**Visual Process Flow**

**POC – Editing Mode**



**Sage 300 ERP (Project Columbus NA (CNA) 2.0)**

Table of Contents

[1 Overview 1](#_Toc398130757)

[1.1 All-In-One Code Framework Coding Standards 1](#_Toc398130758)

[1.2 Software Coding Standards Document 1](#_Toc398130759)

[2 Data Types 1](#_Toc398130760)

[2.1 Alias Data Types 2](#_Toc398130761)

[2.2 Implicitly Typed Local Variables 2](#_Toc398130762)

[3 Namespaces 3](#_Toc398130763)

[3.1 Namespace Segments 3](#_Toc398130764)

[4 Comments 4](#_Toc398130765)

[4.1 XML Comments 4](#_Toc398130766)

[4.2 Copyright Comments 5](#_Toc398130767)

[5 Naming Conventions 5](#_Toc398130768)

[5.1 Redundancy 5](#_Toc398130769)

[5.2 Descriptive Names 6](#_Toc398130770)

[5.3 Method Parameters Names 7](#_Toc398130771)

[5.4 Parameter Driven Generic Routines 7](#_Toc398130772)

[5.5 Technical Terms 8](#_Toc398130773)

[5.6 Company and Product Name 8](#_Toc398130774)

[6 Razor Views 9](#_Toc398130775)

[6.1 Widget Attributes 9](#_Toc398130776)

[6.2 Buttons with Images 9](#_Toc398130777)

[6.3 ID Attribute 10](#_Toc398130778)

[7 Abbreviations 10](#_Toc398130779)

[7.1 Approved Abbreviations and Acronyms 10](#_Toc398130780)

[8 Javascript 13](#_Toc398130781)

[8.1 Using Strict Mode 13](#_Toc398130782)

[8.2 Cache your objects 14](#_Toc398130783)

[8.3 Define and using Enum 14](#_Toc398130784)

[8.4 Don’t Repeat Yourself (DRY) 15](#_Toc398130785)

[8.5 Using “===” and “!==” for primitive values compare 16](#_Toc398130786)

[8.6 Magic Numbers 17](#_Toc398130787)

[9 Jquery 17](#_Toc398130788)

[9.1 Selectors 17](#_Toc398130789)

[9.2 DOM Insertion 18](#_Toc398130790)

[9.3 Caching 19](#_Toc398130791)

[9.4 Chaining 19](#_Toc398130792)

[9.5 Object Literals 20](#_Toc398130793)

[9.6 CSS 20](#_Toc398130794)

[9.7 Avoid multiple $(document).ready() calls 21](#_Toc398130795)

[9.8 this vs. $(this) 21](#_Toc398130796)

[9.9 Use Sub-queries 21](#_Toc398130797)

[9.10 Leverage Event Delegation (a.k.a. Bubbling) 22](#_Toc398130798)

[9.11 AJAX Methods 22](#_Toc398130799)

[9.12 Passing an Attribute Object 23](#_Toc398130800)

[9.13 Use live() 23](#_Toc398130801)

[9.14 Use jQuery.noConflict() 24](#_Toc398130802)

# Overview

## Introduction

The intention of this document is to provide an overview about implementation of Visual Process Flow and its dependencies in Sage 300 Online ERP ecosystem.SVG editor is an open source tool, which provides a comprehensive environment for customization of html SVG Tags. This tool provides various futures which were customized for the Sage project the following features were customized.

1. Storage Capability: By enabling this tool the content of the svg will remain permanently in the user’s browser until the user actually perform “Clean History” on his/her machine. This feature needs to be disabled from configuration file: click here for [demo](http://svg-edit.googlecode.com/svn/branches/stable/editor/svg-editor.html)
2. The resolution of the svg canvas is required to be adjusted to width="670" height="1400", which is the size of the current visual process flow.
3. All SVG customized modules are required to be registered to the config file.

