Web-based spreadsheets and Business Intelligence

Combining a web-based spreadsheet with Pentahos Business Intelligence

Deeb El Harbiti
Abstract

Web-based spreadsheets and Business Intelligence

Deeb El Harbiti

Business Intelligence is becoming more and more an essential element for a company’s planning, development and consistency. A company usually aims to predict and optimize the outcome of different decisions. In order to do that, it requires a tool that can perform scenario analysis or “what if” — analysis.

These analyses can be performed with spreadsheets and today with the tools within business analysis being web-based, the purpose of the project is to combine a web-based spreadsheet with an analysis from a BI server. The BI server provides sample data. Should a combination be possible then a continuing goal would be to connect the BI server with a database and in that way provide the spreadsheet with data to perform “what if” — analysis.

The results indicated that a combination between the web-based spreadsheet and the BI server was possible. However, a few alterations of the spreadsheet are still necessary in order for the combination to work as desired. Further developments of the product involve managing to adapt it to smartphones and tablets.
Sammanfattning

Business Intelligence, informationsanalys eller affärsanalys, är en allt viktigare beståndsdel för organisationer och företags planering och uppföljning. Som företag vill man kunna förutse och optimisera utfallen av olika beslut. För det behövs ett verktyg som kan utföra så kallade scenarioanalyser eller ”what if” – analyser.

Sådana analyser kan utföras med kalkylark och eftersom olika verktyg inom affärsanalyser idag är web-baserade så är syftet med detta projekt att kombinera ett webb-baserat kalkylark med en analys från en BI server. BI servern tillhandahåller exempeldata, men skulle en kombination vara möjlig så skulle ett fortsatt mål vara att koppla ihop BI servern med en databas och på så sätt förse kalkylarket med de data man vill utföra ”what if” - analyser på.

Table of contents

Sammanfattning .................................................................................................................. 0

1. Introduction ..................................................................................................................... 3
   1.1 Spreadsheets .............................................................................................................. 3
   1.2 Business analysis ..................................................................................................... 3
   1.3 Purpose of the project .............................................................................................. 4
   1.4 The setup for this report .......................................................................................... 4

2. Theory ............................................................................................................................ 5
   2.1 Pentaho .................................................................................................................... 5
      2.1.1 BTable ............................................................................................................... 5
   2.2 Business Intelligence and Business Analytics ......................................................... 7
      2.2.1 Business Intelligence (BI) ................................................................................ 7
      2.2.2 Business Analytics (BA) .................................................................................. 7
   2.3 Wickedgrid ............................................................................................................... 7
   2.4 Technologies ............................................................................................................. 8
      2.4.1 AJAX ................................................................................................................ 8
      2.4.2 JavaScript ......................................................................................................... 8
      2.4.3 HTML ............................................................................................................... 9
      2.4.4 CSS .................................................................................................................. 9
      2.4.5 XML ............................................................................................................... 10
      2.4.6 jQuery ............................................................................................................. 10
   2.5 Tomcat ..................................................................................................................... 11

3. Method ............................................................................................................................ 12
   3.1 Step one – Setting up ............................................................................................... 13
   3.2 Step two – Adding a plugin ..................................................................................... 13
      3.2.1 The ‘lib’ folder ................................................................................................. 14
      3.2.2 The ‘plugin.xml’ file ....................................................................................... 14
      3.2.3 The ‘plugin.spring.xml’ file ........................................................................... 14
   3.3 Step three – Forming a combination ........................................................................ 15

4. Result ............................................................................................................................... 16
   4.1 The setup .................................................................................................................... 16
   4.2 The added plugin ...................................................................................................... 18
   4.3 The combination ...................................................................................................... 19
      4.3.1 The combination – First part ........................................................................... 19

5. Discussion & Conclusion ............................................................................................. 20
   5.1 The plugin ................................................................................................................ 20
   5.2 The combination ...................................................................................................... 21
   5.3 Future work .............................................................................................................. 22
Reference .................................................................................................................................................. 23
Appendices 1 – The 'plugin.xml' file ........................................................................................................ 24
Appendices 2 – The 'plugin.spring.xml' file .......................................................................................... 25
Appendices 3 – The 'cpk.xml', 'cpk.spring.xml' and 'settings.xml' files .............................................. 26
1. Introduction

In companies the accountants detail sets of business transactions such as profits and expenses by using large sheets of paper with rows and columns, known as traditional spreadsheets. It lays out a broad set of calculations and numerical values on a single sheet of paper.

