

Session 3-3: Dashboard & Visualization

Table of Contents

Session 3-3: Dashboards and Visualizations – Making Data Shine	3
Introduction.....	3
TIP	7
Creating an Operational Intelligence dashboard.....	9
Getting ready.....	9
How to do it... ..	9
How it works... ..	11
TIP	11
There's more... ..	11
CHANGING DASHBOARD PERMISSIONS.....	11
TIP	12
Using a pie chart to show the most accessed web pages.....	13
Getting ready.....	13
How to do it.....	13
How it works.....	17
There's more.....	18
SEARCHING FOR THE TOP 10 ACCESSED WEB PAGES	18
See also	Error! Bookmark not defined.
Displaying the unique number of visitors	19
Getting ready.....	19
How to do it.....	19
NOTE	22
How it works.....	23
There's more.....	23
COLORING THE VALUE BASED ON RANGES	25
ADDING TRENDS AND SPARKLINES TO THE VALUES	26
NOTE	26
Using a gauge to display the number of errors	27
Getting ready.....	27
How to do it.....	27
How it works.....	29
There's more.....	30

NOTE	30
Charting the number of method requests by type and host	31
Getting ready.....	31
How to do it.....	31
How it works.....	33
Creating a timechart of method requests, views, and response times	34
Getting ready.....	34
How to do it.....	34
How it works.....	36
There's more.....	37
METHOD REQUESTS, VIEWS, AND RESPONSE TIMES BY HOST	37
Using a scatter chart to identify discrete requests by size and response time	39
Getting ready.....	39
How to do it.....	39
How it works.....	41
There's more.....	42
USING TIME SERIES DATA POINTS WITH A SCATTER CHART	42
Creating an area chart of the application's functional statistics.....	44
Getting ready.....	44
How to do it.....	44
How it works.....	45
Using a bar chart to show the average amount spent by category.....	48
Getting ready.....	48
How to do it.....	48
How it works.....	50
Creating a line chart of item views and purchases over time.....	52
Getting ready.....	52
How to do it.....	52
How it works.....	54

Session 3-3: Dashboards and Visualizations – Making Data Shine

In this chapter, we will learn how to build dashboards and create visualizations of your data. We will cover the following recipes:

- Creating an **Operational Intelligence** dashboard
- Using a pie chart to show the most accessed web pages
- Displaying the unique number of visitors
- Using a gauge to display the number of errors
- Charting the number of method requests by type and host
- Creating a timechart of method requests, views, and response times
- Using a scatter chart to identify discrete requests by size and response time
- Creating an area chart of the application's functional statistics
- Using a bar chart to show the average amount spent by category
- Creating a line chart of item views and purchases over time

Introduction

In the previous chapter, we learned all about Splunk's SPL and how it can be leveraged to search and report your data. In this chapter, we're going to build on this knowledge and use some of Splunk's visualization capabilities to make our data shine! You will learn how to create a dashboard through the Splunk UI and proceed to add to it the reports that were built in the previous chapter. Two more dashboards will then be created as a result of the remaining recipes.

Visualizations are a cornerstone for proper data presentation. By visualizing data in a manner that we as humans are accustomed to, you enable the user to better relate to what is being presented and have a proper understanding of how to react. When using Splunk for Operational Intelligence, you will be hard pressed to find a report that is

not visually represented in some fashion. Everyone from front-line staff to C-level executives looks to Splunk's visualizations to make better sense of the data that their systems and applications produce. Through the creation and use of dashboards, these visualizations can then be arranged and centralized to meet the needs of your organization.

About Splunk dashboards

A dashboard represents the most common type of view within Splunk and provides the means to bring together one or more reports and display them on a single page. Each report is placed on a dashboard as a panel and powered by the search you created. Typically, the panels get populated with data once the dashboard is loaded. A report within a panel can display tabular data or one of the number of visualizations we will cover in this chapter.

Using dashboards for Operational Intelligence

In the world of Operational Intelligence, dashboards are one of the key tools to unlock pivotal information and provide a holistic view of systems and applications through a single pane. Dashboards are built to collectively display information to key audiences such as operators, administrators, or executives. They act as a window to how your environment is performing and allow you to obtain the right information at the right time in one place, in order to make timely and actionable decisions.

Enriching data with visualizations

Data on its own can be hard for us, as humans, to make sense of easily and can be extremely tedious to analyze. Visualizations provide a powerful way to bring data to life. Presenting data in a visual context enables those viewing it to better understand the relationship one value has to another, identify patterns, build correlations between datasets, and plot out trends. Colors that we easily relate to can be applied to visualizations in order to direct attention and highlight specific data points. For example, a value within an acceptable range might be colored green, but when this value increases, it might change to yellow and eventually to red when it's in an unacceptable range. Humans

associate red with bad and green with good; therefore, a red value nicely conveys the need for attention to itself.

Let's now apply this to an Operational Intelligence example. Imagine that you have a distributed environment of web servers that are generating large amounts of erratic data. Inside each of these events is a field that represents the response time of when that event occurred. If you were left having to analyze these events row by row in a table, it could take a very long time to find the events with values outside of the norm. Using visualizations such as a scatter chart, you could plot your event data and easily be able to identify these discrete events that lie outside of the primary cluster of events.

Available visualizations

One of the great benefits of Splunk is that there are numerous out-of-the-box visualizations that can be easily overlaid on your data. The type of visualizations and their common usage is outlined in the following table:

Visualization	Common usage
Line chart	This is commonly used to display data over time. If more than one data series is specified, each line on the chart will have a different color. Line charts can be stacked to help understand the relation of a given series of data to the rest of the plotted series.
Area chart	This works in the same way as a line chart, but the area below the line is shaded in color to emphasize quantities.
Column chart	This displays the data values as vertical columns and is most commonly used when the frequency of values needs to be compared. Column charts can also be stacked to highlight the importance of different data types within the chart.
Bar chart	This displays the data values as horizontal bars and works in the same way as column charts, but with its axis reversed.
Pie chart	This is two-dimensional and displays the data values as segments of a pie. It is most commonly used to highlight or compare the proportion of numerical values.

Visualization	Common usage
Scatter chart	This displays the data values as a series of plotted squares. It is commonly used when trying to identify discrete values in data that fall within the confines of regular events.
Single value	This displays the data as a single value and is mostly used to display a sum or total, for example, the total number of errors in the last hour.
Radial gauge	This resembles a speedometer with a needle to signify the current value across an arced range. It is most commonly used on real-time dashboards to draw attention to the current state of a given metric. Thresholds can be defined to signify which values are acceptable (green), escalating (yellow), and severe (red).
Filler gauge	This resembles a thermometer with a liquid-like indicator. As the value changes, the volume of the liquid rises and falls, as well as changing color. As with the radial gauge, it is most commonly found on real-time dashboards and can have custom thresholds defined.
Marker gauge	Similar to the filler gauge, a marker gauge is already filled with values as defined by the thresholds, and it has a sliding marker to signify what the current value is. It is found most commonly on real-time dashboards.
Marker Map	This is commonly used to illustrate geographical distributions of the data. Data points on the map can be charted to highlight distinct counts of values within the same geographic region.
Choropleth Map	Choropleth maps use shading or coloring of specific bounded areas to illustrate differences in the values between different areas on the map. For example, a map of the US may have states shaded or colored differently, in order to represent differing values per state.
Heat map	Tabular values can have a heat map overlay applied to them so that the highest values are shaded red while the lowest values are shaded blue. It is most commonly used when

Visualization	Common usage
	trying to draw attention visually to the variation of values in a table.
Sparkline	Sparkline visualizations are like mini line charts and are applied within a table to each row. They provide insight into the identification of patterns that might not otherwise have been properly represented by the resulting data in the table.

TIP

For more information on the available visualizations, visit <http://docs.splunk.com/Documentation/Splunk/latest/Viz/Visualizationreference>.

You can also checkout the Splunk Web Framework Toolkit app on Splunkbase for examples (<https://splunkbase.splunk.com/app/1613>).

Best practices for visualizations

Here are some best practices to consider when adding visualizations to your dashboards:

- Use visualizations to provide insight in a way that cannot be easily represented by tabular data.
- It can sometimes be useful to have a chart supported with a table, as a table can make finding absolute numbers easier.
- Provide enough information, but not too much. Do not overload the charts with data that is not pertinent to achieving the goal of the visualization.
- Do not overload the dashboards with visualizations. Spread visualizations out among a few dashboards rather than overloading one specific dashboard. This makes things easier for your users to view as well as improves performance.
- Stacked charts are your friend, especially with area charts. If not stacked, area charts can have instances where large values dominate the chart, obscuring other values.
- Scale visualizations properly. Know when it is best to use linear versus log.

- Label your visualizations clearly so that the audience can understand what they are looking at and do not have to assume.
- Use appropriate visualizations for the task at hand. Here is some guidance:
 - **Comparisons over time:** Use line and column charts
 - **Comparisons among items:** Use bar and column charts
 - **Relationships:** Use scatter charts
 - **Distribution:** Use column or bar charts sorted or a scatter chart
 - **Static composition:** Use column charts stacked at 100 percent or a pie chart
 - **Changing composition:** Use column or area charts stacked, or column or area charts stacked at 100 percent
 - **Geographic statistics:** Use marker or choropleth maps
- Make proper use of colors and thresholds when leveraging single value visualizations and gauge charts.
- You can change the orientation of most visualizations; make use of your best judgment as required.

Creating an Operational Intelligence dashboard

Before this chapter gets into everything that is great about visualizations, it is best to first cover the process of creating a dashboard. In this recipe, you will create a dashboard from scratch using the Splunk Web UI that we will then use for other recipes in this chapter.

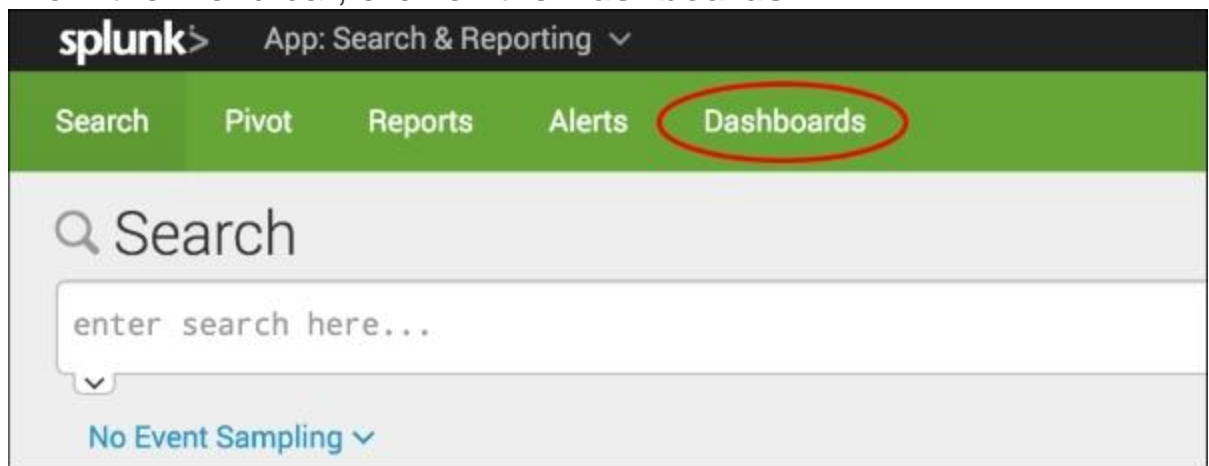
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, Play Time – Getting Data In, and all the completed recipes from Session 3-2, Diving into Data – Search and Report. You should be familiar with navigating the Splunk user interface.

How to do it...