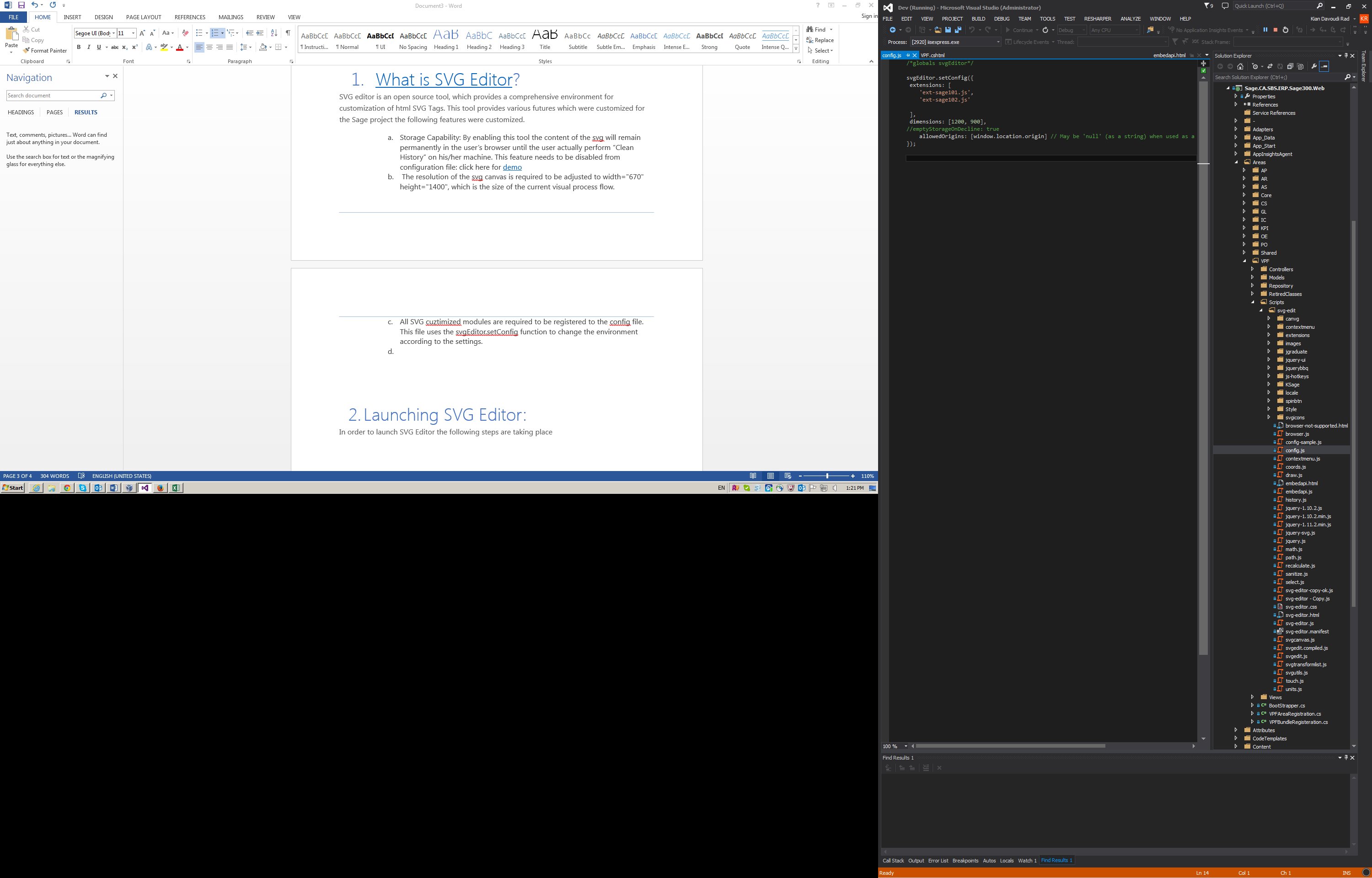


Figure 1:

# Initial launch of SVG Editor

A consistent use of data types will provide a consistent look and feel to the software. Therefore, alias data types and implicitly typed local variables will be required.

SVG Editor used asynchronous call to load itself to the browser, for that reason, an iframe has been used as primary tool to load the svg-editor.html to the iframe.

