HD3C02 – Debugging XML Views

|  |  |
| --- | --- |
| **Product and Focus**  HANA Platform/SAPUI5 | **MOTIVATION**  This case explains methods for debugging XML views.  **PREREQUISITES**  HD3C01 – Hello World |
| **Target Audience**  Undergraduate/Graduate Beginner to Intermediate |
| **Author**  Ross Hightower |
| https://bgoerke.files.wordpress.com/2013/05/section1.png | |

# Debugging XML Views

Debugging XML views is complicated by the fact that the XML code is not executed in the browser in the same way JavaScript in the controller is executed. UI5 generates HTML from the XML code and sends the HTML to the browser. That means that you can’t debug the code in the same way that procedural code is debugged. However, there are some techniques that make it relatively easy to identify typical errors in XML views.

## Hello World App

This case uses the Hello World app developed in case HD3C01 – Hello World so you must have a working version of that app to begin. You will introduce various errors and then observe the errors that are displayed in the browser so you can identify the errors later.

## Debugging Approach

Since you can’t observe the errors directly in the browser you have to learn to identify the symptoms that occur when certain errors in your code occur. There are a few basic categories of coding errors:

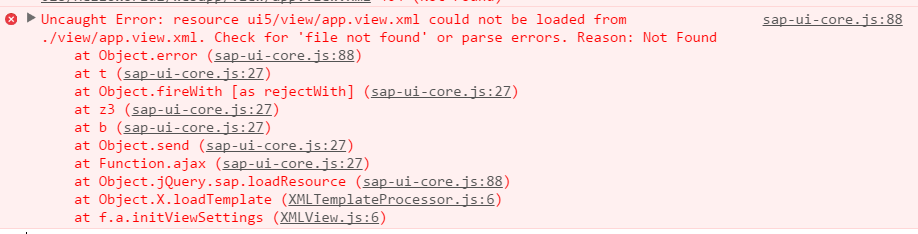
* Errors in locating the view files
* Errors in locating the controller file associated with the view
* Syntax errors in the XML code
* Missing SAPUI5 libraries
* Binding errors between the model and controls in the view

This case is concerned with the first three of the categories. Binding errors will be addressed in the Data Binding case.

|  |
| --- |
| Almost all of the debugging techniques rely on the developer tools in the browser. In almost all browsers you can open the developer tools using the function key F12. All of the examples in this document use Chrome but all browsers have some version of developer tools. In Chrome the developer tools have a number of tabs:    The most important for debugging XML views is the Console. All error messages are written to the Console. There are a couple of errors that are common but are nothing to worry about. The first one is an error indicating that a pre-loader could not be found and the second is that an i18n properties file could not be found. Neither of these will affect the execution of the app and can be safely ignored. |

## Errors in Locating Files

Before identifying the causes of these errors, let’s look at how the error might appear in the browser. One type of location error will look like this:



The error states:

*Uncaught Error: resource ui5/view/app.view.xml could not be loaded from ./view/app.view.xml. Check for ‘file not found’ or parse errors: Reason: Not Found.*

Let’s look at some ways this or similar errors might occur.

### Resource Roots Error

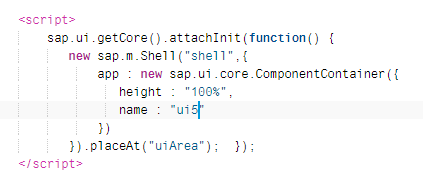
If you remember from the Hello World case, the index.html contains a script called the bootstrap.



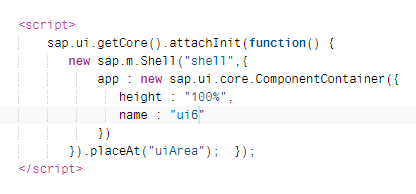
The property data-sap-ui-resourceroots is used to define the root package of the application and is used to locate files in the application. The image below shows the project package structure and the UI5 package structure which you can view by drilling into the sap package. UI5 will look for any files your app requires in the UI5 packages unless you specify otherwise which is he purpose of the resource root configuration.



In the Hello World app, there is a script in the index.html file that loads the component.



Note the ui5 in the name property. This can be read as locate the Component.js file in the project root package. Let’s see what happens if the resource root is not defined correctly. Change the name property to ui6 (so it doesn’t match the data-sap-ui-resourceroots property in the bootstrap) .



Run the application and you will get this error:

|  |
| --- |
| Uncaught Error: failed to load 'ui6/Component.js' from https://openui5.hana.ondemand.com/1.42.6/resources/ui6/Component.js: 404 - Not Found |

Note that the error message states that the Compoent.js file could not be found in the https://openui5.hana.ondemand.com/1.42.6/resources/ui6/ package. **It’s not looking in the project root package in the application but looking for a package in the SAPUI5 library folders**. That’s the indicator that the resource roots is either not defined correctly or you used the wrong value when identifying a file.

Note this same error can affect the references to controller files that appear at the top of view files. In the App.view.xml file the controller associated with the view is identified using the property controllerName:



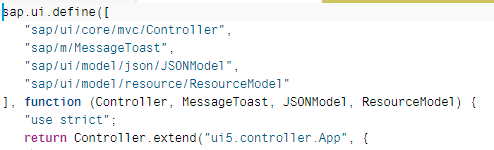
Change this to ui6.controller.App:



The error looks like this

|  |
| --- |
| Uncaught Error: failed to load 'ui6/controller/App.controller.js' from https://openui5.hana.ondemand.com/1.42.6/resources/ui6/controller/App.controller.js: 404 - Not Found |

Again, the key to identifying the error is that UI5 is looking for the file in the UI5 library packages rather than in the project package. Note also that there is a reference to the controller file near the top of the controller file as well:



### Naming Error

Now edit the same view file so that the view name is defined correctly but the name now begins with a lower case a.



