# Dyntaxa – System Documentation

Table of contents

[Dyntaxa – System Documentation 1](#_Toc325448113)

[Övergripande struktur – Ej klar 2](#_Toc325448114)

[Base-Controller 3](#_Toc325448115)

[Navigation tree 5](#_Toc325448116)

[Menu 8](#_Toc325448117)

[Logging 9](#_Toc325448118)

[User Management 10](#_Toc325448119)

[Debug mode 11](#_Toc325448120)

[Language 12](#_Toc325448121)

[Validation 14](#_Toc325448122)

[Error handling 15](#_Toc325448123)

[3rd-party components 16](#_Toc325448124)

[Security 17](#_Toc325448125)

[Cachning – Ej klar 18](#_Toc325448126)

[Settings 19](#_Toc325448127)

[Session – Ej klar 20](#_Toc325448128)

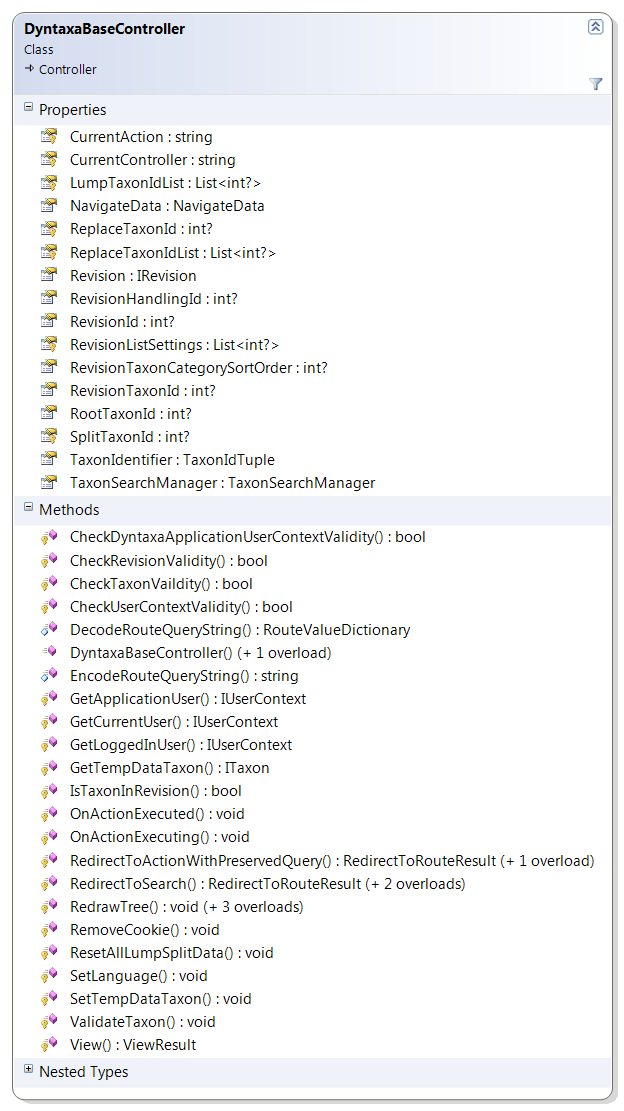
[Testning – Ej klar 21](#_Toc325448129)

## Övergripande struktur – Ej klar

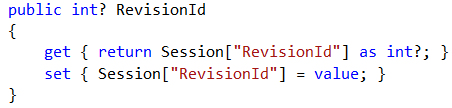
Här borde det finnas information om:

* Tjänster, Löken, etc
* Databas, stored procedures
* Bilder/Diagram som förklarar hur det hänger ihop
* Vilka olika tjänster kommunicerar Dyntaxa med?
* …

## Base-Controller

All controller classes inherit from the DyntaxaBaseController. This class contains methods and properties that are used throughout the system.

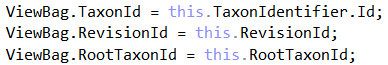
#### Properties

Most properties in DyntaxaBaseController looks like this:  


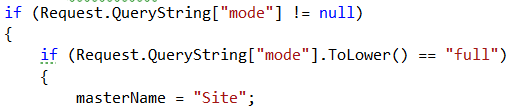
That is, values ​​are stored and retrieved from the session state.

