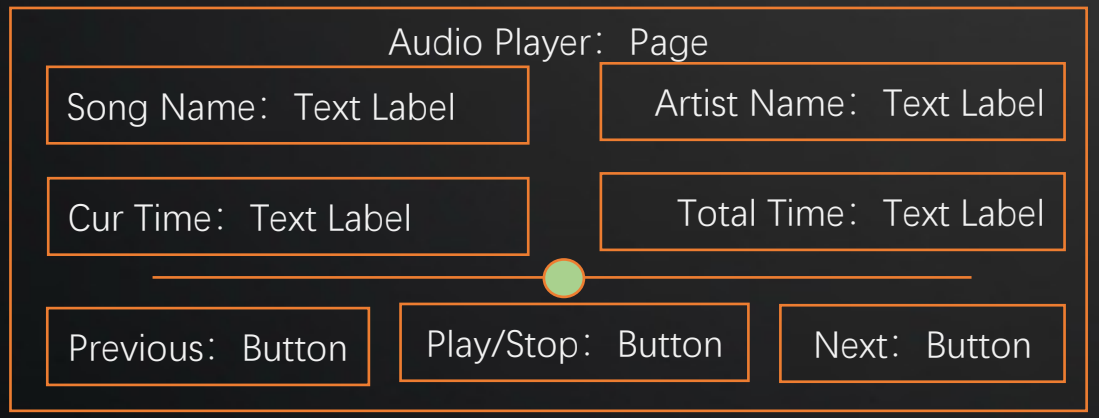
MVC_MVP_MVVM

王丰 （Feng WANG）

01 应用课题

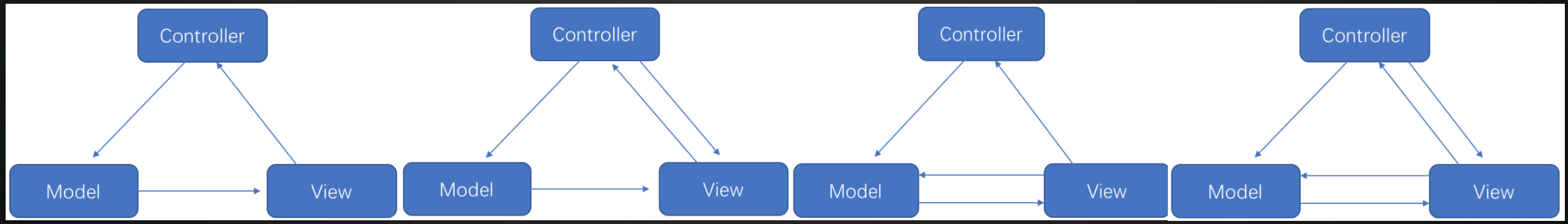



```
IAudioServiceListener
{
    virtual void onPlayingSongChanged (const string& strSongName,
                                       const string& strArtistName);

    virtual void onPlayingTimeChanged (long lTotalTime,
                                       long lCurTime);
};
```

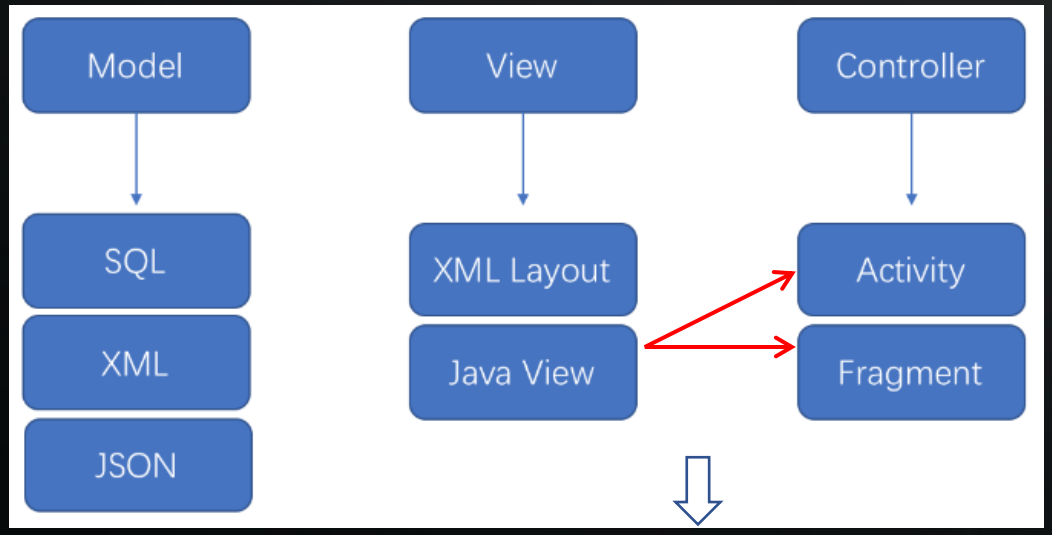
```
<CPage name="AudioPlayer"
    <CCtrlTextLable name="SongName"/>
    <CCtrlTextLable name="ArtistName"/>
    <CCtrlTextLable name="CurTime"/>
    <CCtrlTextLable name="TotalTime"/>
    <CCtrlButton name="Play"/>
    <CCtrlButton name="Stop"/>
    <CCtrlButton name="Previous"/>
    <CCtrlButton name="Next"/>
    <CCtrlProgressBar name="PlayingProgress"/>
</CPage>
```

```
CAudioService
{
Public:
    void            play();
    void            stop();
    void            previous();
    void            next();

    const string&   getPlayingSongName();
    const string&   getPlayingArtistName();
    long            getPlayingCurTime();
    long            getPlayingTotalTime();
};
```

