DashXML

Overview

top
DashXML is a Java web application that communicates with Yellowfin via Web Services. It can be deployed alongside Yellowfin, or on a separate server.

The DashXML framework provides a method for creating dashboards that combine Yellowfin content and custom items without the need for programming. The dashboards are configured in a single XML document that describes each element on each tab as a widget. All layout and styling can be achieved via customised CSS.

Skill Prerequisites

DashXML is designed to be implemented by a highly advanced user. This user would either be a developer, or someone with a strong technical background.

In order to implement a DashXML solution the user would require the following technical skills as a **minimum**:

Implementation	Skills	Description
Basic	• XML • HTML • CSS	This implementation would use reports (tables, charts, multi-charts, maps) called from a Yellowfin instance, with custom images and styling applied. This would **not** include the use of advanced CustomHTML widgets that use custom JavaScript, only purely HTML.
Basic + Custom Widgets	• XML • HTML • CSS • JavaScript	This implementation would use reports (tables, charts, multi-charts, maps) called from a Yellowfin instance, with custom images and styling applied. This could also include the use of advanced CustomHTML widgets that use custom JavaScript and HTML. CustomHTML widgets can be used to pass Yellowfin report results through to other visualisation libraries.
Basic + Third Party Security	• XML • HTML • CSS • Java	This implementation would use reports (tables, charts, multi-charts, maps) called from a Yellowfin instance, with custom images and styling applied. This could also include the use of a Java Servlet Filter to apply source filters based on external user credentials, while still running reporting content as the nominated Yellowfin user (from the web.xml file).

Security

No user security is offered out-of-the-box with DashXML. Third party user security can be implemented using a Java Servlet Filter. This can test access for a user for a given dashboard, apply appropriate source filters for that user, and run the reports as the nominated user specified in the web.xml file.

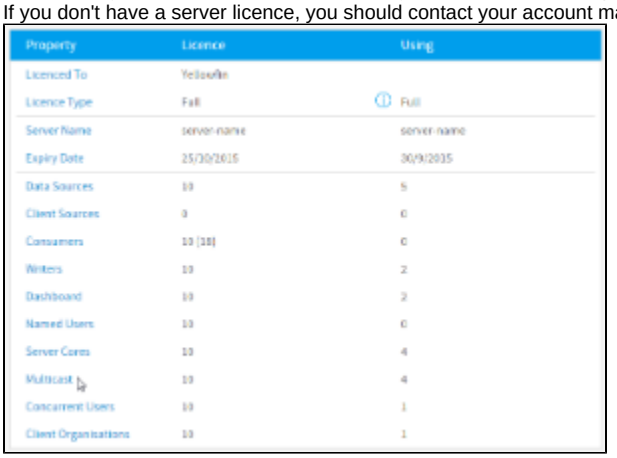
Source filter can be passed for a particular session to enable row-level security. Source filters will need to be defined in the Yellowfin interface, and then referenced within the DashXML.

Limitations

DashXML does not replicate all functionality found in the Yellowfin interface. Some of the functions that can't be replicated within DashXML include:

- Drill Down (on Tables) and Linked Drill Down
- Drill Anywhere (on both Tables or Charts)
- Drill Through (on both Tables or Charts)
- Brushing and Linked Brushing
- Series Selection and Linked Series Selection
- Interaction and Tooltips on Multi-Chart Canvases
- Animated Chart Loading
- Time Series Date Slider and Unit Selection
- Column Formatters (such as Flag Formatter)
- Selectable Layers on Maps
- Collaboration
- Client Organisations
- Yellowfin Dashboards
- Yellowfin Storyboards

Installation

1. Ensure that your version of Yellowfin currently has a server licence (with the correct number of server cores specified).
   If you don't have a server licence, you should contact your account manager.
   

2. Download DashXML from the Support Centre page of the Yellowfin website.
   If you don't have access to this page, you should contact your account manager.
   

3. Ensure your version of Yellowfin is running the latest 7.1 patch.
   Yellowfin releases monthly patches, visit the Support Centre to download the latest.
   Note: early versions of Yellowfin 7.1 (or releases prior to 7.1) will not be compatible with the DashXML framework.