Follow the given steps to create an Operational Intelligence dashboard:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.
3. From the menu bar, click on the **Dashboards** link:



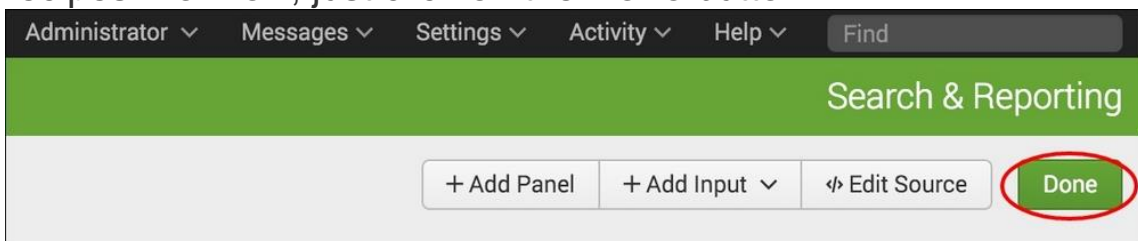
4. On the **Dashboards** screen, click on the **Create New Dashboard** button:



5. The **Create New Dashboard** screen will pop up. Enter [Website Monitoring](#) in the **Title** field. The **ID** field will be automatically populated; leave it as it is. The **Description** field can be left blank for now, but ensure that the **Shared in App** permission is selected. Finally, click on **Create Dashboard**:

A dialog box titled "Create New Dashboard" with a close button (X) in the top right corner. It contains four fields: "Title" with the value "Website Monitoring", "ID" with the value "website_monitoring" and a note "Can only contain letters, numbers and underscores.", "Description" with the value "optional", and "Permissions" with two buttons: "Private" and "Shared in App". At the bottom, there are two buttons: "Cancel" and "Create Dashboard". The "Title", "ID", and "Shared in App" fields are circled in red.

6. The newly created **Website Monitoring** dashboard will appear in edit mode. We will add panels to the dashboard in the next few recipes. For now, just click on the **Done** button:



You have now created a blank dashboard ready to be populated with reports and visualizations.

How it works...

When creating a dashboard through the user interface, Splunk builds the underlying dashboard object code in simple XML for you behind the scenes. Following this, the dashboard object will then be used as a container for the reports you add to it. You can always view the source code of the dashboard by clicking on the **Edit** button and then, from the drop-down menu, you can click on **Edit Source**. The simple XML source code for the dashboard will be displayed in the editor. Dashboards and simple XML will be covered in more detail in the next chapter.

TIP

There are several ways in which dashboards can be created in the Splunk user interface. In this recipe, we essentially created an empty dashboard which is ready to be filled with visualized reports. Splunk also allows dashboards to be created at the time of adding reports, as you will see later in this chapter.

There's more...

When creating a dashboard, the default permission is for it to be private (only accessible by the user who created it). You might wish to share this or other dashboards with other users or groups who have an interest in these reports.

CHANGING DASHBOARD PERMISSIONS

To change the permissions on a dashboard, you must first return to the **Dashboards** screen. This can be accomplished by clicking on the **Dashboards** menu, as outlined in step 3 of this recipe. Within the **Dashboards** screen, you will see the **Website Monitoring** dashboard that you created during the course of this recipe. Under the **Actions** column, you will see a clickable link labeled **Edit**; click on this link. A drop-down panel will appear; click on the item labeled **Edit Permissions**. The resulting pop-up window that appears will allow you to define permissions on a role-by-role basis and limit these permissions to be restrictive within the working application, or globally for all applications.

TIP

When creating dashboards through the GUI in Splunk, the default permission level should be private. However, in this recipe, you selected the **Shared in App** permissions button when creating the dashboard. This ensures that the new dashboard is automatically shared with everyone who has permissions to the application and saves you from having to edit permissions after creating it. On the backend, dashboards with application permissions are stored in the respective application directory structure. Dashboards with private permissions are stored in the respective user directory.

Using a pie chart to show the most accessed web pages

The sample data loaded in Session 3-1, *Play Time – Getting Data In*, provides a wealth of information on how customers are interacting with our online shopping website. In the *Finding the most accessed web pages* recipe in Session 3-2, *Diving into Data – Search and Report*, we saw how to find the most accessed web pages. The output of that recipe was displayed in a tabular format that could be hard for the viewer to grasp the proportional differences between web page access amounts. We will now take a look at how to use pie charts. By taking the same data and visually presenting it using a pie chart now, we will enable the viewer to more easily identify the proportion of requests between the different web pages. Visual representation of data, even if the data is very simple, can lead to better decision making in times of need.

In this recipe, you will use the report named `cp02_most_accessed_webpages`, which you created in Session 3-2, *Diving into Data – Search and Report*. You will graphically display the output of the report using a pie chart and add it to the **Website Monitoring** dashboard that we just created.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*. You should be familiar with the Splunk search bar, the time range picker, and the search tabs (**Events**, **Statistics**, and **Visualization**).

How to do it...

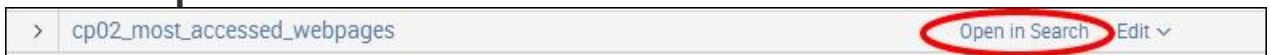
Follow the given steps to use a pie chart to show the most accessed web pages:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.

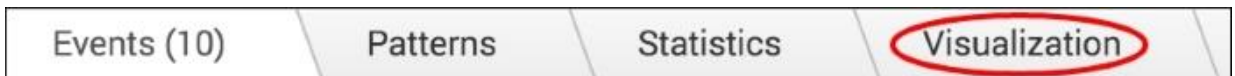
3. From the menu bar, click on the **Reports** link. This will display a list of all the reports we created and saved in Session 3-2, Diving into Data – Search and Report.



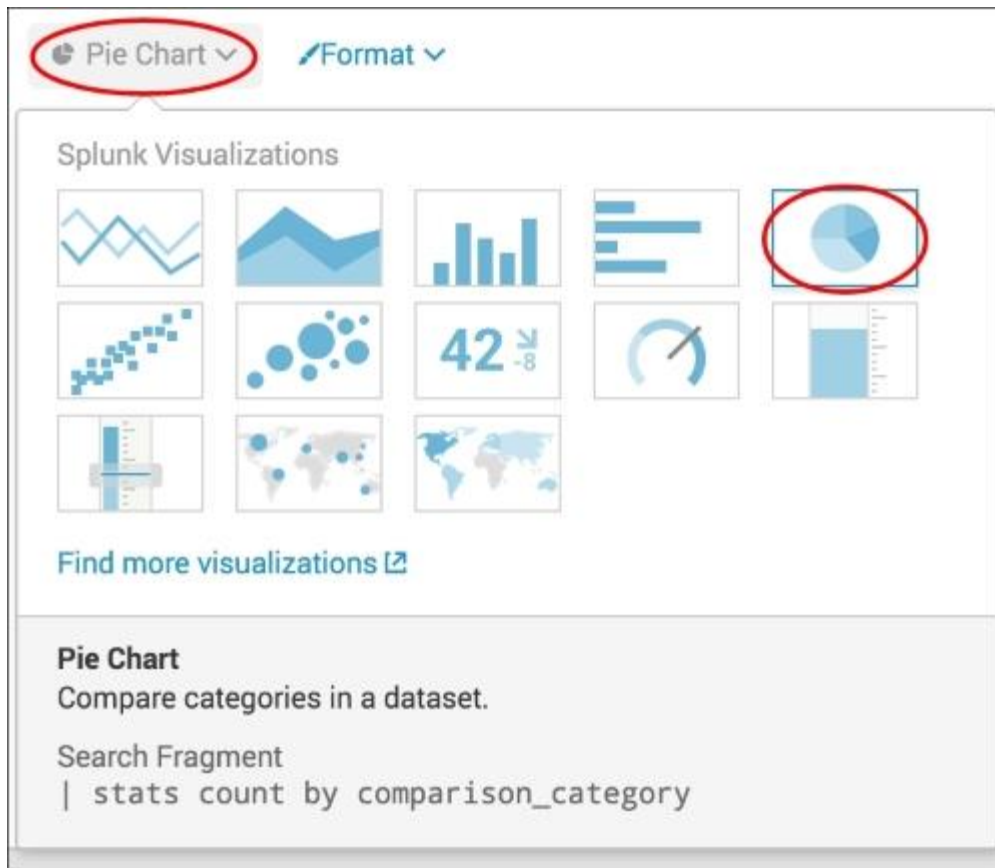
4. Locate the report line item named `cp02_most_accessed_webpages` and click on **Open in Search**:



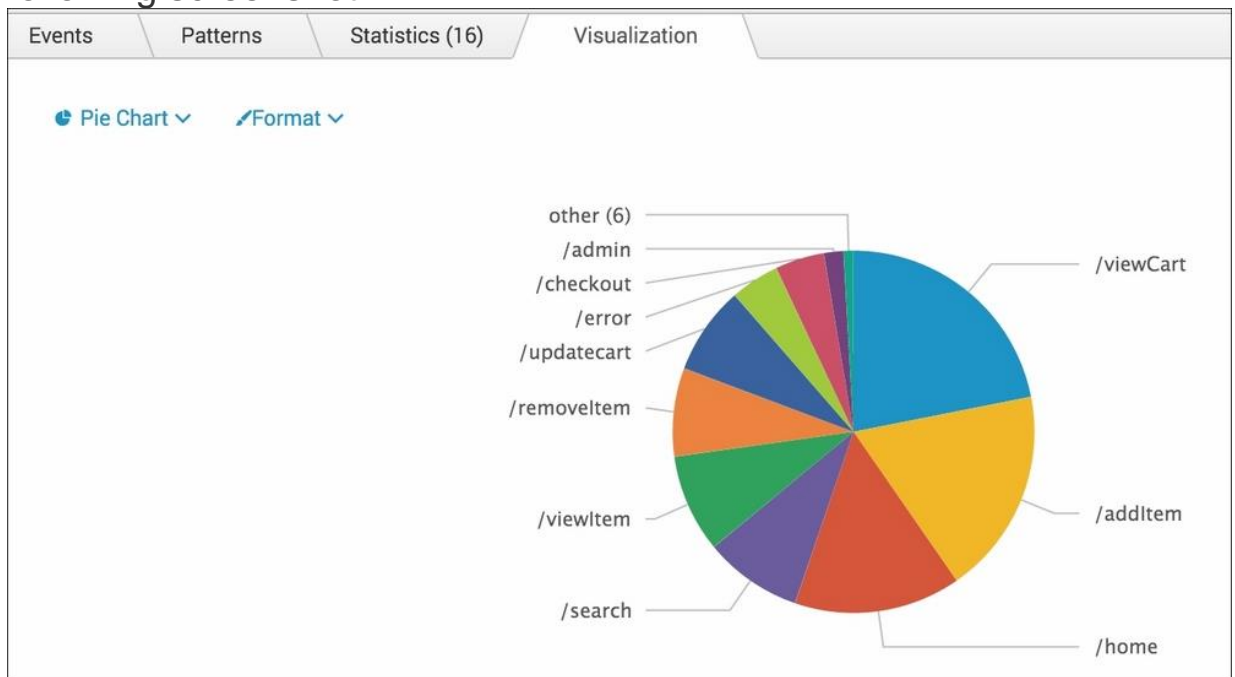
5. Splunk will run the saved report with the search outlined in the following code. This will return a list of pages together with a count field that totals the number of times each page has been accessed.
6. On completion of the search, the results will be displayed within the **Statistics** tab. As we will be creating a pie chart, click on the **Visualization** tab:



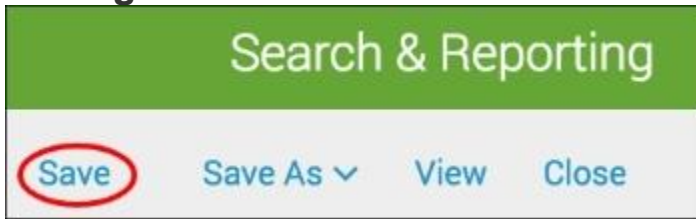
7. As there are a number of visualizations within Splunk, the **Pie** visualization may not be displayed by default within the **Visualization** tab. Click on the dropdown to list the visualization types and then select **Pie Chart**:



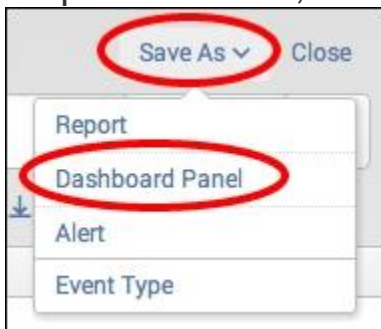
8. Now the data will be visualized as a pie chart, as shown in the following screenshot:



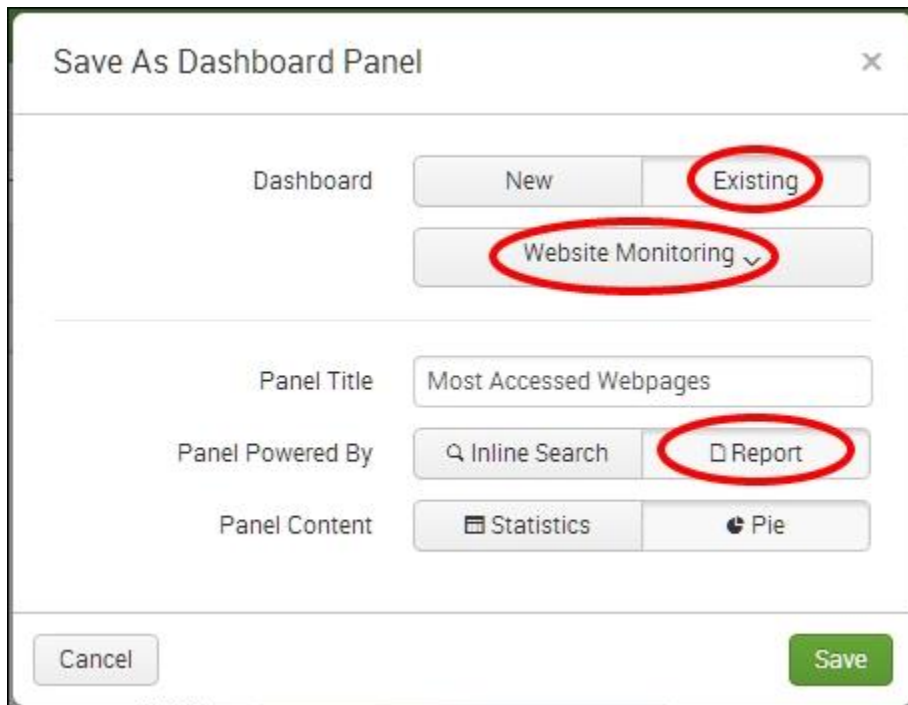
9. Next, save the updated report by clicking the **Savelink**, and then click the **Save** button in the form that pops up. Then click **Continue Editing**:



10. Let's add it to the **Website Monitoring** dashboard you created in the first recipe. Click on **Save As**, and then from the drop-down menu, click on **Dashboard Panel**:



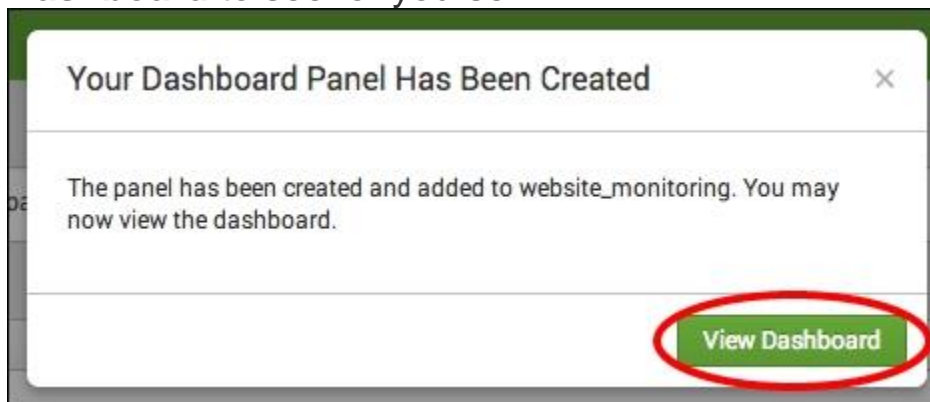
11. The **Save As Dashboard Panel** screen will pop up. Select **Existing** to use an existing dashboard and select the **Website Monitoring** dashboard from the list. For **Panel Title**, enter [Most Accessed Webpages](#) and select **Report** for the **Panel Powered By** field. Then click on **Save**, as shown in the following screenshot:



The 'Save As Dashboard Panel' dialog box contains the following elements:

- Dashboard:** Two buttons, 'New' and 'Existing', with 'Existing' circled in red.
- Category:** A dropdown menu showing 'Website Monitoring', which is circled in red.
- Panel Title:** A text input field containing 'Most Accessed Webpages'.
- Panel Powered By:** Two buttons, 'Inline Search' and 'Report', with 'Report' circled in red.
- Panel Content:** Two buttons, 'Statistics' and 'Pie'.
- Buttons:** 'Cancel' and 'Save' buttons at the bottom.

12. The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself:



The confirmation dialog box contains the following elements:

- Title:** 'Your Dashboard Panel Has Been Created'.
- Message:** 'The panel has been created and added to website_monitoring. You may now view the dashboard.'
- Action:** A green 'View Dashboard' button circled in red.

How it works...

To review how the search works in detail, refer to the *Finding the most accessed web pages* recipe in Session 3-2, Diving into Data – Search and Report.

The **Visualization** tab simply takes the tabular output, which is essentially a value split by another value, and overlays the given

visualization. In this case, it was a total count of events split by the web page name, which you overlaid with the pie chart visualization.

There's more...

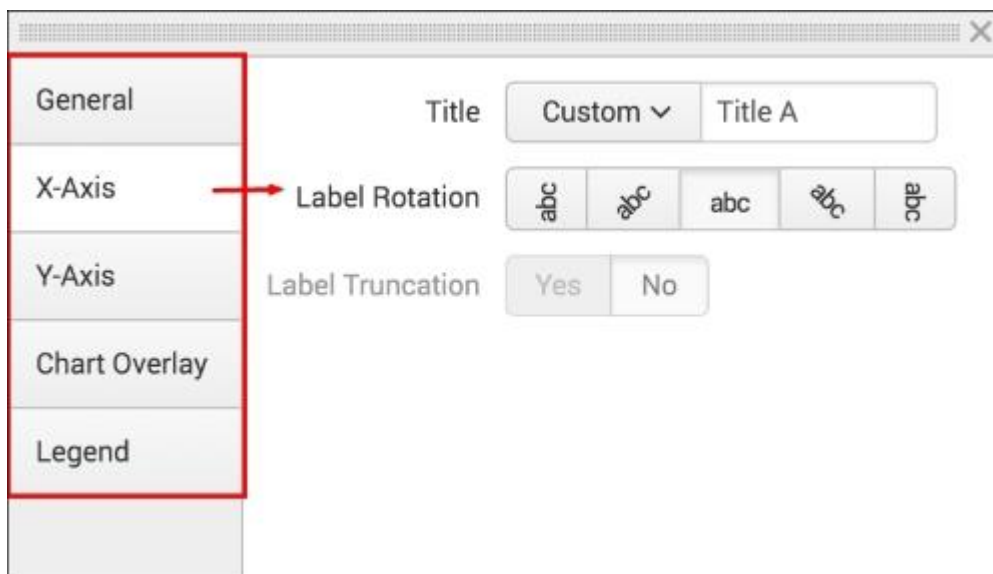
We can further build on the base search to provide different variations of the results and make use of other visualizations.

SEARCHING FOR THE TOP 10 ACCESSED WEB PAGES

If we modify the report search and replace the `stats` command with the `top` command, by default it will display the top 10 web pages:

```
index=main sourcetype=access_combined | top uri_path
```

Here, we modified the report search and replaced the `stats` command with the `top` command. By default, this will display the top 10 web pages. You can then select the **Visualization** tab and choose **Column** to see the results displayed as a column chart. Then, by clicking on **Format**, you can access a menu that allows you to extend the control over the chart by applying specific values, such as customizing the x and y axes, placement or removal of the legend, and more:



Displaying the unique number of visitors

It is always good to understand the number of page views and identify those that are accessed the most, but sometimes it is even better to understand how many of these page views are from unique visitors. Through the web access logs, we can get an understanding of how many unique visitors we have had to our website. For example, it could be helpful to understand whether times of high load are due to the true number of sessions on the website.

In this recipe, you will write a Splunk search to find the unique number of visitors to a website over a given period of time. You will then graphically display this value on a dashboard using the single value visualization.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from [Session 3-1, Play Time – Getting Data In](#). You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you complete all the recipes up until this point.

How to do it...

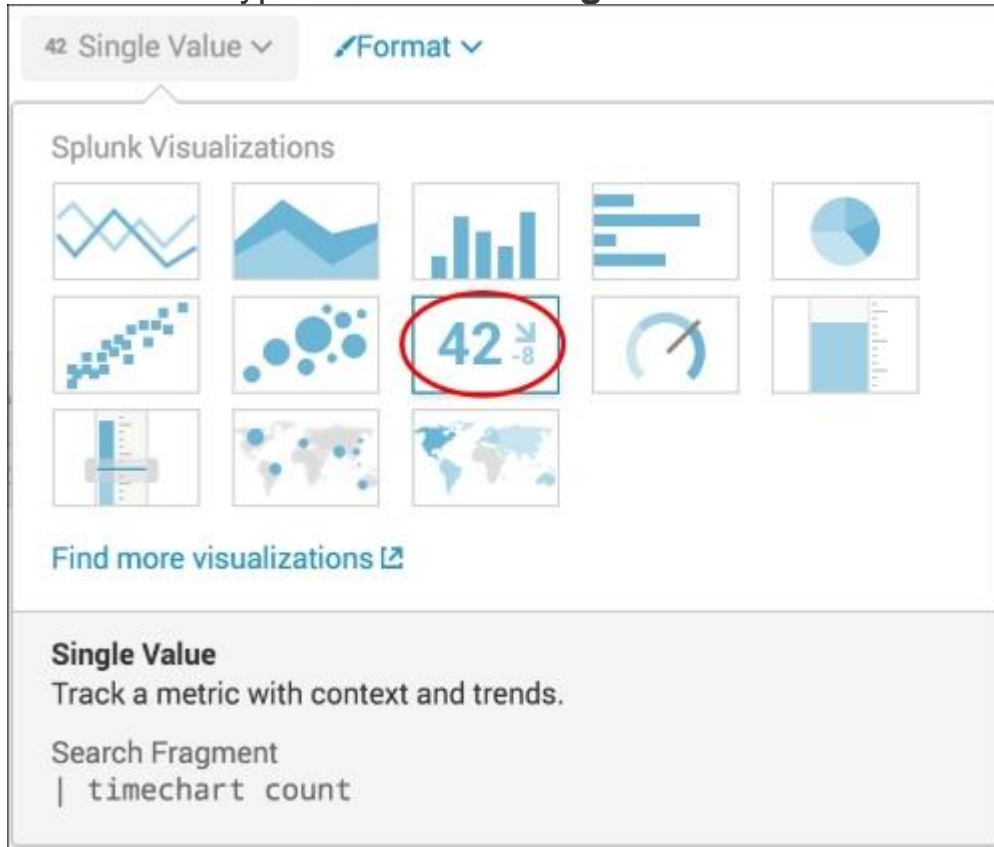
Follow the given steps to display the unique number of website visitors:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.
3. Ensure that the time range picker is set to **Last 24 hours** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*.

