right: {
                anchor: "__container__",
                align: HorizontalAlign.End
            },
            top: {
                anchor: "episode_cover",
                align: VerticalAlign.Top
            }
        })
    Text(this.item?.Overview)
        .id("episode_overview")
        .fontSize($r('sys.float.Body_S'))
        .maxLines(5)
        .textAlign(TextAlign.Start)
        .align(Alignment.TopStart)
        .textOverflow({
            overflow: TextOverflow.Ellipsis
        })
        .margin({
            left: CommonConstants.SPACE_12,
            top: CommonConstants.SPACE_4
        })
        .alignRules({
            left: {
                anchor: "episode_cover",
                align: HorizontalAlign.End
            },
            right: {
                anchor: "__container__",
                align: HorizontalAlign.End
            },
            top: {
                anchor: "episode_name",
                align: VerticalAlign.Bottom
            },
            bottom: {
                anchor: "__container__",
                align: VerticalAlign.Bottom
            }
        })
}
.padding({
    left: CommonConstants.SPACE_16,
```

```
            right: CommonConstants.SPACE_16,
        })
        .width(CommonConstants.FULL_PERCENT)
        .height(100)
        .onClick(() => {
            let args: DetailArgs = {
                id: this.item!.Id!,
                name: this.item!.Name!
            }
            let sheet = new ShowEpisodeSheet(this.getUIContext(), args)
            sheet.show()
        })
    }
}
import { LoadType } from "@jellyfin-harmony/core";
import { ShowSeasonArgs } from "./ShowSeasonArgs";
import { ListViewModel } from "../../../common/ListViewModel";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/23 17:13
 * @Version V1.0
 * @Description
 */
export class ShowSeasonViewModel extends ListViewModel {

    private readonly args: ShowSeasonArgs

    constructor(context: Context, args: ShowSeasonArgs) {
        super(context)
        this.args = args
    }

    loadData(type: LoadType): Promise<void> {
        return this.repository.getEpisodes(this.args.id, this.args.seasonId)
            .then((items) => {
                this.dataSource.initData(items)
            })
    }

}
import { BaseItemDto, ImageType } from "@jellyfin-harmony/sdk"
import { CommonConstants } from "../../common/CommonConstants"
```

```
import { Repository } from "../../../data/Repository"
import { ShowSeasonArgs } from "./season/ShowSeasonArgs"
import router from "@ohos.router"
import { DetailArgs } from "../DetailArgs"
import { ShowEpisodeSheet } from "../../../sheet/ShowEpisodeSheet"

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/23 12:12
 * @Version V1.0
 * @Description
 */
@Component
export struct ShowNextUp {

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

  @Require nextDto?: BaseItemDto

  build() {
    Column({space: CommonConstants.SPACE_12}) {
      Text("接下来")
        .fontSize($r('sys.float.Title_S'))
      Image(this.repository?.buildImage(this.nextDto!, ImageType.Primary))
        .alt($r('app.media.alt'))
        .autoResize(true)
        .objectFit(ImageFit.Cover)
        .width(CommonConstants.FULL_PERCENT)
        .aspectRatio(1.5)
        .borderRadius($r('app.float.lg_border_radius'))
        .onClick(() => {
          let args: DetailArgs = {
            id: this.nextDto!.Id!,
            name: this.nextDto!.Name!
          }
          let sheet = new ShowEpisodeSheet(this.getUIContext(), args)
          sheet.show()
        })
      Text(this.nextDto?.Name)
    }
    .margin({
      left: CommonConstants.SPACE_16,
      right: CommonConstants.SPACE_16,
```

```
        })
        .alignItems(HorizontalAlign.Start)
    }

}

@Component
export struct ShowSeasons {

    @Prop seasonDto?: Array<BaseItemDto> = undefined

    @Prop args: DetailArgs

    build() {
        Column({space: CommonConstants.SPACE_12}) {
            Text("季")
                .fontSize($r('sys.float.Title_S'))
                .margin({
                    left: CommonConstants.SPACE_16,
                    right: CommonConstants.SPACE_16
                })
            List({space: CommonConstants.SPACE_12}) {
                ForEach(this.seasonDto, (dto: BaseItemDto) => {
                    ListItem() {
                        ShowSeasonItem({
                            args: this.args,
                            seasonDto: dto
                        })
                    }
                })
            }
            .contentStartOffset(CommonConstants.SPACE_16)
            .contentEndOffset(CommonConstants.SPACE_16)
            .scrollBar(BarState.Off)
            .listDirection(Axis.Horizontal)
            .width(CommonConstants.FULL_PERCENT)
            .height(CommonConstants.ITEM_HEIGHT)
        }
        .alignItems(HorizontalAlign.Start)
        .width(CommonConstants.FULL_PERCENT)
    }
}