1.1 Spreadsheets

Since the start and increase of computers usage throughout the years, the usage of spreadsheets increased and went from physical to electronic form. Early spreadsheet computer programs such as VisiCalc were used to tabularise and organise data. Implemented as a standalone program, electronic spreadsheets were executed on a data processor [1]. These spreadsheets organized information into rows and columns inside an electronic archive. They can be used to view and add data to help review business issues just as their physical counterpart. Instead of adding the data by hand a formula can be used with the electronic spreadsheets, therefore when alterations to a spreadsheet cell are made, for example a changing a number in the cell, the outcome of the entire setup can be seen instantly. This is an advantage with the desktop electronic spreadsheets. A popular desktop electronic spreadsheet is Microsoft's Excel spreadsheet [2]. It allows the user to tabularise cells of the spreadsheet and to enter data into those cells. Although users are allowed to enter links to the web in a cell, Excel just like most electronic spreadsheet applications are not designed with considerations to the web.

Today, with the great usage of the web, web-based spreadsheets or online spreadsheets has become more relevant. They are needed to include data only accessible through the web and make full use of the capabilities of the web.

A web-based spreadsheet is displayed and edited in a web browser. With the arrival of web technologies such as AJAX a new generation of web-based spreadsheets began to be developed. These spreadsheets have many of the features seen in the desktop versions and are provided with an internet application user experience.

1.2 Business analysis

The purpose for a company is usually to be successful and receive as big profit as possible. An essential part for achieving this is to make the right decisions. Most companies all over the world review their businesses from time to time in order to make the decisions that will improve the company. Defining solutions to business problems and the pursuing discipline of finding business needs is what business analysis is about. These solutions could include policy development, strategic planning or organizational changes. However today with the current computerised world solutions often contain a software system development component.
Businesses are mostly run through computers, meaning that profits, revenues, salaries etc. are computerised. This means that when a company is about to be evaluated through a business analysis, it has to organise all this data and revise it. This is done using computers and the larger the company is, the more data there is to process. Time is money in business, and to save money it would be convenient for a company to have that data processed as effectively as possible. For this purpose, there are some technologies developed, such as Business Intelligence (BI).

1.3 Purpose of the project

Omicron is an official partner to Pentaho (a Business Intelligence software company described in section 2.1) in Sweden, both as reseller and integrator [3]. One of the services the company offers is solutions that involve business analysis. Something that is missing today is a well working web-based page, free of charge, where you can combine output data from a Business Intelligence analysis with calculations in an environment that resembles for example an excel document. The issues that occur with the lack of a web-based spreadsheet of this type can for example be difficulties with performing "what if" - analysis.

The main purpose of this project is to, in a webpage, combine a spreadsheet and a BI analysis. The webpage will be similar to an excel-document and one will be able to place an analysis from a BI system there. Further on, by studying different methods on how, try to implement "what if" - analysis. By implementing this, it could ease the way for methods to perform different prognosis and risk analysis.

1.4 The setup for this report

This report is set up in a way that it first provides background information about the technologies, programing languages etc. that is encountered during the project. Thereafter it proceeds to the method section, which describes the approach taken with the technologies mentioned to succeed with this project. This is followed up by the result section were the acquired results are shown. Finally the report ends with a section including discussions and conclusions, regarding the results from the previous sections.
2. Theory

2.1 Pentaho

Pentaho is a Business Intelligence software company and is today the lead provider of Open Source Business Analytics. It is a comprehensive data integration and business analytics platform that addresses the barriers that blocks an establishment’s ability to obtain value from all its data [4]. The platform includes a spectrum of tools to easily visualise, analyse, predict, explore and report any data, it also simplifies preparing it. Pentaho provides businesses with a translation of data into value. The company provides solutions within industries such as healthcare, retail, government and financial services.

2.1.1 BTable

BTable is a table added as a plugin in Pentaho that can receive data from different data sources and databases and provide data analysis among other things. A part of the project’s purpose is to be able to place analysis of data from the BTable, received from a connected database, into the web-based spreadsheet.

Figure 1. Shows how a BTable looks like and a sample of the data it provides.
Figure 2. Shows another example of the BTable.

As seen in the figures above there are already existing data inside the BTable. It’s a sample data provided by the BTable. This is the data that should be placed inside an online spreadsheet. If that succeeds the next step is to connect a database to the Business Intelligence server, and thereafter place an analysis from the Business Intelligence server inside the spreadsheet.
2.2 Business Intelligence and Business Analytics

2.2.1 Business Intelligence (BI)

BI is the set of processes, data, applications, technical architectures and technologies which are used by businesses to support analysis, presentation and collection of business information [5]. Both current and predictive views by business operations are provided by the BI technologies. Functions commonly used Business Intelligence technologies are business performance management, complex event processing, online analytical processing, process mining, reporting, analytics, data mining, prescriptive analytics, predictive analytics and are capable of handling great quantities of occasionally unstructured and structured data to help develop, find and if not, generate new strategic business opportunities. The goal is to let this big data be easily interpreted. By applying effective comprehended strategies and finding innovative opportunities a business can be provided with a long consistency and a competitive market lead.