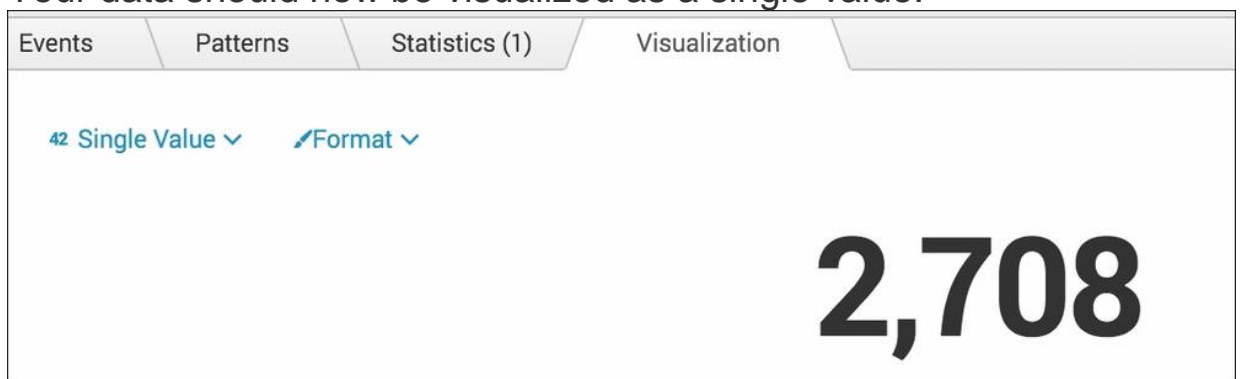
```
index=main sourcetype=access_combined | stats dc(JSESSIONID)
```

4. Splunk will return a single value that represents the distinct count (unique) of values in the field named `JSESSIONID`.
5. Click on the **Visualization** tab.

6. As there are a number of visualizations within Splunk, the **Single Value** visualization might not be displayed by default within the **Visualization** tab. Click on the dropdown that lists the visualization types and select **Single Value**:



7. Your data should now be visualized as a single value:



8. Save this search by clicking on **Save As** and then on **Report**. Name the report `cp03_unique_visitors` and click on **Save**. On the next screen, click on **Add to Dashboard**.
9. You will now add this to the **Website Monitoring** dashboard. Select the button labeled **Existing**, and from the drop-down menu that

appears, select the **Website Monitoring** dashboard. For the **Panel Title** field value, enter **Unique Visitors** and select for the **Panel Powered By** field as **Report**. Then, click on **Save**:

Save As Dashboard Panel

Dashboard: New, Existing

Website Monitoring

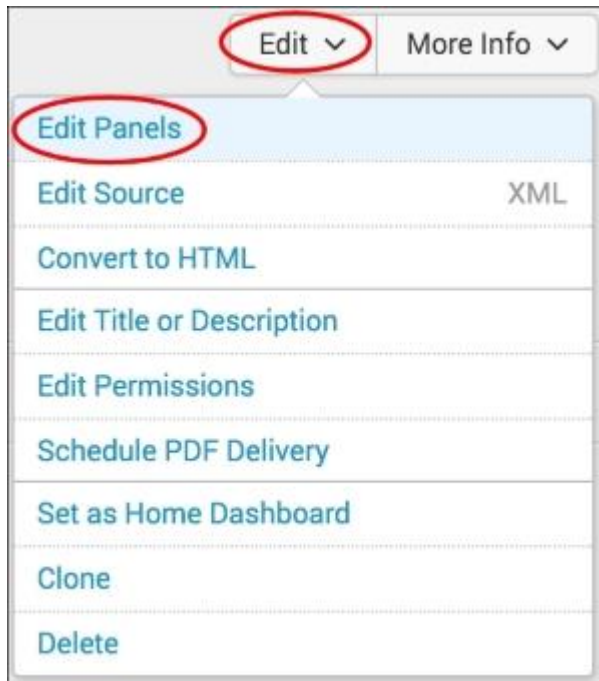
Panel Title: Unique Visitors

Panel Powered By: Inline Search, Report

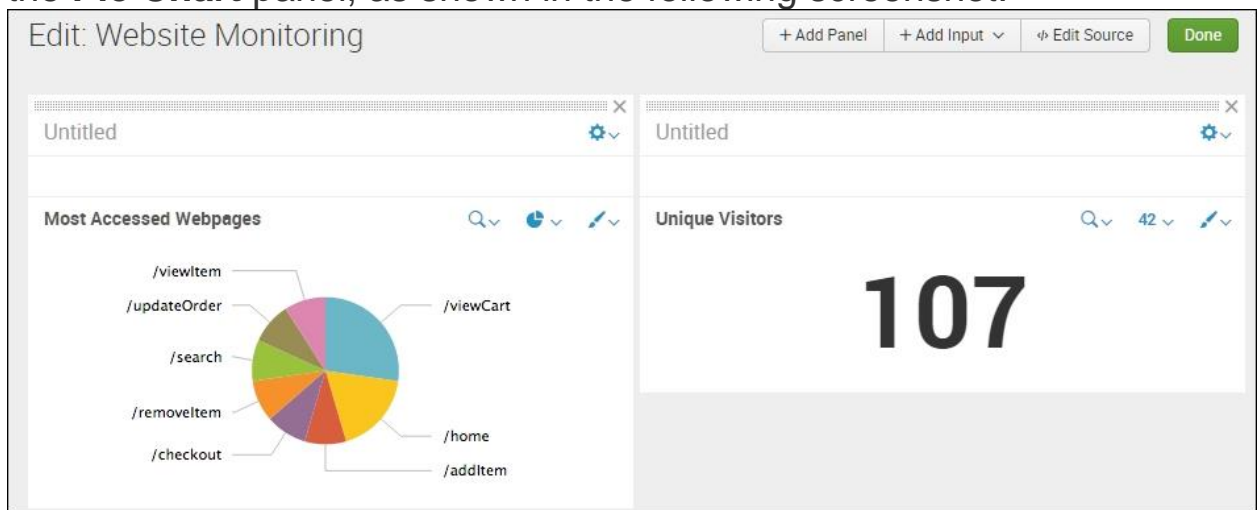
Panel Content: Statistics, Single Value

Cancel, Save

10. The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself. The single value visualization should be placed below the pie chart you created in the previous recipe.
11. You will now arrange the dashboard such that the **Pie Chart** panel and the **Single Value** panel are side by side. Click on the **Edit** button, and from the drop-down menu, select **Edit Panels**:



12. A gray bar will now appear at the top of your panel. Using this bar, click-and-drag the panel to be aligned onto the same row as the **Pie Chart** panel, as shown in the following screenshot:



13. Finally, click on **Done** to save the changes to your dashboard.

NOTE

You will learn more about the functions and features of the Dashboard Editor in the next chapter. For the purpose of this chapter, you will simply be moving panels around on a dashboard.

How it works...

Let's break down the search piece by piece:

Search fragment	Description
<code>index=main sourcetype=access_combined</code>	You should now be familiar with this search from the earlier recipes in this book.
<code> stats dc(JSESSIONID)</code>	Using the <code>stats</code> command, you call the distinct count (<code>dc</code>) function to count the total number of unique values for the field named <code>JSESSIONID</code> . The <code>JSESSIONID</code> field is chosen because each visitor to the website is given a random session identifier whose value is stored in this field. The <code>clientip</code> field, for example, was not chosen here as you can have multiple users coming to a website from the same IP address through the use of NAT (short for Network Address Translation).

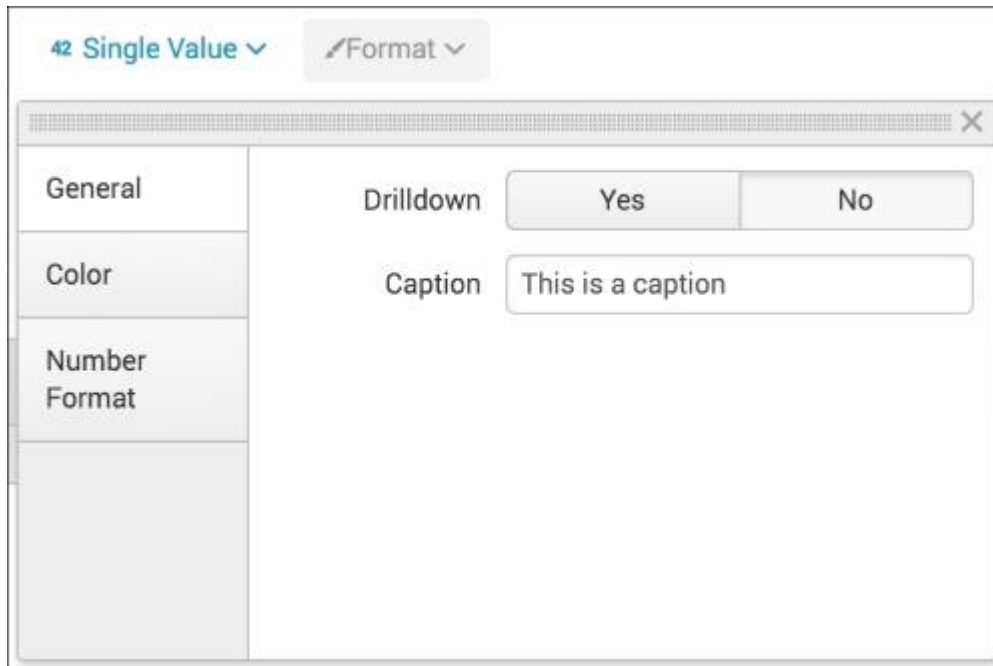
The **Visualization** tab simply takes the numeric output of the `stats` command and overlays the given visualization. In this case, you overlaid the single value visualization on a distinct count of visitor sessions.

There's more...

A single value on the dashboard is very useful, but providing some visual colors and context to the value can prove even more useful.

Adding labels to a single value panel

Run the same search from this recipe, and when the search completes, click on the **Visualization** tab and choose the **Single Value** visualization type. Next, click on the **Format** button, and in the drop-down menu that appears, you have the option to enter a **Caption** text value:



The screenshot shows a configuration window for the 'Single Value' visualization. At the top, there is a header with '42 Single Value' and a 'Format' button. Below this is a sidebar with tabs: 'General', 'Color', 'Number Format', and an empty tab. The 'General' tab is selected. In the main area, there are two settings: 'Drilldown' with 'Yes' and 'No' buttons, and 'Caption' with a text input field containing 'This is a caption'.

After entering your desired values, click *Enter*. The changes will appear immediately, as shown in the following screenshot:



You can now save this single value report as a panel on a dashboard, as you did earlier, but can leave the **Panel Title** field empty as the description of the value is now part of the data itself.

COLORING THE VALUE BASED ON RANGES

After adding labels, it can be useful to provide some visual color to the numeric value displayed, based on a given range within which the number might be. This can be done directly from the **Single Value** formatting option by clicking on the **Color** tab and selecting **Use Colors** as follows:

42 Single Value ▾ Format ▾

General

Color

Number Format

Use Colors **Yes** No

Color by Value Trend

Ranges

from min	to	0	■
from 0	to	30	■ ×
from 30	to	70	■ ×
from 70	to	100	■ ×
from 100	to	max	■

+ Add Range

Color Mode ☒ ☐ 42 42

From this screen you can set the colors and ranges that you wish to use for your value. You can also select whether you want a color mode

where the number is colored or the background is colored. Once you have made your selection, click anywhere on the page.

ADDING TRENDS AND SPARKLINES TO THE VALUES

The **Single Value** visualization is also able to display trends and sparklines. However, for this to happen, the search must be based on the `timechart` command. We can modify the search from the recipe to illustrate this functionality, as follows:

```
index=main sourcetype=access_combined | timechart span=1h  
dc (JSESSIONID)
```

Run this modified search and when the search completes, click on the **Visualization** tab and choose the **Single Value** visualization type. Next, click on the **Format** button. On the **General** tab, you will now notice that there are many more options available to select from, specifically around trending. Leave the default options selected, but select **Show trend in Percent**. Now select the **Color** tab from the formatting box. Again, you will notice some additional options. Select **Color by Trend** and then select **Apply**. You will now see the **Single Value** visualization represented as a number, with a Sparkline illustrating the last 24 hours and also a downward or upward trend percentage compared against the previous hour:



NOTE

For further information on the formatting options available for the Single Value visualization, visit http://docs.splunk.com/Documentation/Splunk/latest/Viz/Visualizationreference#Single_value_visualizations.