4. Stop your Yellowfin instance.
   There are some changes to files related to your Yellowfin instance that you will have to update, and in order for the changes to be implemented correctly, Yellowfin must not be running.

5. Edit your Yellowfin web.xml file.
   Navigate to `\Yellowfin\appserver\webapps\ROOT\WEB-INF\` and edit the web.xml file. You will need to uncomment the following section in order to enable Web Services.

Update your file from this:	To this:
<!-- Uncomment this section to enable JAX-WS Web Services. Java 1.6+ only <listener> <listener-class> com.sun.xml.ws.transport.http.servlet.WSServletContextListener </listener-class> </listener> <servlet> <servlet-name>WebServices</servlet-name> <servlet-class> com.sun.xml.ws.transport.http.servlet.WSServlet </servlet-class> </servlet> <servlet-mapping> <servlet-name>WebServices</servlet-name> <url-pattern>/webservices/*</url-pattern> </servlet-mapping><br-->	<!-- Uncomment this section to enable JAX-WS Web Services. Java 1.6+ only --> <listener> <listener-class> com.sun.xml.ws.transport.http.servlet.WSServletContextListener </listener-class> </listener> <servlet> <servlet-name>WebServices</servlet-name> <servlet-class> com.sun.xml.ws.transport.http.servlet.WSServlet </servlet-class> </servlet> <servlet-mapping> <servlet-name>WebServices</servlet-name> <url-pattern>/webservices/*</url-pattern> </servlet-mapping>

   Save your changes.

6. Unzip your DashXML file.
   You are able to install DashXML two ways:
   a. into your Yellowfin directory, so that it runs on the same Tomcat instance, or
   b. into a completely separate Tomcat instance which may sit on a different server to Yellowfin if required.

Yellowfin Tomcat	Separate Tomcat
Copy the DashXML folder (from your downloaded zip) into your `\Yellowfin\appserver\webapps\` directory. You should now have both a `ROOT` and `DashXML` directory sitting within `webapps`.	Copy the DashXML folder (from your downloaded zip) into your `\Tomcat\webapps\` directory. You will now have a DashXML directory sitting within webapps.

7. Edit your DashXML web.xml file.
   Navigate to `\webapps\DashXML\WEB-INF\` and edit the web.xml file. You will need to make three changes to this file, in order for DashXML to be able to connect to your Yellowfin instance.
   a. Set your Yellowfin URL.

```
<init-param>
  <param-name>PathToServer</param-name>
  <param-value>http://server-name:7171</param-value> <!-- update to Yellowfin server -->
</init-param>
```

   b. Set your Yellowfin username.

```
<init-param>
  <param-name>Username</param-name>
  <param-value>admin@yellowfin.com.au</param-value> <!-- update to Yellowfin username -->
</init-param>
```

   c. Set your Yellowfin password.

```
<init-param>
  <param-name>Password</param-name>
  <param-value>test</param-value> <!-- update to Yellowfin password -->
</init-param>
```

8. Start Yellowfin.

Getting Started

In order to get started with DashXML walk through the following steps to familiarise yourself with the development process.

1. Edit your dashboard.xml file.
   Navigate to `\DashXML\WEB-INF\` and edit the dashboard.xml file. This is the file you will use to build your dashboards. To start with, all you need is this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application-definition>

        <!-- Start Dashboard -->
    <dashboard>
        <id>1</id>


        </dashboard>

</application-definition>
```

2. Edit your dashboard.css file.
   Navigate to `\DashXML\css\` and edit the dashboard.css file. This is the file you will use to style your dashboards. To start with, it will look like this:

```
html {
        height: 100%;
}
body {
        font-family: sans-serif;
        background-color: #F5F5F5;
        background-image: url("../images/Background.jpg");
        background-size: 100% 100%;
        background-repeat: no-repeat;
        background-attachment: fixed;
        width: 100%;
        margin: 0px;
        position: relative;
        height: 100%;
}
.main {
        width: 1275px;
        margin: auto;
        border: 0px solid #FFFFFF;
        background-color: #F5F5F5;
        padding: 0px;
        position: absolute;
        left: 0px;
        right: 0px;
        height: auto;
        border-right: 3px solid rgba(51, 51, 51, 0.5);
        border-left: 3px solid rgba(51, 51, 51, 0.5);
        border-bottom: 3px solid rgba(51, 51, 51, 0.5);
    -webkit-background-clip: padding-box; /* ensures transparency in border for Safari */
    background-clip: padding-box; /* ensures transparency in border for IE9+, Firefox 4+, Opera,
Chrome */
}
.logo{
        position: relative;
        top: 0px;
        left: 0px;
        width: 200px;
}
.copyright {
        font-size: 11px;
        color: #555555;
```

```
        text-align: center;
        float: left;
        width: 100%;
}
/* Report Styles */
.reportDisplay {
        position: relative;
        page-break-inside: avoid;
}
.reportHtml {
        height: 100%;
        width: 100%;
}
.reportTitle {
        padding: 0px 0px 0px 0px;
        color: #555555;
        font-size: 24px;
        text-align: center;
}
.reportHolder {
        height: 450px;
        margin-bottom: 50px;
}
.banner {
        width: 1024px;
        float: left;
        overflow: hidden;
        margin-bottom: 25px;
}
.wide {
        width: 1024px;
        float: left;
        overflow: hidden;
}
.wide .reportHolder, .halfSummary .reportHolder {
        height: 300px;
}
.half{
        width: 510px;
        float: left;
        overflow: hidden;
}
.halfSummary{
        width: 510px;
        float: left;
        overflow: hidden;
        height: 400px;
}
.textWidget {
        color: #666666;
}

/* sub tabs */
.subTabs {
        clear: both;
}
/* Sub Tab Button Selectors */
.subTabSelection {
        width: 200px;
        display: table;
        border-collapse: separate;
        table-layout: fixed;
        overflow: hidden;
        position: relative;
        border-spacing: 0px 1px;
        background-color: #CCCCCC;
        float: left;
        height: 100%;
}
.subTabContainer {
        float: left;
```

```
        width: 1023px;
        padding: 25px 25px 10px 25px;
        margin-top: -230px;
        border-left: 1px solid #CCCCCC;
        background-color: #FFFFFF;
}
.subTabSelection .subTabSelector {
        display: table-row;
        background: #F5F5F5;
        cursor: pointer;
        position: relative;
}
.subTabSelection .subTabMain {
        height: 90px;
        overflow: hidden;
        position: relative;
        padding: 10px;
}
.subTabSelection  .selected {
        background: #FFFFFF;
}
.subTabSelection .subTabTitle{
        color: #555555;
        font-size: 18px;
        line-height: 22px;
        text-align: center;
        margin-top: 65px;
}
.subTabSelection .subTabIcon {
        position: absolute;
        top: 6px;
        left: 45px;
        width: 110px;
        text-align: center;
}
/* Inactive Icons */
.subTabSelection .subtabCompactIcon, .subTabSelection .subtabCompactIcon:after{
        content: url("../images/Compact_Off.png");
}
.subTabSelection .subtabHybridIcon, .subTabSelection .subtabHybridIcon:after{
        content: url("../images/Hybrid_Off.png");
}
.subTabSelection .subtabMiniIcon, .subTabSelection .subtabMiniIcon:after{
        content: url("../images/Minivan_Off.png");
}
.subTabSelection .subtabSedanIcon, .subTabSelection .subtabSedanIcon:after{
        content: url("../images/Sedan_Off.png");
}
.subTabSelection .subtabSuvIcon, .subTabSelection .subtabSuvIcon:after {
        content: url("../images/SUV_Off.png");
}
.subTabSelection .subtabSummaryIcon, .subTabSelection .subtabSummaryIcon:after{
        content: url("../images/Summary_Off.png");
}
/* Active/Selected Icons */
.subTabSelector.selected .subtabCompactIcon, .subTabSelector.selected .subtabCompactIcon:after {
        content: url("../images/Compact_On.png");
}
.subTabSelector.selected .subtabHybridIcon, .subTabSelector.selected .subtabHybridIcon:after{
        content: url("../images/Hybrid_On.png");
}
.subTabSelector.selected .subtabMiniIcon, .subTabSelector.selected .subtabMiniIcon:after{
        content: url("../images/Minivan_On.png");
}
.subTabSelector.selected .subtabSedanIcon, .subTabSelector.selected .subtabSedanIcon:after{
        content: url("../images/Sedan_On.png");
}
.subTabSelector.selected .subtabSuvIcon, .subTabSelector.selected .subtabSuvIcon:after{
        content: url("../images/SUV_On.png");
}
.subTabSelector.selected .subtabSummaryIcon, .subTabSelector.selected .subtabSummaryIcon:after{
```

