Overview

Please note that this repo contains the following:

- Lightning component LNE_VisualforceContainer (the container component)
- Static resource LNE_GeneralResources (static resource used by component)

along with demo materials:

- DEMO_CustomCount__c custom field on Contact
- TEST_PostMessageParent - starting point for the demo
- TEST_PostMessageParent__c custom controller
- TEST_PostMessageChild - child page contained within the demo

Deploying Demo

@TODO: make a separate deployment for just the component and static resource

1: run `ant makeCredentials` to generate the credentials file

```
ant makeCredentials
```

2: run `ant test` to do a test deploy

```
ant test
```

3: run `ant deploy` to deploy the demo to your org

```
ant deploy
```

4: login and navigate to /apex/TEST_PostMessageParent

Helpers

LNE_PostMessage2

LNE_PostMessage is a JavaScript class that provides a way to send a message from one window to another.

By creating the LNE_PostMessage2, you can dispatch that event to another window and then

receive and parse to have the exact same message dispatched.

It also provides additonal helper functions to specify the page it was sent from, and message type - along with filter functions to validate on the receiving side.

The LNE_PostMessage is available from any visualforce page by including:

```
<apex:includeScript
value='{!URLFOR($Resource.LNE_GeneralResources,"js/events/LNE_PostMessage2.js")}' />
```

Example dispatch:

```
var pageName = 'LNE_TestPostMessage';
var method = 'saveAttempted';
var isSuccessful = true;
var data = { userId: 'someId', someSetting: 23 };
var m = new LNE_PostMessage( pageName, method, isSuccessful, data );
```

Or all in one line

```
new LNE_PostMessage( 'LNE_TestPostMessage','saveComplete',true,
{src:window.location.href}).dispatch( parent );
```

To receive events, all that is needed is to listen for 'message' events on the window:

```
//-- all postMessages are dispatched as window level events
//-- of type 'message'
window.addEventListener( "message", handleResultDispatch, false );

function handleResultDispatch( evt ){
    var postMessage = LNE_PostMessage.parse( evt );

    if( postMessage ){
        postMessage.matchesPageMessage( 'LNE_TestPostPage','saveAttempted' ){
        console.log( 'user:' + postMessage.data.userId );
        console.log( 'someSetting:' + postMessage.data.someSetting );
    }
}
```

for more info, please see: https://developer.mozilla.org/en-US/docs/Web/API/Window/PostMessage

LNE_PostMessage methods

constructor(pageName:String, messageType:String, isSuccessful:Boolean,

payload:Object|String)

```
 Constructs an LNE Post Message (payload).
 @param pageName - String - name of the page
 @param messageType - String - arbitrary name of the message type to be sent.
 @param isSuccessful (Boolean) - whether the call was successful or not
 @param payload (String|Object) - payload to be provided (will be converted to
string)
```

dispatch(targetWindow:Window, targetDomain:String = '*'):void

```
 Dispatches the event.
 @param targetWindow (Window) - target window to dispatch from. i.e. parent
 @param targetDomain (String) - target domain to accept the request, defaults to '*'
 @return void
```

parse(evt:PostMessageEvent):LNE_PostMessage

```
 Parses a dispatched Event)
 @param evt (string - postMessage Event String)
 @return boolean - whether it was successful (true) or not (false)
```