#### OnActionExec…

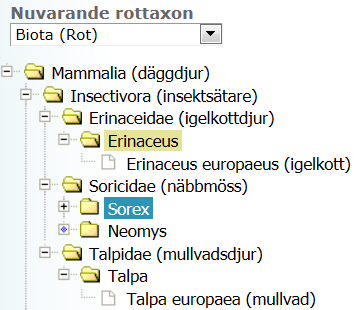
OnActionExcecuted(…) and OnActionExecuting(…) is called every time a page is loaded. Data that is used in most views is added to the ViewBag object:



OnActionExecuting (...) handles the Query string parameters "mode" and "lang". The parameters tell the system which master file and what language to be used. When one of these values is set it is saved in the session, and not changed until the user sends a new value in the Query string or performs a specific action to change this.



## Navigation tree



The tree uses a 3rd-party JavaScript Component named Dyntatree. <http://wwwendt.de/tech/dynatree/>

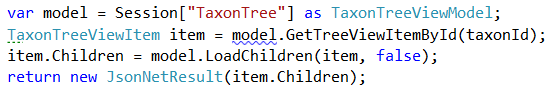
When a tree is loaded for the first time, the root taxon will be loaded and its immediate children. Other taxon is loaded as soon as you click on the icon to the left of the taxon name: 

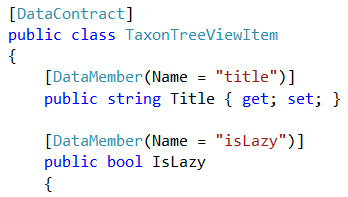
When the icon shows  or  , the nearby children are already loaded.

#### Ajax loading

When the user clicks the icon, an Ajax call to the following function is performed:  

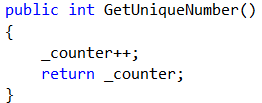

What then occurs on the server side is that the clicked taxon closest children is loaded and added to the tree model. Then the children is returned as a JSON array.



Each child is formatted correctly in JSON and uses the properties that DynaTree has defined since that TaxonTreeViewItem uses WCF Data Contract as follows:  


#### ID values

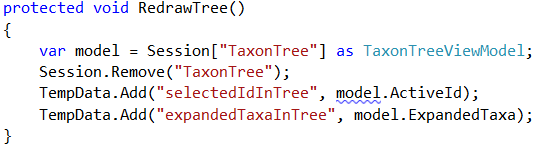
In order for an element in the tree to be identified DynaTree requires that each node has a unique id. As a taxon can be found in several places in the tree, a taxon must be identified by a unique ID value. TaxonId does not work.

When a node is created in the tree, it gets a unique value from a counter that is increased by one every time it's called.  


#### Redrawing

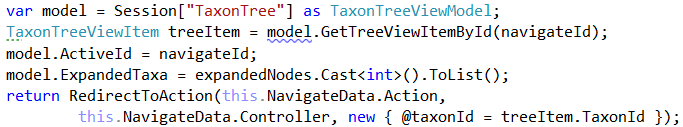
When a user is in a revision and has for example updated a scientific name, the tree must be redrawn. This is done by 1. Call RedrawTree() 2. Reload the page.

What happens is that the tree is removed from the session so that at a reload of the page the tree is rebuilt with the refreshed data. RedrawTree() also makes sure to save which taxon was chosen and which nodes were expanded so that the state can be restored.



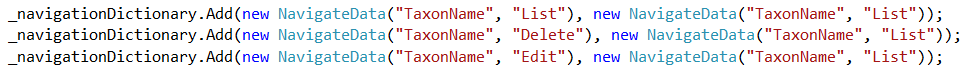
#### Navigation

When a user clicks on a taxon in the tree, it will send a POST call to the following function:  


Somewhat simplified (error handling is removed, etc.), the following happens:

I.e. 1. The model is loaded from the session cache. 2. The node that was clicked is retrieved from the tree model. 3. The clicked node is set as active. 4. Expanded nodes is set. 5. The user is redirected to the page stored in NavigateData.

TreeNavigationManager is the class that defines to which page the user should be routed when a node is selected in the tree.  


The class initializes a Dictionary that contains the mappings as follows:

Every time a page is loaded the following lines of code is executed:  

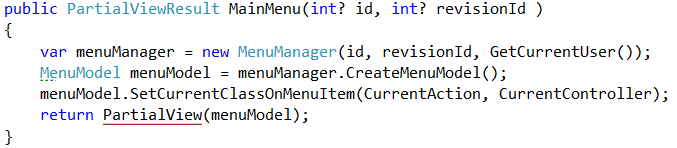

NavigateData then contains the page to be loaded when the user clicks on a taxon in the tree.

