Recipe 16

Creating Charts and Graphs with Highcharts

Problem

As the old cliché goes, "A picture is worth a thousand words," and charts are no exception. With a chart or graph, we can present data to our customers and clients in a meaningful and often more attractive way.

There are many options for creating charts, from server-side solutions that generate images to systems that that run with Adobe Flash. We need a simple and effective way to create charts and graphs that doesn't require Flash, since it doesn't work on iOS devices, but we also don't want to use resources on our server to generate images.

Ingredients

- jQuery
- Highcharts[7]
- QEDServer

Solution

The Highcharts JavaScript library lets us easily create interactive and readable charts and graphs. It works across platforms, and since it runs on the client's machine, it doesn't require any special configuration on our servers. The interface built into Highcharts is highly interactive and customizable, letting us present data in a number of ways. In this recipe, we'll build and customize a simple chart and then build a more complex one using some remote data.

Our sales team has developed an affiliate program for our company's shopping site. We've been tasked to develop an interface for our affiliates, and we want to show their data in a visual way with graphs and charts. We'll use Highcharts to build these. But first, let's look at what it takes to get a simple chart displayed on our page.

7. http://www.highcharts.com/

Building a Simple Chart

Let's build a simple pie chart so we can get acquainted with Highcharts and its various options. First, let's build a simple HTML document and include the necessary JavaScript files. We'll need highcharts.js, which we can get from the Highcharts website, and we'll need the jQuery library, since Highcharts relies on it. Although other sections of this book use jQuery 1.7, Highcharts requires jQuery version 1.6.2.

Download highcharts/example_chart.html

```
<script type="text/javascript"
  src="/jquery.js">
</script>
<script type="text/javascript" src="/highcharts.js"></script>
```

Now that we have Highcharts loaded, let's build a chart. Highcharts requires us to create a , which it will use to hold the chart, so let's create one in the <body>. We'll give the an id so we can reference it with our JavaScript code, like this:

Download highcharts/example_chart.html

```
<body>
  <div id="pie-chart"></div>
</body>
```

All the magic is done by creating a new instance of the Highcharts.Chart class and passing it some options. Highcharts has many options for configuring a chart, and this configuration can very quickly get long and unwieldy. To keep it simple, we'll create a variable called chartOptions and set some values on it that will be expected by Highcharts.

Download highcharts/example_chart.html

```
$(function() {
  var chartOptions = {};

  chartOptions.chart = {
    renderTo: "pie-chart"
  };
  chartOptions.title = {text: "A sample pie chart"};
  chartOptions.series = [{
    type: "pie",
    name: "Sample chart",
    data: [
      ["Section 1", 30],
      ["Section 2", 50],
      ["Section 3", 20]
    ]
  }];
  var chart = new Highcharts.Chart(chartOptions);
});
```

The first value we set is a chart property that contains information about the chart. This is where we pass the ID of the we created earlier. We set a title for the chart with some sample text. Finally, the series property is an array that contains an object for each type of chart you want to render. Highcharts allows us to pass any number of objects that will be rendered on top of each other. Each object defines a chart type, a name, and a dataset. The format of this data changes depending on the type of chart we're using. For the pie chart, the data is a two-dimensional array where the inner arrays are pairs of X and Y data.

With just a few lines of code, we have a chart that looks like Figure 23, *Our simple pie chart*, on page 121. Let's go a little further now and explore some additional options.

Customizing Our Chart's Appearance

Highcharts supports pie graphs, line graphs, area graphs, and scatter plots, and the extensibility of the graph types lets us create any number of more interesting graphs.

Consider our chartOptions variable from before. We can define a property on it called plotOptions, which is an object containing a number of settings for modifying how the graph is drawn. Let's define some options on our pie chart from earlier.

We can set options for all charts by defining them in the series property on our chartOptions object, but we can also define options for each chart type. Let's customize our pie chart by changing the appearance of the labels that point to each section of the chart.

