December 2015

UNISYS

Developing Agility

The Quarterly Newsletter for Unisys EAE and Agile Business Suite Clients

Contents

Application Development: It's All About Communication

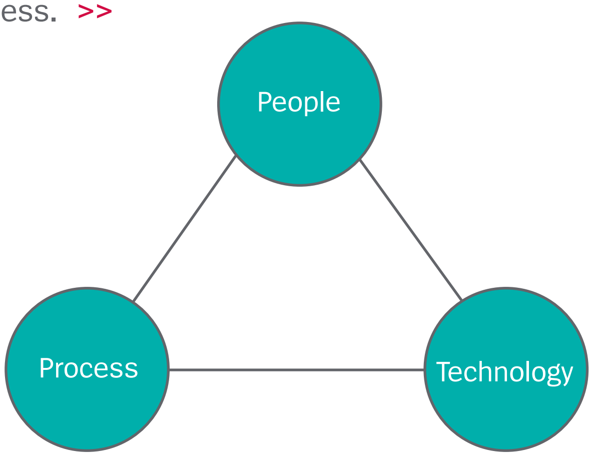
By Maarten Schneider, Worldwide Marketing Manager, EAE and AB Suite, Unisys

It's a software-driven world, one where we rely on a sea of apps to manage, coordinate, and support nearly every aspect of our lives.

In this world, established markets, like banks, airlines, hotels, and taxi services, are seeing their business models disrupted by agile newcomers boasting an "app first" mentality.

So, what qualities do these nimble startups have in common? In most cases, teams from the business and IT sides of the house work closely around a single, well-defined goal. And sometimes, business people are actually the ones who design and develop the apps.

Speed and innovation clearly power success in today's market. And to keep pace with this new breed of competition, established organizations must think and act like a startup. And to do so as effectively as possible, it's crucial to find an optimal way to coordinate people, technology, and process. >>



Not a subscriber of Developing Agility? Don't miss the next issue – subscribe today.
To see earlier issues of Developing Agility, please visit the archive.

People

A clear understanding of a project's requirements and open communication between the business and application development teams are integral ingredients for success. And while today's business leaders know much more about IT and application development than they did decades ago, there is still work to be done.

That's why organizations should consider teaching business people programming concepts and showing them how to articulate their application requirements through clear "user stories."

The same is true for application developers. The better educated they are about the businesses they support, the better equipped they'll be to roll out applications at a speed the organization demands.

Technology

Nothing changes faster than technology. While such disruptive trends as mobility, big data, and the Internet of Things all hold promise, the pressure to transform these new capabilities into business benefits still lies on our shoulders.

One of the reasons we designed Agile Business Suite (AB Suite™) in the first place was to help developers benefit from these and other innovations. The best example is of course the replacement of our Enterprise Application Environment (EAE) framework with Microsoft® Visual Studio®.

Another example is our recent release of the Client Framework, a DataModel based interface that enables developers to create UIs using their preferred technologies, like Windows Presentation Foundation (WPF) and Windows Communication Foundation (WCF).

And, the latest integration with additional elements in Microsoft Team Foundation Server (TFS), such as those for version control, test, build, and release management, opens doors to further optimize the application development process.

Process

The application development process should unite people and technology by documenting and streamlining the actions required to bring new capabilities into production. But, taking a process from idea to implementation means coordinating a potentially dizzying number of steps.

That's where Application Lifecycle Management (ALM) comes into play. ALM starts by defining business requirements and priorities before the project is transitioned to the development team.

It emphasizes consistent communication between business and development teams throughout the project, helping developers understand the project's timeline and if it's necessary to move from Waterfall development to an Agile approach.

ALM also advocates for open communication between developers and testers – the closer they work, the less time it should take to test a new release. In fact, more and more organizations are building test scripts during – or even before – the development phase.

The third important communication driven by ALM happens between developers and operations specialists, a concept known as DevOps. When operations knows when a new release has to get into production – and what special requirements must be addressed – they can get ready without unneeded delays.

>>

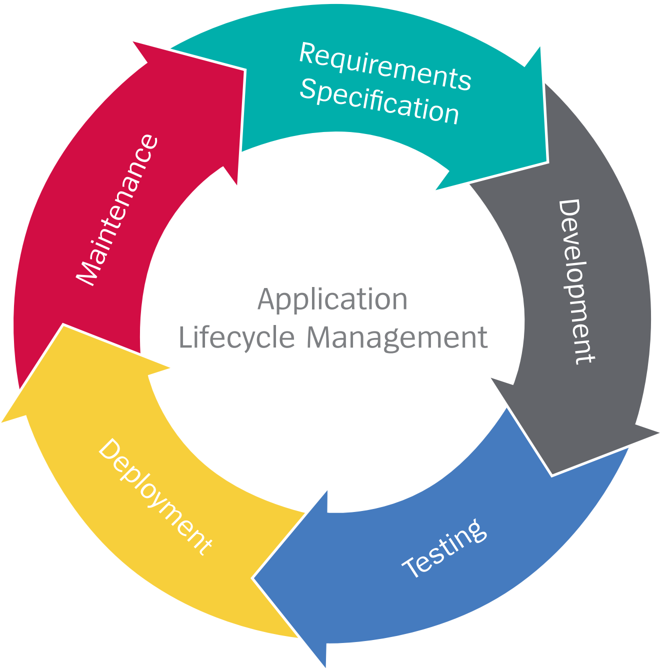
With the help of ALM, the application development process gains a sharpened focus on connecting people, which leads to better applications delivered at a faster pace.

AB Suite and TFS: A Strong Combination for ALM

The AB Suite team leverages the TFS package to support the communications processes – spanning idea, production, and maintenance – that support ALM. All stakeholders get a unified view of the same information, leading to a development process that is transparent to everyone involved.

For example, special dashboards enable line-of-business heads and project managers to see, at a high level, which projects are on track and which might be falling behind schedule. End users have an easy means of reporting wishes and bugs. Developers receive insight into all of the background information that's pertinent to their specific tasks. Project managers can monitor progress and re-route work items to keep the release on schedule. Testers can track their work and report results to the appropriate stakeholders. And, operations team members have the ability to see when a release is almost ready and prepare to get it into production.

With the implementation of AB Suite in combination with TFS, organizations can embrace ALM and make it a guiding principle that results in fast cycles and a transparent development process.