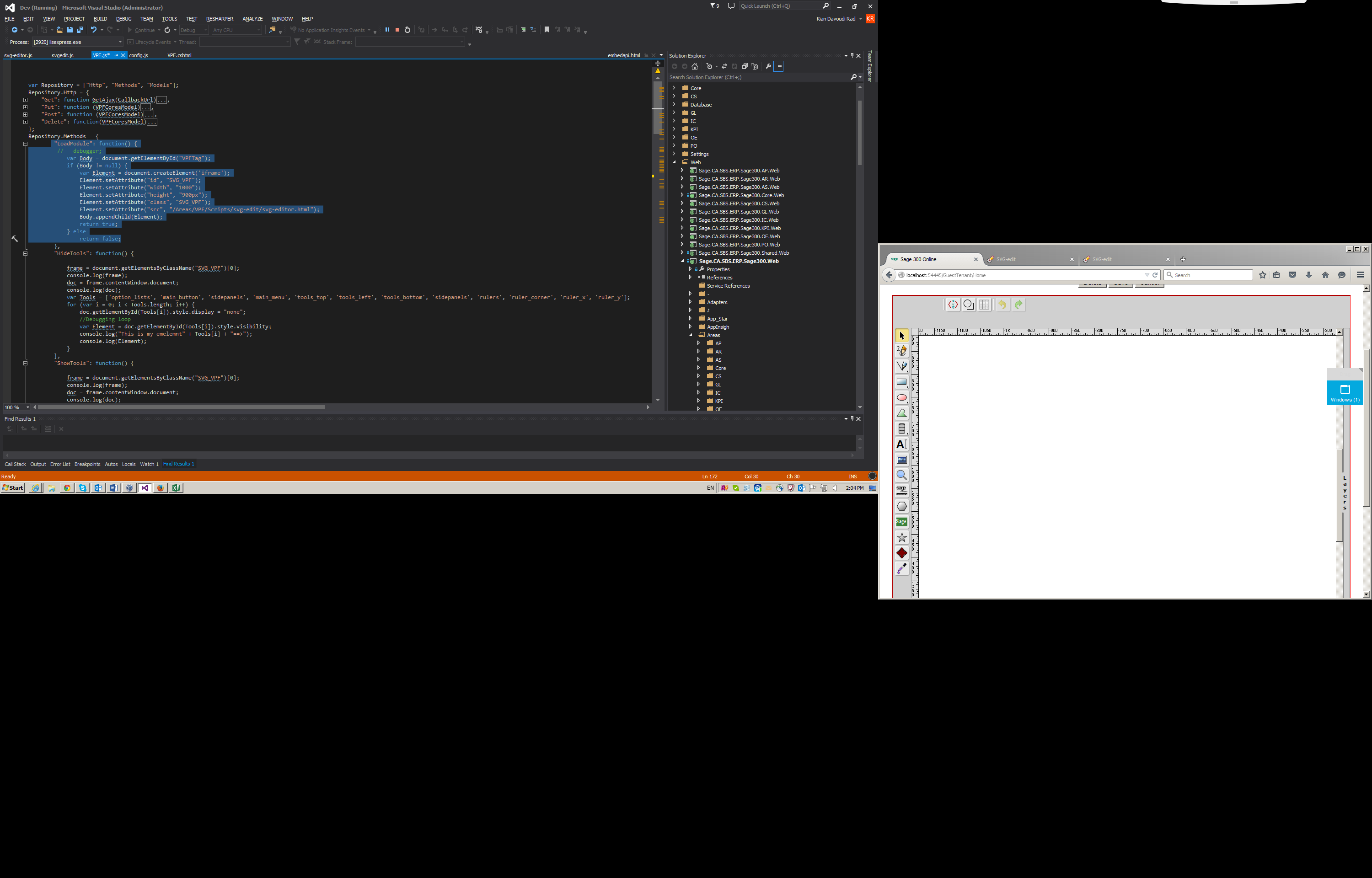


Figure2:

Upon successful load of iframe, Hiding/Showing SVG tools in editing mode: The SVG Editor is shipped with many good features but for visual process flow development, some of these features are not necessary. In order to address this item, the JS calls were used to hide these items in the Editing mode. For this reason, Hide Tools/Show Tools method were used. Then ajax Get call is used to obtain the Data from the controller in an Jason format. The saving option uses this feature to identify the identity of the template.

## Event Listener

In order to open the controller from an SVG element, I decided to use a Javascript listener call, which takes three parameters, area name, controller name and parameter. These three items are used to build a url that is being used to call the controller from visual process flow.

javascript:EventListenter\_VPF('AP','VendorGroups','');

## Sage 101

Intention of this module was to produce a tool for users to be able to assign controllers to SVG items. This module uses a drop dwon menu to provide user with the name of module and on second level, it will provide the name of the controller to the user. The software automatically, builds the javascript call and assigns this to the svg element.

## Sage 102

Sage 102: This module was also developed based on the Sage 101. The difference is that this module will use Ajax call to a controller, in order to provide a svg pre build modules to the svg editor. Upon successful transfer of module from the controller, the module will use the SVG editor to import this item as a group to the svg canvas.

## Loading data into the svg editor

In order open a visual process flow into SVG editor, EmbeddedSVGEdit method is being used, this function takes the iframe of the svg editor as an input parameter and initiates a new object.