@Component
```

```
@Reusable
struct ShowSeasonItem {

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

    @Require seasonDto?: BaseItemDto

    @Require args?: DetailArgs

    aboutToAppear(): void {

    }

    build() {
        Column() {
            Image(this.repository?.buildImage(this.seasonDto!, ImageType.Primary))
                .alt($r('app.media.alt'))
                .objectFit(ImageFit.Cover)
                .autoResize(true)
                .borderRadius($r('app.float.lg_border_radius'))
                .width(CommonConstants.ITEM_WIDTH)
                .aspectRatio(CommonConstants.ITEM_RATIO)
            Text(this.seasonDto?.Name)
        }
        .width(CommonConstants.ITEM_WIDTH)
        .height(CommonConstants.ITEM_HEIGHT)
        .onClick(() => {
            let args: ShowSeasonArgs = {
                id: this.args!.id,
                name: this.args!.name,
                seasonId: this.seasonDto!.Id!
            }
            router.push({
                url: "pages/detail/show/season/ShowSeasonPage",
                params: args
            })
        })
    }

    aboutToDisappear(): void {
    }
}
import { BaseItemDto } from "@jellyfin-harmony/sdk";
```

```
export interface ShowInfo {

  show: BaseItemDto

  nextUp?: BaseItemDto

  seasons?: Array<BaseItemDto>
}
import { CommonConstants } from '../../common/CommonConstants'
import { DetailArgs } from '../DetailArgs'
import { DetailFooter, DetailHeader } from '../DetailComponent'
import { ShowViewModel } from './ShowViewModel'
import { window } from '@kit.ArkUI'
import { ShowInfo } from './ShowInfo'
import { ShowNextUp, ShowSeasons } from './ShowComponent'
import { ScrollRefreshView } from '@jellyfin-harmony/core'
import { ToolBar } from '@jellyfin-harmony/core'

let showStorage: LocalStorage = new LocalStorage();
@Entry(showStorage)
@Component
struct ShowPage {

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

  private args = this.getUIContext().getRouter().getParams() as DetailArgs

  private viewModel = new ShowViewModel(getContext(), this.args)

  aboutToAppear(): void {
    showStorage.setOrCreate("viewModel", this.viewModel)
  }

  @Builder
  showScrollView(info?: ShowInfo) {
    Scroll(new Scroller()) {
      Column({space: CommonConstants.SPACE_12}) {
        DetailHeader({itemDto: info?.show})
        // 季及接下来
        if (info?.nextUp) {
          ShowNextUp({
            nextDto: info.nextUp
          })
        }
```

```
            if (info?.seasons) {
              ShowSeasons({
                args: this.args,
                seasonDto: info.seasons
              })
            }
            DetailFooter({itemDto: info?.show})
          }
          .padding({
            bottom: px2vp(this.avoidArea?.bottomRect.height)
          })
        }
        .align(Alignment.Top)
        .width(CommonConstants.FULL_PERCENT)
        .height(CommonConstants.FULL_PERCENT)
    }

    build() {
        Stack() {
            ScrollRefreshView({
                source: this.viewModel,
                itemLayout: (info: ESObject) => {
                    this.showScrollView(info)
                }
            })
            ToolBar({
                color: $r('sys.color.font_on_primary')
            })
        }
        .alignContent(Alignment.Top)
        .width(CommonConstants.FULL_PERCENT)
        .height(CommonConstants.FULL_PERCENT)
    }

    aboutToDisappear(): void {
    }

}
import { LoadType } from "@jellyfin-harmony/core";
import { SimpleRepository } from "@jellyfin-harmony/core";
import { DetailViewModel } from "../DetailViewModel";
import { ShowInfo } from "./ShowInfo";

/**
```

```
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/21 22:11
 * @Version V1.0
 * @Description
 */
export class ShowViewModel extends DetailViewModel implements SimpleRepository {

  async loadData(type: LoadType): Promise<Object> {
    await this.getUserItem()
    let show = await this.repository.loadShow(this.args.id)
    let info: ShowInfo = {
      show: show
    }
    let nextUps = await this.repository.getNextUp(this.args.id)
    let seasons = await this.repository.getSeasons(this.args.id)
    if (nextUps && nextUps.length > 0) {
      info.nextUp = nextUps[0]
    }
    info.seasons = seasons
    return info
  }

}
/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/19 23:20
 * @Version V1.0
 * @Description
 */
export interface DetailArgs {

  id: string

  name: string

}
import { router } from "@kit.ArkUI"
import { BaseItemDto, BaseItemKind, BaseItemPerson, ImageType, PersonKind } from
"@jellyfin-harmony/sdk"
import { CommonConstants } from "../common/CommonConstants"
import { DetailViewModel } from "./DetailViewModel"
import { Repository } from "../../data/Repository"
```