Pushing ClearPath – and AB Suite – Forward

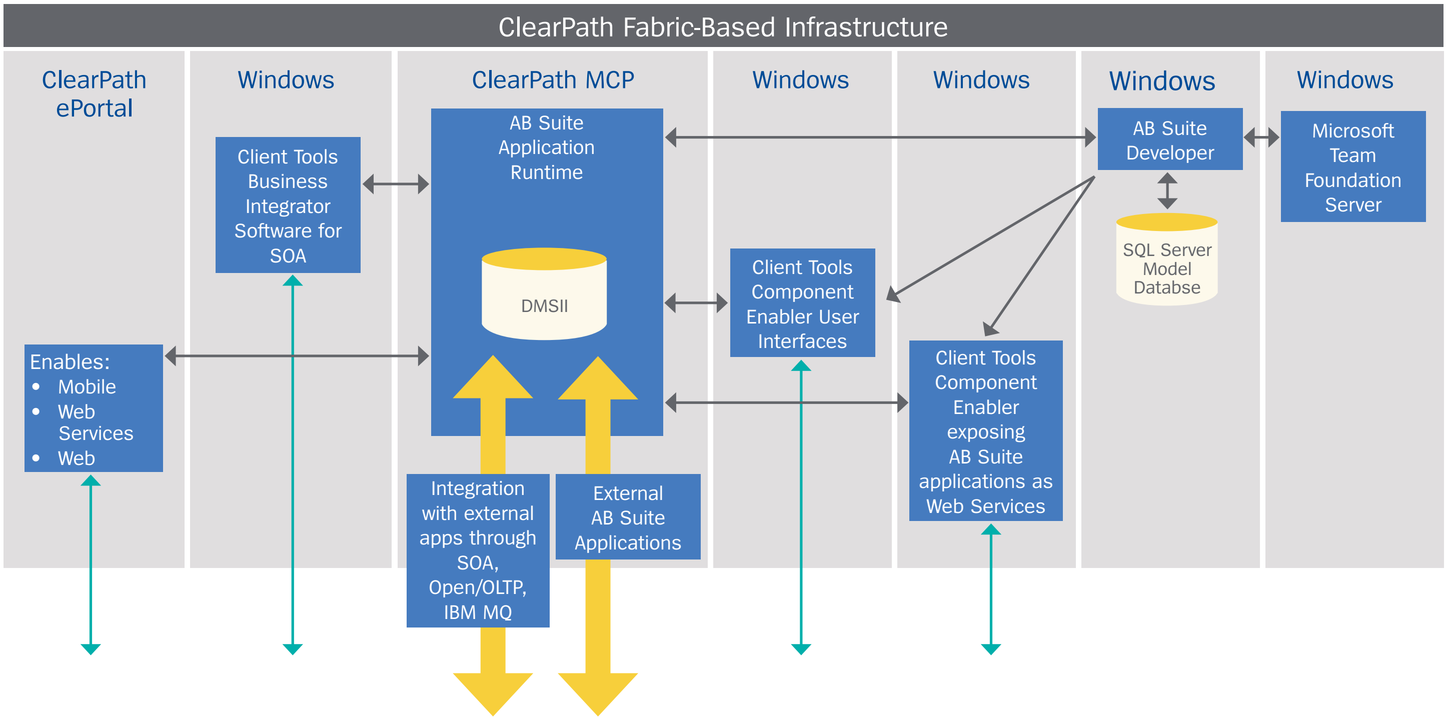
Welcome to the era of ClearPath Forward™.

In a move designed to unite the best of two worlds into one, ClearPath Forward blends established ClearPath® strengths and new innovations with the engineering ingenuity behind the *Forward!* by Unisys™ enterprise computing platform.

As a result of this change, what we had referred to as "the fabric" will now be known as ClearPath Forward. But this is much more than a simple name change: with ClearPath Forward, we're extending the capabilities of the fabric-based infrastructure with new features designed to add more value to the ClearPath environment.

And in doing so, we've strengthened the integration between the ClearPath MCP and Microsoft Windows® environments, making it possible for the multiple elements within the AB Suite application landscape – the core of which runs on MCP, while other aspects run on Windows – to function much more elegantly and provide the advanced capabilities your organization needs to excel.

>>


ClearPath Fabric-Based Infrastructure
ClearPath ePortal
Windows
ClearPath MCP
Windows
Windows
Windows
Windows
Client Tools Business Integrator Software for SOA
AB Suite Application Runtime
DMSII
AB Suite Developer
Microsoft Team Foundation Server
SQL Server Model Databse
Client Tools Component Enabler User Interfaces
Enables:
• Mobile
• Web Services
• Web
Client Tools Component Enabler exposing AB Suite applications as Web Services
Integration with external apps through SOA, Open/OLTP, IBM MQ
External AB Suite Applications

A More Secure Fabric

With the secure fabrics included in ClearPath Forward, it's easier to share data in near real time between ClearPath systems and Windows servers, leading to better agility, responsiveness, and time to value.

Secure fabrics make this possible by allowing you to segment and isolate traffic over the environment's high-speed interconnect, which is currently 56Gb Sixteen Data Rate (SDR) InfiniBand. This creates dedicated data lanes that subdivide quality of service and deliver predictable, secure communication for both virtualized and partitioned workloads.

Virtualization, Extended

ClearPath Forward streamlines virtual environments and helps to boost workload density by offering a plug-in that integrates with VMware® vCenter™ and enables you to manage, stop, and start Windows and Linux® partitions directly from vCenter. You get a streamlined, single-pane-of-glass management and user experience – as well as the ability to view the status of these partitions right in the vCenter portal, helping you unify management across environments.

Additionally, the SUSE and Red Hat® Enterprise Linux® operating systems supported by ClearPath Forward now allow Linux containers to be implemented using Docker® software. Blueprints and gold images for the supported Linux operating systems include the code modules that make them ready to support Docker containers.

Increased Flexibility

In addition to delivering increased control, greater predictability, and simplified management, ClearPath Forward significantly expands on the high levels of flexibility already available in the ClearPath environment.

In addition to your host MCP environment, you now have the option to divide each system's Enterprise Partitionable Platform (EPP) into a maximum of 36 unique environments, allowing all of the many Windows elements required to create an AB Suite application to be deployed within the ClearPath Forward environment.

Increasing the number of available partitions also gives you more ways to utilize the deliverables offered under the ClearPath Software Series umbrella. For instance, you could deploy multiple instances of ClearPath MCP Developer Studio and use these unique development environments to stage different MCP releases, test application updates, or provide individual development resources to your team.

So now, you have the freedom to create as many MCP development environments as you have EPPs. One could include an AB Suite 4.0 solution in MCP 16.0. Right next to it could be a test partition for the same solution, but on AB Suite 5.0 running in the MCP 17.0 environment. Numerous other combinations, like EPPs for testing web services or a mobility solution that leverages the ClearPath ePortal for MCP EPP, are possible, as well.

A Big Step for AB Suite

The elements we've built into ClearPath Forward – including the high-speed interconnect, secure fabrics, expanded EPP size, and compatibility with ClearPath MCP Developer Studio, to name a few – all play an integral role in taking your AB Suite development, test, and Runtime environments to the next level.

For example, when you pair ClearPath Forward with MCP 17.0, you can dramatically accelerate the AB Suite Build process, cutting the time it needs to execute by as much as half. And by moving your AB Suite solution into the ClearPath Forward environment, you'll gain tighter control over the entire software channel, so it's easier to manage application availability SLAs.

But this is just the start. We designed ClearPath Forward with the express purpose of redefining what's possible in a ClearPath system. And as we add more to the environment, your AB Suite applications will be in a prime position to reap the rewards.

Q&A: Job Service North Dakota on EAE and Microsoft Team Foundation Server

We ended engineering-level support for the legacy, UREP-based EAE version control software on June 30, 2015. In place of the tool, we're offering an MSSCCI API interface to Microsoft Team Foundation Server (TFS) to support version control activities in the EAE environment.

One of the first EAE users to make the switch to TFS was Job Service North Dakota. An organization dedicated to providing services that address North Dakota's current and emerging workforce needs, Job Service North Dakota helps residents find suitable, sustainable employment, works with businesses to recruit qualified candidates, and administers the state's unemployment benefits program.

We recently spoke with Brian Heitkamp, Database Administrator at Job Service North Dakota, about the transition to TFS and how the new tool has impacted the organization's version control activities and overall development process.

Developing *Agility*: Please give us a brief description of your development environment.

Brian Heitkamp: We're running EAE Developer 3.3.3412 and EAE 3.3.3413 for the ClearPath MCP Runtime. We have three code repositories. One allows our developers to create and modify objects. One is a User Acceptance Testing (UAT) environment. And the third is a production repository that contains the code running in our production environment.

Our development environment has three gen sets defined and three different generated runtime environments – the code is the same in each, but the data is different. Our UAT environment also has three gen sets and three generated runtime environments, once again with the same code but different data. Because we use Remote Database Backup (RDB), the production environment has three gen sets to support the RDB environment.

