

```

package dr.utils.spellChecker
{
    import com.gskinner.spelling.SPLTagFlex;
    import com.gskinner.spelling.SPLWordListLoader;
    import com.gskinner.spelling.SpellingDictionary;
    import com.gskinner.spelling.SpellingDictionaryEvent;
    import com.gskinner.spelling.SpellingHighlighter;

    import dr.model.state.AppModel;

    import flash.display.BitmapData;
    import flash.display.DisplayObject;
    import flash.display.DisplayObjectContainer;
    import flash.events.Event;
    import flash.text.TextField;

    import mx.core.Application;

    [Bindable]
    /**
     * A wrapper for the implementation of the spell checker library.
     * @author Bryan Elkins
     */
    public class SpellChecker
    {
        //-----
        //
        // Properties
        //
        //-----

        /**
         * @private
         * Singleton instance of this class.
         */
        private static var instance:SpellChecker;

        /**
         * @private
         * The loader for the word list.
         */
        protected var listLoader:SPLWordListLoader;

        /**
         * @private
         * The target property for a spell check, stored when a spell check is attempted
         * before the word list is loaded. If the class is initialized by calling
         * getInstance() plenty of time before anything can be checked, then this
         * should hopefully be unnecessary.
         */
        protected var deferredTarget:Object;
    }
}

```

```

/**
 * @private
 * Flag for whether the word list is loaded.
 */
protected var wordListLoaded:Boolean;

//-----
// libraryUrl
//-----

/**
 * @private
 * Holder for libraryUrl
 */
protected var _libraryUrl:String;

/**
 * The url for loading the library. Setting this property also loads the library
 * at the specified url.
 */
public function get libraryUrl():String
{
    return _libraryUrl;
}

/**
 * @private
 */
public function set libraryUrl(value:String):void
{
    if (_libraryUrl != value)
    {
        _libraryUrl = value;

        // Setting the url for the listLoader initiates the load internally.
        if (!listLoader)
            listLoader = new SPLWordListLoader();
        listLoader.url = AppModel.instance.config.spellCheckUrl;
    }
}

//-----
// target
//-----

/**
 * @private
 * Holder for target.
 */
protected var _target:Object;

/**
 * SPL only looks through the first layer of children to find a text field, and our
 * project sometimes needs to go deeper than that. This method looks to see
 * which text field should be passed to SPL, since SPL won't go deep enough to

```

```

* look for one.
*
* <p>This method also stores a target for checking if a check has been attempted
* before the word list is loaded</p>
*
* @see TextHighlighter::target
*/
protected function get target():Object
{
    return _target;
}

/**
* @private
*/
protected function set target(value:Object):void
{
    if (_target != value)
    {
        var tf:TextField;
        if (value)
        {
            tf = value as TextField;
            if (!tf)
            {
                var doc:DisplayObjectContainer = value as DisplayObjectContainer;
                if (doc)
                    tf = getBiggestTextField(doc);
            }

            // If there's a text field available, check it. If not, do nothing.
            if (tf)
            {
                // Setting the target here initiates the checking process.
                if (wordListLoaded)

                    // Right now, this is set up to check a single component at a time. Uncommenting the line
                    // below allows for multiple checking and highlighting, but no removal of the highlighter
                    // instances has been implemented.
                    //
                    //var highlighter:SpellingHighlighter = new SpellingHighlighter(
                    //    tf, SpellingDictionary.getInstance());
                    //
                    tagFlex.target = tf;

                // An event listener for the load complete was already added in the
                // constructor. Once the list is loaded, this target will be grabbed and
                // set in onWordListLoaded().
                else
                    deferredTarget = tf;
            }
        }
        _target = tf;
    }
}