```
        content: url("../images/Summary_On.png");
}
/* Map Navigation Buttons */
.mapNav {
        display: block !important;
        position: relative;
        width: 120px;
        height: 90px;
        right: 10px;
        bottom: 160px;
        color: #393737;
        text-align: center;
        float: right;
}
.mapUp {
        position: absolute;
        top: 6px;
        right: 40px;
        content: url("../images/mapNavUp.png");
        cursor: pointer;
}
.mapDown {
        position: absolute;
        top: 59px;
        right: 40px;
        content: url("../images/mapNavDown.png");
        cursor: pointer;
}
.mapLeft {
        position: absolute;
        top: 7px;
        right: 92px;
        content: url("../images/mapNavLeft.png");
        cursor: pointer;
}
.mapRight {
        position: absolute;
        top: 7px;
        right: 39px;
        content: url("../images/mapNavRight.png");
        cursor: pointer;
}
.mapZoomIn {
        position: absolute;
        top: 6px;
        right: 3px;
        content: url("../images/mapZoomIn.png");
        cursor: pointer;
}
.mapZoomOut {
        position: absolute;
        top: 49px;
        right: 3px;
        content: url("../images/mapZoomOut.png");
        cursor: pointer;
}

/* Report Loading */
.loadingDiv {
    position: absolute;
    width: 100%;
    height: 100%;
    background-color: #F5F5F5;
    display: block;
    top: 0px;
    left: 0px;
        z-index: 1000;
}
.loadingDiv:before {
    content: "\f110";
        font-family: FontAwesome;
```

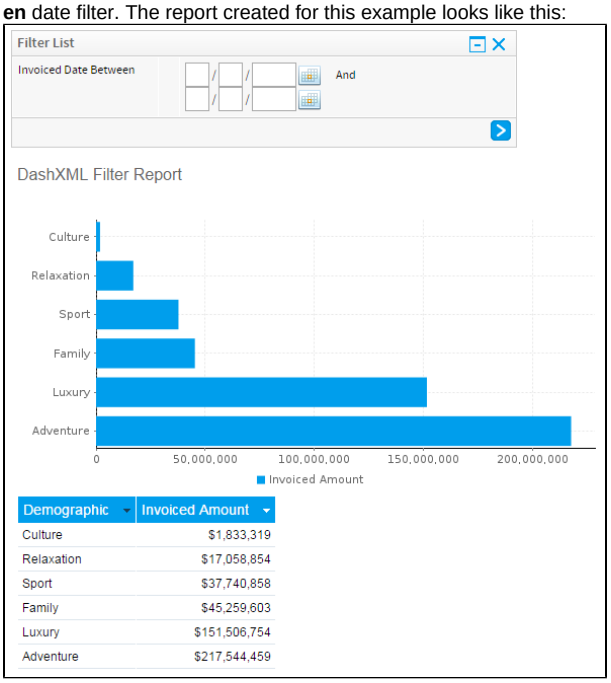
```
        font-style: normal;
        font-weight: normal;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
        font-size: 22px;
        color: #444444;
        -webkit-animation: fa-spin 2s infinite linear;
    animation: fa-spin 2s infinite linear;
        position: absolute;
        top: calc(50% - 10px);
        left: calc(50% - 50px);
}
.loadingDiv:after {
    content: "Loading";
        color: #444444;
        font-size: 22px;
        font-family: sans-serif;
        margin-left: 30px;
        position: absolute;
        top: calc(50% - 10px);
        left: calc(50% - 50px);
}
@-webkit-keyframes fa-spin {
  0% {
    -webkit-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(359deg);
    transform: rotate(359deg);
  }
}
@keyframes fa-spin {
  0% {
    -webkit-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(359deg);
    transform: rotate(359deg);
  }
}