```
import LazyDataSource from "../common/LazyDataSource"
import { PlayerArgs } from "../player/PlayerArgs"

@Component
export struct DetailOperate {

  @LocalStorageProp("viewModel") viewModel?: DetailViewModel = undefined

  @Require @Prop itemDto?: BaseItemDto = undefined

  @State collecting: boolean = false

  @State collected: boolean = false

  @State finishing: boolean = false

  @State finished: boolean = false

  aboutToAppear(): void {
    if (this.viewModel) {
      // TODO fix me
      // these value not set
      this.finished = this.viewModel.isFinished()
      this.collected = this.viewModel.isFavourite()
    }
  }

  build() {
    Row({space: CommonConstants.SPACE_8}) {
      Button({type: ButtonType.Capsule}) {
        SymbolGlyph($r('sys.symbol.play_fill'))
          .fontColor([$r('sys.color.font_on_primary')])
          .fontSize($r('sys.float.Title_M'))
      }
      .id('episode_play')
      .width(80)
      .height(32)
      .onClick(() => {
        this.playOrNot()
      })

      Blank()

      Button({type: ButtonType.Circle}) {
```

```
            if (this.finishing) {
              SymbolGlyph($r('sys.symbol.loading'))
                .fontSize($r('sys.float.Title_M'))
            } else {
              SymbolGlyph($r('sys.symbol.checkmark'))
                .fontColor([this.finished ? Color.Red : Color.Gray])
                .fontSize($r('sys.float.Title_M'))
            }
          }
          .id('episode_finish')
          .backgroundColor($r('sys.color.ohos_id_color_button_normal'))
          .enabled(!this.finishing)
          .padding(CommonConstants.SPACE_4)
          .onClick(() => {
            this.finishOrNot()
          })

          Button({type: ButtonType.Circle}) {
            if (this.collecting) {
              SymbolGlyph($r('sys.symbol.loading'))
                .fontSize($r('sys.float.Title_M'))
            } else {
              SymbolGlyph(this.collected ? $r('sys.symbol.heart_fill') : $r('sys.symbol.heart'))
                .fontColor([this.collected ? Color.Red : Color.Gray])
                .fontSize($r('sys.float.Title_M'))
            }
          }
          .id('episode_collect')
          .backgroundColor($r('sys.color.ohos_id_color_button_normal'))
          .enabled(!this.collecting)
          .padding(CommonConstants.SPACE_4)
          .onClick(() => {
            this.collectOrNot()
          })
        }
      }

      private playOrNot() {
        let args: PlayerArgs = {
          id: this.itemDto!.Id!,
          name: this.itemDto!.Name!,
          type: this.itemDto!.Type!
        }
        router.pushUrl({url: "pages/player/PlayerPage", params: args})
```

```
    }

    private finishOrNot() {
        this.finishing = true
        this.viewModel?.finishOrNot()
            .then((finished) => {
                this.finishing = false
                this.finished = finished
            })
            .catch((error: Error) => {
                this.finishing = false
            })
    }

    private collectOrNot() {
        this.collecting = true
        this.viewModel?.favouriteOrNot()
            .then((collected) => {
                this.collecting = false
                this.collected = collected
            })
            .catch((error: Error) => {
                this.collecting = false
            })
    }

    aboutToDisappear(): void {

    }

}

@Component
export struct DetailHeader {

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

    @Prop itemDto: BaseItemDto

    @State collecting: boolean = false

    @State collected: boolean = false

    @State finishing: boolean = false
```

```
@State finished: boolean = false

aboutToAppear(): void {
}

build() {
    Column({space: CommonConstants.SPACE_12}) {
        Image(this.repository?.buildImage(this.itemDto, ImageType.Backdrop))
            .alt($r('app.media.alt'))
            .objectFit(ImageFit.Cover)
            .autoResize(true)
            .width(CommonConstants.FULL_PERCENT)
            .aspectRatio(1.5)

        Column({space: CommonConstants.SPACE_12}) {

            Text(this.itemDto.Name)
                .fontSize($r('sys.float.Title_M'))

            Row({space: CommonConstants.SPACE_8}) {
                Text(this.itemDto.ProductionYear?.toString())

                Text(Math.round(((this.itemDto.RunTimeTicks!) / 600000000)) + "min")

                Text(this.itemDto.OfficialRating)

                Text(this.itemDto.CommunityRating?.toString())

            }

            Row() {
                Text(this.itemDto.MediaSources?.[0].MediaStreams?.[0].DisplayTitle)
            }

            if (this.itemDto.Type === BaseItemKind.Movie) {
                DetailOperate({
                    itemDto: this.itemDto
                })
            }

            Text(this.itemDto.Overview)

            Row({space: CommonConstants.SPACE_8}) {
```

```
            Text("风格")
            Text(this.itemDto.Genres?.join(","))
          }