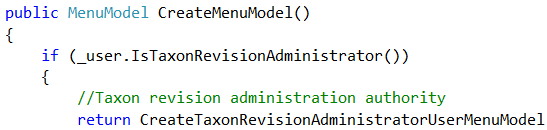
## Menu



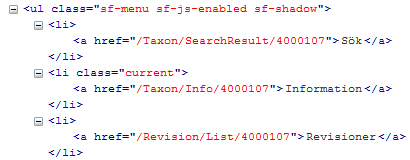
The menu uses a 3rd-party JavaScript component called Superfish.   
[http://users.tpg.com.au/j\_birch/plugins/superfish/#](http://users.tpg.com.au/j_birch/plugins/superfish/)

#### Creation

The code that executes when the menu is created is as follows:

In CreateMenuModel (..) the menu is created in different ways depending on the permissions the user has and if she is in revision mode or not.

…

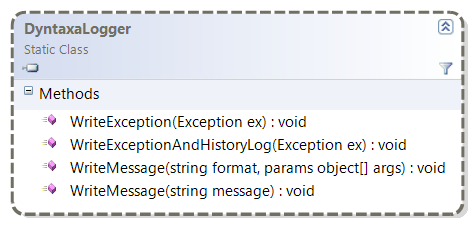
The rendered HTML-code looks like this:  


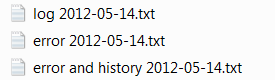
When the page has finished loading Superfish is applied on the list. The menu works even when the user has turned off JavaScript in the browser.

#### Navigation

The links are already generated when the page is rendered, which means that when we click on a menu item, we come to the page directly via a GET call without any redirects.

## Logging

Logging is performed using one of the methods of the static class DyntaxaLogger:

Three different types of files are created:  


WriteException(…) writes ”error - .txt”  
WriteExceptionAndHistoryLog(…) writes ”error and history - .txt”  
WriteMessage(…) writes “log - .txt”

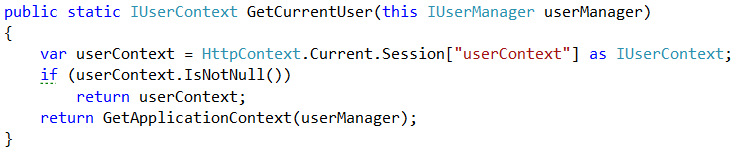
For each day, a file of each type is created. The files are saved in a folder called Temp.

## User Management

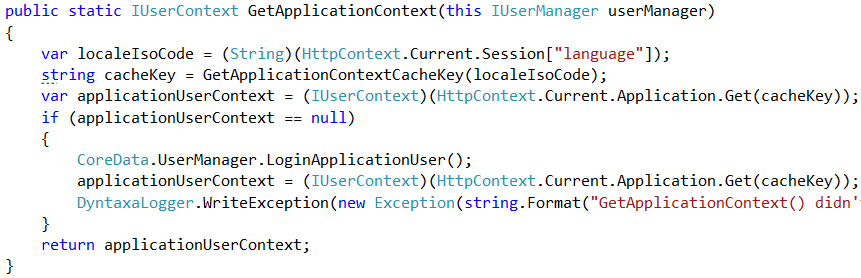
When Dyntaxa is running in public mode when no user is logged in then the "DyntaxaUser" is used as UserContext when calls to web services are made. When a user is logged on then that UserContext is used in calls to the web services.

To retrieve a UserContext the method GetCurrentUser(..) is commonly used:  

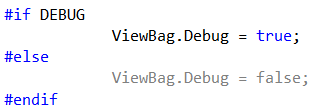

The method first checks if there is a user logged in, by obtaining a value from the session state with the key "userContext". If there was a user then it is returned, otherwise "DyntaxaUser" is returned.

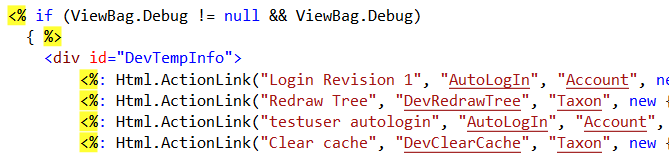


Application User "DyntaxaUser" is logged in once for each language Dyntaxa support. Once the "DyntaxaUser" is logged on it will be saved in the Application cache for quick access.

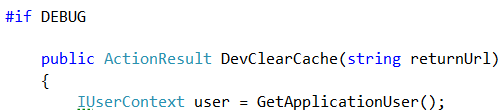


## Debug mode

In DyntaxaBaseController we execute the following code in OnActionExecuting (..) on each page load:

In Site.master we use this information to render a <div> tag with links to features that is used for debugging purposes.