/* Popularity Slider */
.popularityRows {
        padding-top: 15px;
        width: 510px;
}
.popularityRowSUV, .popularityRowcompact, .popularityRowsedan, .popularityRowhybrid, .
popularityRowmini-van {
        text-align: right;
        width: 100%;
}
.dimName {
        float: left;
        color: #555555;
        font-size: 20px;
        margin-right: 15px;
        width: 80px;
        margin-top: 3px;
}
.description {
        text-align: right;
    clear: both;
        color: #555555;
        font-size: 14px;
        line-height: 22px;
        padding-bottom: 15px;
        width: 500px;
}
.loadingBar {
```

```
        float:left;
        width: 410px;
    height: 10px;
    background-color: #FFFFFF;
        border-radius: 3px;
        margin-top: 10px;
        box-shadow: 1px 1px #999999;
}
.progressBar {
    height:10px;
    background-color: #CCCCCC;
        border-left: 1px solid #CCCCCC;
        border-right: 0px;
        border-top-left-radius: 3px;
        border-bottom-left-radius: 3px;
    float:left;
        margin-left: -1px;
        box-shadow: 1px 1px #999999;
}

.handle {
    width: 18px;
    height: 18px;
    margin-top:-4px;
        margin-left: -4px;
    float:left;
        border-radius: 10px;
        box-shadow: 3px 1px #999999;
        background-color: #0051A4;
}
```

3. Locate a report in Yellowfin.
   In this scenario we're going to create a dashboard with a single report with a date filter. Run Yellowfin and find or create a report with a **betwe en** date filter. The report created for this example looks like this:



4. Find the Filter and Report UUIDs.
   DashXML references content components via their system UUIDs. In order to locate these for our example you will need to:
   a. Run the report after applying some filter values (any value will do, just don't leave them blank).
   b. Open the Report Details window.
   c. Copy the Report URL
   d. Paste the Report URL into a text file and locate the UUIDs as follows:

URL	UUID	Description

http://localhost/RunReport.i4?reportUUID=39fc0f87-77c8-4a5b-aec3-2a04ed78bae3&primaryOrg=1&clientOrg=1&filterUUID9d1dc0bb-ab9a-4852-b408-a9a6b94af66a=2015-09-01%5C%7C2015-09-30		This is the full Report URL. From here you can locate the Report UUID and any Filter UUIDs you need.
http://localhost/RunReport.i4?reportUUID=**39fc0f87-77c8-4a5b-aec3-2a04ed78bae3**&primaryOrg=1&clientOrg=1&filterUUID9d1dc0bb-ab9a-4852-b408-a9a6b94af66a=2015-09-01%5C%7C2015-09-30	39fc0f87-77c8-4a5b-aec3-2a04ed78bae3	This is the Report UUID, which will be referenced in the DashXML in order to display the results of the report on the dashboard.
http://localhost/RunReport.i4?reportUUID=39fc0f87-77c8-4a5b-aec3-2a04ed78bae3&primaryOrg=1&clientOrg=1&filterUUID**9d1dc0bb-ab9a-4852-b408-a9a6b94af66a**=2015-09-01%5C%7C2015-09-30	9d1dc0bb-ab9a-4852-b408-a9a6b94af66a	This is the Filter UUID, which will be referenced in the DashXML in order to pass filter values through to the report displayed on the dashboard.

5. Develop your dashboard in the dashboard.xml file.
   Here we need to complete the following in order to create a simple dashboard using our sample report and filter.
   a. Set up the Dashboard
      To begin, you will need the `<application-definition>` and `<dashboard>` elements. As we're creating our first dashboard, the ID can be '1'.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application-definition>

        <!-- Start Dashboard -->
    <dashboard>
        <id>1</id>

        </dashboard>

</application-definition>
```

   b. Prepare the User Prompt date filter
      In order to ask the user for filter values you will need to include the `<filter>` element. What we've done in this example is set up four possible options for the user for a date filter; All Time, Year, Month, Custom.