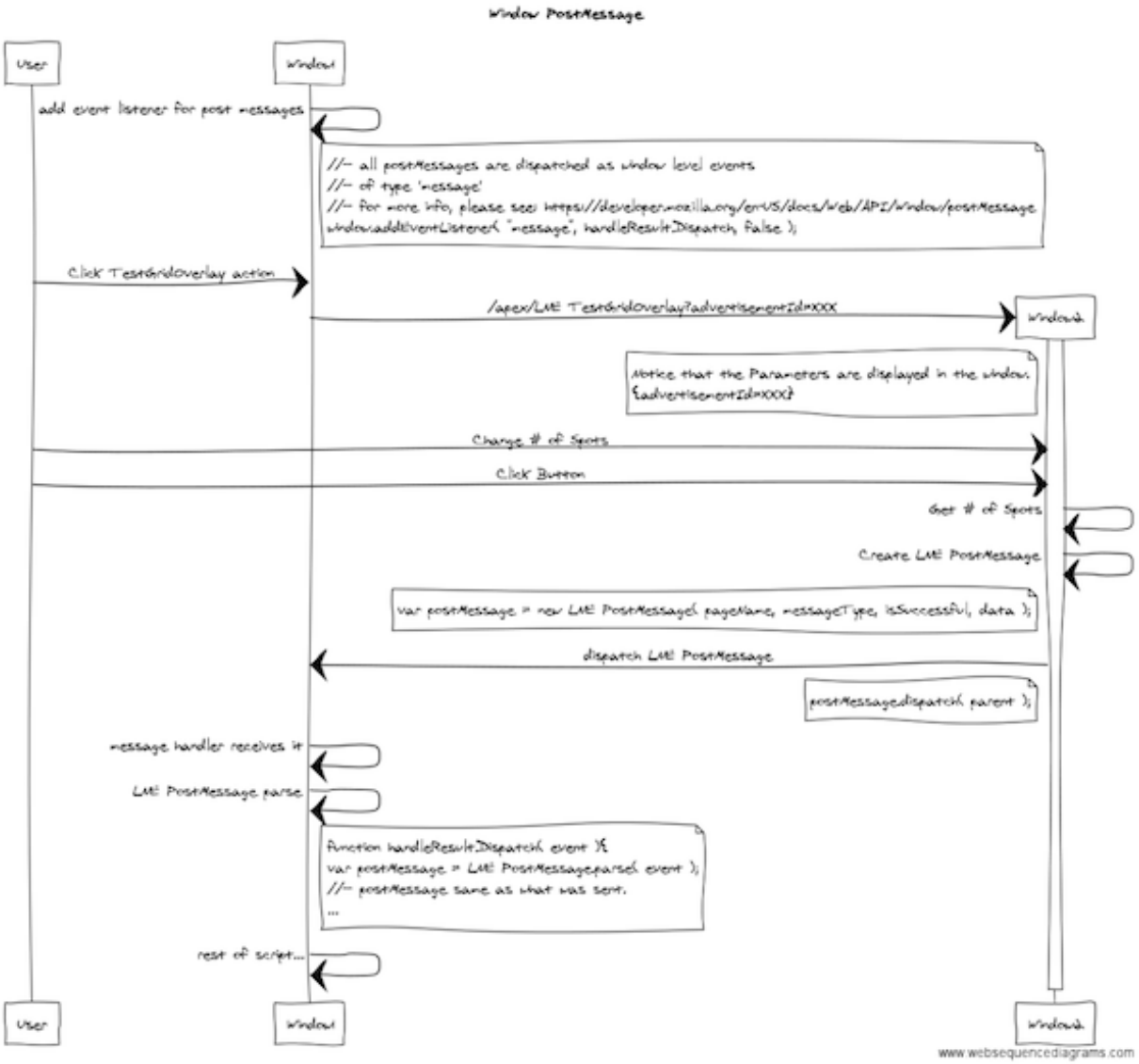
LNE_PostMessage.getMessageOrigin(evt:PostMessageEvent):String

```
 Determines the origin of a PostMessage event.
 @return String
```

matchesPageMessage(pageName:String, messageType:String):boolean

```
 *  Whether it matches both the page and the message type
 *  @param pageName (String)
 *  @param messageType (String)
 *  @return boolean - whether the pageName and the messageType both match in a case
insensitive manner.
```

Sequence Diagram:



[Web Sequence Diagram]

.

LNE_MessagePostOffice

LNE_MessagePostOffice is a JavaScript class that provides a very simple way to monitor PostMessages (but mostly geared for managing LNE_PostMessage2 messages)

To listen for LNE_PostMessage2 post messages, all that is needed is the following:

1: Create an instance of the postOffice

```
//-- instantiate with the scope object (to represent 'this' in any handling)
this.postOffice = new LNE_MessagePostOffice(this);
```

2: Add event listener for any LNE_PostMessages based on messageType

```
this.postOffice.addTypeHandler( 'testMessages', function( postMessage ){
    //-- @invariant: an LNE_PostMessage2 was received and has 'messageType' =
'testMessage';
    //-- @invariant: the EXACT object provided in LNE_PostMessage2.data is available
here
    //-- as postMessage.data
)
```

Repeat this for as many messageTypes as you would like to handle.

3: Optional: add handler for any postMessage that the type is not recognized for

```
this.postOffice.addTypeHandler( null, function( postMessage ){
    console.error( 'an unknown postMessage.type was received:' +
postMessage.messageType );
});
```

4: Listen for postMessages on the window

```
this.postOffice.listenForPostEvents( window );
```

For an example page that communicates please see

/apex/TEST_PostMessageParent

LNE_MessagePostOffice methods

constructor(scope:Object)

```
 Constructs an LNE Message Post Office
 example: this.postOffice = new LNE_MessagePostOffice(this);
 @param scope - Object - The object that represents 'this' within the handlers.
```

addTypeHandler(messageType:[null|string], handler:function):void

```
 Handler for any LNE_PostMessage2 post event that has a matching message type. (or
catchall handler if null)
 example: this.postOffice.addTypeHandler( 'testMessage', function(postMessage){} );
 @param handler (function(LNE_PostMessage2)) - function that will execute
 @return void
```

listenForPostevents(window:Window):void

```
 Initiates the PostOffice for listening for PostMessages on that window.
 example: this.postOffice.listenForPostEvents( window );
 @param w (Window) window to listen to post messages on.
 @return void
```