DA: How many developers do you have? What's the size of your model?

BH: We have eight developers currently. A full extract of our model file is 203 MB, and at TFS conversion time, it contained 14,147 versionable objects.

DA: When did you start using TFS with EAE?

BH: We began using TFS in August of 2015.

DA: Tell us a little about how your developers reacted to TFS. What's been the biggest adjustment for them?

BH: The main difference was learning how to do things they were familiar with when using the UREP-based version control product. For example, adding an object now requires them to do so from the business segment. An object must now be checked out before they can copy it. And there's a bit of a difference in how the GO command works. These are some minor issues that will be addressed by Unisys soon.

>>

DA: Overall, do you feel that you had to make big adjustments in your environment, as well as your approach to source control and releasing changes?

BH: We did have to make adjustments in our process, but nothing big. Our release process had the most changes because we made extensive use of the UREP labeling feature, which didn't translate well to TFS branching.

DA: What aspects of TFS have your developers found particularly useful?

BH: Because we use TFS branches to contain our development, UAT, and production code, our developers can see what versions of objects are in which code repositories. They can then use TFS to compare the versions to identify the differences in each.

DA: Looking back, what was the biggest challenge in the process of moving from the UREP-based version control product to TFS?

BH: We ran into some issues around getting ICs installed. We also had some trouble when we did our container operations as part of the process to migrate our existing EAE models into TFS, but Unisys was quick to provide a workaround for that.

DA: What recommendations would you offer to sites that are just now beginning their move to TFS?

BH: It's a good idea to download and explore TFS first, so your developers have a chance to play with it before going live. Maybe even set up a test environment that uses TFS.

Participating in the Unisys hosted three-day Version Control Workshop was a worthwhile exercise, as well.

It's also important to understand and document the processes and procedures that you use in the UREP world, so you can quickly and easily convert them to the TFS world. Along those same lines, be sure to practice the steps needed to move to TFS. And don't wait too long after the workshop to execute the migration.

Thanks again to Brian for taking the time to speak with us. If you're curious about how TFS can benefit your EAE environment, please email us for additional information.

Using the Client Framework to Develop Lightweight Front Ends

*By Joaquin Zschuschen, AB Suite Consultant, Unisys Netherlands
and Nigel Tunnicliffe, Consultant, Unisys UK*

The Agile Business Suite Client Framework creates a number of new possibilities for developing modern front-end applications. Windows Presentation Foundation (WPF), Microsoft Model-View-Controller (MVC), and Windows Communication Foundation (WCF) are supported out of the box, making it a suitable complement to most modern development practices.

In the web and mobile application development worlds, JavaScript frameworks like jQuery, Knockout, Angular, and Microsoft WinJS are often used to create applications that run completely on the client, minimizing the requirements placed on front-end servers without demanding extra installations. These frameworks have the ability to use Model View ViewModel (MVVM) patterns and data-bind the results of a web service call to deliver highly responsive and interactive web sites or mobile applications.

So, how can the Client Framework make use of these JavaScript frameworks?

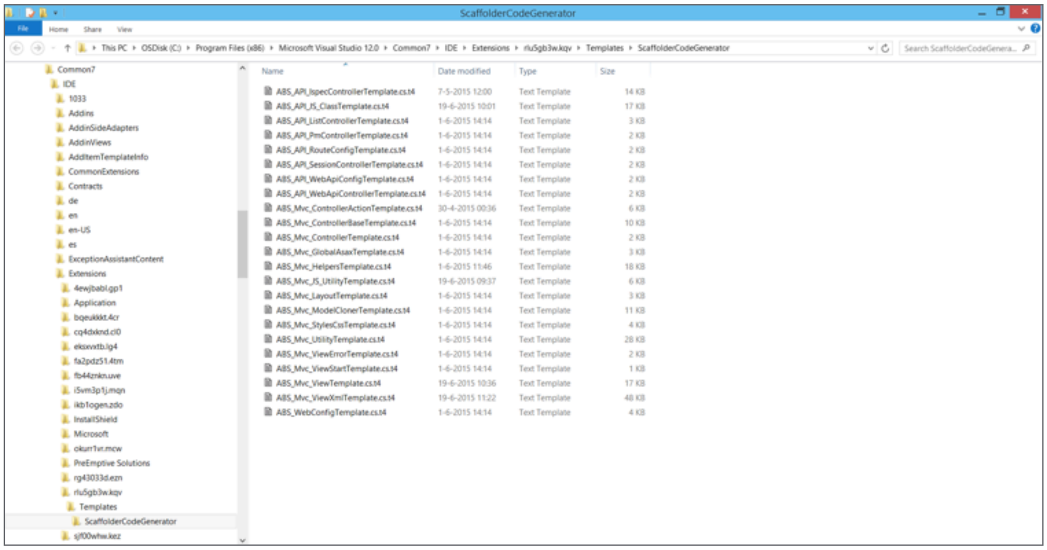
The Client Framework and JavaScript

MVC-generated C# models from the Client Framework use technology called T4 code generation to scaffold controllers and views. With T4, you code a template that generates code. So, when integrating JavaScript into the Client Framework, you have to develop unique T4 templates for each JavaScript framework. And if needed, specific requirements from clients can be integrated by customizing the T4 templates. In this example, we will be customizing the MVC templates to create a JavaScript web application using a REST service and Knockout data-binding. We've used bootstrap for the GUI in order to make the web site responsive to various screen sizes.

Note that depending on where you installed Visual Studio, the T4 templates may be found under "extensions," such as: C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\Extensions\.

There, under a generated id you will find \Templates\ScaffolderCodeGenerator.

>>



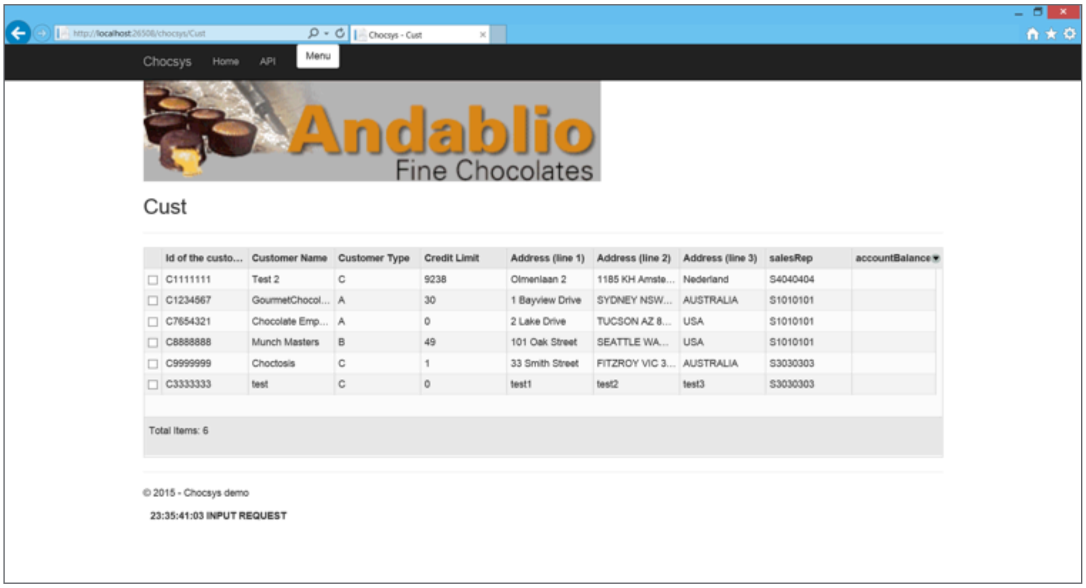
This PC ▸ OSDisk (C:) ▸ Program Files (x86) ▸ Microsoft Visual Studio 12.0 ▸ Common7 ▸ IDE ▸ Extensions ▸ rlu5gb3w.kqv ▸ Templates ▸ ScaffolderCodeGenerator

```
Common7
  IDE
    1033
    Addins
    AddinSideAdapters
    AddinViews
    AddItemTemplateInfo
    CommonExtensions
    Contracts
    de
    en
    en-US
    es
    ExceptionAssistantContent
    Extensions
      4ewjbabl.gp1
      Application
      bqeukkkt.4cr
      cq4dxknd.cl0
      eksxvxtb.lg4
      fa2pdz51.4tm
      fb44znkn.uve
      i5vm3p1j.mqn
      ikb1ogen.zdo
      InstallShield
      Microsoft
      okurr1vr.mcw
      PreEmptive Solutions
      rg43033d.ezn
      rlu5gb3w.kqv
        Templates
          ScaffolderCodeGenerator
      sjf00whw.kez
```