2.2.2 Business Analytics (BA)

BA on the other hand refers to the technologies and skills to improve business planning and gain insights by reviewing and constantly explore past business performances. Based on statistical procedures and data, Business Analytics centres on an understanding of the performances of a business and a development of modern visions [5]. Traditionally Business Intelligence centres on advised business planning and evaluating past performances, using a dependable set of metrics. In other words BA is needed to change the business while BI is needed to run it.

2.3 Wickedgrid

Background studies on web-based spreadsheets have shown that there are a few different kinds already existing. The better and more complete ones are however liable to charge. The objective of this project is to be able to create a spreadsheet that is fully functional and free of charge that can be available for the open-source community.

As mentioned before, many web-based spreadsheets exist. Some of them provide an open-source\(^1\) code that generates what resembles a shell for a spreadsheet. In the beginning of the project these open-source codes were studied and tested in order to see what they provided in terms of functionality, technique, and layout etc. Some desired results were provided by a particular sheet, namely Wickedgrid.

---
\(^1\) Open-source generally refers to a computer program that has its source code accessible for the general public, either to use or adjust it from its primary design.
Wickedgrid is an AJAX (described in Section 2.4.1) component spreadsheet that provides an experience familiar to Microsoft Excel for web applications using Java [7]. It runs entirely in the browser, has an open and flexible API\(^2\) and support for JSON\(^3\), XML (described in Section 2.4.5) and HTML (described in Section 2.4.3). To manage creation of sheets, editing process and viewings it uses simple HTML tables and components, providing stability even in older web browsers.

Based on the above, the project proceeded with further development of the already existing source code for this spreadsheet.

### 2.4 Technologies

#### 2.4.1 AJAX

AJAX, which is short for Asynchronous JavaScript And XML, is an arrangement of techniques within web development. Many of the web mechanisations on the client side are used by AJAX to construct asynchronous web operations. An advantage when using AJAX is that data can be recovered and sent asynchronously through web operations without allowing the existing page to have its activities and display interfered with. Moreover AJAX lets web pages and applications adjust their content vigorously without needing to have the complete page reloaded, doing so by unlinking the presentation layer from the data interchange layer. Because of the advantages of JSON being inherent to JavaScript, recent applications usually substitute XML for JSON in practice.

AJAX is not representing a single technology, but rather a group of them, such as [8]:

- CSS and HTML that can be used in combination to style information and mark it up for presentations.
- XMLHttpRequest objects needed to arrange for data exchange asynchronously between server and browser to avoid pages from being fully reloaded.
- XML and JSON, used for data transactions.
- The DOM (Document Object Model) that is accessed with JavaScript. It allows the user to interact with accessible data, and for dynamic displays. (The DOM is described further in section 2.5.6)
- JavaScript, used to bring all this together.

#### 2.4.2 JavaScript

JavaScript is an untyped, advanced, interpreted and active run-time programming language, often shortened as JS. It is known as the language of the web. JS is, along with CSS and HTML, one of the three main technologies of the content production for the World Wide Web [9]. It is engaged by the majority of websites and endorsed by all modern web browsers, excluding the necessity for plugins. JS is a multi-paradigm

---

2 API, application programming interface, is a set of tools and arrangements for building application software.

3 JSON is an open-standard file format. It uses human-readable text to send out data.
programming language, prototype-based and supports imperious, practical and object-oriented programming styles. This programing language has an API for operations with arrays, text, regular expressions and dates. However, it does not incorporate for any I/O, for example storage, graphics facilities or networking, which makes it depend on the hosting environment to have it inserted. In this project JS was mostly used to bring everything together in the web-based spreadsheet, but also for providing the spreadsheet with formula bars, sheet bars, toolbars and a grid-like user interface, contributing with essential features such as sorting and merging cells, style and text editing, freezing, inserting and deleting columns and rows.

Even though there are strong apparent resemblances between Java and JavaScript, which comprises the respective standard libraries, language name and syntax, the two languages are dissimilar and their design varies significantly.

### 2.4.3 HTML

As mentioned before HTML, short for HyperText Markup Language, is one of the three main technologies for World Wide Web. When creating web pages and applications, the standard markup language to use is HTML. The documents that web browsers render into web pages are HTML documents that are sent from a local storage or a web server. The construction of a web page is described by HTML originally and semantically including indications for the documents appearance. The construction blocks for HTML pages are HTML elements. The rendered web page can with HTML constructs have images and objects inserted into it. In the project HTML is therefore used to create the “body” of the spreadsheet. In other words, it is used to build the structure for the spreadsheet.