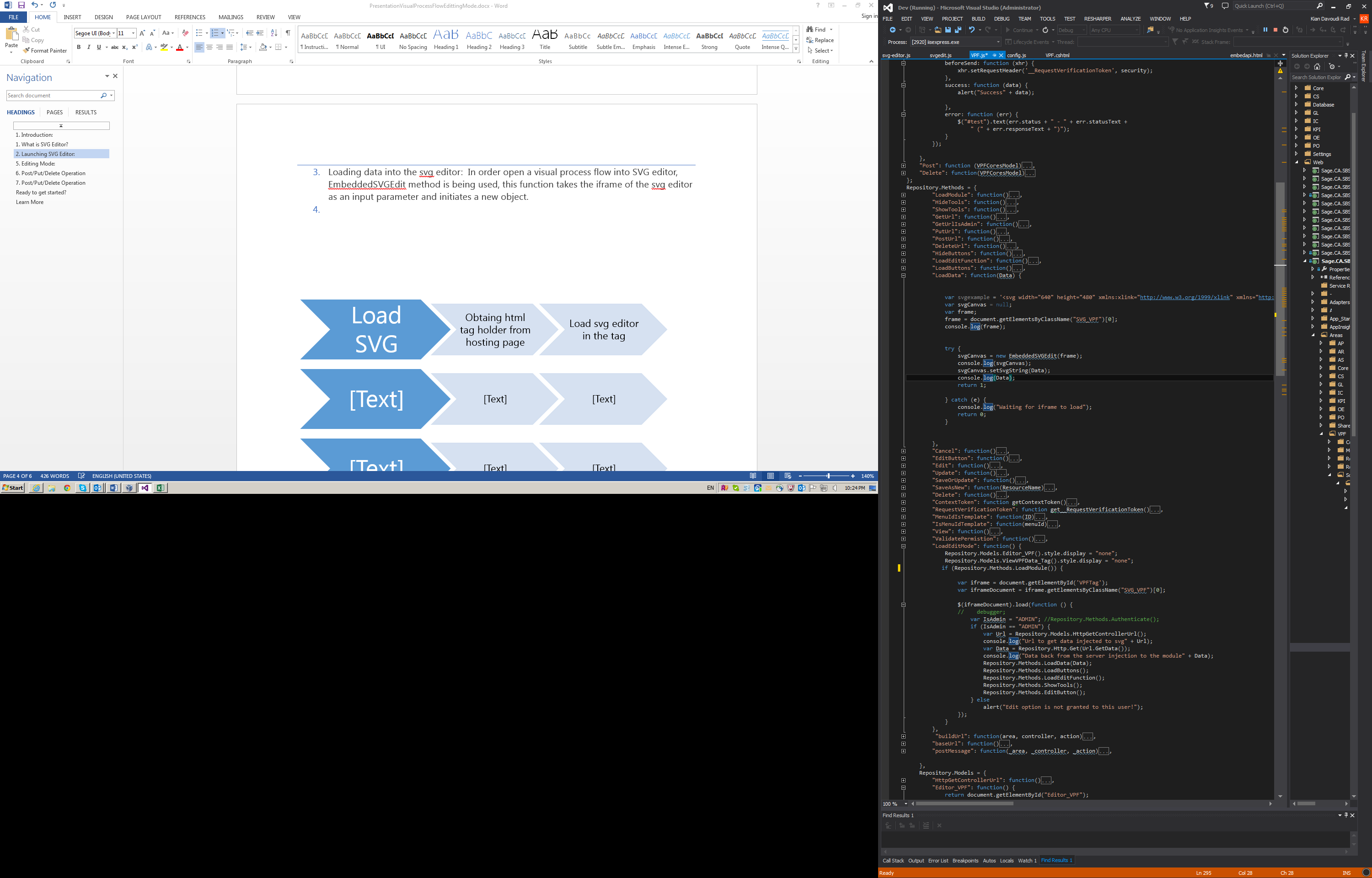


Figure 3:

## Extracting data from SVG canvas.

In order to extract data from the canvas the same producer needs to take a place as the loading steps, with difference in the method call, which this time will be getSvgString. Then the data will be posted as a Put to a controller.

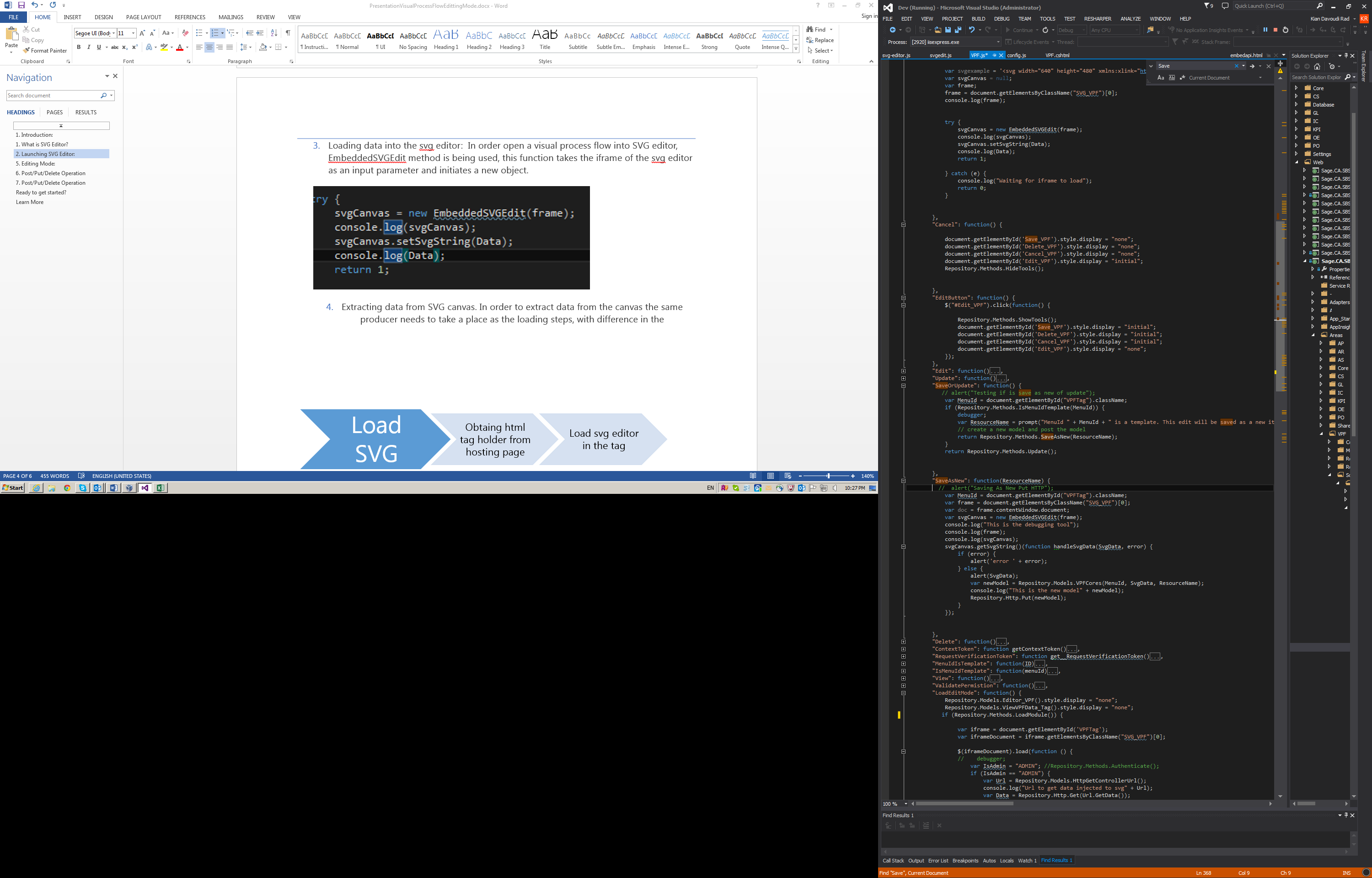


Figure4:

## Performing Delete and Update:

These two operations are used to evaluate the administration privilege.