          Row({space: CommonConstants.SPACE_8}) {
            Text("导演")
            Text(this.itemDto.People?.filter((people) => {
              return people.Type === PersonKind.Director
            })?.map((person) => {
              return person.Name
            })?.join(","))
          }

          Row({space: CommonConstants.SPACE_8}) {
            Text("作者")
            Text(this.itemDto.People?.filter((people) => {
              return people.Type === PersonKind.Writer
            })?.map((people) => {
              return people.Name
            }).join(","))
          }
        }
        .padding({
          left: CommonConstants.SPACE_16,
          right: CommonConstants.SPACE_16
        })
        .alignItems(HorizontalAlign.Start)
      }
      .width(CommonConstants.FULL_PERCENT)
    }

    aboutToDisappear(): void {

    }

}

@Component
export struct DetailFooter {

    @Require itemDto?: BaseItemDto

    private personDataSource: LazyDataSource<BaseItemPerson> = new LazyDataSource();
```

```
    aboutToAppear(): void {
        if (this.itemDto?.People) {
            this.personDataSource.appendArrayData(this.itemDto.People)
        }
    }

    build() {
        Column({space: CommonConstants.SPACE_12}) {
            Text("演职人员")
                .fontSize($r('sys.float.Title_S'))
                .margin({
                    left: CommonConstants.SPACE_16
                })

            List({space: CommonConstants.SPACE_12}) {
                LazyForEach(this.personDataSource, (person: BaseItemPerson, index) => {
                    ListItem() {
                        PersonItem({person: person})
                    }
                })
            }
            .contentStartOffset(CommonConstants.SPACE_16)
            .contentEndOffset(CommonConstants.SPACE_16)
            .listDirection(Axis.Horizontal)
            .width(CommonConstants.FULL_PERCENT)
            .scrollBar(BarState.Off)
            .height(220)
        }
        .alignItems(HorizontalAlign.Start)
    }

    aboutToDisappear(): void {

    }

}

@Reusable
@Component
struct PersonItem {

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

    @Require person?: BaseItemPerson
```

```
    build() {
      Column() {
        Image(this.repository?.buildPersonImage(this.person!, ImageType.Primary))
          .alt($r('app.media.alt'))
          .borderRadius($r('app.float.lg_border_radius'))
          .objectFit(ImageFit.Cover)
          .autoResize(true)
          .width(120)
          .height(180)
        Text(this.person?.Name)
          .maxLines(1)
          .ellipsisMode(EllipsisMode.END)
          .height(20)
        Text(this.person?.Role)
          .maxLines(1)
          .ellipsisMode(EllipsisMode.END)
          .height(20)
      }
      .alignItems(HorizontalAlign.Start)
      .width(120)
      .height(220)
    }

}

import { UserItemDataDto } from "@jellyfin-harmony/sdk";
import { AppViewModel } from "../common/AppViewModel";
import { DetailArgs } from "./DetailArgs";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/21 21:08
 * @Version V1.0
 * @Description
 */
export class DetailViewModel extends AppViewModel {

    protected readonly args: DetailArgs

    protected userItemDto?: UserItemDataDto = undefined

    constructor(context: Context, args: DetailArgs) {
```

```
        super(context)
        this.args = args
    }

    protected getUserItem(): Promise<UserItemDataDto> {
        return this.repository.getUserItem(this.args.id)
            .then((dto) => {
                this.userItemDto = dto
                return dto
            })
    }

    isFinished(): boolean {
        return this.userItemDto?.Played === true
    }

    async finishOrNot(): Promise<boolean> {
        if (this.userItemDto?.Played) {
            this.userItemDto = await this.repository.unmarkPlayed(this.args.id)
        } else {
            this.userItemDto = await this.repository.markPlayed(this.args.id)
        }
        return this.userItemDto!.Played!
    }

    isFavourite(): boolean {
        return this.userItemDto?.IsFavorite === true
    }

    async favouriteOrNot(): Promise<boolean> {
        if (this.userItemDto?.IsFavorite) {
            this.userItemDto = await this.repository.unmarkFavorite(this.args.id)
        } else {
            this.userItemDto = await this.repository.markFavorite(this.args.id)
        }
        return this.userItemDto!.IsFavorite!
    }

}
import { AddressInfo } from '@jellyfin-harmony/core'
import { ToolBar } from '@jellyfin-harmony/core'
import { ListRefreshView } from '@jellyfin-harmony/core'
import { CommonConstants } from '../../common/CommonConstants'
import { EditAddressViewModel } from './EditAddressViewModel'
```

```
@Entry
@Component
struct EditAddressPage {

  private viewModel = new EditAddressViewModel(getContext())