### 2.4.4 CSS

In this project, CSS is a style sheet language used to make the spreadsheet look good. This means that it is used for things such as adding a front colour, a background image, etc. CSS, short for Cascading Style Sheets, is therefore used for describing the appearance of documents written in a markup language [10]. As mentioned earlier, CSS is one of the main technologies used by almost all websites for the purpose of creating user webpages that are visually pleasing and also for user interfaces for web applications and many mobile applications. However, the user interfaces and the visual styles of the web pages written in HTML is not the only thing CSS can be used for. XML documents, counting plain XML (described in section 2.4.5), XUL4 and SVG5 are other parts the language can be applied to, and to rendering in speech (artificial production of human speech) and other media also.

As mentioned before CSS is used to allow the separation of content and appearance,

---

4 XUL, XML User Interface Language, is a user interface markup language developed by Mozilla.
5 SVG, Scalable Vector Graphics, is a vector image format that is XML-based. It is used for two-dimensional graphics.
counting fonts, layout and colours which are what the language is mainly designed for. When this separation is allowed it improves the content accessibility and arranges for more control and flexibility in the requirements of appearance characteristics. It also reduces the recurrences in the structural content and the complexity within and finally it allows several HTML pages sharing formats through specifications of relevant CSS code in distinct CSS files.

### 2.4.5 XML

XML, short for Extensible Markup Language, is a language that defines the rules for documents encoded, in a format that is readable for both humans and machines. The main goals with the design of XML underline usability, simplicity and generality across the internet. XML is a textual data format with a robust maintenance for different human languages. Even though the language is designed with documents in centre, XML is commonly used for the illustration of arbitrary data structures, for example those which are used in web services. Although many computer operators have developed several APIs to support the handling of XML data, there still exists a numerous of schema systems (a particular type of XML document), to support in the characterisation of languages that are XML-based.

Nevertheless, this language has become a general tool for the interchange of data over the Internet and in this project XML is used when an online spreadsheet is added as a plugin to Pentahos Business Intelligence server.

### 2.4.6 jQuery

The technique used for the web-based spreadsheet is jQuery. Thus, one of the reasons for choosing to work with Wickedgrid is that the spreadsheet has support for this technique as it is an AJAX application. jQuery is a free of charge, open-source, JavaScript library multi-platform software that is developed to simplify HTML’s client-side scripting and the furthermore generally utilized JavaScript library by a great margin [9]. The jQuery library is simply a single JavaScript file holding all of its trivial AJAX functions, event, DOM and effects.

The syntax for jQuery is designed with intention to make it smoother to generate animations, navigate documents, choose DOM elements, improve AJAX applications and manage events. Developers are also provided by jQuery with capabilities to make plugins on top of the JavaScript library. Therefore the developers are allowed to create abstractions, a system for organizing complexity of computer systems, for low-level animation and interaction, high-level widgets and advanced effects. Because of the interchangeable approach to the jQuery library, the construction of powerful dynamic web pages and web applications are enabled.

The Document Object Model, or DOM, is a tree-structure illustration of all essentials of a web page. Furthermore, jQuery is fundamentally a DOM manipulation library;
therefore the syntax for discovering, manipulating and choosing these DOM elements is simplified by jQuery. For instance, jQuery can be used to change one or more of an element’s attribute (such as visibility, colour, etc.), make an element respond to an event (as a mouse click) or in a document finding an element with a certain asset.

However, jQuery also offers an archetype for event management that goes beyond plain DOM element management and selection. The event call-back function characterization and the event task are therefore done in a single step and place in the code.

2.5 Tomcat

The ASF (Apache Software Foundation) has developed an open-source Java servlet container, named Apache Tomcat and often mentioned as Tomcat Server [11]. As a web container, Tomcat is the part of the web server that interacts with Java servlets. It is in charge of handling a servlets lifecycle, mapping a URL to a specific servlet and confirming that the requester of the URL has the accurate access-rights. Moreover, Tomcat applies many Java EE (Java Platform, Enterprise Edition) specifications counting Java servlet, Java EL (Java Unified Expression Language) and Java server pages among other things. Furthermore, Java code can run in a HTTP web server environment provided by Tomcat. As open-source software Tomcat is sustained by an open community of developers.

Tomcat is in this project mainly used to start, restart and stop the BI server. In other words, the BI server is started as a window service. Whenever an alteration to the BI server is done, for example adding a plugin, the server has to be restarted, using Tomcat, in order for the changes to appear. Tomcat keeps logs of the launches to the BI server and after an ended session the logs shows specifications of the aforementioned, such as warnings and errors. This can be used to debug scripts uploaded to the BI server.
3. Method

First a literature study about Pentaho and web-based spreadsheets was done. The literature study regarding the spreadsheets implied looking for already existing web-based spreadsheets and see if they were available in open-source. The ones that existed in open-source were reviewed in order to find the sheet that would be most fitted to combine with a BI analyse in Pentaho.