02 MVC (Model-View-Controller)



Android



- MVC框架模式最早由Trygve Reenskaug 于1978年在Smalltalk-80系统上首次提出。
- 经过多年的发展，演变出不同版本，但核心没变依旧还是三层模型Model-View-Controller。
- 箭头→代表的是一种事件流向，直接或间接持有对方，比如，view可以通过注册监听器的形式得到model发来的事件。
- Conroller：有说是UI 输入源控制，有说是业务逻辑控制。
- Model：有说是数据层，有说是封装对数据的操作。
- View：有说是资源信息，有说是包含显示逻辑代码。

02 MVC (Model-View-Controller)

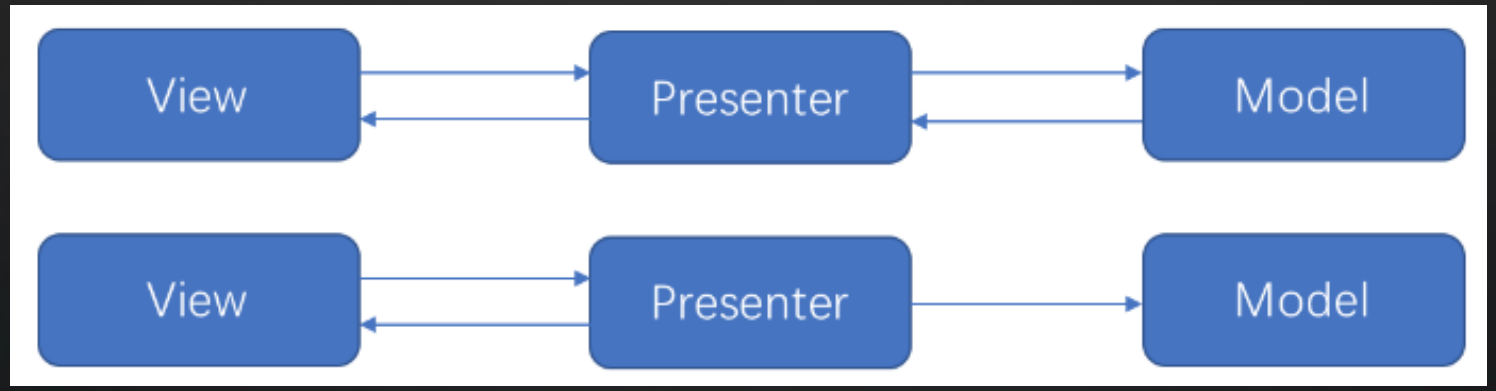
借助HMI框架的强有力支持

```
CPageAudioPlayer
    : public CPage
    , public IAudioServiceListener
{
Public:
    CPageAudioPlayer();

    void        onBtnPlayClick();
    void        onBtnStopClick();
    void        onBtnPreviousClick();
    void        onBtnNextClick();

    virtual void onPlayingSongChanged(const string& strSongName,
                                      const string& strArtistName);
    virtual void onPlayingTimeChanged(long lTotalTime,
                                      long lCurTime);

private:
    CCtrlTextLable      m_txtSongName;
    CCtrlTextLable      m_txtArtistName;
    CCtrlTextLable      m_txtCurTime;
    CCtrlTextLable      m_txtTotalTime;
    CCtrlProgressBar    m_pgbPlayingProgress;
    CCtrlButton         m_btnPlay;
    CCtrlButton         m_btnStop;
    CCtrlButton         m_btnPrevious;
    CCtrlButton         m_btnNext;
};
```