Name	Date modified	Type	Size
ABS_API_IspecControllerTemplate.cs.t4	7-5-2015 12:00	Text Template	14 KB
ABS_API_JS_ClassTemplate.cs.t4	19-6-2015 10:01	Text Template	17 KB
ABS_API_ListControllerTemplate.cs.t4	1-6-2015 14:14	Text Template	3 KB
ABS_API_PmControllerTemplate.cs.t4	1-6-2015 14:14	Text Template	2 KB
ABS_API_RouteConfigTemplate.cs.t4	1-6-2015 14:14	Text Template	2 KB
ABS_API_SessionControllerTemplate.cs.t4	1-6-2015 14:14	Text Template	2 KB
ABS_API_WebApiConfigTemplate.cs.t4	1-6-2015 14:14	Text Template	2 KB
ABS_API_WebApiControllerTemplate.cs.t4	1-6-2015 14:14	Text Template	2 KB
ABS_Mvc_ControllerActionTemplate.cs.t4	30-4-2015 00:36	Text Template	6 KB
ABS_Mvc_ControllerBaseTemplate.cs.t4	1-6-2015 14:14	Text Template	10 KB
ABS_Mvc_ControllerTemplate.cs.t4	1-6-2015 14:14	Text Template	2 KB
ABS_Mvc_GlobalAsaxTemplate.cs.t4	1-6-2015 14:14	Text Template	3 KB
ABS_Mvc_HelpersTemplate.cs.t4	1-6-2015 11:46	Text Template	18 KB
ABS_Mvc_JS_UtilityTemplate.cs.t4	19-6-2015 09:37	Text Template	6 KB
ABS_Mvc_LayoutTemplate.cs.t4	1-6-2015 14:14	Text Template	3 KB
ABS_Mvc_ModelClonerTemplate.cs.t4	1-6-2015 14:14	Text Template	11 KB
ABS_Mvc_StylesCssTemplate.cs.t4	1-6-2015 14:14	Text Template	4 KB
ABS_Mvc_UtilityTemplate.cs.t4	1-6-2015 14:14	Text Template	28 KB
ABS_Mvc_ViewErrorTemplate.cs.t4	1-6-2015 14:14	Text Template	2 KB
ABS_Mvc_ViewStartTemplate.cs.t4	1-6-2015 14:14	Text Template	1 KB
ABS_Mvc_ViewTemplate.cs.t4	19-6-2015 10:36	Text Template	17 KB
ABS_Mvc_ViewXmlTemplate.cs.t4	19-6-2015 11:22	Text Template	48 KB
ABS_WebConfigTemplate.cs.t4	1-6-2015 14:14	Text Template	4 KB

Knockout Demo

For this article, we adjusted the Chocsys application from the AB Suite student guide by creating a master-detail view for every Ispec. For example, it displays a list of “Cust” records in a grid and opens a detail view whenever a specific record is selected. We used the koGrid, a Knockout add-in that provides client-side filtering and column selection without the need to rebind and collect new data, to add a nice, responsive feel to the application.



Cust

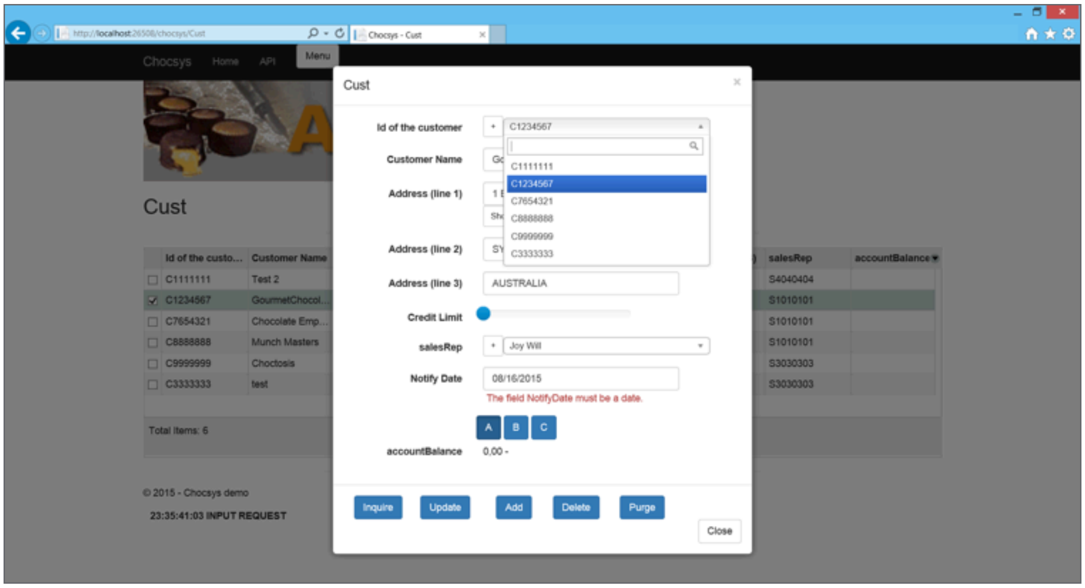
	Id of the custo...	Customer Name	Customer Type	Credit Limit	Address (line 1)	Address (line 2)	Address (line 3)	salesRep	accountBalance
□	C1111111	Test 2	C	9238	Olmenlaan 2	1185 KH Amste...	Nederland	S4040404	
□	C1234567	GourmetChocol...	A	30	1 Bayview Drive	SYDNEY NSW...	AUSTRALIA	S1010101	
□	C7654321	Chocolate Emp...	A	0	2 Lake Drive	TUCSON AZ 8...	USA	S1010101	
□	C8888888	Munch Masters	B	49	101 Oak Street	SEATTLE WA...	USA	S1010101	
□	C9999999	Choctosis	C	1	33 Smith Street	FITZROY VIC 3...	AUSTRALIA	S3030303	
□	C3333333	test	C	0	test1	test2	test3	S3030303	

Total Items: 6

© 2015 - Chocsys demo

23:35:41:03 INPUT REQUEST

Selecting a specific individual opens a popup showing the detailed contents. The GUI elements include a widget for the dropdown that accommodates the listbox elements from AB Suite, as well as a slider for the numeric fields. In addition, client-side validation is included. Remember: these screens are completely generated by the T4 templates, so once they are developed no more coding is needed, only generation. >>



Cust

Id of the customer: C1234567
C1111111
C1234567
C7654321
C8888888
C9999999
C3333333

Customer Name: Go

Address (line 1): 1 E

Sh

Address (line 2): SY

Address (line 3): AUSTRALIA

Credit Limit

salesRep: Joy Will

Notify Date: 08/16/2015

The field NotifyDate must be a date.