The central aspects looked for in the sheets were the technology used, the supported programming languages, the level of development and the format or in other words the layout and how similar to familiar spreadsheets, for example Microsoft Excel, they were.

The study regarding Pentaho involved learning more about Pentaho as a business analytics platform and how this platform worked. Readings were done to see if there was any possibility for the desired combination in that environment and if yes, then how.

The planned technique to use for the project was jQuery and therefore Wickedgrid, which is an AJAX spreadsheet, was a suitable open-source spreadsheet to modify and develop further. First the spreadsheet was going to be added to the BI server in order to establish that a combination could be made. The procedure of producing the combination of an online spreadsheet with a BI server could be divided into three steps. These steps are described in the three following sections.

Figure 3. Shows that tomcat has started, meaning that a connection to the BI server is established. This is done inside a terminal on the desktop.
3.1 Step one – Setting up

The first step was to set everything up, which was to download the BI server, modify settings for Tomcat, and adding Wickedgrid to Tomcats “webapps” folder. Modifying the settings for Tomcat involved setting a path for the Java_Home environment variable to the Java Runtime Environment (JRE) in order for Tomcat to run. Then the BI server was started in order to see if it worked. By adding the online spreadsheet to the “webapps” folder it could be accessed through Tomcat but it still didn’t interact with the BI server. (See Fig. 4 below, for a visualised explanation).

![Figure 4. A sketch of the situation in step one. Both the sheet and the BI server can be accessed through Tomcat, but there is still no connection between the two.](image)

3.2 Step two – Adding a plugin

To be able to make the described combination, the spreadsheet had to be integrated into Pentahos BI server. The way this step was accomplished was to add Wickedgrid, the desired spreadsheet, as a plugin into the BI server. In order to add a plugin to the BI server, some specific conditions had to be met. These conditions were that the plugin contained a lib folder with (JAR) files and a plugin.xml file among other things, a more thorough explanation will be provided further on. Since Wickedgrid didn’t fulfil these conditions it had to be modified such that it could be added as a plugin.

In order to be able to add a plugin to the BI server, already existing plugins were studied. The Pentaho community also provided information about how plugins are added [12]. In this case the plugin is added locally, which basically means that the folder that contains the files for the plugin in question is placed in the BI servers “system” folder that is located on the computer.
3.2.1 The ‘lib’ folder

One requirement is to create a ‘lib’ folder. The plugin folder needs to contain a ‘lib’ folder in which the binary (JAR) files needed by the plugin can be placed. This means that libraries necessary for the plugin don’t have to be deployed by the server. Therefore plugin libraries can be moved out of the plugin ‘lib’ folder and be placed in one of the servers ‘lib’ folders as needed.

3.2.2 The ‘plugin.xml’ file

Another requirement is the ‘plugin.xml’ file. This file is needed to define the extension points the plugin has used (See Appendix 1).

In the creation of a ‘plugin.xml’ file some attributes are needed. These are:

- A title – That means the human readable title of the plugin, in this case wiik3 (wiik3 refers to Wickedgrid).
- A name – That is the unique id of the plugin inside the system.
- A loader – Which can either be an overriding loader or a default loader. The loader tells the system whether to use the overriding class loader, which allows the plugin’s ‘lib’ folder to override (JAR) files that are included in the system, or the default class loader. In this case the default class loader was used.

A definition of a lifecycle-listener class in the plugin.xml is needed for the plugin to respond to plugin lifecycle events, such as:

- Loaded – This is an event fired after the plugin has registered with the BI server.
- Unloaded – An event fired once the plugin is about to be unregistered.
- Init – An event fired just before the plugin is about to be registered with the BI server.

An implementation of ‘IPluginLifecycleListener’ needs to be created and configured into the plugin.xml (like; <lifecycle-listener class="my.plugin.MyLifecycleLister"/> see Appendix 1) in order to hook into these events.

3.2.3 The ‘plugin.spring.xml’ file

The plugin.spring.xml is another necessary file when adding a plugin into the BI server. This file handles the internal configurations as well as exposing beans, a type of a generic object that hold information, to the Pentaho system (see Appendix 2).

After meeting the necessary conditions for the plugin Wickedgrid was added to the BI server and the second step was done. Figure 5 below shows the current state after adding the spreadsheet as a plugin to the BI server. (BTable in the figure below is the one described in Section 2.1.1)

---

6 The extension points are interfaces or abstract classes. Those interfaces classify contracts of what needs to be implemented.
3.3 Step three – Forming a combination

The third step in the project was to adjust the web-based spreadsheet such that it could interrelate with the BI server and receive data from the database. This last step could be divided into two smaller parts.