Download highcharts/example_chart.html

```
var pieChartOptions = {
  dataLabels: {
    style: {
      fontSize: 20
    },
    connectorWidth: 3,
    formatter: function() {
      var label = this.point.name + " : " + this.percentage + "%";
      return label;
    }
  }
};

chartOptions.plotOptions = {
  pie: pieChartOptions
};
```

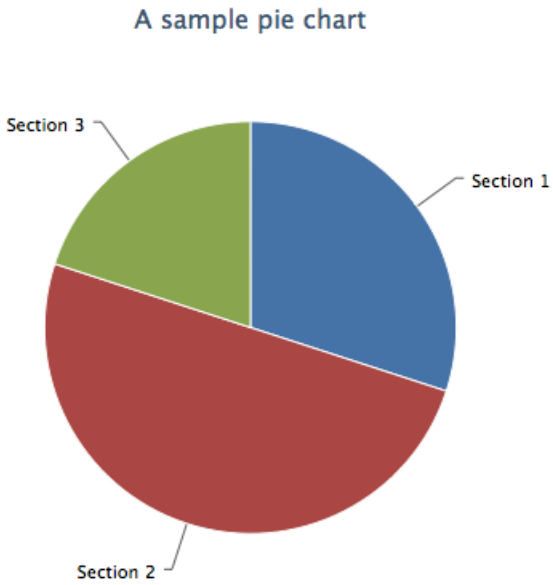

Figure 23—Our simple pie chart

We first increase the font size to make them more visible. Then we increase the connector width to match the font size. Lastly, we create a function that returns a newly formatted label with our desired information. The default labels showed only the point name, so we changed it to show the percentage as well. Our finished graph looks like the one in Figure 24, *Our finished pie chart*, on page 122.

The plotOptions property has a ton of options; refer to the Highcharts documentation on the plotOptions property to see them all.[8]

Now that we know how to create and configure a simple chart, let's use Highcharts to model our affiliate data.

Modeling the Affiliate Data Sets

Our affiliate program tracks quite a bit of data. In most cases, data sets are best represented by a varying types of graphs. To explore another type of graph, we're going to model some customer data that comes through. The customer data includes information such as names, locations of the customer, and age. This kind of information is useful for profiling customers and making

8. http://www.highcharts.com/ref/#plotOptions

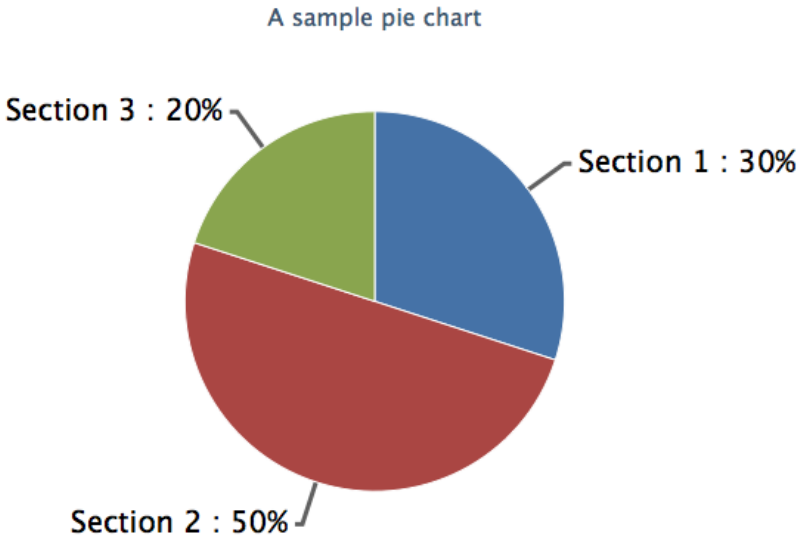

Figure 24—Our finished pie chart

assumptions on how to market products. It's our job to transform this raw data into a graph that our marketing folks can quickly analyze before they dig into the hard data.