该View代码可以自动生成

```
void CPageAudioPlayer::CPageAudioPlayer()
{
    addComponent(L"SongName",  &m_txtSongName);
    ……

    loadResource("AudioPlayer.xml");

    m_btnPlay.addListener(L"onClick",  onBtnPlayClick);
    ……
   // 以上代码自动生成

    CAudioService::getInstance()->addListener(this);
    m_txtSongName.setContent(CAudioService::getInstance()->getPlayingSongName());
    ……
}

void CPageAudioPlayer::onBtnPlayClick() // 函数体代码框架自动生成
{
    CAudioService::getInstance()->play();

    m_btnPlay.setVisiable(FALSE);
    m_btnStop.setVisible(TRUE);
}

void CPageAudioPlayer::onPlayingTimeChanged(long lTotalTime, long lCurTime)
{
    m_txtCurTime.setContent(lCurTime);
    m_txTotalTime.setContent(lTotalTime);
    m_pgbPlayingProgress.setProgressInfo(lCurTime, lTotalTime);
}
```

03 MVP (Model-View-Presenter)



- 根据MVC的发展来看，把MVP当成MVC来看也不为过。
- Model和View解耦，都是通过Presenter完成。
- Google的MVP实现方案是把业务逻辑放在presenter中，弱化Model。
- Contract 契约类用于定义同一个界面的view的接口和presenter的具体实现，这是Google MVP与其他实现方式的一个不同。Presenter（ Contract ）持有View。
- View持有Presenter，并调用Presenter的方法，实现Contract契约类中定义的view接口， Contract传递数据给view。
- View的内部实现，可能会使用Model中定义的数据结构。
- UI的改变多的情况下，会有非常多的跟UI相关的case，这样就会造成View的接口会很庞大而且变化。

03 MVP (Model-View-Presenter)

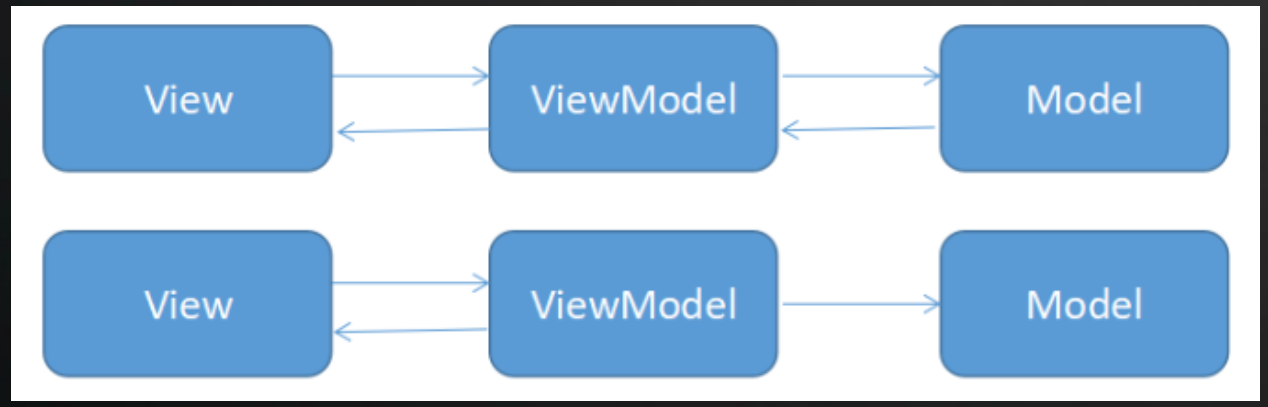
```
CPageProxyAudioPlayer
    : public CPageProxy
    , public IAudioServiceListener
{
Public:
    virtual void onBuildControlTreeComplete();

    void        onBtnPlayClick();
    void        onBtnStopClick();
    void        onBtnPreviousClick();
    void        onBtnNextClick();

    virtual void onPlayingSongChanged(const string& strSongName,
                                      const string& strArtistName);
    virtual void onPlayingTimeChanged(long lTotalTime,
                                      long lCurTime);
};
```