The first part was to create a dashboard for the plugin. This dashboard could be created by using Pentaho’s app builder, where HTML, JS, and CSS scripts were uploaded. The app builder runs these scripts and creates the desired dashboard [13]. The HTML scripts generated the page for the spreadsheet, while the JS and CSS scripts adjust the functions, layout, format, etc. of the sheet (see Section 4.3 for further explanation).

The second part was to construct new and modify already existing HTML, JS and CSS scripts for the spreadsheet in order for it to be able to receive data and analysis directly from the BTable.

Figure 6 shows a sketch of the combination between the spreadsheet and BI server. It also shows how data from the BTable is placed in the spreadsheet. This data is received from a database as seen in this figure.
4. Result

The results for this project are divided after each step described in the method part.

4.1 The setup

First off was the setup part, where the BI server was test started in order to check that it works. Furthermore, the online spreadsheet was also tested by accessing it through Tomcat after placing it in the “webapps” folder, in order to see that the open-source code that was going to be developed really worked. The figures below shows the results for the first part.

Figure 7, shows the Pentaho platform. It can be accessed when the BI server has started. The plugins already existing in Pentaho are seen under the tool tab marked in red in the figure. The goal is to add Wickedgrids (renamed as wiik3 in Pentaho) to the tools.

Figure 8, shows Wickedgrid, or wiik3, when its accessed with Tomcat. This is just to see that there is a sheet to work with. It still requires some modifications in order to work as desired. Furthermore, in this phase it doesn’t interact with the BI server, it’s only available through Tomcat.
Figure 7. The platform for the BI server is shown here. The tools tab marked in red shows the existing plugins so far.

Figure 8. Shows the web-based spreadsheet when it’s accessed through Tomcat. Here it doesn’t yet interact with the BI server.
4.2 The added plugin

The second step was to add the spreadsheet to the BI server. The figures below show the results of the procedures done in the method section regarding the plugin conditions that had to be met.

In Figure 9, it is shown that the spreadsheet, wiik3, now is available under the tool tab. The spreadsheet is now interacting with Pentaho and can be accessed through the BI server platform.

Figure 9. Shows the Pentaho platform after the plugin has been added to the BI server. The spreadsheet named wiik3, marked in red in the figure, is now shown under the tool tab.

Figure 10, shows what happens when the spreadsheet is tried to be accessed through the BI server, by clicking on “wiik3” under the tool tab shown above (see Figure. 9). As Figure 10 shows, there’s nothing there. That’s because the plugin required a dashboard, which was managed in the last step of the process described in the method section.
4.3 The combination

As mentioned in the method section the last step of the process could be divided into two smaller parts. One part was to construct the dashboard for the spreadsheet in order for the sheet to appear inside the BI server. The second part was to modify the spreadsheet with the purpose of making it possible to place a BI analysis inside the sheet and make it possible to perform a “what if” – analysis. Unfortunately, because of obstacles encountered during the process and restricted time, the second part of the last step was never accomplished. A more thorough explanation will be covered in the discussion part.

4.3.1 The combination – First part

In this part the dashboard for the spreadsheet was created. Pentaho’s BI server works in a certain way and in order to get a visualisation of the spreadsheet within the server a dashboard was created through Pentaho’s app builder, which is a plugin creator instrument. The HTML, JS and CSS files that generate the online spreadsheet are placed in the app builder which creates a dashboard suitable for the BI server.

Figure 11 shows the results after creating the dashboard for the spreadsheet. As seen in the figure below the sheet isn’t appearing as it did in Figure 8 (see results 4.1). The reasons for this will be discussed in the next section.
Figure 11. Shows the result when accessing the spreadsheet through the BI server after a dashboard is created.

5. Discussion & Conclusion

The purpose of the project was to combine a web-based spreadsheet with an analysis from a BI server. The literature study that was done showed that this type of combination is possible and that it does exist, however they are liable to charge. This combination was going to be free of charge, available as an open-source. The main purpose of a combination like this is to be able to make “what if” - analysis. That is a type of a scenario analysis, where the outcomes of changes made to a data are shown straight after. This allows for the analysis of future events, in order to answer questions like, what happened, how regularly, how many, where the difficulties are and what action is required. Tools for predicting and optimising the outcome are truly useful in the business world.