#### Security

To ensure that the debug functions cannot be called in the Release mode, they are only defined if the compiler constant DEBUG is set.  


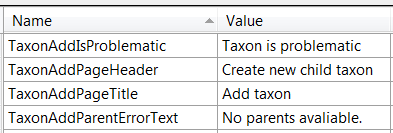
## Language

Dyntaxa currently supports English and Swedish.

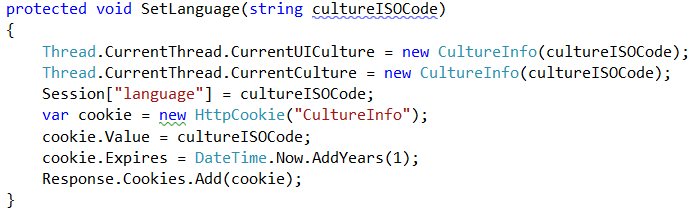
#### Resource files

In the file DyntaxaResource.resx the resource strings are defined and their value in English. In DyntaxaResource.sv.resx there are the same resource strings with Swedish values.



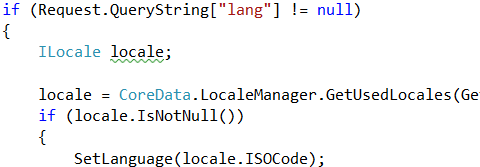


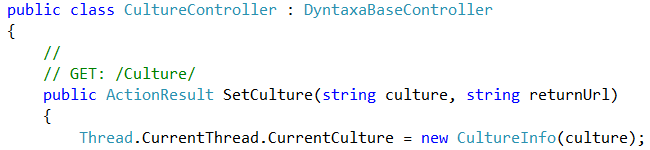
#### Change language

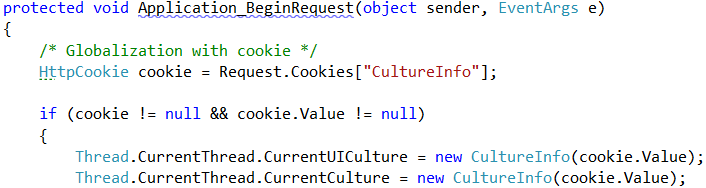
The following function in DyntaxaBaseController is used to change the language.  
  
The function saves the current language in the Session state, and also writes a cookie so the user gets the right language the next time she visits Dyntaxa.

The user can change the language in two ways:

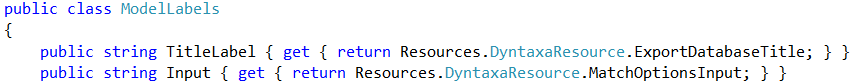
1. Use the Query string parameter "lang" to set which language to use.

This is handled in OnActionExecuting (..) each time a page is loaded as follows:  
  
Once the language is changed, that language is used until a language change occurs again. The language is stored in session state.

1. Make a call to the Action SetCulture (..) where you specify which language to use.  
   

Every time Application\_BeginRequest is executed the language on the current thread is set:  


#### Labels

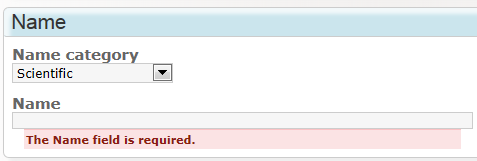
To access the resource strings in a uniform manner, almost every view model has an inner class called ModelLabels where all resources are defined.  


In the view you write like this:  

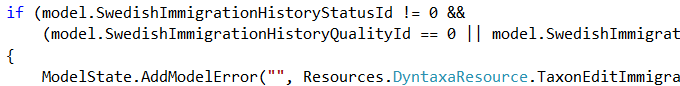

## Validation

Dyntaxa uses the built-in validation features in ASP.Net MVC. For example, if a name is required you add a attribute to the property like this:  

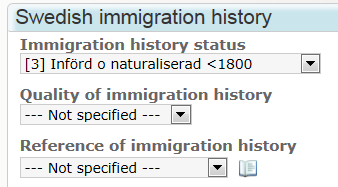

When the user clicks on save, a validation error displays at the top of the page and next to the field that didn't pass validation:  

To handle validation errors caused by more than one property, we use the function ModelState.AddModelError(..). For example the following code will create a validation error if we have chosen a value in the first box, but not in the other two:

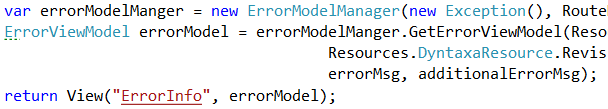
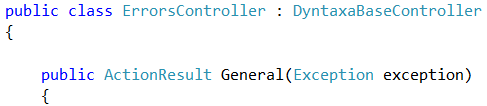