## Numbering:

In order to distinguish between the templates and the custom designed template a numbering algorithm took place, this algorithm evaluates first four digits of the ID. The Algorithm distinguishes between 10811010. The first four digits correspond to the menu Id of the parent item. 1081 and 10 corresponds to the template ID and the second 10 corresponds to the numbering if templates for that particular screen. For example 1081 is parent ID for AP module in the menudetail.xml and 10 is used for AP and the second 10 is being used to distinguish that is not a custom designed item. This contributes to reduction of update and Create new by reducing these two methods into a one semi-automated algorithm which is able to distinguish that only custom designed items are eligible for POST operation and the rest will be a put operation.

# Overview of SVG Editor:

SVG editor provides a powerful tool for drawing propose. This tool takes advantage of standard html tags. In particular it has been focused on svg object.

Fundamental features which were studied in this POC are listed as following:

## Loading svg editor:

The tool uses asynchronous JavaScript and xml calls to revive the internal dependencies. One main reason for using iframe is to ensure that modules are loaded properly.

## Interaction to Controllers:

Across this implementation, many asynchronous calls such as Post, Delete, Put and Get calls have been implemented to accommodate movement of data from the storage layer to svg canvas. For example: the following code ensures successful opening of SVG editor and upon this success, the svg data will be loaded to canavs.

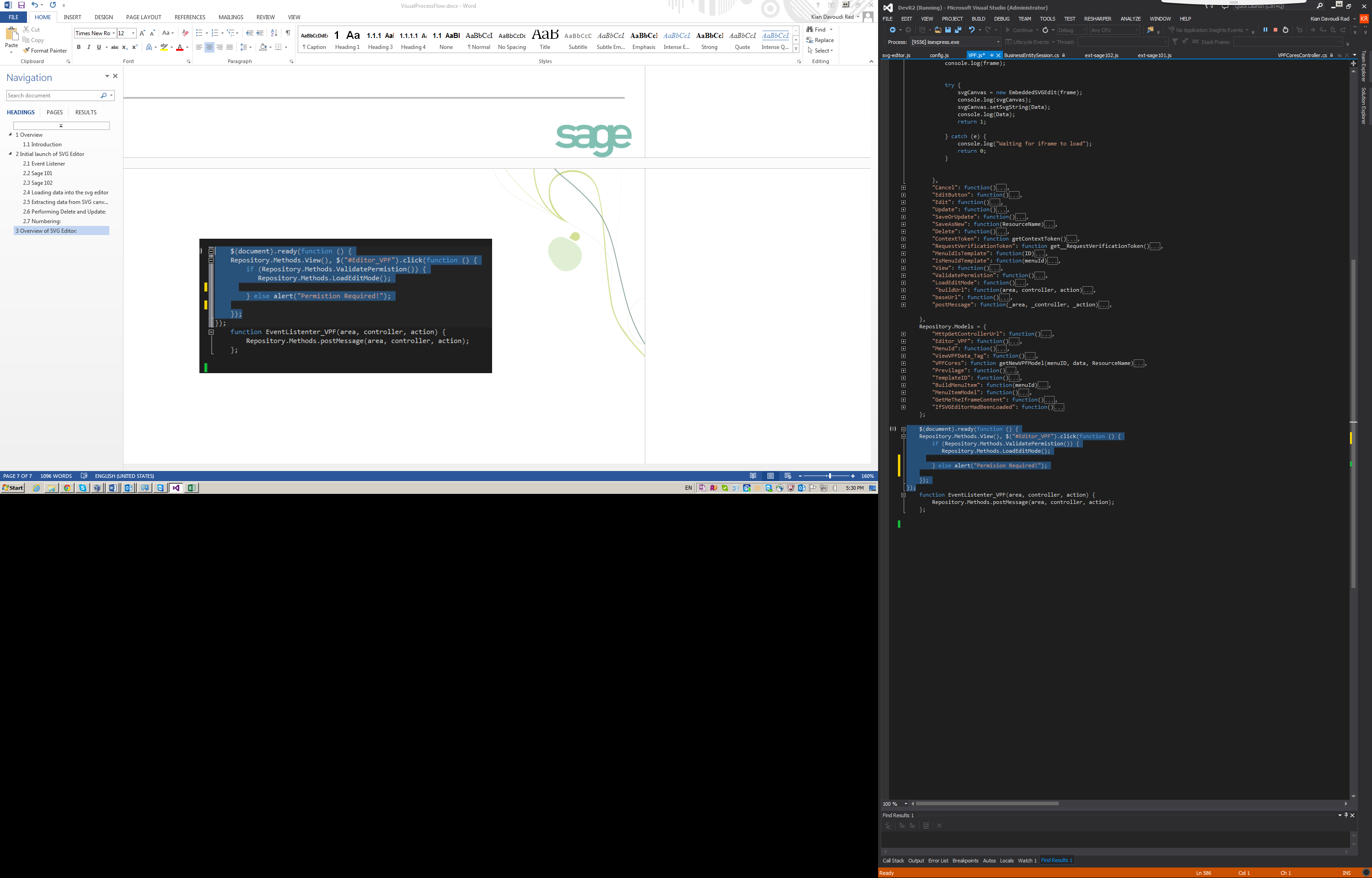


Figure 4:

## Privilege acquisition:

Another set of functionality, was also implemented to accommodate the user access privilege. During POC, one attribute was added to the data model, to embed the user Privilege to access a flow. Inspired by current implementation of home page, I used a tag holder within cs html. This html tag is being update by a Data model retrieved from the controller.