  @Builder
  private itemView(addressInfo: Object, index: number) {
    AddressItem({address: addressInfo as AddressInfo})
  }

  build() {
    Column() {
      ToolBar({
        title: $r('app.string.prefer_address')
      })
      ListRefreshView({
        source: this.viewModel,
        itemLayout: this.itemView,
        listAttribute: (attr) => {
          attr.divider = {
            color: Color.Transparent,
            strokeWidth: CommonConstants.SPACE_12
          }
          attr.width = CommonConstants.FULL_PERCENT
          attr.height = CommonConstants.FULL_PERCENT
        },
      })
        .width(CommonConstants.FULL_PERCENT)
        .layoutWeight(1)
    }
  }

}


@Component
@Reusable
struct AddressItem {

  @Prop address: AddressInfo

  build() {
```

```
        Row({space: CommonConstants.SPACE_12}) {
          SymbolGlyph($r('sys.symbol.worldclock'))
            .fontSize(24)
            .fontColor([$r('sys.color.ohos_id_color_text_primary')])
          Text(this.address.address)
            .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(() => {
        })

    }
}
import { LoadType } from "@jellyfin-harmony/core";
import { ListViewModel } from "../../common/ListViewModel";
/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/12/5 20:51
 * @Version V1.0
 * @Description
 */
export class EditAddressViewModel extends ListViewModel {

    loadData(type: LoadType): Promise<void> {
        return this.repository.getAddresses()
            .then((addresses) => {
                this.dataSource.initData(addresses)
            })
    }

}
import { ServerInfo } from '@jellyfin-harmony/core'
import { ToolBar } from '@jellyfin-harmony/core'
import { ListRefreshView } from '@jellyfin-harmony/core'
import { CommonConstants } from '../../common/CommonConstants'
```

```
import { EditServerViewModel } from './EditServerViewModel'

@Entry
@Component
struct EditServerPage {

  private viewModel = new EditServerViewModel(getContext())
  @Builder
  private itemView(serverInfo: Object, index: number) {
    ServerItem({server: serverInfo as ServerInfo})
  }

  build() {
    Column() {
      ToolBar({
        title: $r('app.string.prefer_server')
      })
      ListRefreshView({
        source: this.viewModel,
        itemLayout: this.itemView,
        listAttribute: (attr) => {
          attr.divider = {
            color: Color.Transparent,
            strokeWidth: CommonConstants.SPACE_12
          }
          attr.width = CommonConstants.FULL_PERCENT
          attr.height = CommonConstants.FULL_PERCENT
        },
      })
        .width(CommonConstants.FULL_PERCENT)
        .layoutWeight(1)
    }
  }
}


@Component
@Reusable
struct ServerItem {

  @Prop server: ServerInfo

  build() {
```

```
        Row({space: CommonConstants.SPACE_12}) {
          SymbolGlyph($r('sys.symbol.externaldrive'))
            .fontSize(24)
            .fontColor([$r('sys.color.ohos_id_color_text_primary')])
          Text(this.server.serverName)
            .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(() => {
        })

    }
}
import { LoadType } from "@jellyfin-harmony/core";
import { ListViewModel } from "../../common/ListViewModel";
/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/12/5 20:51
 * @Version V1.0
 * @Description
 */
export class EditServerViewModel extends ListViewModel {

    constructor(context: Context) {
        super(context)
    }

    loadData(type: LoadType): Promise<void> {
        return this.repository.getServers()
            .then((servers) => {
                this.dataSource.initData(servers)
            })
    }

}
import { UserInfo } from '@jellyfin-harmony/core'
```

```
import { ToolBar } from '@jellyfin-harmony/core'
import { ListRefreshView } from '@jellyfin-harmony/core'
import { CommonConstants } from '../../common/CommonConstants'
import { EditUserViewModel } from './EditUserViewModel'

@Entry
@Component
struct EditUserPage {

  private viewModel = new EditUserViewModel(getContext())

  aboutToAppear(): void {

  }

  @Builder
  private itemView(userInfo: Object, index: number) {
    UserItem({user: userInfo as UserInfo})
  }

  build() {
    Column() {
      ToolBar({
        title: $r('app.string.prefer_user')
      })
      ListRefreshView({
        source: this.viewModel,
        itemLayout: this.itemView,
        listAttribute: (attr) => {
          attr.divider = {
            color: Color.Transparent,
            strokeWidth: CommonConstants.SPACE_12
          }
          attr.width = CommonConstants.FULL_PERCENT
          attr.height = CommonConstants.FULL_PERCENT
        },
      })
        .width(CommonConstants.FULL_PERCENT)
        .layoutWeight(1)
    }
  }

}
```

```
@Component
@Reusable
struct UserItem {

    @Prop user: UserInfo

    build() {

        Row({space: CommonConstants.SPACE_12}) {
            SymbolGlyph($r('sys.symbol.person'))
                .fontSize(24)
                .fontColor([$r('sys.color.ohos_id_color_text_primary')])
            Text(this.user.userName)
                .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(() => {
        })

