**<u>BrowseViewController</u>**

- User taps on instance of SPVideoItemViewCell which launches SPVideoReel.
- **SPVideoReel *videoReel** = [[**SPVideoReel alloc**] initWithGroupType:**groupType groupTitle**:title **videoFrames**:videoFrames **videoStartIndex**:videoIndex **andChannelID**:channelID];
- **SPVideoReel** is passed knowledge of:
    - **GroupType:** (Stream, Likes, Personal Roll, ChannelDashboard, ChannelRoll)
    - **Group Title**: (Stream, Likes, Channel Zero, Adrenaline, etc...0
    - **Video Frames**: Array of pre-fetched videoFrames for a particular GroupType
    - **Video Start Index**: Video that was tapped within videoFrames
    - **ChannelID (optional)**: Only used if GroupType = ChannelDashbaord/ChannelRoll

**<u>SPVideoReel</u>**

- After initialization, app runs through setup methods. In a nutshell,
    - iVars are initialized with info
    - Observers are initialized
    - Scroll View is initialized
    - SPOverlayView is initialized
    - Gestures (up/down/pinch/single-tap) are initialized
    - VideoPlayers are initialized
    - AirPlay is initialized
- After videoPlayers are initialized in **setupVideoPlayers**, the video-loading method, **currentVideoDidChangeToVideo:** is hit and initialzied video at chosen **videoStartIndex**
    - Video at **videoStartIndex** begins extraction
    - Queues up next few videos in **SPVideoExtractor** for extraction
    - Connects SPVideoScrubber to chosen SPVideoPlayer object
    - Changes state of SPVideoOverlay to show everything about chosen video
    - Changes labels, buttons, and images to display info for chosen video
    - Disable overlayTimer - NSTimer that hides overlay. Re-activated when chosen video is extracted and playback is initialized

**<u>The video loading/extraction process</u>**

- When **currentVideoDidChangeToVideo:** is activated, videos are cued for extraction. The number of videos depends on the the iPad's hardware limitations.
- The call to the extractor occurs on the **queueMoreVideos**: method.
- All remaining extractions are cancelled, to make sure the newest video is going to be extracted.
    - **[[SPVideoExtractor sharedInstance] cancelRemainingExtractions];**
- Video begins extraction in this method: **[self extractVideoForVideoPlayer:position]**
- If the video is loaded on hard disk (offline-mode) then extraction is skipped and SPVideoPlayer for the current video is loaded with the downloaded video.
- If video isn't on disk, then video is sent to extractor by way of SPVideoPlayer, using SPVideoPlayer's **queueVideo**: method.
- In **queueVideo:,** a check is performed to see if the video was previously extracted, but dropped from memory due to hardware/memory constraints. If this is true (within a specific time window), the video is automatically loaded back into SPVideoPlayer and playback is re-initialized.
- If the video isn't cached, the video is then sent to the extractor: **[[SPVideoExtractor sharedInstance] queueVideo:[_videoFrame video]];**
- The video is added to a queue, and when its turn appears, the video is extracted via an invisible UIWebView. Once extraction is successful, the mp4 is saved on the **extractedURL** property of the **Video** NSManagedObject (e.g., **video.extractedURL =/path/to/mp4**).

- CoreDataUtility is initialized and the **video.extractedURL** change is committed into the Store. Once the change has been committed in CoreDataUtility's **saveContext:** method, a notification is sent via NSNotificationCenter. All SPVideoPlayers are observing this notification. Only the SPVideoPlayer whose frame.video.providerID matches the one sent back in the notification loads the video in its AVPlayer.
- The method that is reached from this notification is SPVideoPlayer's **loadVideo:** method.
- If offlineMode is enabled, the extractedURL from the video object in the NSNotificaiton is used to download the video using SPVideoDownloader.
- If offlineMode is disabled, **setupPlayerForURL:**  method is hit, which instantiates AVPlayer, AVPlayerLayer, AVPlayerAsset, and AVPlayerItem. All of these AVFoundation classes are using to initiate video playback.
- Now, this **setupPlayerForURL:** is also accessed from a previously cached video (e.g., in the situation when video is dropped from memory due to memory constraints). If the video is accessed in this situation, the video begins playback at the elapsedTime where video playback ended, using AVPlayer's **seekToTime:** method.
- If this is the first time the video is setup, the video stream is initiated and a double-tap gesture for toggling playback is initialized.
- A notification, **AVPlayerItemDidPlayToEndTimeNotification**,  is also added to monitor for when playback ends. When this occurs, the method **itemDidFinishPlaying:** on SPVideoPlayer is hit, and the next video is loaded (e.g., continuous playback).
- This notificaiton also allows for the restart-playback button to appear if a user ever goes back to a previously finished video.