      The **All Time** filter has been defined within the XML to cover a 200 year range, effectively covering all data in our database. The **Year** and **Month** filters have been set using predefined ranges within the XML that will be dynamically set in relation to the current date when the report is run. The **Custom** range has been left as completely User Prompt, meaning that the user will be provided with date pickers to choose their start and end dates.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application-definition>
        <!-- Start Dashboard -->
    <dashboard>
        <id>1</id>
                <!-- Year Filters -->
        <filter>
            <sync>true</sync>
            <datatype>DATE</datatype> <!-- This is the data type expected for the filter, in
this case we're looking for dates. -->
            <name>dateFilter</name> <!-- This is the name of the filter, to be referenced by
reports in their filterMap elements -->
            <operator>BETWEEN</operator> <!-- This is the date filter operator that we used in
Yellowfin. This means we are expecting a start and end date. -->
            <styleClass>topFilter</styleClass> <!-- This is the CSS class name we've assigned
to the filter for styling purposes from our dashboard.css file. -->
                        <options>  <!-- This is the first option available to the user - a
hard coded date range to cover all dates. -->
                                <title>All Time</title>
                                <value>1900-01-01\|2100-01-01</value>
                                <defaultOption>true</defaultOption>
                        </options>
                        <options>  <!-- This is the second option available to the user - a
predefined range that covers the year to date based on current date. -->
                                <title>Year</title>
                                <typeCode>PREDEF</typeCode>
                                <value>YEARTODATE</value>
                        </options>
                        <options>  <!-- This is the third option available to the user - a
predefined range that covers the month to date based on the current date.-->
                                <title>Month</title>
                                <typeCode>PREDEF</typeCode>
                                <value>MONTHTODATE</value>
                        </options>
                        <options>  <!-- This is the last option available to the user - a user
prompt date calendar picker where they can define two custom dates. -->
                                <title>Custom</title>
                                <prompt>true</prompt>
                        </options>
        </filter>

        </dashboard>

</application-definition>
```

c. Set up the Report and Filter Map

In order to display the report and pass through the filter values that the user selects you will have to include the `<report>` and `<filterMap>` elements. We've used the UUIDs we identified earlier in order to populate the elements, defined the report to display as a CHART, and styled it with the canvasChart class from our css file. The filter map has linked the filter UUID we found on the report, to the dateFilter that we defined in the previous step.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application-definition>
        <!-- Start Dashboard -->
    <dashboard>
        <id>1</id>
                <!-- Year Filters -->
        <filter>
            <sync>true</sync>
            <datatype>DATE</datatype>
            <name>dateFilter</name>
            <operator>BETWEEN</operator>
            <styleClass>topFilter</styleClass>
                        <options>
                                <title>All Time</title>
                                <value>1900-01-01\|2100-01-01</value>
                                <defaultOption>true</defaultOption>
                        </options>
                        <options>
                                <title>Year</title>
                                <typeCode>PREDEF</typeCode>
                                <value>YEARTODATE</value>
                        </options>
                        <options>
                                <title>Month</title>
                                <typeCode>PREDEF</typeCode>
                                <value>MONTHTODATE</value>
                        </options>
                        <options>
                                <title>Custom</title>
                                <prompt>true</prompt>
                        </options>
        </filter>

                <!-- Canvas Chart -->
                <report>
                        <display>CHART</display>  <!-- This is the display type for the
report. CHART means that the report will be displayed as a chart. -->
                        <uuid>39fc0f87-77c8-4a5b-aec3-2a04ed78bae3</uuid>  <!-- This is the
report UUID used to retrieve the report results from Yellowfin. -->
                        <styleClass>canvasChart</styleClass>  <!-- This is the CSS class we've
assigned to the filter for styling purposes from our dashboard.css file. -->
                        <filterMap>
                                <filterUUID>9d1dc0bb-ab9a-4852-b408-a9a6b94af66a</filterUUID>
<!-- This is the filter UUID used to pass filter values to the correct filter on the report in
Yellowfin. -->
                                <mapToFilter>dateFilter</mapToFilter>  <!-- This is the name
of the filter the values are coming from, defined in the filter element earlier. -->
                        </filterMap>
                </report>

        </dashboard>

</application-definition>
```