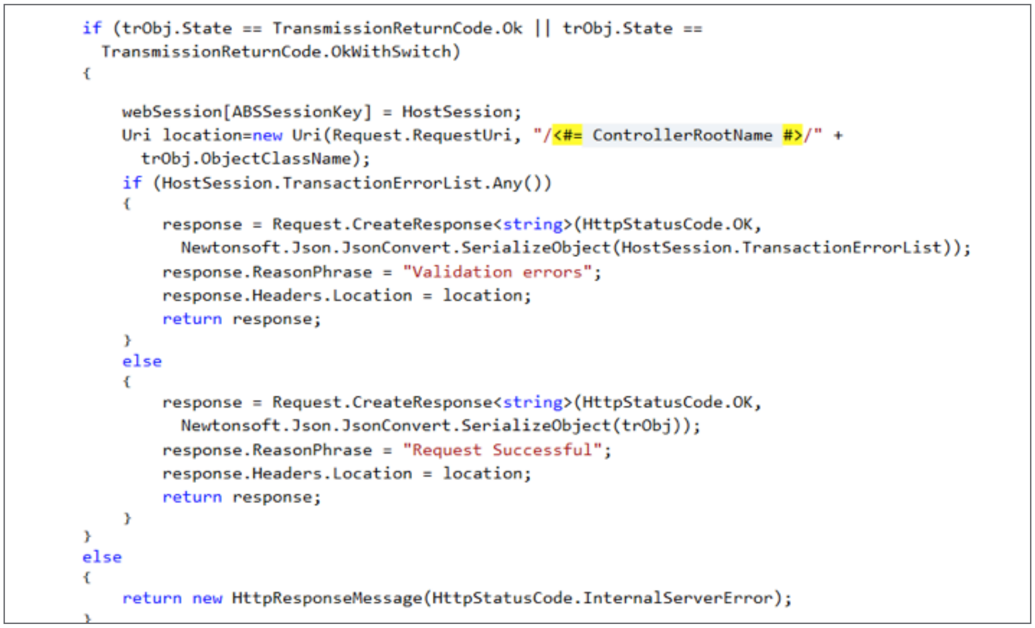
A B C

accountBalance 0,00 -

T4 Templating

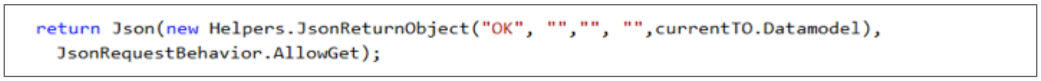
As noted earlier, some customization must be done to the T4 templates when generating an application. In this case, only a few templates require customization.

As Knockout has the ability to data-bind the results of a REST service, only a small change to the controller is necessary to accommodate validation by the binding engine.



```
if (trObj.State == TransmissionReturnCode.Ok || trObj.State ==
  TransmissionReturnCode.OkWithSwitch)
{
    webSession[ABSSessionKey] = HostSession;
    Uri location=new Uri(Request.RequestUri, "/<#= ControllerRootName #>/" +
      trObj.ObjectClassName);
    if (HostSession.TransactionErrorList.Any())
    {
        response = Request.CreateResponse<string>(HttpStatusCode.OK,
          Newtonsoft.Json.JsonConvert.SerializeObject(HostSession.TransactionErrorList));
        response.ReasonPhrase = "Validation errors";
        response.Headers.Location = location;
        return response;
    }
    else
    {
        response = Request.CreateResponse<string>(HttpStatusCode.OK,
          Newtonsoft.Json.JsonConvert.SerializeObject(trObj));
        response.ReasonPhrase = "Request Successful";
        response.Headers.Location = location;
        return response;
    }
}
else
{
    return new HttpResponseMessage(HttpStatusCode.InternalServerError);
}
```

In order for Knockout to process data from the service, it is modified within the controller and sent back as Json data.



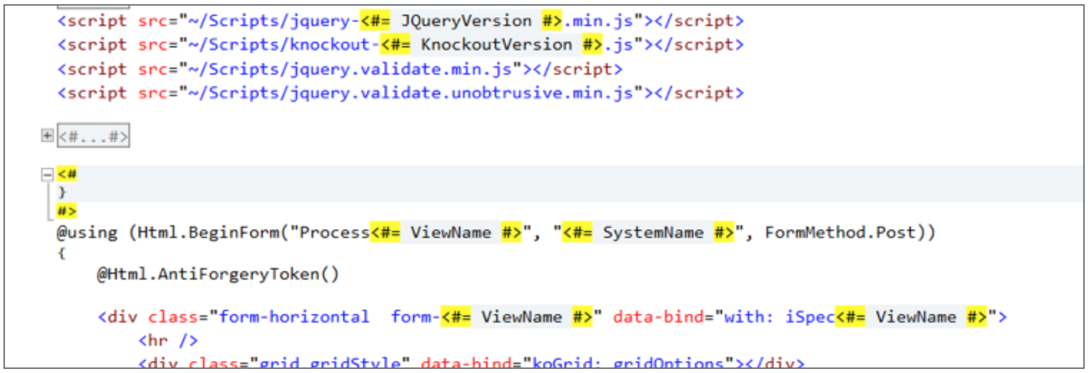
```
return Json(new Helpers.JsonReturnObject("OK", "","", "",currentTO.Datamodel),
  JsonRequestBehavior.AllowGet);
```

Some extra data items are also included to support the processing of service calls.

The majority of the changes occur on the view and JavaScript model. We'll examine them in the remainder of the article. >>

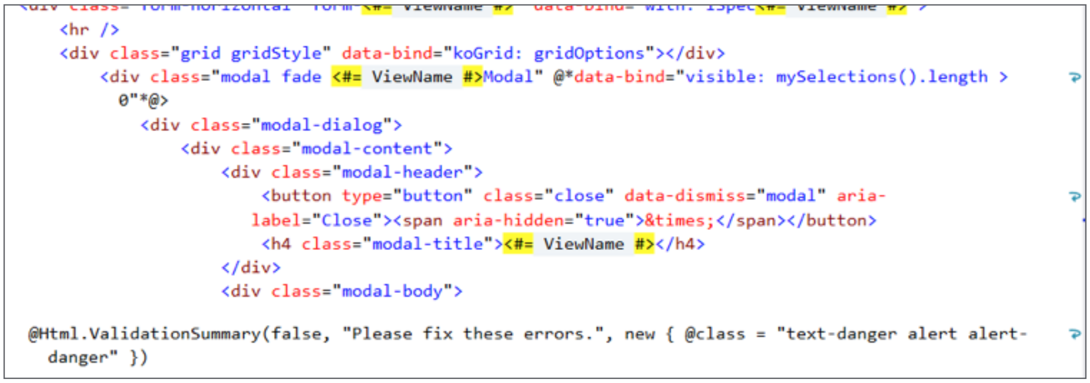
MVVM View (HTML)

The MVVM pattern decouples the view from the logic and the data. However, the ViewModel binds the view to the data, meaning the T4 template for the view and JavaScript must be adjusted to fit the needs of Knockout. To adjust the view, we must first include references to the JavaScript files and then bind the form to the JavaScript model using “with:” binding.