Using a gauge to display the number of errors

Not every user interaction with a website goes smoothly. There are times when the accessed pages report an unsuccessful status code.

Understanding this number and being able to apply acceptable low, medium, and high thresholds enables a better understanding of the current user experience when there are a higher number of errors than acceptable.

In this recipe, you will write a Splunk search to find the total number of errors over a given period of time. You will then graphically represent this value on the dashboard using a radial gauge.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from [Session 3-1, Play Time – Getting Data In](#). You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you complete all the recipes up until this point.

How to do it...

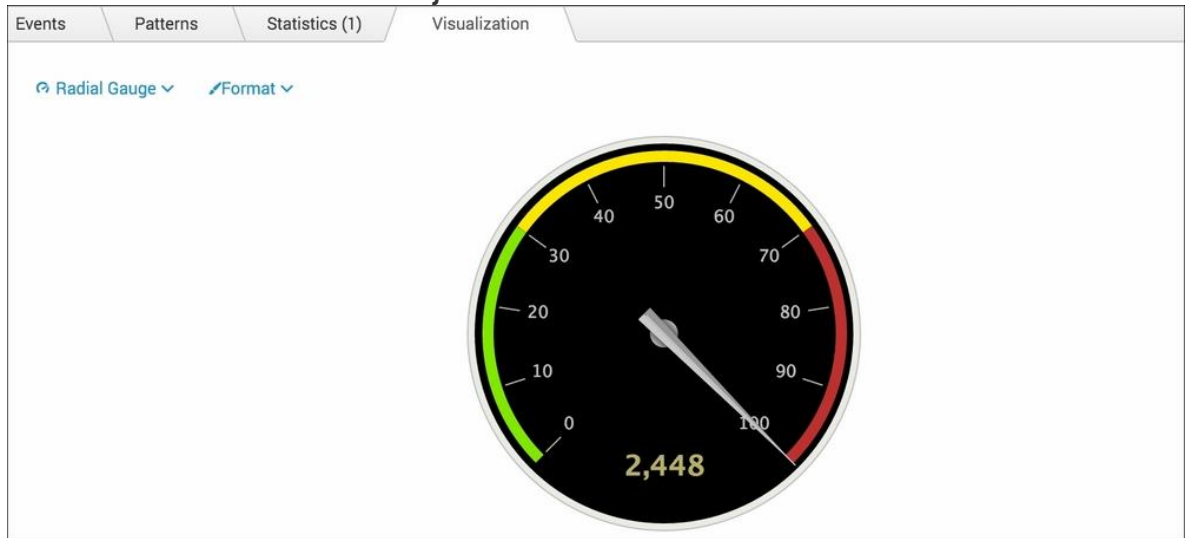
Follow the given steps to use a gauge visualization to display the number of web access errors:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.
3. Ensure that the time range picker is set to **Last 24 hours** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*.

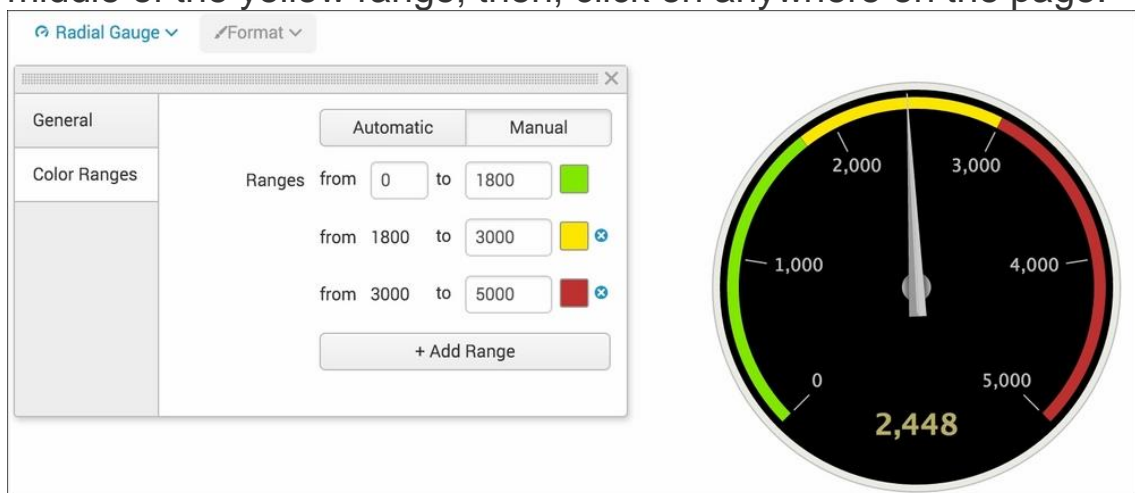
```
index=main sourcetype=access_combined NOT status="200" | stats  
count
```

4. Splunk will return the total count of events where the status was anything but successful.

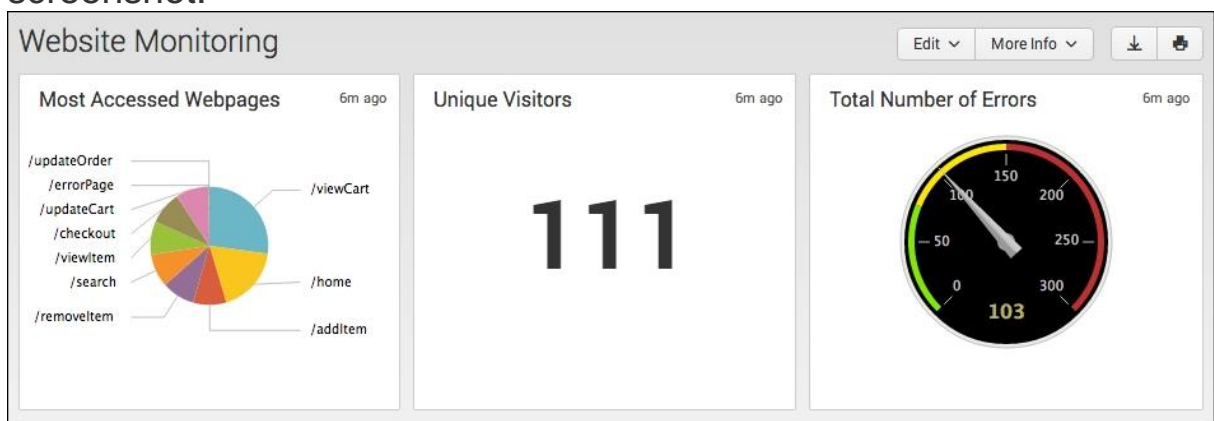
5. Click on the **Visualization** tab.
6. As there are a number of visualizations within Splunk, the **Single Value** visualization might not be displayed by default within the **Visualization** tab. Click on the dropdown that lists the visualization types and select **Radial Gauge**.
7. Your data should now be visualized as a gauge and the needle on the gauge is likely all the way into the red—this is because the thresholds need to be adjusted:



8. To adjust the thresholds on the radial gauge, click on the **Format** button. Then, from the drop-down menu, click on the **Color Ranges** tab, and finally, click on the **Manual** button.
9. From within the **Manual Color Ranges** screen, you can now adjust the values to acceptable amounts such that the needle sits in the middle of the yellow range; then, click on anywhere on the page:



10. Save this report by clicking on **Save As** and then on **Report**. Name the report `cp03_webaccess_errors` and click on **Save**. On the next screen, click on **Add to Dashboard**.
11. You will now add this report to the **Website Monitoring** dashboard. Select the button labeled **Existing**, and from the drop-down menu that appears, select the **Website Monitoring** dashboard. For the **Panel Title** field value, enter **Total Number of Errors** and select for the panel to be powered by **Report**; then, click on **Save**.
12. The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself. The radial gauge visualization should now be positioned on the dashboard below the previous two panels.
13. Arrange the dashboard so that the radial gauge panel is to the right of the single value panel. Click on the **Edit** button, and from the drop-down menu, select **Edit Panels**. Move the radial gauge panel accordingly.
14. Finally, click on **Done** to save the changes to your dashboard. The dashboard should now look like the following screenshot:



How it works...

Let's break down the search piece by piece:

Search fragment	Description
<code>index=main</code> <code>sourcetype=access_combined</code> <code>NOT status="200"</code>	You should be familiar with this search from the earlier

Search fragment	Description
	recipes in this chapter. However, we added the search criteria to not return any event where the <code>status</code> field is equal to <code>200</code> (success).
<code> stats count</code>	Using the <code>stats</code> command, we count the total number of events that are returned.

The **Visualization** tab simply takes the numeric output of the `stats` command and overlays the given visualization. In this case, you overlaid a radial gauge visualization on the total count of events that were not successful.

There's more...

In this recipe, we did not use the other two types of gauges: the filler gauge and the marker gauge. It is advisable to try out the other gauges, as they might be the preferred single value visualization for your intended audience.

NOTE

For more information on the available gauge visualizations, visit <http://docs.splunk.com/Documentation/Splunk/latest/Viz/Visualizationreference#Gauges>.

Charting the number of method requests by type and host

In our environment, where multiple hosts are responding to web requests for customers who browse the website, it is good to get an idea of the current number of each method request split by the host. Methods relate to request/response actions between a customer's web client and our web hosts. Having this type of information can enable us to understand if these requests are properly being balanced across the hosts or if one host is receiving the majority of the load.

In this recipe, you will write a Splunk search to chart the number of method requests split by type and host. You will then graphically represent these values on a dashboard, using a column chart.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from **Session 3-1, *Play Time – Getting Data In***. You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you also complete all the recipes up until this point.

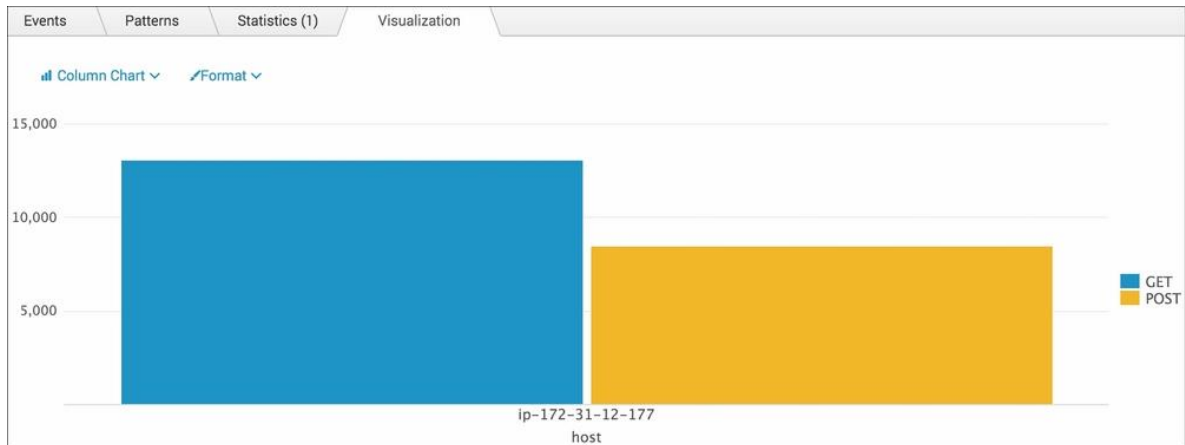
How to do it...