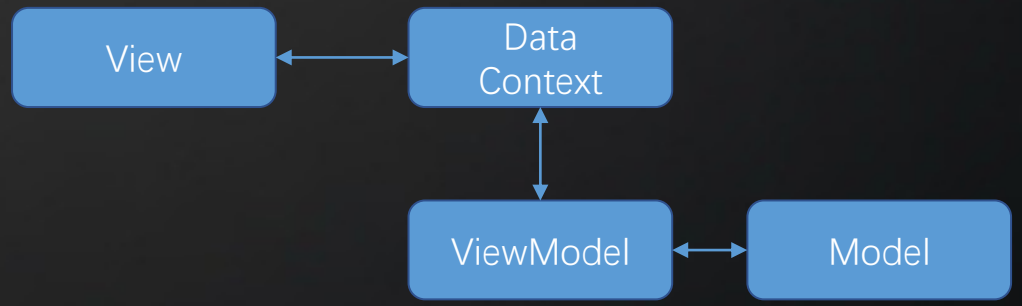
该View代码可以自动生成
不在是真正意义上的View
更具有Presenter的特性

HMI框架的能力进一步强化

```
void CPageProxyAudioPlayer::onBuildControlTreeComplete()
{
    findComponent(L"Play")->addListener(L"onClick",  onBtnPlayClick);
    ……
   // 以上代码自动生成

    findComponent(L"SongName")->setProperty(L"content",
                                        CAudioService::getInstance()->getPlayingSongName());
    ……
}

void CPageProxyAudioPlayer::onBtnPlayClick() // 函数体代码框架自动生成
{
    CAudioService::getInstance()->play();
}

void CPageProxyAudioPlayer::onPlayingTimeChanged(long lTotalTime, long lCurTime)
{
     findComponent(L"CurTime")->setProperty(L"content", lCurTime);
     findComponent(L"TotalTime")->setProperty(L"content", lTotalTime);
     findComponent(L"PlayingProgress")->setProperty (L"pos", lCurTime);
     findComponent(L"PlayingProgress")->setProperty (L"max", lTotalTime);
}
```

04 MVVM (Model-View-ViewModel)





- 最早是微软的WPF引出MVVM概念。
- 看起来跟MVP一样，但实现机制，有根本的区别。
- View可以包含资源信息和显示逻辑代码。
- View不再使用Model中的任何信息。
- View和ViewModel不再双向持有，通过Data-Binding，做双向的桥梁。
- Google Android新版本已经支持Data-Binding机制，不过使用起来还是有些复杂，不友好。

04 MVVM (Model-View-ViewModel)

HMI框架的能力达到最优化

```
<DataContext name="AudioPlayer"
    <DataProperty name="SongName"/>
    <DataProperty name="ArtistName"/>
    <DataProperty name="CurTime"/>
    <DataProperty name="TotalTime"/>
    <Command name="Play"/>
    <Command name="Stop"/>
    <Command name="Previous"/>
    <Command name="Next"/>
</DataContext >
```

```
<CPage name="AudioPlayer"
    <CCtrlTextLable name="SongName" content_bind="SongName"/>
    <CCtrlTextLable name="ArtistName" content_bind="ArtistName"/>
    <CCtrlTextLable name="CurTime" content_bind="CurTime"/>
    <CCtrlTextLable name="TotalTime" content_bind="TotalTime"/>
    <CCtrlButton name="Play" onClick_bind="Play"/>
    <CCtrlButton name="Stop" onClick_bind="Stop"/>
    <CCtrlButton name="Previous" onClick_bind="Previous"/>
    <CCtrlButton name="Next" onClick_bind="Next"/>
    <CCtrlProgressBar name="PlayingProgress" pos_bind="CurTime"  max_bind="TotalTime"/>
</CPage>
```

```
CImpViewModelAudioPlayer
    : public CBaseViewModel
    , public IAudioServiceListener
{
Public:
    virtual void initialize();
    virtual void doCommand(const string& strCommand);

     virtual void onPlayingSongChanged(
                                const string& strSongName,
                                const string& strArtistName);

    virtual void onPlayingTimeChanged(long lTotalTime,
                                            long lCurTime);
};
```

```
void CImpViewModelAudioPlayer::initialize()
{
    CContextProxy::updateDataProperty( L"SongName",
                                        CAudioService::getInstance()->getPlayingSongName());
    ......
}
void CImpViewModelAudioPlayer::doCommand(const string& strCommand)
{
    if (strCommand == L"Play") CAudioService::getInstance()->play();
}
void CImpViewModelAudioPlayer ::onPlayingTimeChanged(long lTotalTime, long lCurTime)
{
    CContextProxy::updateDataProperty(L"CurTime", lCurTime);
    CContextProxy::updateDataProperty(L"TotalTime", lTotalTime);
}
```

THANK YOU