5.1 The plugin

Understanding Pentaho and how it works was necessary in order to be able to make the desired combination. The BI server works in a specific way and knowledge about the platform is required in order to understand how to proceed with the construction. For example, Pentaho had its own setup for adding a plugin. It included scripting several XML files, but also JS, CSS and JAR files. However, it didn’t exist any debugging
program and therefore errors in the scripts couldn’t be found until the BI server was started and the scripts were run. The errors could then be located by the aid of Tomcat logs. Tomcat kept logs for every launching of the BI server that showed what happened during the process, for example, it showed if and where warnings and errors occurred. Debugging the scripts like this was time consuming.

Secondly, Pentaho used certain formats for retrieving and delivering data, such as XML, XLS\(^7\), CSV\(^8\) or JSON. The spreadsheet however was constructed with HTML, JS and CSS files and because of that it didn’t appear inside the BI server when it first was added as a plugin. This was because the files for the spreadsheet couldn’t be read by the BI server.

### 5.2 The combination

Before creating a dashboard with Pentaho’s app builder, another method was tried. That method included converting the HTML, JS and CSS files that constructed the spreadsheet, into files which were readable for Pentaho. That didn’t succeed and the app builder became the next solution to be tried. The acquired result of that was seen in Figure 11, in Section 4.3.1. However, the desired result can be seen in Figure 12 below.

![Figure 12](image)

*Figure 12. Shows the desired results for the combination of the spreadsheet and the BI server.*

As seen in Figure 11 and Figure 12 there are some differences between them, parts of what is shown in Figure 12 is missing in Figure 11. The reason for that, for the acquired result to be so different from the desired result, is that the JS and CSS files are never

\(^7\) XLS, eXcel Spreadsheet, is a file format developed by Microsoft for operations with Microsoft Excel.

\(^8\) CSV, Comma-Separated Values, files supplies text and numbers in pure text.
found in the constructed dashboard. The parts that are seen missing in Figure 11 are the parts of the script that the JS and CSS files generate.

HTML code is written inside the app builder when creating the dashboard. Inside this HTML code there are calls for particular JS and CSS functions. However, the app builder inside Pentaho doesn’t manage to locate the search path for the requested functions and proceeds without them. Thereby the result acquired is what is shown in Figure 11 and not what’s shown in Figure 12.

The next step in the process became trying to locate the search path for the JS and CSS functions.

5.3 Future work

The desired combination seems to be achievable. After locating the search paths for the JS and CSS files a combination of a web-based spreadsheet and a BI server would be created.

The next step would be to use jQuery and modify the spreadsheet in order for it to be able to retrieve an analysis from the BI server and moreover be able to perform “what if” - analysis. Additionally, some modifications of the spreadsheet are required in order for it to function properly. Thereafter a database could be connected to the BI server to provide the spreadsheet with data for analysing.

Later on the future work of this product could involve managing to adapt it to smartphones and tablets. Since the technique used for the spreadsheet is jQuery, the adaption should be achievable, as jQuery eliminates cross-browser incompatibilities.
Reference


Appendices 1 – The ‘plugin.xml’ file

<?xml version="1.0" encoding="UTF-8"?>
<plugin title="wiik3" loader="DEFAULT">
    <overlays>
        <overlay id="startup.wiik3"
            resourcebundle="content/wiik3/resources/lang/messages">
            <menubar id="toolsmenu">
                <menuitem id="wiik3" label="${Launcher.wiik3}"
                    command="mantleXulHandler.openUrl('${Launcher.wiik3}',('${Launcher.wiik3_TOOLTIP}','plugin/wiik3/api/default')")"/>
            </menubar>
        </overlay>
    </overlays>

    <!--lifecycle-lisener class="wiik3.MyLifecycleListener"/-->
    <lifecycle-listener class="pt.webdetails.cpk.CpkLifecycleListener"/>