Follow the given steps to chart the number of method requests by type and host:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.
3. Ensure that the time range picker is set to **Last 24 hours** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*.

```
index=main sourcetype=access_combined | chart count by  
host,method
```


4. Splunk will return a tabulated list of the total counts for each method request split by host.
5. Click on the **Visualization** tab.
6. Click on the dropdown that lists the visualization types and select **Column**.
7. Your data should now be visualized as shown in the following screenshot:



8. Save this report by clicking on **Save As** and then on **Report**. Name the report **cp03_methods_by_host** and click on **Save**. On the next screen, click on **Add to Dashboard**.
9. You will now add this to the **Website Monitoring** dashboard. Select the button labeled **Existing**, and from the drop-down menu that appears, select the **Website Monitoring** dashboard. For the **Panel Title** field value, enter **Method Requests by Type and Host** and select **Report** in the **Panel Powered By** field; then, click on **Save**.
10. The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself.
11. Edit the dashboard to position the column chart visualization below the previously added panels.

How it works...

Let's break down the search piece by piece:

Search fragment	Description
<code>index=main sourcetype=access_combined</code>	You should now be familiar with this search from the earlier recipes.
<code> chart count by host,method</code>	The <code>chart</code> command simply performs a count of events split by host and method. This produces the total count of each method for a given host.

The **Visualization** tab simply takes the tabulated output of the `stats` command and overlays the given visualization. In this case, you overlaid a column chart visualization on the total count for each method split by host.

Creating a timechart of method requests, views, and response times

Having the right single values displayed on a dashboard can be beneficial to understanding key metrics, but can also be limiting in providing true operational intelligence on how the different metrics of our website affect one another. By plotting values such as the number of method requests, the number of total views, and the average response times over a given time range, you can begin to understand if there is any correlation between these numbers. This can be very beneficial in understanding things such as if the average response time of pages is growing due to the number of active `POST` requests to the website or if one type of request is making up for the majority of the total number of requests at that given time.

In this recipe, you will create a Splunk search using the `timechart` command to plot values over a given time period. You will then graphically represent these values using a line chart.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from [Session 3-1, Play Time – Getting Data In](#). You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you also complete all the recipes up until this point.

How to do it...

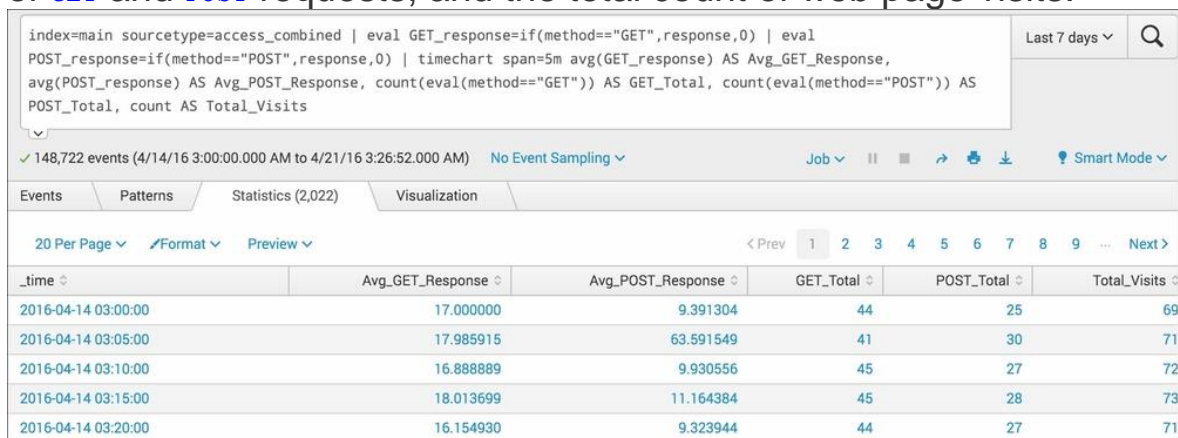
Follow the given steps to create a timechart of method requests, views, and response times:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.

- Ensure that the time range picker is set to **Last 7 days** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*:

```
index=main sourcetype=access_combined | eval
GET_response=if(method=="GET",response,0) | eval
POST_response=if(method=="POST",response,0) | timechart span=5m
avg(GET_response) AS Avg_GET_Response, avg(POST_response) AS
Avg_POST_Response, count(eval(method=="GET")) AS GET_Total,
count(eval(method=="POST")) AS POST_Total, count AS
Total_Visits
```

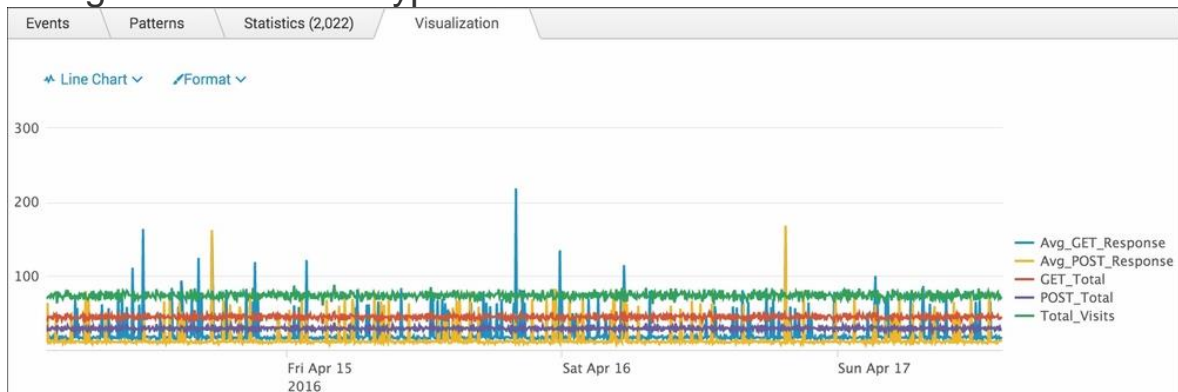
- Splunk will return a time series chart of values for the average response time of **GET** and **POST** requests, the count of **GET** and **POST** requests, and the total count of web page visits:



The screenshot shows the Splunk search interface with the search bar containing the query. Below the search bar, it indicates 148,722 events for the time range 4/14/16 3:00:00.000 AM to 4/21/16 3:26:52.000 AM. The 'Visualization' tab is selected, displaying a table with the following data:

_time	Avg_GET_Response	Avg_POST_Response	GET_Total	POST_Total	Total_Visits
2016-04-14 03:00:00	17.000000	9.391304	44	25	69
2016-04-14 03:05:00	17.985915	63.591549	41	30	71
2016-04-14 03:10:00	16.888889	9.930556	45	27	72
2016-04-14 03:15:00	18.013699	11.164384	45	28	73
2016-04-14 03:20:00	16.154930	9.323944	44	27	71

- Click on the **Visualization** tab and select **Line** from the drop-down listing of visualization types to visualize the data as a line chart:



- Save this report by clicking on **Save As** and then on **Report**. Name the report `cp03_method_view_reponse` and click on **Save**. On the next screen, click on **Add to Dashboard**.
- You will now add this report to the **Website Monitoring** dashboard. Select the button labeled **Existing**, and

from the drop-down menu that appears, select the **Website Monitoring** dashboard. For the **Panel Title** field value, enter `Website Response Performance` and select **Report** in **Panel Powered By** field; then, click **Save**.

8. The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself. The line chart visualization should now be positioned on the dashboard below the previously added panels.
9. Arrange the dashboard so that the line chart panel is on the right-hand side of the column chart panel created in the previous recipe. Click on the **Edit** button, and from the drop-down menu, select **Edit Panels**. Move the line chart panel accordingly.
10. Finally, click on **Done** to save the changes to your dashboard.

How it works...

Let's break down the search piece by piece:

Search fragment	Description
<code>index=main sourcetype=access_combined</code>	You should now be familiar with this search from the earlier recipes in this book.
<code> eval GET_response=if(method=="GET",response,0)</code>	Using the <code>eval</code> command, we create a new field called <code>GET_response</code> , whose value is based on the return value of the <code>if</code> function. In this case, if the method is <code>GET</code> , then the value returned is the value of the <code>response</code> field; otherwise, the value returned is <code>0</code> .
<code> eval POST_response=if(method=="POST",response,0)</code>	Using the <code>eval</code> command, we create a new field, called <code>POST_response</code> , whose value is based on the return value of the <code>if</code> function. In this case, if the method is <code>POST</code> ,

Search fragment	Description
	then the value returned is the value of the <code>response</code> field; otherwise, the value returned is <code>0</code> .
<pre>timechart span=5m avg(GET_response) AS Avg_GET_Response, avg(POST_response) AS Avg_POST_Response, count(eval(method=="GET")) AS GET_Total, count(eval(method=="POST")) AS POST_Total, count AS Total_Visits</pre>	<p>Using the <code>timechart</code> command, we first specify a span of five minutes. Next, we calculate the average value for the given span of <code>GET_response</code> and <code>POST_response</code>. Next, we count the total number of <code>GET</code> and <code>POST</code> events. Finally, the total number of events, both <code>GET</code> and <code>POST</code>, are counted. Note that we make use of the <code>AS</code> operator to rename the fields so that they are meaningful and easy to understand when displayed on our chart.</p>

The **Visualization** tab takes the time series output of the `timechart` command and overlays the given visualization. In this case, you overlaid the line chart visualization.

There's more...

In this recipe, we looked at the values represented as a whole across our web server environment. However, in instances like ours, where web traffic is balanced across multiple servers, it is a good idea to split the values based on their respective hosts.

METHOD REQUESTS, VIEWS, AND RESPONSE TIMES BY HOST

It is very easy to obtain a more granular view of events split by the host where the events are occurring. All we need to do is add the `by` clause to the end of our previous Splunk search, as follows:

```
index=main sourcetype=access_combined | eval
GET_response=if(method=="GET",response,0) | eval
POST_response=if(method=="POST",response,0) | timechart
span=5m avg(GET_response) AS Avg_GET_Response,
avg(POST_response) AS Avg_POST_Response,
count(eval(method=="GET")) AS GET_Total,
count(eval(method=="POST")) AS POST_Total, count AS
Total_Visits by host
```

As simple as this is, we can now visualize the values broken down by the host on which these values originated. In a distributed environment, this can be most crucial in understanding where latency or irregular volumes exist.

Using a scatter chart to identify discrete requests by size and response time

As shown by the recipes up until this point, there is vast intelligence that can be attained by building visualizations that summarize the current application state, analyze performance data over time, or compare values to one another. However, what about those discrete events that appear off in the distance at odd or random times? These events might not be correctly reflected when looking at a column chart, single value gauge, or pie chart, as to most calculations, they are just a blip in the radar somewhere off in the distance. However, there could be times where these discrete events are indicative of an issue or simply the start of one.

In this recipe, you will write a very simple Splunk search to plot a few elements of web request data in a tabular format. The real power comes next, when you will graphically represent these values using a scatter chart.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from [Session 3-1, Play Time – Getting Data In](#). You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you complete the recipes up until this point.

How to do it...

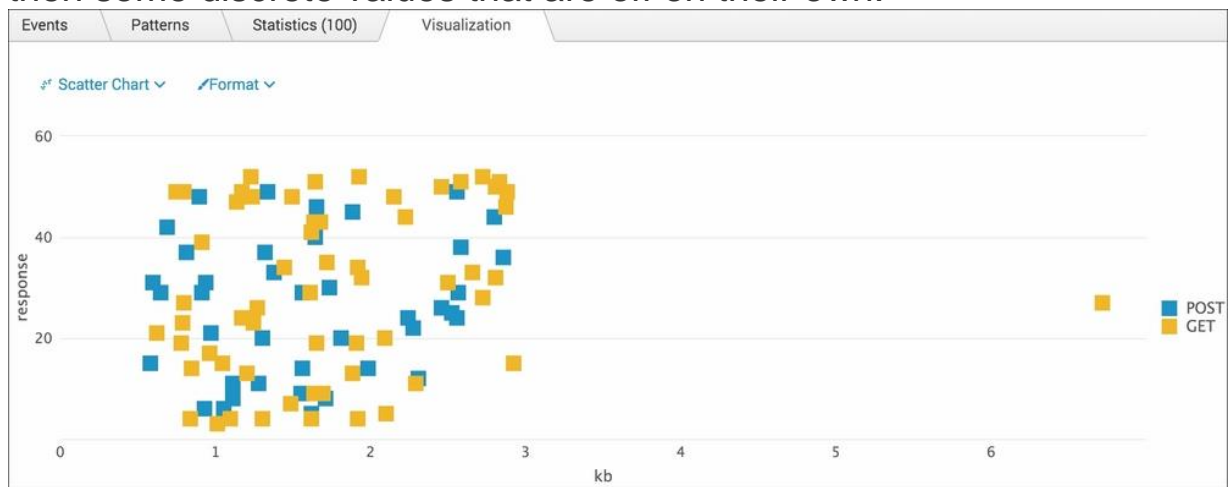
Follow the given steps to use a scatter chart to identify discrete requests by size and response time:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.