Save your changes.

d. Load the DashXML content.
Yellowfin stores the DashXML content definitions from the dashboard.xml file in memory. In order to ensure your changes are used you need to refresh this by running the `reloadcontent.jsp` file. To do this load the following URL in your browser: http://<location of Yellowfin>/DashXML/reloadcontent.jsp so for example: **http://localhost:8080/DashXML/reloadcontent.jsp**

If run correctly, you will get the following message:

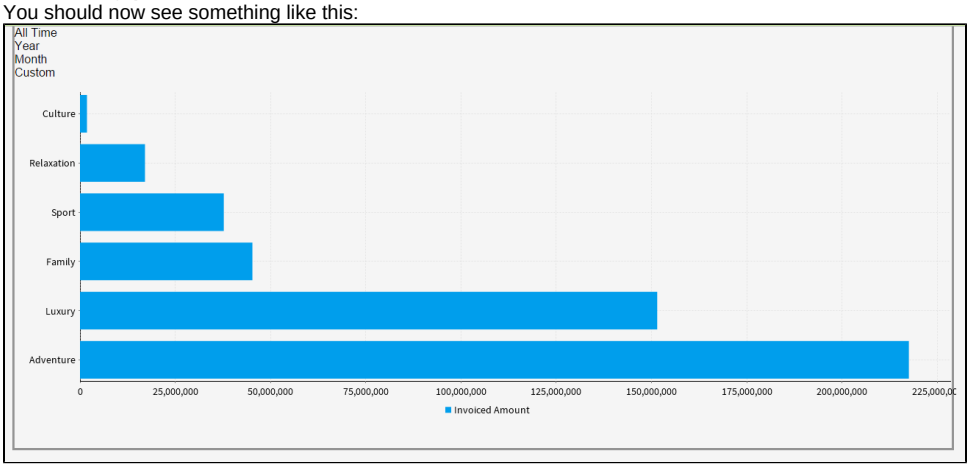
Reloading Content...
Content Loaded Successfully and deployed.

If run incorrectly, you will get an error. This one means the UUID is incorrect:

Reloading Content...
Content had errors

0) The report 39fc0f87-77c8-4a5b-aec3-2a04ed78bae3 set in the configuration file was not found in the Yellowfin server.

e. Load your DashXML Dashboard.
To view your dashboard load the following URL in your browser:
http://<location of Yellowfin>/DashXML/dashboard.jsp?id=<dashboard ID number> so for example: **http://localhost:80/DashXML/dashboard.jsp?id=3**
You should now see something like this:



As you can see, the dates need some formatting.