    <content-types>
        <static-paths>
            <static-path url="/wiik3/res" localFolder="resources"/>
            <static-path url="/wiik3/js" localFolder="plugins"/>
            <static-path url="/wiik3/JSClass" localFolder="chart/JSClass"/>
            <static-path url="/wiik3/swf" localFolder="chart/charts"/>
            <static-path url="/wiik3/widgets" localFolder="widgets"/>
            <static-path url="/wiik3/static" localFolder="static"/>
            <static-path url="/wiik3/resources" localFolder="resources"/>
        </static-paths>
        <content-type>
            <title>wiik3</title>
            <description></description>
        </content-type>
        <content-type type="wiik3" mime-type="text/html">
            <title>Wiik3 Config</title>
            <description>Wiik3 Config File</description>
            < Operations>
                <operation>
                    <id>RUN</id>
                </operation>
                <operation>
                    <id>NEWWINDOW</id>
                </operation>
            </Operations>
        </content-type>
    </content-types>
    <content-generator
        id="wiik3"
        title="wiik3"
        type="wiik3"
        class="pt.webdetails.cpk.CpkContentGenerator"/>
</plugin>
Appendices 2 – The 'plugin.spring.xml' file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ws="http://jax-ws.dev.java.net/spring/core"
    xmlns:wss="http://jax-ws.dev.java.net/spring/servlet"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-2.5.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util-2.5.xsd
    http://jax-ws.dev.java.net/spring/core.xsd
    http://jax-ws.dev.java.net/spring/servlet.xsd">
    <context:annotation-config/>
    <!-- REST resources -->
    <bean id="api" class="org.Pentaho.platform.web.servlet.JAXRSPluginServlet"/>
    <bean id="wiik3.api" class="pt.webdetails.cpk.CpkApi"/>
    <context:annotation-config/>
    <!-- Content Generator -->
    <bean id="cpk" class="pt.webdetails.cpk.CpkContentGenerator" scope="prototype"/>
    <!-- Content Generator -->
    <bean id="wiik3" class="it.biztech.wiik3.wiik3ContentGenerator" scope="prototype"/>
    <bean id="file" class="it.biztech.wiik3.api.FileApi"/>
    <bean id="olap" class="it.biztech.wiik3.api.OlapApi"/>
</beans>
```
Appendices 3 – The 'cpk.xml', ‘cpk.spring.xml’ and ‘settings.xml’ files

**cpk.xml:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<cpk>
    <elementTypes defaultElement="main">
        <elementType name="Dashboard" class="pt.webdetails.cpk.elements.impl.DashboardElement">
            <elementLocations>
                <elementLocation path="dashboards/admin" isRecursive="true" pattern=".*\.wcdf" adminOnly="true"/>
                <elementLocation path="dashboards/" isRecursive="true" pattern=".*\.wcdf" adminOnly="false"/>
            </elementLocations>
        </elementType>
        <elementType name="Kettle" class="pt.webdetails.cpk.elements.impl.KettleJobElement">
            <elementLocations>
                <elementLocation path="endpoints/kettle/admin" isRecursive="true" pattern=".*\.kjb" adminOnly="true"/>
                <elementLocation path="endpoints/kettle/" isRecursive="true" pattern=".*\.kjb" adminOnly="false"/>
            </elementLocations>
        </elementType>
        <elementType name="Kettle" class="pt.webdetails.cpk.elements.impl.KettleTransformationElement">
            <elementLocations>
                <elementLocation path="endpoints/kettle/admin" isRecursive="true" pattern=".*\.ktr" adminOnly="true"/>
                <elementLocation path="endpoints/kettle/" isRecursive="true" pattern=".*\.ktr" adminOnly="false"/>
            </elementLocations>
        </elementType>
    </elementTypes>
    <info>
        <CPK_version>4</CPK_version>
        <description>wik3 - A Drill-Anywhere Component For CDE</description>
        <name>wik3</name>
        <author_email></author_email>
        <author_name></author_name>
        <company_name>Biz Tech</company_name>
        <company_url>www.biztech.it</company_url>
        <creation_date>2017-03-02</creation_date>
        <version>1.5</version>
    </info>
</cpk>
```
**cpk.spring.xml:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ws="http://jax-ws.dev.java.net/spring/core"
      xmlns:wss="http://jax-ws.dev.java.net/spring/servlet"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
                        http://www.springframework.org/schema/context
                        http://www.springframework.org/schema/context/spring-context-2.5.xsd
                        http://www.springframework.org/schema/util
                        http://www.springframework.org/schema/util/spring-util-2.5.xsd
                        http://jax-ws.dev.java.net/spring/core
                        http://jax-ws.dev.java.net/spring/servlet"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd"
      xsi:schemaLocation="http://www.springframework.org/schema/context
                        http://www.springframework.org/schema/context/spring-context-2.5.xsd"
      xsi:schemaLocation="http://www.springframework.org/schema/util
                        http://www.springframework.org/schema/util/spring-util-2.5.xsd"
      xsi:schemaLocation="http://jax-ws.dev.java.net/spring/core
                        http://jax-ws.dev.java.net/spring/servlet">
  <context:annotation-config />
  <bean id="ICpkEnvironment"
        class="pt.webdetails.cpk.CpkPentahoEnvironment" scope="prototype"/>
  <bean id="IPluginCall"
        class="pt.webdetails.cpf.InterPluginBroker"
        scope="prototype"/>
</beans>
```

**settings.xml:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<settings>
  <cde-styles>
    <path>system/wiik3/resources/styles</path>
  </cde-styles>
  <cde-datasources/>
  <cde-components/>
  <!-- paths from solution with cde component definitions to be loaded-->
  <path>system/wiik3/resources/components</path>
  <path>wiik3/components</path>
</cde-components>
</settings>
```