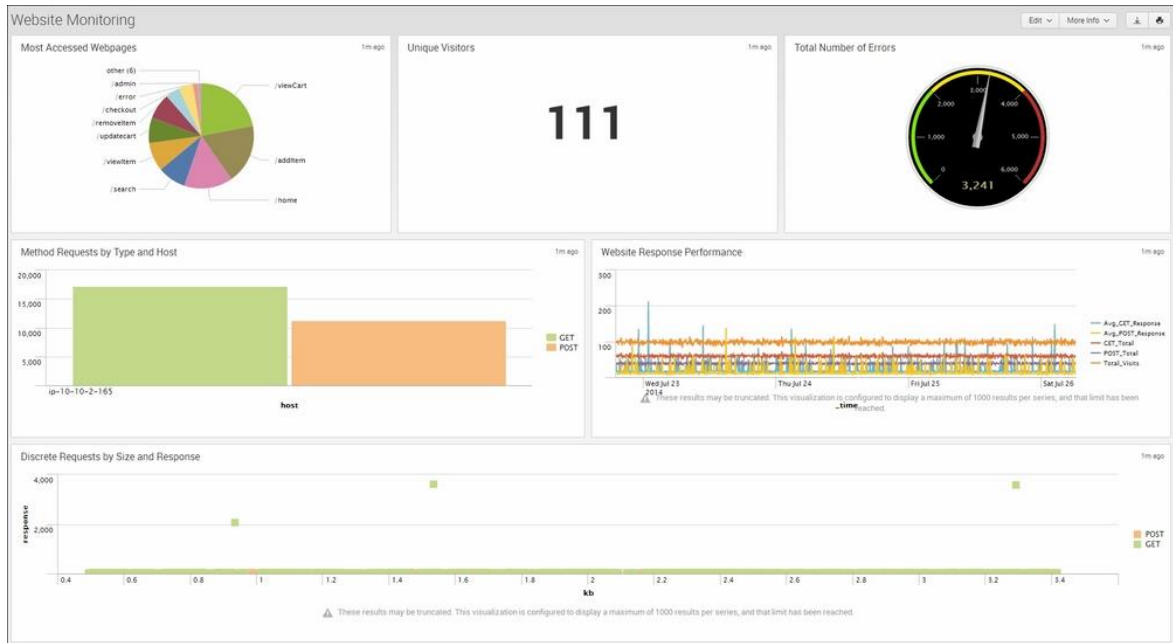
3. Ensure that the time range picker is set to **Last 24 hours** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*.

```
index=main sourcetype=access_combined | eval kb=bytes/1024 |  
table method kb response
```

4. Splunk will return a tabulated list of the `method`, `kb`, and `response` fields for each event.
5. Click on the **Visualization** tab and select **Scatter** from the drop-down list of visualization types to see the data represented as a scatter plot chart. You should see the cluster of normal activity and then some discrete values that are off on their own:



6. Save this report by clicking on **Save As** and then on **Report**. Name the report `cp03_discrete_requests_size_response` and click on **Save**. On the next screen, click on **Add to Dashboard**.
7. You will now add this to the **Website Monitoring** dashboard. Select the button labeled **Existing**, and from the drop-down menu that appears, select the **Website Monitoring** dashboard. For the **Panel Title** field value, enter **Discrete Requests by Size and Response** and select **Report** in **Panel Powered By**; then, click on **Save**.
8. The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself. The scatter chart visualization should now be positioned on the dashboard below the previously added panels:



How it works...

Let's break down the search piece by piece:

Search fragment	Description
<code>index=main</code> <code>sourcetype=access_combined</code>	You should now be familiar with this search from the earlier recipes in this book.
<code> eval kb=bytes/1024</code>	Using the <code>eval</code> command, we convert the size of the request from bytes to kilobytes. For presentation purposes, this makes it easier to read and relate.
<code> table method kb response</code>	Using the <code>table</code> command, we plot our data points that will be represented on the scatter chart. The first field, <code>method</code> , presents the data that appears in the legend. The second field, <code>kb</code> , represents the x axis value. Finally, the third

Search fragment	Description
	field, <code>response</code> , represents the yaxis value.

There's more...

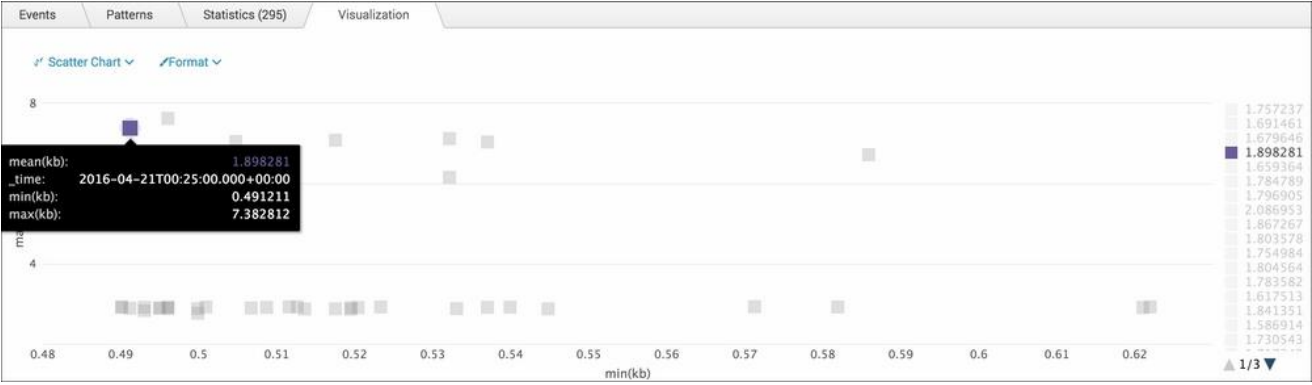
Aside from simply plotting the data points for a scatter chart in tabular form, you can leverage the `timechart` command and its available functions to better identify and provide more context to these discrete values.

USING TIME SERIES DATA POINTS WITH A SCATTER CHART

The Splunk search you ran in this recipe can be modified to make use of the `timechart` command and all the functions it has to offer. Using the **Visualization** tab and scatter chart, run the following Splunk search over the last 24 hours:

```
index=main sourcetype=access_combined | eval kb=bytes/1024 |  
timechart span=5m mean(kb) min(kb) max(kb)
```

As you can see, with the `timechart` command, you first bucket the events into 5-minute intervals, as specified by the span parameter. Next, the `mean`, `min`, and `max` values of the `kb` field for that given time span are calculated. This way, if there is an identified discrete value, you can see more clearly what drove that span of events to be discrete. An example of this can be found in the following screenshot. In this scatter chart, we have highlighted one discrete value that is far outside the normal cluster of events. You can see why this might have stood out using the min and max values from this event series:



Creating an area chart of the application's functional statistics

Understanding not only how your web page is performing and responding to requests but also how underlying applications that you rely on is critical to the success of any website. You need to have a constant pulse on how the application is behaving and if any trends are emerging or correlations are being observed between interdependent pieces of data. The experience a customer has with your website is reliant on the constant high performance of all its components.

In this recipe, you will write a Splunk search using the `timechart` command to plot web application memory and response time statistics over a given time period. You will then graphically present these values using an area chart.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from **Session 3-1, *Play Time – Getting Data In***. You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you complete all the recipes up until this point.

How to do it...

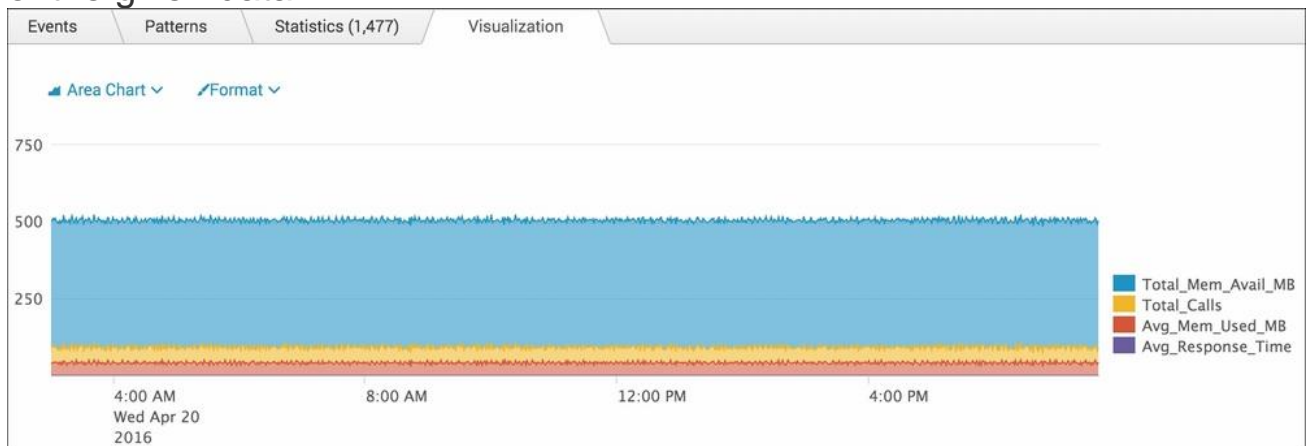
Follow the given steps to create an area chart of an application's functional statistics:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.
3. Ensure that the time range picker is set to **Last 24 hours** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*.

```
index=main sourcetype=log4j | eval  
mem_used_MB=(mem_used/1024)/1024 | eval  
mem_total_MB=(mem_total/1024)/1024 | timechart span=1m
```

```
values(mem_total_MB) AS Total_Mem_Avail_MB, count AS
Total_Calls, avg(mem_used_MB) AS Avg_Mem_Used_MB,
avg(response_time) AS Avg_Response_Time
```

- Splunk will return a time series chart of values for the average response time of `GET` and `POST` requests, the count of `GET` and `POST` requests, and the total count of web page visits.
- Click on the **Visualization** tab and select **Area** from the drop-down list of visualization types to see the data represented as an area chart. Note how the data is stacked for better visual representation of the given data:



- Save this report by clicking on **Save As** and then on **Report**. Name the report `cp03_webapp_functional_stats` and click on **Save**. On the next screen, click on **Add to Dashboard**.
- You will now add this to the **Website Monitoring** dashboard. Select the button labeled **Existing**, and from the dropdown menu that appears, select the **Website Monitoring** dashboard. For the **Panel Title** field value, enter `Web Application Functional Statistics` and select **Report** in **Panel Powered By**; then, click on **Save**.
- The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself.

How it works...

Let's break down the search piece by piece:

Search Fragment	Description
<code>index=main sourcetype=log4j</code>	In this example, we search for our

Search Fragment	Description
	application's logs that have the <code>log4j</code> sourcetype.
<pre> eval mem_used_MB=(mem_used/1024)/1024</pre>	Using the <code>eval</code> command, we calculate the amount of memory currently being used, in megabytes.
<pre> eval mem_total_MB=(mem_total/1024)/1024</pre>	Using the <code>eval</code> command again, we calculate the total amount of memory that is available for use, in megabytes.
<pre> timechart span=1m values(mem_total_MB) AS Total_Mem_Avail_MB, count AS Total_Calls, avg(mem_used_MB) AS Avg_Mem_Used_MB, avg(response_time) AS Avg_Response_Time</pre>	Using the <code>timechart</code> command, we first specify a span of 1 minute for our events. Next, we use the <code>values</code> function to retrieve the value stored in the <code>mem_total_MB</code> field. The <code>count</code> function is then used to calculate the total number of function calls during the given time span. The average function is then called twice, to calculate the average amount of memory used and the average response time for the function call during the given time span. Note that we make use of

Search Fragment	Description
	the <code>AS</code> operator to rename the fields so that they are meaningful and easy to understand when displayed on our chart.