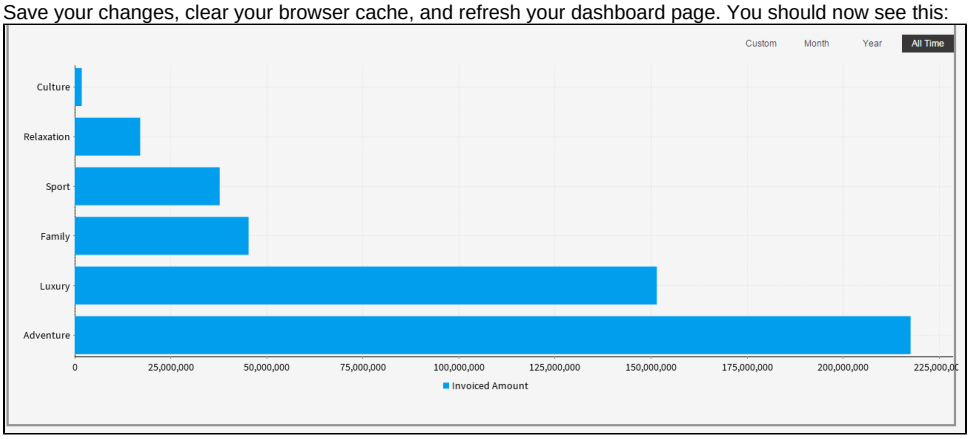
f. Apply styles to the filters.
Return to your dashboard.css file and apply the following styles. We found the classes required through dev tools within our browser.

```
/* Top Nav Filters */
.topFilter {
        height: 24px;
        line-height: 24px;
        padding-top: 14px;
        font-size: 12px;
}
.topFilter .filterOption {
        text-align: center;
        float: right;
        color: #393737;
        cursor: pointer;
        width: 70px;
        margin-left: 5px;
}
.topFilter .selected {
        background-color: #393737;
        color: #FFFFFF;
}
.topFilter .filterOption .customEntry {
        background-color : #393737;
        color: #FFFFFF;
        position: absolute;
        overflow: hidden;
        z-index: 100;
        margin-top: 5px;
        width: 115px;
}
.topFilter .filterOption .customEntry  input{
        background-color : #FFFFFF;
        border: 0px;
        font-family: arial, sans-serif;
        font-size: 12px;
        color: #393737;
        height: 24px;
        width: 70px;
        line-height: 24px;
        margin: 12px 12px 0px 12px;
        padding: 0px 10px;
```

```
        text-align: center;
}
/* Date Entry */
.submitCustom {
        float: left;
        line-height: normal;
        margin: 10px 3px 10px 12px;
}
.cancelCustom {
        float: right;
        line-height: normal;
        margin: 10px 12px 10px 3px;
}
/* Date Picker */
.ui-datepicker {
    background-color : #FFFFFF;
    border: 1px solid #393737;
        color: #393737;
        font-family: arial, sans-serif;
        font-size: 12px;
        margin-top: 5px;
}
.ui-datepicker a{
        color: #393737;
}
.ui-datepicker .ui-datepicker-prev {
        left: 10px;
        top: 6px;
}
.ui-datepicker .ui-datepicker-next {
        right: 10px;
        top: 6px;
}
.ui-datepicker .ui-datepicker-prev:before {
        content: "\f104";
        color: #393737;
        display: inline-flex;
        font-family: FontAwesome;
        font-style: normal;
        font-weight: normal;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
        font-size: 18px;
        cursor: pointer;
}
.ui-datepicker .ui-datepicker-next:before {
        content: "\f105";
        color: #393737;
        display: inline-flex;
        font-family: FontAwesome;
        font-style: normal;
        font-weight: normal;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
        font-size: 18px;
        cursor: pointer;
}
.ui-datepicker .ui-datepicker-prev-hover:before {
        content: "\f104";
        color: #393737;
        display: inline-flex;
        font-family: FontAwesome;
        font-style: normal;
        font-weight: normal;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
        font-size: 18px;
        cursor: pointer;
}
.ui-datepicker .ui-datepicker-next-hover:before {
        content: "\f105";
```

```
        color: #393737;
        display: inline-flex;
        font-family: FontAwesome;
        font-style: normal;
        font-weight: normal;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
        font-size: 18px;
        cursor: pointer;
        text-align: right;
}
.ui-datepicker .ui-datepicker-prev-hover {
        left: 10px;
        top: 6px;
}
.ui-datepicker .ui-datepicker-next {
        text-align: right;
}
.ui-datepicker .ui-datepicker-next-hover {
        right: 10px;
        top: 6px;
        text-align: right;
}
```

Save your changes, clear your browser cache, and refresh your dashboard page. You should now see this:



Try out the filters and ensure they update your report after each selection.

You have now completed a very simple dashboard. From here you can design dashboards and styling based on your business requirements. Explore the widgets available to DashXML here:

- Widget Overviews
- Widget Examples
- Widget XML

top