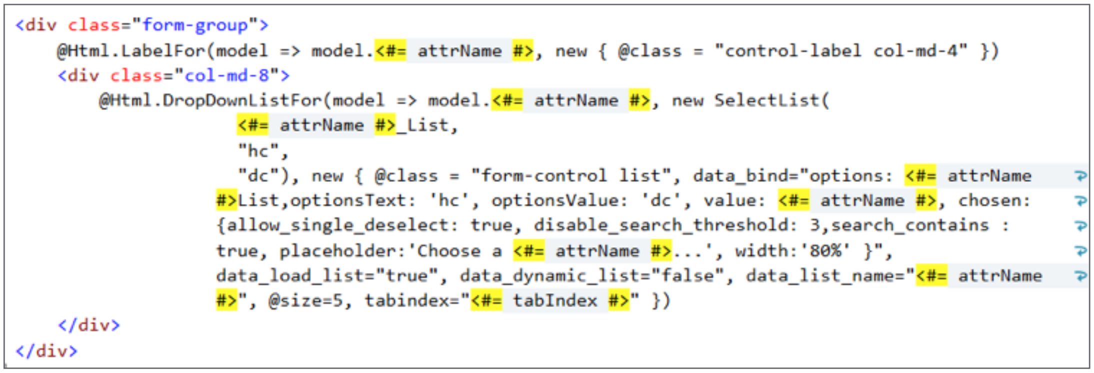
```
<script src="~/Scripts/jquery-<#= JQueryVersion #>.min.js"></script>
<script src="~/Scripts/knockout-<#= KnockoutVersion #>.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

<#...#>

<#
}
#>
@using (Html.BeginForm("Process<#= ViewName #>", "<#= SystemName #>", FormMethod.Post))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal  form-<#= ViewName #>" data-bind="with: iSpec<#= ViewName #>">
        <hr />
        <div class="grid gridStyle" data-bind="koGrid: gridOptions"></div>
```

Since we’re using a master-detail view, we’re able to create a grid with the popup for the detail view included via standard bootstrap functionality. >>



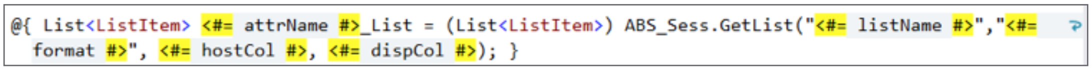
```
<div class="form-horizontal  form-<#= ViewName #>" data-bind="with: iSpec<#= ViewName #>">
    <hr />
    <div class="grid gridStyle" data-bind="koGrid: gridOptions"></div>
        <div class="modal fade <#= ViewName #>Modal" @*data-bind="visible: mySelections().length >
          0"*@>
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-
                       label="Close"><span aria-hidden="true">&times;</span></button>
                        <h4 class="modal-title"><#= ViewName #></h4>
                    </div>
                    <div class="modal-body">

 @Html.ValidationSummary(false, "Please fix these errors.", new { @class = "text-danger alert alert-
   danger" })
```

In the details view, we loop over the properties of the model generated from the Client Framework. And just like with MVC, the proper HTML elements are generated – many through HTML helpers with attributes added for bootstrap. That said, a different HTML input element is created depending on the property. For instance, a list property – one that reads from the XML metadata file generated during a Client Framework conversion – will be transformed into a dropdown list.



```
<div class="form-group">
    @Html.LabelFor(model => model.<#= attrName #>, new { @class = "control-label col-md-4" })
    <div class="col-md-8">
        @Html.DropDownListFor(model => model.<#= attrName #>, new SelectList(
                     <#= attrName #>_List,
                     "hc",
                     "dc"), new { @class = "form-control list", data_bind="options: <#= attrName 
                   #>List,optionsText: 'hc', optionsValue: 'dc', value: <#= attrName #>, chosen: 
                   {allow_single_deselect: true, disable_search_threshold: 3,search_contains : 
                   true, placeholder:'Choose a <#= attrName #>...', width:'80%' }",
                   data_load_list="true", data_dynamic_list="false", data_list_name="<#= attrName 
                   #>", @size=5, tabindex="<#= tabIndex #>" })
    </div>
</div>
```

To enable the dropdown to work as intended, a list generated by the new Glb.List element introduced in the Client Framework is used for the auto-complete function.



```
@{ List<ListItem> <#= attrName #>_List = (List<ListItem>) ABS_Sess.GetList("<#= listName #>","<#= 
  format #>", <#= hostCol #>, <#= dispCol #>); }
```

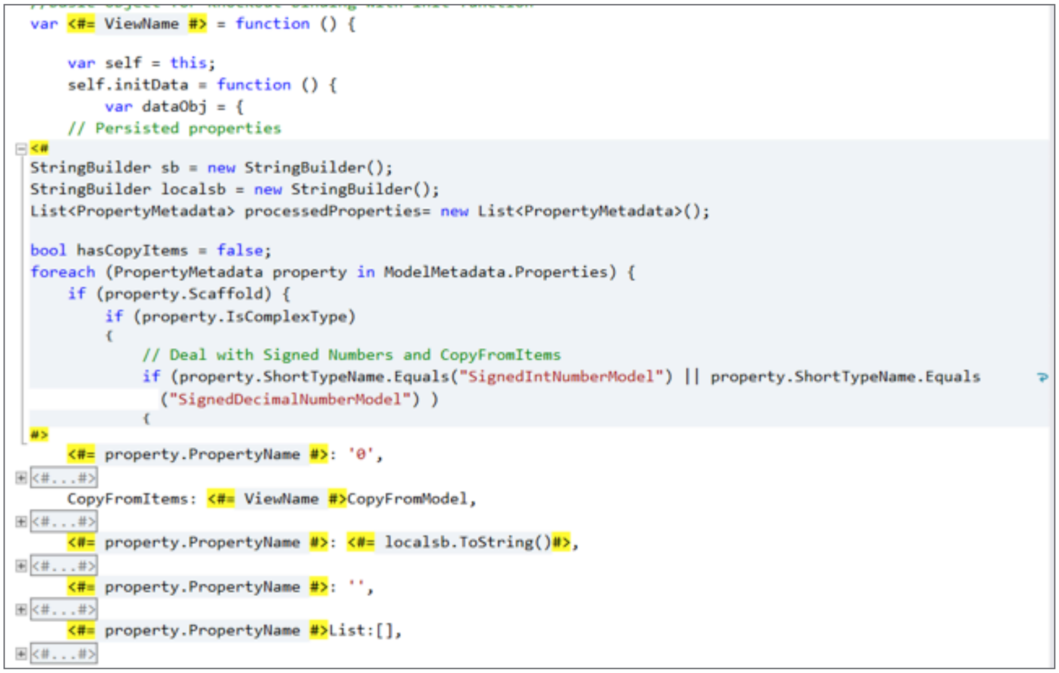
In addition, we performed some extra JavaScript work to glue the dropdown to the list REST call.

The grid itself does not have too many extra elements in the view because we will be using the binding engine from Knockout. The columns are read from the provided JavaScript model.