    }
}
import { LoadType } from "@jellyfin-harmony/core";
import { ListViewModel } from "../../common/ListViewModel";
/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/12/5 20:51
 * @Version V1.0
 * @Description
 */
export class EditUserViewModel extends ListViewModel {

    loadData(type: LoadType): Promise<void> {
        return this.repository.getUsers()
            .then((users) => {
                this.dataSource.initData(users)
            })
```

```
    }

}
import { BaseItemDto, BaseItemKind } from "@jellyfin-harmony/sdk"
import { CommonConstants } from "../../common/CommonConstants"
import { FavouriteViewModel } from "./FavouriteViewModel"
import { BreakpointTypeEnum } from "@jellyfin-harmony/core"
import { ListRefreshView } from "@jellyfin-harmony/core"
import { HomeToolBar } from "../../../widget/bar/HomeToolBar"
import { CategoryToolBar } from "../../../widget/bar/CategoryToolBar"
import { EpisodeItem } from "../../../widget/media/EpisodeItem"
import { MediaItem } from "../../../widget/media/MediaItem"
import { GroupInfo } from "../../../entity/GroupInfo"

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/7 20:27
 * @Version V1.0
 * @Description Main
 */
@Component
export struct FavouriteComponent {

    private viewModel = new FavouriteViewModel(getContext())

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

    @Builder
    favouriteGroupBuilder(group: GroupInfo, index: number) {
        FavouriteGroup({groupInfo: group})
    }

    build() {
        Column() {

            HomeToolBar({
                title: $r('app.string.home_favourite')
            })

            ListRefreshView({
                source: this.viewModel,
                itemLayout: this.favouriteGroupBuilder,
                listAttribute: (attr) => {
```

```
            attr.divider = {
              color: Color.Transparent,
              strokeWidth: 16
            }
            attr.width = CommonConstants.FULL_PERCENT
            attr.height = CommonConstants.FULL_PERCENT
          },
        })
          .width(CommonConstants.FULL_PERCENT)
          .layoutWeight(1)
      }
    }
}

@Component
@Reusable
struct FavouriteGroup {

  @Require @Prop groupInfo: GroupInfo

  aboutToReuse(params: Record<string, Object>): void {
    this.groupInfo = params
  }

  build() {
    Column() {
      // title
      CategoryToolBar({
        itemDto: this.groupInfo.group
      })
      // list
      List({space: 8}) {
        ForEach(this.groupInfo.items, (item: BaseItemDto) => {
          ListItem() {
            if (item.Type === BaseItemKind.Episode) {
              EpisodeItem({item: item})
            } else {
              MediaItem({item: item})
            }
          }
        })
      }
      .contentStartOffset(CommonConstants.SPACE_16)
      .contentEndOffset(CommonConstants.SPACE_16)
```

```
            .width('100%')
            .scrollBar(BarState.Off)
            .listDirection(Axis.Horizontal)
        }
        .alignItems(HorizontalAlign.Start)
        .width('100%')
    }

    aboutToRecycle(): void {

    }

}
import { ListViewModel } from "../../common/ListViewModel";
import { LoadType } from "@jellyfin-harmony/core";
import { BaseItemDto, BaseItemKind } from "@jellyfin-harmony/sdk";
import { GroupInfo } from "../../../entity/GroupInfo";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/14 21:24
 * @Version V1.0
 * @Description
 */
export class FavouriteViewModel extends ListViewModel {

    async loadData(type: LoadType): Promise<void> {
        let includeTypeArray = [BaseItemKind.Movie, BaseItemKind.Series, BaseItemKind.Episode]
        let items = await this.repository.loadFavourite(includeTypeArray)
        let groupMap = new Map<string, GroupInfo>()
        items.forEach((item) => {
            let groupInfo = groupMap.get(item.Type!)
            if (!groupInfo) {
                let name: string
                if (item.Type === BaseItemKind.Movie) {
                    name = "电影"
                } else if (item.Type === BaseItemKind.Series) {
                    name = "电视剧"
                } else if (item.Type === BaseItemKind.Episode) {
                    name = "集"
                } else {
                    name = item.Type!
                }
```