Remember that JavaScript and XML are case sensitive. If you run the app now you will get this error:

|  |
| --- |
| Uncaught Error: failed to load 'ui5/controller/app.controller.js' from ./controller/app.controller.js: 404 - Not Found |

The difference between this error and the previous error is that UI5 is looking for the file in the correct location (./controller) but still can’t find it. One problem is that you have an error in your XML (we’ll cover this later) but another cause is the there is a naming problem. In this case, it’s because the file is named App.view.xml and in the app we used the name app (with a lower case a). This same error can occur if you

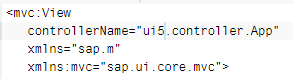
* misspell the view name,
* misspell the package name,
* misspell the word view or xml in the file name.

A similar error can occur if the names of controller files are incorrect.

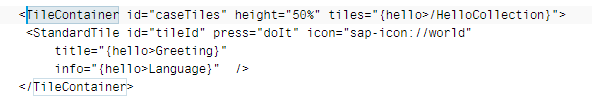
**Naming errors can be very difficult to spot, even when you are looking very carefully.**

### Missing UI5 Library

At the top of view files the modules (or libraries) that contain definitions of the controls used in the view are referenced. In the Hello World App view two modules are referenced: sap.m and sap.ui.core.mvc.

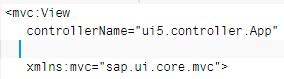


If a required module is not listed, then UI5 will not be able to find the definition of the affected control. In the App view the TileContainer control is defined in the sap.m module. You can tell this because sap.m is the default module (there is not prefix defined after xmlns as there is for the sap.ui.core.mvc module) and the Button control has no prefix:



On the other hand, you can tell the View control comes from the sap.ui.core.mvc module because it includes the mvc: prefix assigned to that module.

Now delete the reference to the sap.m library.



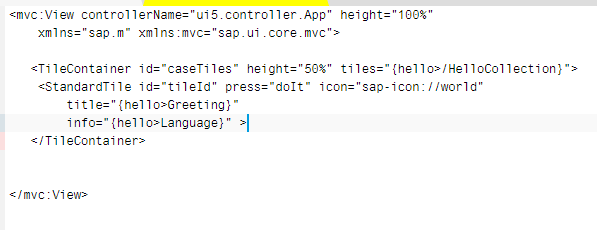
The error looks like this:

|  |
| --- |
| library-preload.js:522 Uncaught Error: failed to load 'null/TileContainer.js' from https://openui5.hana.ondemand.com/1.42.6/resources/null/TileContainer.js: 404 - Not Found |

UI5 is indicating that it can’t find a specific control which is a dead giveaway that the module with that control was not referenced (null means that there is no value).

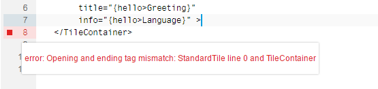
## XML Syntax Errors

XML syntax errors can very difficult to spot even for experienced developers. Try to locate the error in the small bit of XML code:



The problem is that the StandardTile control is not closed. All XML elements must have an opening and closing tag. For elements that have not content, you can use the short cut /> to close it. The StandardTile control tag ends with > instead of />.

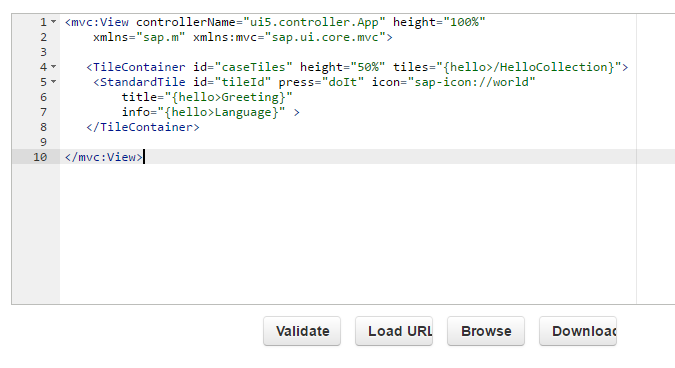
Fortunately, there are tools that make it much easier. The first is the editor which can often identify errors. If you edit the code to create the error mentioned above, you will see a red square appear in the margin. If you hover the mouse pointer over this, the editor will tell what it thinks the error is.



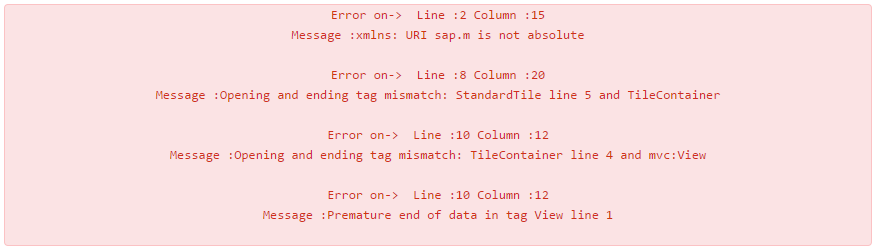
Sometimes the editor can’t identify the error. In that case, you can use an XML validator to find the errors.

<http://codebeautify.org/xmlvalidate>

Copy the XML code into the validator field and click Validate.



The first and last errors listed are not an error you need to worry about. Notice that it identified the error with the StandardTile element and identified the line on which the error occurred.



## Exercise

Debug the sample project called DebugViews provided by your instructor. To import the project, right-click your student package and click Import→Archive.