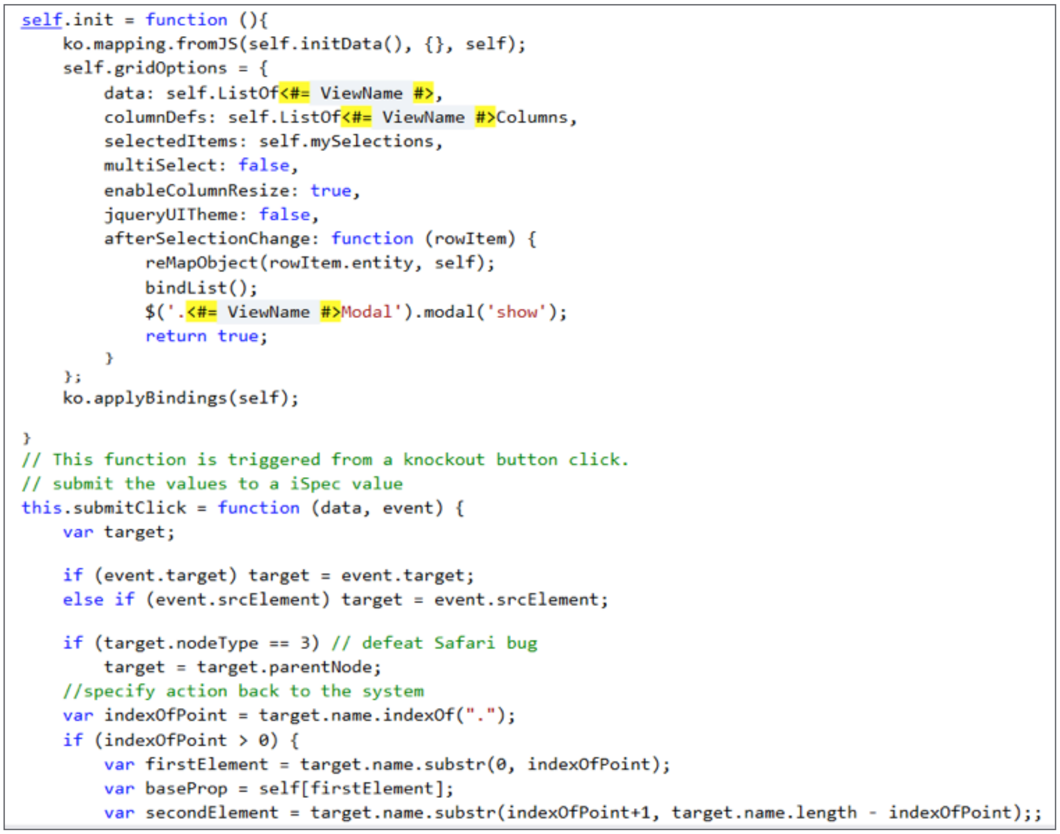
That's the sum total of the changes made to the view itself. This view is also the location to make further adjustments, should you need to address individual client requirements. >>

MVVM ViewModel (JavaScript)

As it serves as the glue between the Model (data) and the View, the ViewModel handles the majority of the processing in the MVVM. It includes bindable properties, as well as some basic Knockout specific methods. For example, the object is initialized via self.initData. And, it is connected to the REST service using a submit handler that is present with the callback to bind the result to the model (this.submitClick). >>

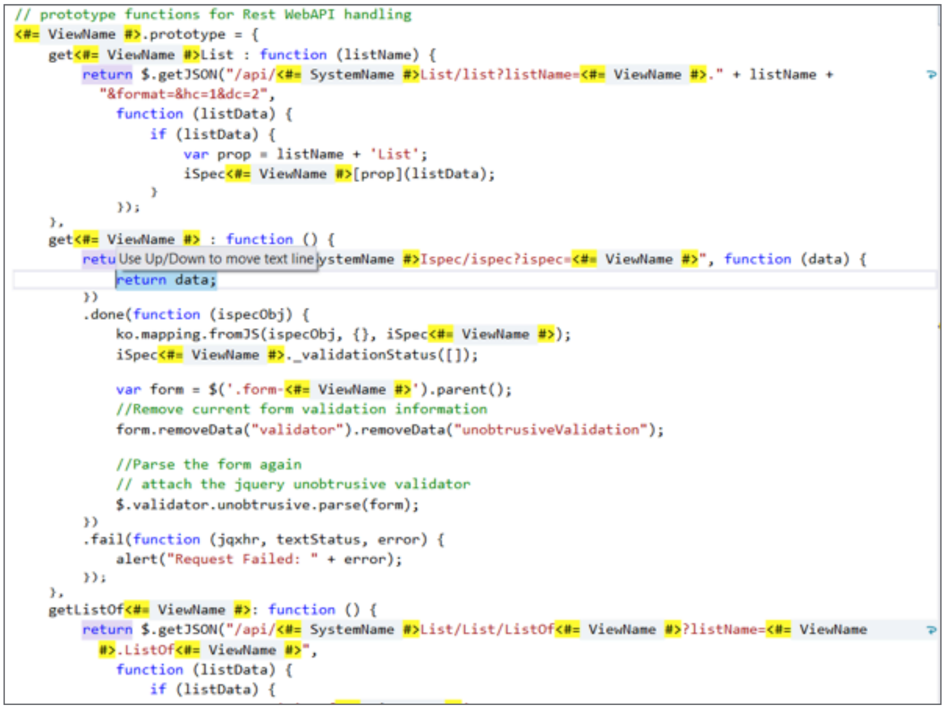


```
var <#= ViewName #> = function () {

    var self = this;
    self.initData = function () {
        var dataObj = {
    // Persisted properties
<#
StringBuilder sb = new StringBuilder();
StringBuilder localsb = new StringBuilder();
List<PropertyMetadata> processedProperties= new List<PropertyMetadata>();

bool hasCopyItems = false;
foreach (PropertyMetadata property in ModelMetadata.Properties) {
    if (property.Scaffold) {
        if (property.IsComplexType)
        {
            // Deal with Signed Numbers and CopyFromItems
            if (property.ShortTypeName.Equals("SignedIntNumberModel") || property.ShortTypeName.Equals
              ("SignedDecimalNumberModel") )
            {
#>
    <#= property.PropertyName #>: '0',
<#...#>
    CopyFromItems: <#= ViewName #>CopyFromModel,
<#...#>
    <#= property.PropertyName #>: <#= localsb.ToString()#>,
<#...#>
    <#= property.PropertyName #>: '',
<#...#>
    <#= property.PropertyName #>List:[],
<#...#>
```



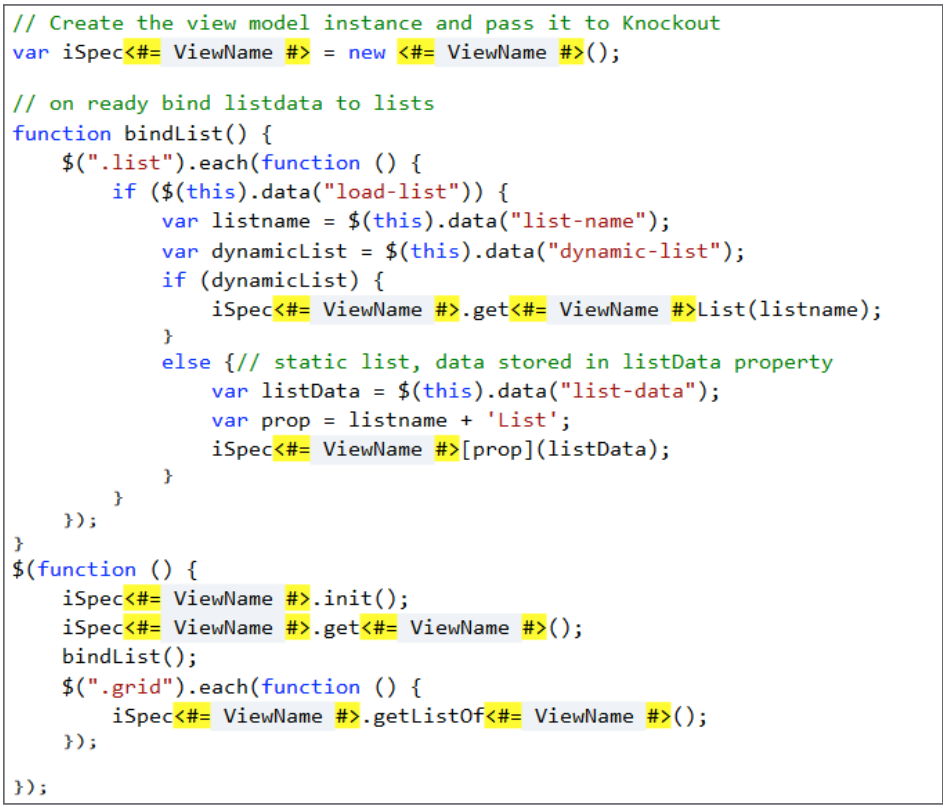
```
self.init = function (){
    ko.mapping.fromJS(self.initData(), {}, self);
    self.gridOptions = {
        data: self.ListOf<#= ViewName #>,
        columnDefs: self.ListOf<#= ViewName #>Columns,
        selectedItems: self.mySelections,
        multiSelect: false,
        enableColumnResize: true,
        jqueryUITheme: false,
        afterSelectionChange: function (rowItem) {
            reMapObject(rowItem.entity, self);
            bindList();
            $('.<#= ViewName #>Modal').modal('show');
            return true;
        }
    };
    ko.applyBindings(self);

}
// This function is triggered from a knockout button click.
// submit the values to a iSpec value
this.submitClick = function (data, event) {
    var target;

    if (event.target) target = event.target;
    else if (event.srcElement) target = event.srcElement;

    if (target.nodeType == 3) // defeat Safari bug
        target = target.parentNode;
    //specify action back to the system
    var indexOfPoint = target.name.indexOf(".");
    if (indexOfPoint > 0) {
        var firstElement = target.name.substr(0, indexOfPoint);
        var baseProp = self[firstElement];
        var secondElement = target.name.substr(indexOfPoint+1, target.name.length - indexOfPoint);;
```