```

```

//-----
// tagFlex
//-----

/**
 * @private
 * Holder for tagFlex
 */
protected var _tagFlex:SPLTagFlex;

/**
 * Component needed for using SPL in Flex.
 */
protected function get tagFlex():SPLTagFlex
{
    if (!_tagFlex)
    {
        _tagFlex = new SPLTagFlex();
        _tagFlex.target = this.target;
        Application.application.addChild(_tagFlex);
    }
    return _tagFlex;
}

//-----
//
// Methods
//
//-----

/**
 * Constructor. Do not call directly. Call getInstance() instead.
 */
public function SpellChecker(singletonEnforcer:SingletonEnforcer)
{
    // This listener fired when the library is loaded. The library is loaded when the
    // "libraryUrl" property is set.
    SpellingDictionary.getInstance().addEventListener(
        SpellingDictionaryEvent.CHANGED_MASTER_WORD_LIST, onChangedMasterWordList);

    SpellingHighlighter.defaultUnderlinePattern = getUnderlinePattern();
}

/**
 * Set the underline pattern. This one is a little more opaque than the original
 * default value.
 */
protected function getUnderlinePattern():BitmapData
{
    var bmpd:BitmapData = new BitmapData(10, 4, true, 0);
    const RED:uint = 0xffff0000;
    const LIGHT_RED:uint = 0x77ff0000;

    // Make a pattern that is, approximately, little circles with spaces between them.

```

```

    bmpd.setPixel32(0, 0, LIGHT_RED);
    bmpd.setPixel32(0, 1, RED);
    bmpd.setPixel32(0, 2, RED);
    bmpd.setPixel32(0, 3, LIGHT_RED);
    bmpd.setPixel32(1, 0, RED);
    bmpd.setPixel32(1, 1, RED);
    bmpd.setPixel32(1, 2, RED);
    bmpd.setPixel32(1, 3, RED);
    bmpd.setPixel32(2, 0, RED);
    bmpd.setPixel32(2, 1, RED);
    bmpd.setPixel32(2, 2, RED);
    bmpd.setPixel32(2, 3, RED);
    bmpd.setPixel32(3, 0, LIGHT_RED);
    bmpd.setPixel32(3, 1, RED);
    bmpd.setPixel32(3, 2, RED);
    bmpd.setPixel32(3, 3, LIGHT_RED);

    return bmpd;
}

/**
 * The SPL was designed to be used as a single instance per application.
 * This method can be called to initialize the spell checker,
 * since it takes a few seconds to load the word list.
 *
 * @return A singleton instance of this class.
 */
public static function getInstance():SpellChecker
{
    if (!instance)
    {
        instance = new SpellChecker(new SingletonEnforcer());
    }
    return instance;
}

/**
 * Run spell checking on the specified object.
 * @param <code>target</code> An object for the spell checker to evaluate.
 */
public function check(target:Object):void
{
    this.target = target;
}

/**
 * Get the text field from inside a display object container.
 * @param <code>doc</code> DisplayObjectContainer that should contain a TextField for
 * spell checking.
 */
protected function getBiggestTextField(doc:DisplayObjectContainer):TextField
{
    // Extract the text field from the container and its children as necessary.
    var numKids:uint = doc.numChildren;
    var biggestTf:TextField;

```

```

var biggestSize:Number = 0;
for (var i:uint = 0; i < numKids; i++)
{
    var kid:DisplayObject = doc.getChildAt(i);
    if (kid is DisplayObjectContainer)
    {
        var kidTF:TextField = getBiggestTextField(kid as DisplayObjectContainer);
        if (kidTF && !biggestTf)
            biggestTf = kidTF;
    }
    else if (kid is TextField)
    {
        var tempTfRef:TextField = kid as TextField;

        // This is how the "size" is determined in SPL; just copying the logic here.
        var tempSize:Number = tempTfRef.width + tempTfRef.height;

        if (tempSize > biggestSize)
        {
            biggestSize = tempSize;
            biggestTf = tempTfRef;
        }
    }
}
return biggestTf;
}

/**
 * Handle when the master word list has been loaded.
 */
protected function onChangedMasterWordList(e:Event):void
{
    // When implementing other language loading, this event listener may need
    // to remain, instead of being removed here, and the loader may need to
    // not be nulled out.
    SpellingDictionary(e.target).removeEventListener(
        SpellingDictionaryEvent.CHANGED_MASTER_WORD_LIST,
        onChangedMasterWordList);
    listLoader = null;
    wordListLoaded = true;
    if (deferredTarget)
    {
        tagFlex.target = deferredTarget;
        deferredTarget = null;
    }
}

} // End class
}

/**
 * Class to enforce use of getInstance() to fetch singleton of SpellChecker.
 */
class SingletonEnforcer {}

```