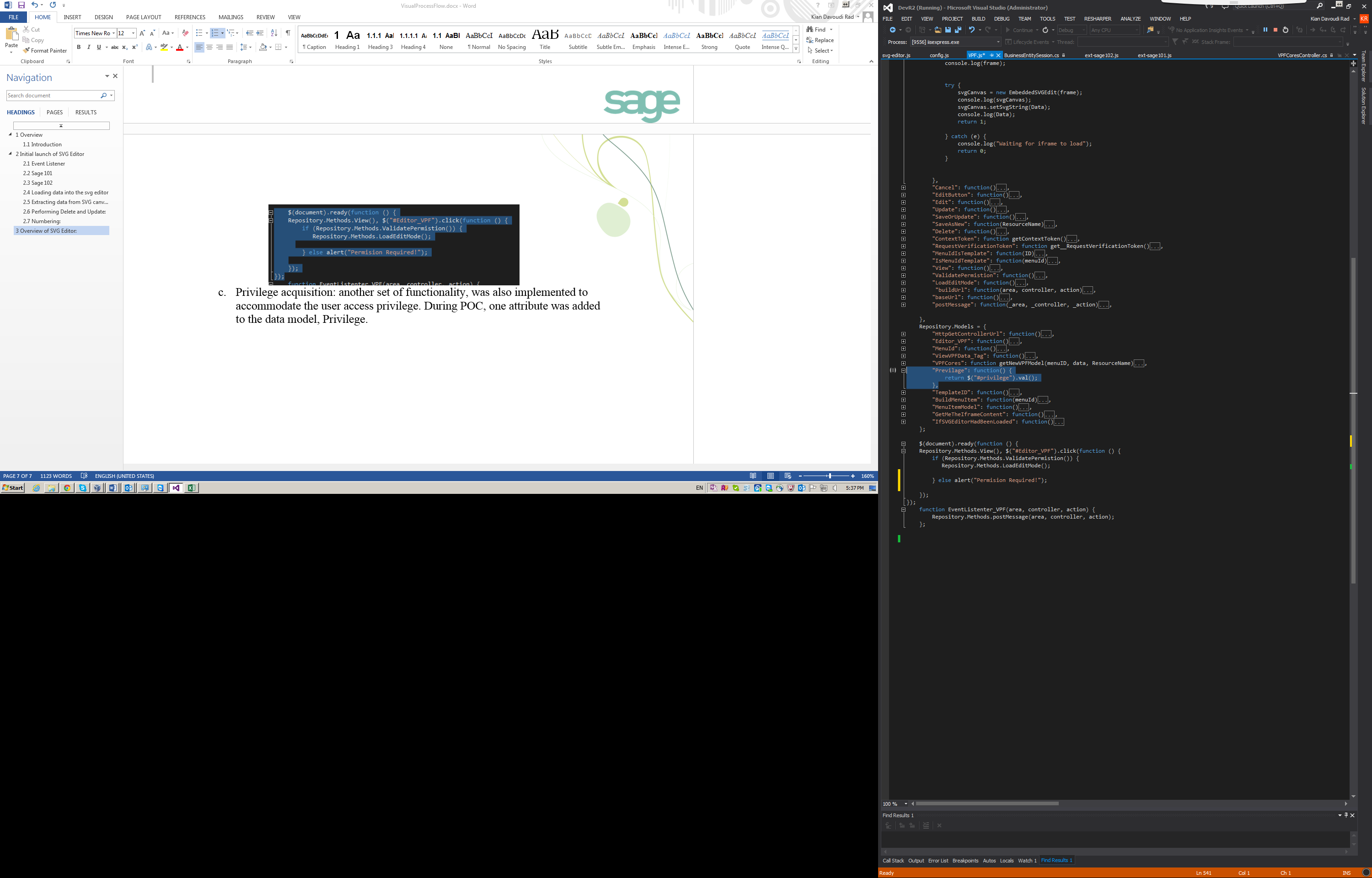
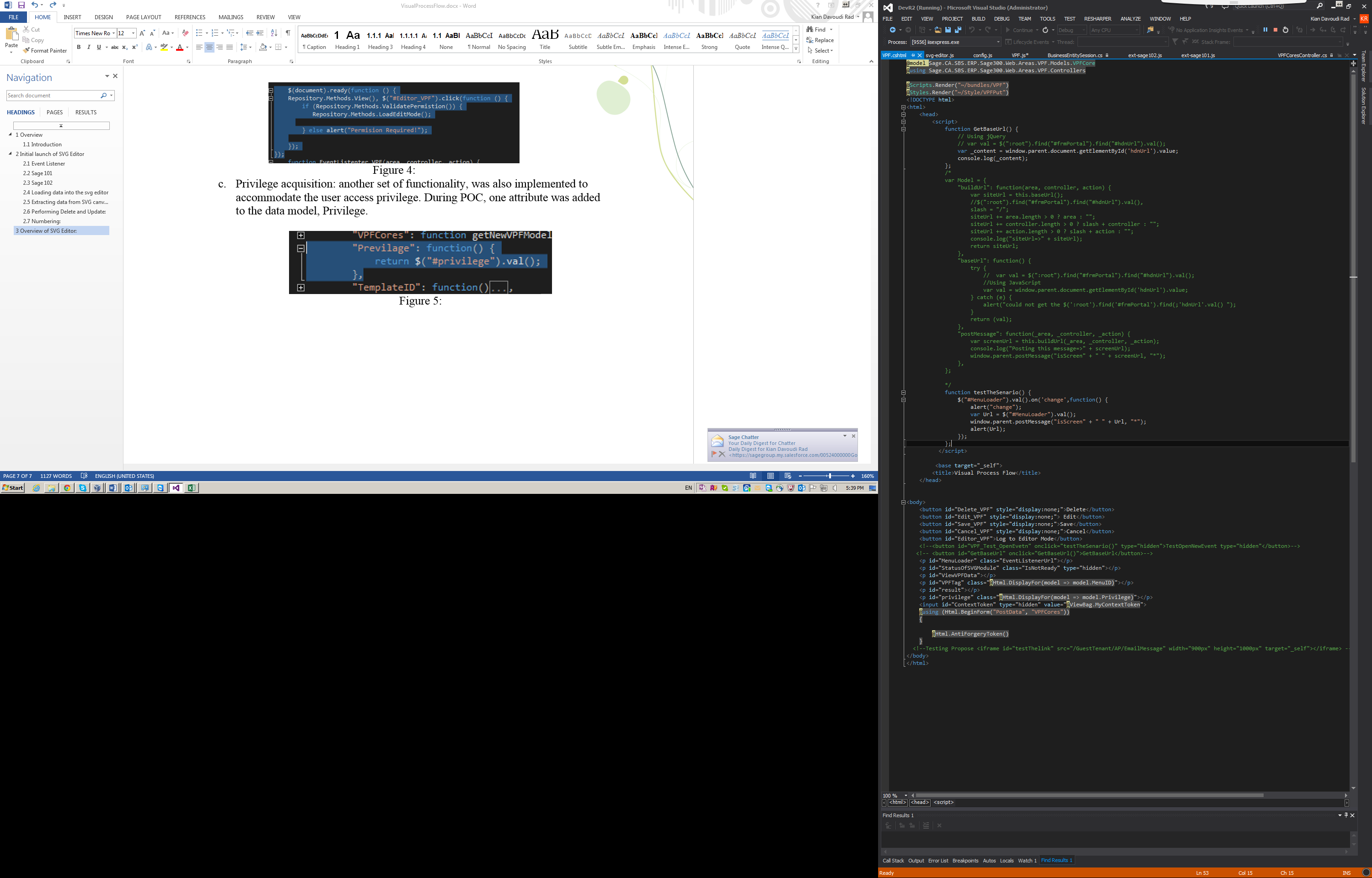
 