Then, some prototype functions are added to the JavaScript object to handle the REST service calls, including functions that retrieve the object's data, list the dropdown elements, and transmit data back to the AB Suite system.



```
// prototype functions for Rest WebAPI handling
<#= ViewName #>.prototype = {
    get<#= ViewName #>List : function (listName) {
        return $.getJSON("/api/<#= SystemName #>List/list?listName=<#= ViewName #>." + listName +
          "&format=&hc=1&dc=2",
            function (listData) {
                if (listData) {
                    var prop = listName + 'List';
                    iSpec<#= ViewName #>[prop](listData);
                }
            });
    },
    get<#= ViewName #> : function () {
        retu Use Up/Down to move text line ystemName #>Ispec/ispec?ispec=<#= ViewName #>", function (data) {
            return data;
        })
        .done(function (ispecObj) {
            ko.mapping.fromJS(ispecObj, {}, iSpec<#= ViewName #>);
            iSpec<#= ViewName #>._validationStatus([]);

            var form = $('.form-<#= ViewName #>').parent();
            //Remove current form validation information
            form.removeData("validator").removeData("unobtrusiveValidation");

            //Parse the form again
            // attach the jquery unobtrusive validator
            $.validator.unobtrusive.parse(form);
        })
        .fail(function (jqxhr, textStatus, error) {
            alert("Request Failed: " + error);
        });
    },
    getListOf<#= ViewName #>: function () {
        return $.getJSON("/api/<#= SystemName #>List/List/ListOf<#= ViewName #>?listName=<#= ViewName
          #>.ListOf<#= ViewName #>",
            function (listData) {
                if (listData) {
```

And finally, a bit of jQuery code glues the view to the data received from the web service. >>



```
// Create the view model instance and pass it to Knockout
var iSpec<#= ViewName #> = new <#= ViewName #>();

// on ready bind listdata to lists
function bindList() {
    $(".list").each(function () {
        if ($(this).data("load-list")) {
            var listname = $(this).data("list-name");
            var dynamicList = $(this).data("dynamic-list");
            if (dynamicList) {
                iSpec<#= ViewName #>.get<#= ViewName #>List(listname);
            }
            else {// static list, data stored in listData property
                var listData = $(this).data("list-data");
                var prop = listname + 'List';
                iSpec<#= ViewName #>[prop](listData);
            }
        }
    });
}
$(function () {
    iSpec<#= ViewName #>.init();
    iSpec<#= ViewName #>.get<#= ViewName #>();
    bindList();
    $(".grid").each(function () {
        iSpec<#= ViewName #>.getListOf<#= ViewName #>();
    });

});
```

Summary

With these modifications to the standard MVC T4 templates, it's possible to create modern, responsive, and lightweight applications for the Web and mobile devices. This brings a new perspective to classic AB Suite applications by opening them up to modern, front-end application development practices without losing the strength of AB Suite's model-based development environment.

Customizing the Client Framework to use JavaScript frameworks creates an extra avenue for web and mobile application development. While this example shows the power of Knockout combined with a REST service, there are certainly more options open to you. For more, see the example we created that uses the WinJS JavaScript library in a similar fashion to how we employed Knockout.

And if you'd like to learn more about building modern front ends with the Client Framework, please email us at ABSuite@unisys.com.

Another Successful Year with AB Suite in Universities Across Latvia

By Astra Kikute, Director of Human Resources and Contract Operations, Baltic Technology Group

Since 2006, Baltic Technology Group (BTG) has partnered with universities across Latvia to introduce students to the Agile Business Suite development environment and give the newest generation of programmers the skills they need to become attractive candidates to today's businesses.

At the beginning of this academic year, however, this partnership faced a serious challenge. It had become quite hard to retain the same number of students in the AB Suite development courses we offer at the University of Latvia, Daugavpils University, and Ventspils University College, because a number of interesting study courses were recently added to the curriculum by Microsoft and other large IT vendors.

In light of this challenge, we were very pleased this summer to learn about the release of AB Suite Express – a free download of the AB Suite package that includes AB Suite Developer, the AB Suite Runtime for Windows, the Sample system, and Builder.

Giving students an opportunity to "test drive" the software before signing up for the course helped to increase interest, as well as enrollment. Also, the new Agile Business Suite Client Framework, which simplifies the process of building state-of-art user interfaces, is another factor that convinced students to join the AB Suite development course.

Based on student feedback, we determined that giving participants several methods to access the AB Suite development environment was another way to increase enrollment. So in addition to a remote connection to the BTG AB Suite study server, we also made virtual machines (VMs) available to students.

When coupled with this simplified approach to accessing the training environment, the enhancements built into AB Suite 5.0 have created a renewed interest in the AB Suite development course at the University of Latvia and Daugavpils University. Ventspils University College plans to offer the course during the upcoming spring semester, and we are quite confident that it will be a success there, as well.

To keep this momentum going, the BTG development and training teams look forward to joining the AB Suite 6.1 field test next year and introducing the newest AB Suite innovations to students across Latvia.

To learn more about BTG, please visit their web site.

Info Center

New additions to our libraries of How To documents, white papers, and other useful information include:

- **How To:** AB Suite Application Lifecycle Management using Visual Studio Team Foundation Server **(New)**
- **How To:** Set up Team Foundation Server (TFS) for use with AB Suite **(New)**
- **How To:** Use Registry Keys with Windows Runtime **(Updated)**

To view these and other resources, simply go to public.support.unisys.com and choose "Documentation" in the "Public Information" box located on the left-hand side of the screen. No special login is needed.

We also encourage you to view the list of available AB Suite training courses. These courses are a great educational resource and include a lot of graphics, interactivities, simulations, and demonstrations with voice-over narration.

Please be sure to visit the ClearPath Customer Education homepage to view the AB Suite Course Catalog and other helpful education and training resources.

To stay up to date on the latest happenings in the ClearPath world, please subscribe to the ClearPath Connection™ newsletter.

Quick reminder that the next major AB Suite release will be AB Suite 6.1. Naming this release AB Suite 6.1, not AB Suite 6.0, was an internal decision and does not have any impact on our users. Its availability is planned for Q4 2016.

And if you're looking to explore AB Suite on an evaluation basis, please download AB Suite Express today! This free download includes the full AB Suite package, as well as a "getting started" course designed to help you begin exploring everything AB Suite has to offer in no time.

Specifications are subject to change without notice. No warranties of any nature are extended by this document. Unisys cannot accept any financial or other responsibility that may be the result of recipient's use of the information in this document or, including direct, indirect, special, or consequential damages.