```
            groupInfo = {
                group: {
                    Name: name,
                },
                items: new Array<BaseItemDto>()
            }
            groupMap.set(item.Type!, groupInfo)
        }
        groupInfo.items!.push(item)
    })
    let groupArray = new Array<GroupInfo>()
    includeTypeArray.forEach((type) => {
        let groupInfo = groupMap.get(type)
        if (groupInfo) {
            groupArray.push(groupInfo)
        }
    })
    this.dataSource.initData(groupArray)
  }
}
import { BaseItemDto } from '@jellyfin-harmony/sdk';
import { BreakpointTypeEnum } from '@jellyfin-harmony/core';
import { CommonConstants } from '../../common/CommonConstants';
import { MainViewModel } from './MainViewModel';
import { ListRefreshView } from '@jellyfin-harmony/core';
import { HomeToolBar } from '../../../widget/bar/HomeToolBar';
import { CategoryToolBar } from '../../../widget/bar/CategoryToolBar';
import { MediaItem } from '../../../widget/media/MediaItem';
import { EpisodeItem } from '../../../widget/media/EpisodeItem';
import { GroupInfo } from '../../../entity/GroupInfo';

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/7 20:27
 * @Version V1.0
 * @Description Main
 */
@Component
export struct MainComponent {

  private viewModel = new MainViewModel(getContext())

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

```
  @Builder
  private mainGroup(groupInfo: GroupInfo, index: number) {
    MainNormalGroup({
      groupInfo: groupInfo
    })
  }

  build() {
    Column() {
      HomeToolBar({
        title: $r('app.string.home_main')
      })

      ListRefreshView({
        source: this.viewModel,
        itemLayout: this.mainGroup,
        listAttribute: (attr) => {
          attr.divider = {
            color: Color.Transparent,
            strokeWidth: CommonConstants.SPACE_12
          }
          attr.width = CommonConstants.FULL_PERCENT
          attr.height = CommonConstants.FULL_PERCENT
        },
      })
        .width(CommonConstants.FULL_PERCENT)
        .layoutWeight(1)
    }
  }

  aboutToDisappear(): void {

  }

}

@Component
@Reusable
struct MainNormalGroup {

  @Require @Prop groupInfo: GroupInfo

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

```
        this.groupInfo = params
    }

    build() {
        Column() {
            // title
            CategoryToolBar({itemDto: this.groupInfo?.group})
            // list
            List({space: CommonConstants.SPACE_12}) {
                ForEach(this.groupInfo!.items, (item: BaseItemDto) => {
                    ListItem() {
                        if (this.groupInfo.group?.CollectionType) {
                            MediaItem({item: item})
                        } else {
                            EpisodeItem({item: item})
                        }
                    }
                })
            }
            .contentStartOffset(CommonConstants.SPACE_16)
            .contentEndOffset(CommonConstants.SPACE_16)
            .width(CommonConstants.FULL_PERCENT)
            .scrollBar(BarState.Off)
            .listDirection(Axis.Horizontal)
        }
        .width(CommonConstants.FULL_PERCENT)
    }

    aboutToRecycle(): void {

    }
}
import { LoadType } from "@jellyfin-harmony/core";
import { GroupInfo } from "../../../entity/GroupInfo";
import { ListViewModel } from "../../common/ListViewModel";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/12 21:42
 * @Version V1.0
 * @Description
 */
export class MainViewModel extends ListViewModel {
```

```
    constructor(context: Context) {
        super(context)
    }

    async loadData(type: LoadType): Promise<void> {
        let groupArray = new Array<GroupInfo>()
        let resumeItems = await this.repository.getResumeItems()
        if (resumeItems && resumeItems.length > 0) {
            groupArray.push({
                group: {
                    Name: "继续观看"
                },
                items: resumeItems
            })
        }
        let nextUpItems = await this.repository.getNextUp()
        if (nextUpItems && nextUpItems.length > 0) {
            groupArray.push({
                group: {
                    Name: "接下来"
                },
                items: nextUpItems
            })
        }
        let normalItems = await this.repository.getMediaList()
        if (normalItems) {
            for (let item of normalItems) {
                let items = await this.repository.getLatestMedia(item.Id)
                if (items && items.length > 0) {
                    groupArray.push({
                        group: item,
                        items: items
                    })
                }
            }
        }
        this.dataSource.initData(groupArray)
    }

}
import { GridAttr } from "@abner/refresh";
import { BaseItemDto, ImageType } from "@jellyfin-harmony/sdk";
import { MediaViewModel } from "./MediaViewModel";
```

```
import { Repository } from "../../../data/Repository"
import { BreakpointTypeEnum } from "@jellyfin-harmony/core";
import { CommonConstants } from "../../common/CommonConstants";
import { GridRefreshView } from "@jellyfin-harmony/core";
import { HomeToolBar } from "../../../widget/bar/HomeToolBar";
import { MediaListArgs } from "../../media/MediaListArgs";

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/7 20:27
 * @Version V1.0
 * @Description Main
 */
@Component
export struct MediaComponent {

  private viewModel = new MediaViewModel(getContext())

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

  @Builder
  mediaItem(item: BaseItemDto, index: number) {
    MediaItem({
      item: item
    })
  }

  build() {
    Column() {
      HomeToolBar({
        title: $r('app.string.home_media')
      })