Figure 5:

## URL builders:

Since asynchronous calls have been used as primary tool to transfer data bisectionally between a controller and a view, there was a need to accommodate a url builder. In addition to the url builder, an other set of functions were also implemented to perform the actual ajax call and retrieve the data model back to the associated call. Next two figures demonstrate this implementation.

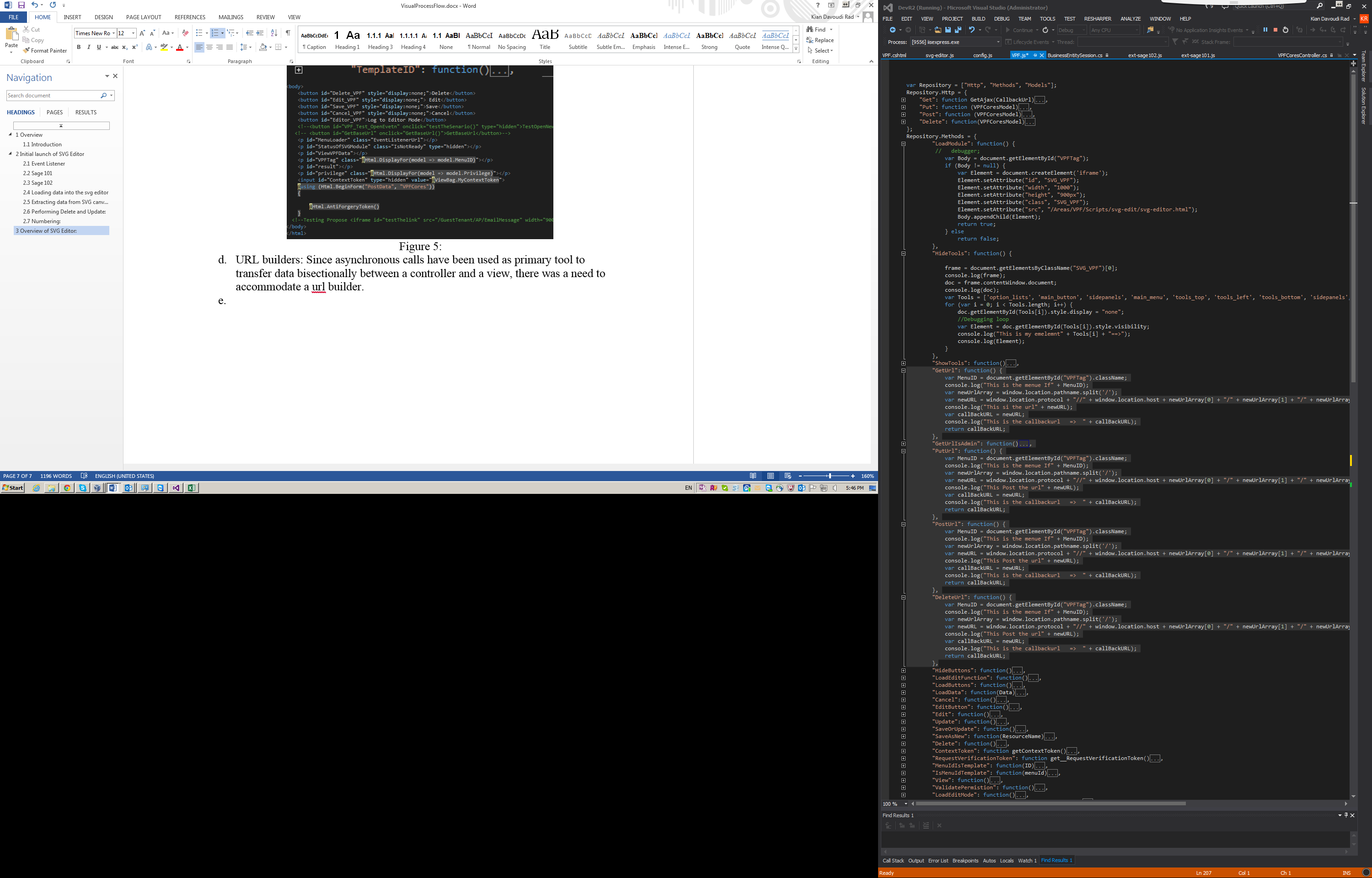


Figure 6: Demosterating the url builders

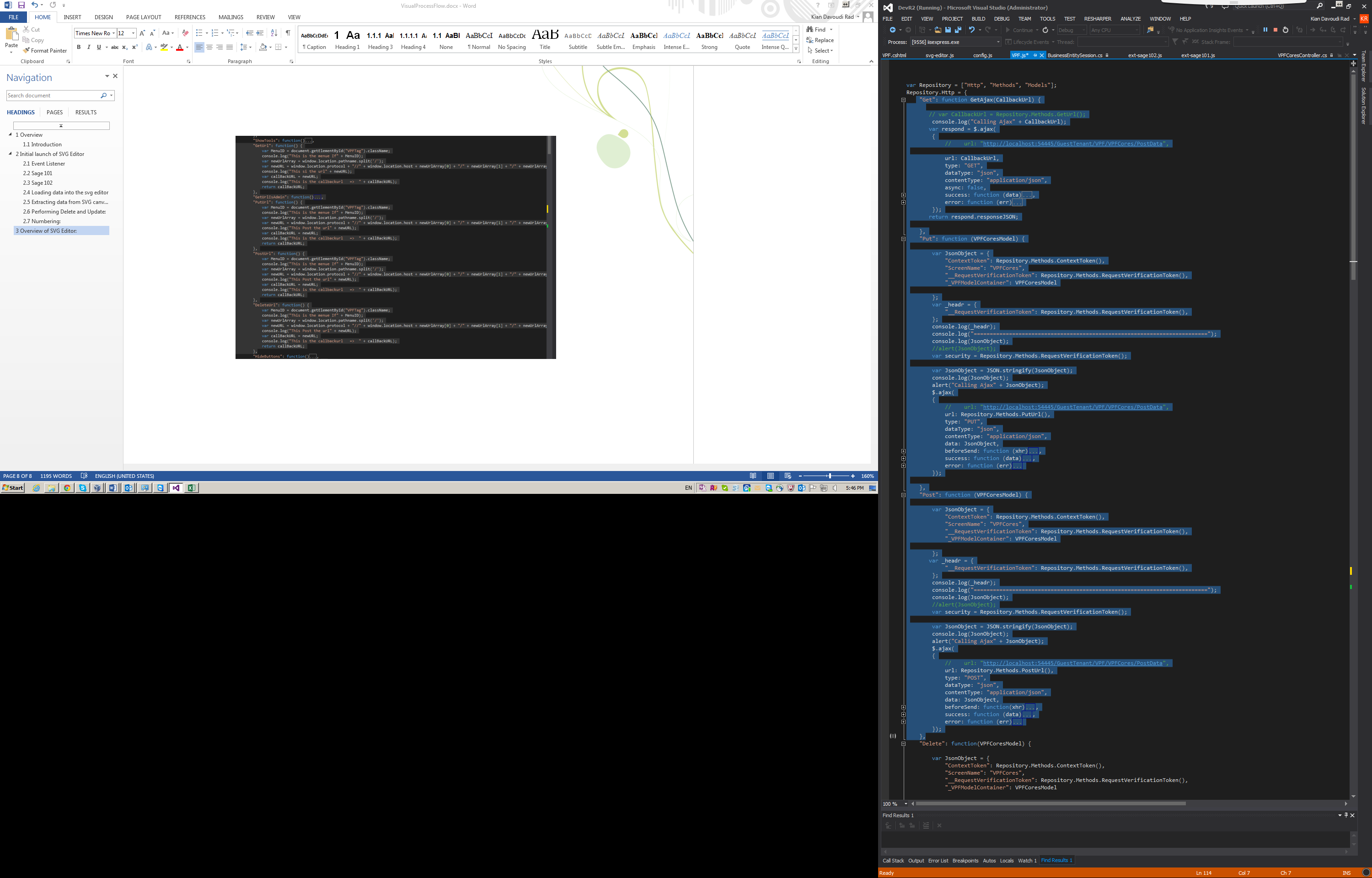


Figure 7: Demosteartes implementation of asyncronous calls