The **Visualization** tab takes the time series output of the `timechart` command and overlays the given visualization. In this case, you overlaid an area chart.

Using a bar chart to show the average amount spent by category

Throughout this chapter, you have been building visualizations to provide insight into the operational performance of our e-commerce website. It can also be useful to understand the customers' views and the factors that might drive them to the website. This type of information is traditionally most useful for product or marketing folks. However, it can also be useful to gain an understanding around whether an item is increasing in popularity and if this could ultimately lead to additional customers and heavier load on the site.

In this recipe, you will write a Splunk search to calculate the average amount of money spent, split out by product category. You will then graphically present this data using a bar chart on a new **Product Monitoring** dashboard.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from [Session 3-1, Play Time – Getting Data In](#). You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you complete all the recipes up until this point.

How to do it...

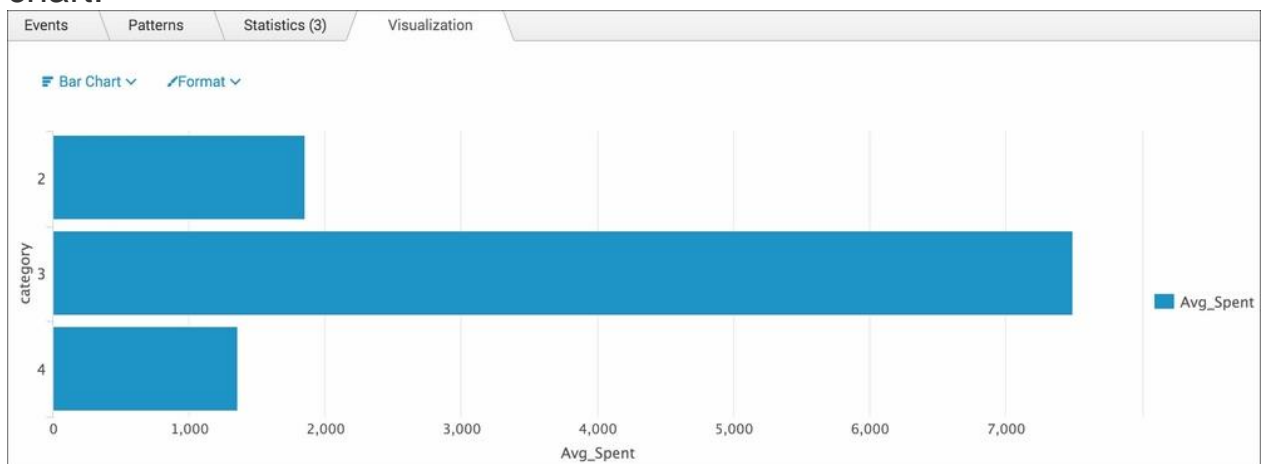
Follow the given steps to use a bar chart to show average amount spent by category:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.

3. Ensure that the time range picker is set to **Last 24 hours** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*.

```
index=main sourcetype=log4j | transaction sessionId maxspan=30m  
| search requestType="checkout" | stats avg(total) AS Avg_Spent  
by category
```

4. Splunk will return a tabulated list, detailing the category and the associated average amount spent.
5. Click on the **Visualization** tab and select **Bar** from the drop-down list of visualization types to see the data represented as a bar chart:



6. Save this report by clicking on **Save As** and then on **Report**. Name the report `cp03_average_spent_category` and click on **Save**. On the next screen, click on **Add to Dashboard**.
7. You will now add this to a new **Product Monitoring** dashboard. Select the button labeled **New** and enter the dashboard title **Product Monitoring**. For the **Panel Title** field value, enter `Average Spent by Category` and select **Report** in the **Panel Powered By** field; then, click on **Save**:

Save As Dashboard Panel

×

Dashboard

New

Existing

Dashboard Title

Product Monitoring

Dashboard ID ?

product_monitoring

Can only contain letters, numbers and underscores.

Dashboard Description

optional

Dashboard Permissions

Private

Shared in App

Panel Title

Average Spent by Category

Panel Powered By

🔍 Inline Search

📄 Report

Panel Content

📊 Statistics

📊 Bar

Cancel

Save

8. The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself.

How it works...

Let's break down the search piece by piece:

Search Fragment	Description
<code>index=main sourcetype=log4j</code>	In this example, we search for our application's logs that have the <code>log4j</code> sourcetype.
<code> transaction sessionId maxspan=30m</code>	Using the <code>transaction</code> command, we group together all the events that share the same <code>sessionId</code> in a 30-minute span.

Search Fragment	Description
<pre> search requestType="checkout" paymentReceived="Y"</pre>	Using the <code>search</code> command, we limit the grouped results to those that have only a <code>checkout</code> event and where the payment was received. In this visualization, a purchase does not qualify for consideration if it did not successfully process.
<pre> stats avg(total) AS Avg_Spent by category</pre>	Using the <code>stats</code> command, we calculate the average total amount spent by category. Note that we make use of the <code>AS</code> operator to rename the field so that it is meaningful and easy to understand when displayed on our chart.

The **Visualization** tab simply takes the time series output of the `stats` command and overlays the given visualization. In this case, you overlaid a bar chart visualization.

Creating a line chart of item views and purchases over time

Continuing on from the last recipe, we further improve our understanding of customer activities by now looking at a chart of item views and actual purchases over a given time period. This will allow you to understand if the customers who view an item actually follow through with purchasing the given item.

In the last recipe of this chapter, you will write a Splunk search to chart item views and purchases over a given time period. You will then graphically present this data using a line chart.

Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from **Session 3-1, *Play Time – Getting Data In***. You should be familiar with the Splunk search bar, the time range picker, and the **Visualization** tab. It is not required, but is advisable, that you complete all the recipes up until this point.

How to do it...

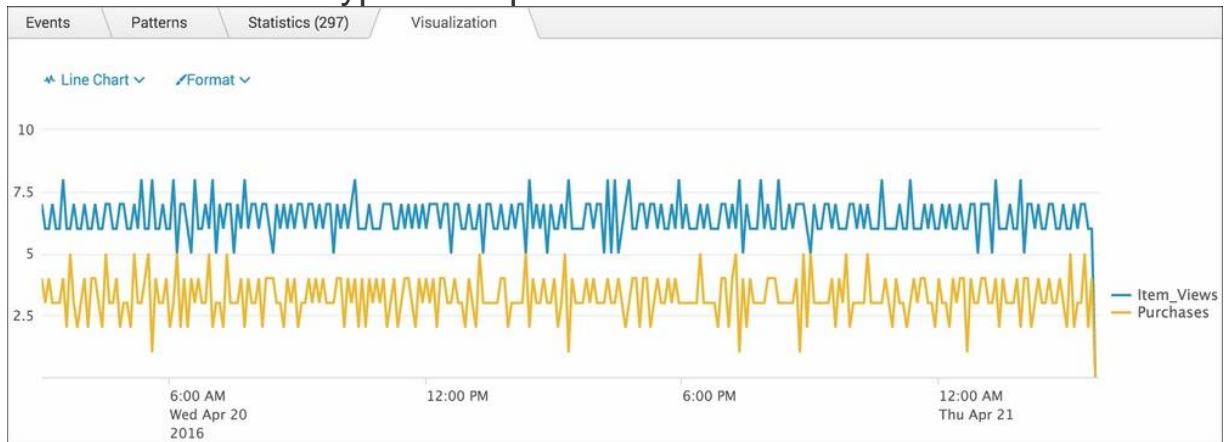
Follow the given steps to create a line chart of item views and purchases over time:

1. Log in to your Splunk server.
2. Select the default **Search & Reporting** application.
3. Ensure that the time range picker is set to **Last 24 hours** and type the following search into the Splunk search bar. Then, click on **Search** or hit *Enter*.

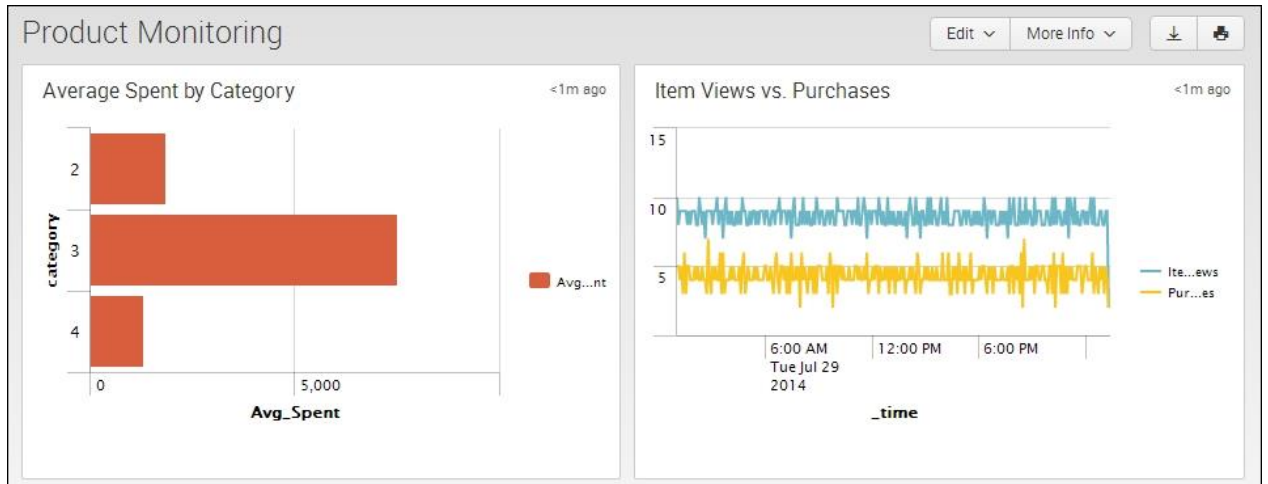
```
index=main sourcetype=access_combined | timechart span=5m  
count(eval(uri_path="/viewItem")) AS Item_Views,  
count(eval(uri_path="/checkout")) AS Purchases
```

4. Splunk will return a time series-based chart, listing the count of item views and the count of purchases over the given time period.

- Click on the **Visualization** tab and select **Line** from the drop-down list of visualization types to represent the data as a line chart:



- Save this report by clicking on **Save As** and then on **Report**. Name the report **cp03_item_views_purchases** and click on **Save**. On the next screen, click on **Add to Dashboard**.
- You will now add this to the **Product Monitoring** dashboard. Select the button labeled **Existing**, and from the drop-down menu that appears, select the **Product Monitoring** dashboard. For the **Panel Title** field value, enter **Item Views vs. Purchases** and select **Report** in **Panel Powered By**; then, click on **Save**.
- The next screen will confirm that the dashboard has been created and the panel has been added. Click on **View Dashboard** to see for yourself.
- Arrange the dashboard so that the line chart panel is to the right of the bar chart panel created in the previous recipe. Click on the **Edit button**, and from the drop-down menu, select **Edit Panels**. Move the line chart panel accordingly.
- Finally, click on **Done** to save the changes to your dashboard:



How it works...

Let's break down the search piece by piece:

Search fragment	Description
<code>index=main sourcetype=access_combined</code>	In this example, we search for our application's logs that have the <code>log4j</code> sourcetype.
<code> timechart span=5m count(eval(uri_path="/viewItem")) AS Item_Views, count(eval(uri_path="/checkout")) AS Purchases</code>	Using the <code>timechart</code> command, we count the total number of times an item is viewed and the total number of purchases that occur.