      GridRefreshView({
        source: this.viewModel,
        itemLayout: this.mediaItem,
        gridAttribute: (attr: GridAttr) => {
          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
            },
        })
            .width(CommonConstants.FULL_PERCENT)
            .layoutWeight(1)
    }
  }
}

@Reusable
@Component
struct MediaItem {

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

  @Require item?: BaseItemDto = undefined

  build() {
    Stack() {
      Image(this.repository?.buildImage(this.item!, ImageType.Primary))
        .alt($r('app.media.alt'))
        .autoResize(true)
        .objectFit(ImageFit.Cover)
        .borderRadius($r('app.float.lg_border_radius'))
        .width('100%')
        .aspectRatio(1.5)
    }
    .onClick(() => {
      let args: MediaListArgs = {
        id: this.item!.Id!,
        name: this.item!.Name!,
        type: this.item!.CollectionType!
      }
      this.getUIContext().getRouter().pushUrl({url:    "pages/media/MediaListPage",    params:
args})
    })
  }

}
import { LoadType } from "@jellyfin-harmony/core";
import { ListViewModel } from "../../common/ListViewModel";
/**
```

```
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/13 21:56
 * @Version V1.0
 * @Description
 */
@Observed
export class MediaViewModel extends ListViewModel {

    loadData(type: LoadType): Promise<void> {
        return this.repository.loadMedia()
            .then((items) => {
                this.dataSource.initData(items)
            })
    }

}
import { Repository } from "../../../data/Repository"
import { HomeToolBar } from "../../../widget/bar/HomeToolBar"
import { CommonConstants } from "../../common/CommonConstants"
import { WebTool} from '@jellyfin-harmony/core'

/**
 * @Author peerless2012
 * @Email peerless2012@126.com
 * @DateTime 2024/11/7 20:27
 * @Version V1.0
 * @Description Prefer
 */
@Component
export struct PreferComponent {

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

    @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}) {
      HomeToolBar({
        title: $r('app.string.home_prefer')
      })

      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.externaldrive'), $r('app.string.prefer_server'), () => {
        this.getUIContext().getRouter().pushUrl({url: "pages/edit/server/EditServerPage"})
      })
      this.preferItemBuilder($r('sys.symbol.worldclock'), $r('app.string.prefer_address'), () => {
        this.getUIContext().getRouter().pushUrl({url: "pages/edit/address/EditAddressPage"})
      })
```

```
            this.preferItemBuilder($r('sys.symbol.person'), $r('app.string.prefer_user'), () => {
                this.getUIContext().getRouter().pushUrl({url: "pages/edit/user/EditUserPage"})
            })
            this.preferItemBuilder($r('sys.symbol.doc_plaintext'), $r('app.string.prefer_privacy'), () =>
{
                // TODO
                WebTool.openBrowser(getContext(),
"https://agreement-drcn.hispace.dbankcloud.cn/index.html?lang=zh&agreementId=1571961528
242951104")
            })
        }
        .alignItems(HorizontalAlign.Start)
        .width(CommonConstants.FULL_PERCENT)
        .height(CommonConstants.FULL_PERCENT)
    }

}


import { window } from '@kit.ArkUI';
import { FavouriteComponent } from './favourite/FavouriteComponent';
import { MainComponent } from './main/MainComponent';
import { PreferComponent } from './prefer/PreferComponent';
import { MediaComponent } from './media/MediaComponent';
import { CommonConstants } from '../common/CommonConstants';

@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() {
          MainComponent()
        }
        .tabBar(this.TabBuilder($r('app.string.home_main'), 0, $r('sys.symbol.house_fill')))

        TabContent() {
          MediaComponent()
        }
        .tabBar(this.TabBuilder($r('app.string.home_media'),                                    1,
$r('sys.symbol.externaldrive_fill')))

        TabContent() {
          FavouriteComponent()
        }
        .tabBar(this.TabBuilder($r('app.string.home_favourite'), 2, $r('sys.symbol.heart_fill')))

        TabContent() {
          PreferComponent()
        }
        .tabBar(this.TabBuilder($r('app.string.home_prefer'),                                   3,
$r('sys.symbol.square_fill_grid_2x2')))
      }
      .expandSafeArea([SafeAreaType.SYSTEM], [SafeAreaEdge.BOTTOM])
      .barWidth('100%')
      .barHeight(50)
```

```
        .animationDuration(0)
        .backgroundColor($r('sys.color.background_primary'))
        .onChange((index: number) => {
          this.currentIndex = index;
        })
        .padding({
          top: px2vp(this.currentAvoidArea?.topRect.height),
          bottom: px2vp(this.currentAvoidArea?.bottomRect.height)
        })
    }

    aboutToDisappear(): void {
    }

}
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})
            })
    }
```

```
}
```