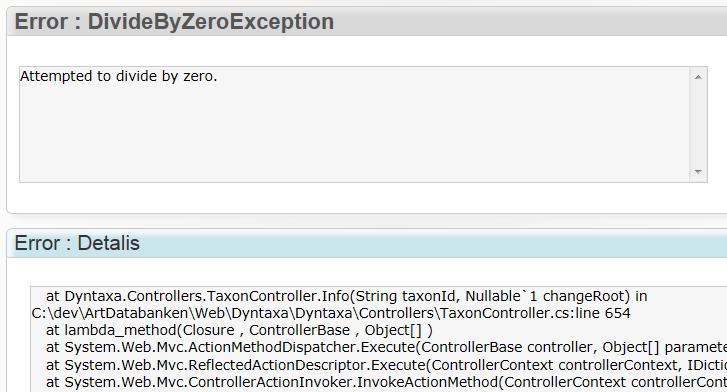




## Error handling

When unexpected errors occurs (not validation), we show the "ErrorInfo" view. It can be accessed in two ways:

1. For maximum control of what to display, you can create a error model and display the view.  
    
2. All Exception that is not handled, get to a function named Application\_Error(..) defined in Global.asax.cs.   
     
     
   That function will redirect the user to an Action named General contained in the Error Controller.   
     
   That function shows the "ErrorInfo" view for the user:



Logging of the error is done with this call:



## 3rd-party components

The following components is used:

**JavaScript components**

|  |  |
| --- | --- |
| **Name** | **Address** |
| DynaTree | <http://wwwendt.de/tech/dynatree/doc/dynatree-doc.html> |
| DataTables | <http://datatables.net/> |
| jQuery | <http://jquery.com/> |
| jQuery UI | <http://jqueryui.com/> |
| jQuery BlockUI | [http://jquery.malsup.com/block/#](http://jquery.malsup.com/block/) |
| jQuery Cookie | <https://github.com/carhartl/jquery-cookie> |
| jQuery Subcookie | <http://blog.valugi.ro/2009/09/13/subcookies-in-jquery/> |
| Superfish | [http://users.tpg.com.au/j\_birch/plugins/superfish/#](http://users.tpg.com.au/j_birch/plugins/superfish/) |
| Drop-down-check-list | <http://code.google.com/p/dropdown-check-list/> |

**.Net components**

|  |  |
| --- | --- |
| **Name** | **Address** |
| Json.Net | <http://james.newtonking.com/projects/json-net.aspx> |
| RhinoMock | <http://ayende.com/wiki/Rhino+Mocks.ashx> |
| MvcContrib | <http://mvccontrib.codeplex.com/> |
| Office Interop Excel |  |

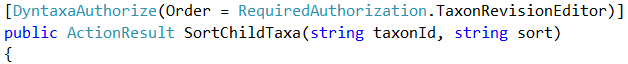
## Security

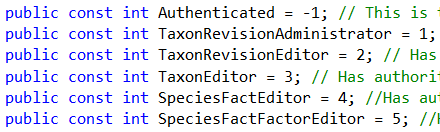
#### Användare

Här kan det stå text om UserAdmin…

#### Authorization

To prevent users from accessing pages they are not authorized to access, we decorate the Actions that should be protected with the DyntaxaAuthorize attribute. The attribute specifies the permission required to enable the user to access the page. If the user does not have permission, she is redirected to the Login page.



The following permissions exist:  


#### Request validation

To protect against script injection attacks we use the built in request validation mechanism in ASP.Net. When we want a property to allow HTML-characters we decorate it with the AllowHtml-attribute.



## Cachning – Ej klar

* Vad cachas?
* Var sker cachning?
* Hur länge cachas objekt?

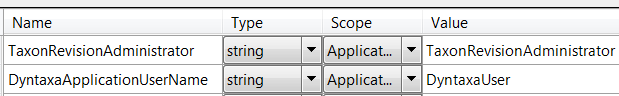
## Settings

Settings in Dyntaxa are done in two different files.

#### Web.Config

Here are general settings for the web application. Eg how long the session state is in memory:  


#### DyntaxaSettings.settings

Here are specific settings for Dyntaxa such as name of the user account to be used in public view:  


## Session – Ej klar

* Vad sparas?
* Hur länge?

## Testning – Ej klar

* Enhetstester
* Andra tester