We want to be able to glance at the data and understand how old the customers are. Let's use a bar graph so that it's easily to see the mean and the most frequent value. We'll create something that looks like Figure 25, *Our customer data bar graph*, on page 123.

To get started, let's create a new HTML document with jQuery and Highcharts included in it. We'll be working with JSON data and Ajax requests, so place this new HTML file in the public directory of your QEDServer installation.

Download highcharts/affiliates.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Affiliate Customer Data</title>
  <script type="text/javascript"
    src="/jquery.js">
  </script>
  <script type="text/javascript" src="/highcharts.js"></script>
</head>
```

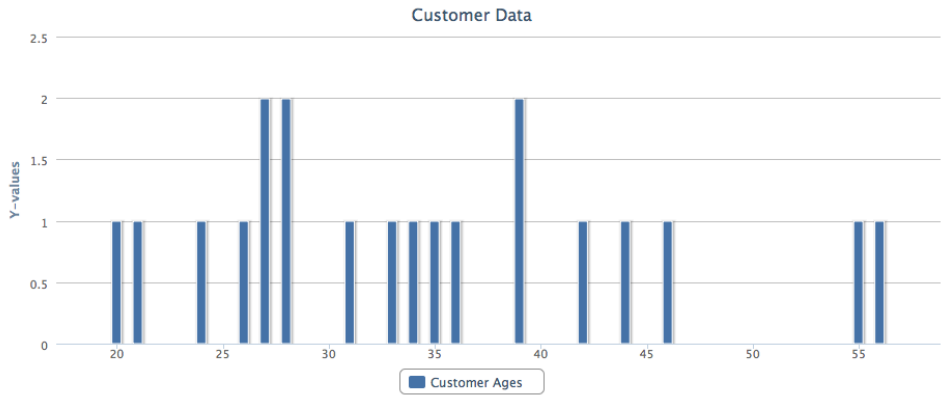

Figure 25—Our customer data bar graph

```
<body>
  <div id="customer-data"></div>
</body>

</html>
```

Within this file, we'll create a <script> and set up our new instance of the Highcharts.Chart class. Let's set a few simple options, including the chart's title and the target element on our page where the chart will go.

Download highcharts/affiliates.html

```
var options = {
  chart: {
    renderTo: "customer-data"
  },
  title: {
    text: "Customer Data"
  },
  credits: {
    enabled: false
  }
};
```

Now that our document is ready to go, let's do some work with our data.

Showing the Customer Data

Normally, we'd get our customer data from a backend system, but for the purpose of this recipe, we've created some sample data you can use. You can find it in the book's source code, which you can download from the book's website.

Remember that we can't just pull in regular JSON data from a remote server because of the security restrictions of the web browser. Our index.html page and our data file have to be hosted on the same web server. Place this sample data file in a folder called sample_data within the public folder that QEDServer uses. This way, QEDServer can serve it from http://localhost:8080/sample_data/customer_data.json, and our page can consume it properly.

To show the ages in a bar graph, we need to pair an age with the number of times it has occurred. Right now, we have only a list of ages. Let's write some JavaScript to collect the ages and sum up the frequencies. We will make a request to get our customer data and do all our work inside of the success callback, which gets invoked when we get data back from our Ajax request.

Download highcharts/affiliates.html

```
$.getJSON('/sample_data/customer_data.json', function(data) {

  var ages = [];

  $.each(data.customers, function(i, customer) {
    if (typeof ages[customer.age] === "undefined") {
      ages[customer.age] = 1;
    } else {
      ages[customer.age] += 1;
    }
  });

  var age_data = [];

  $.each(ages, function(i, e) {
    if (typeof e !== "undefined") {
      age_data.push([i, e]);
    }
  });
});
```

Here, we used an array to store some intermediate data. The ages array uses ages as indexes and stores the number of occurrences for that age. Then, we look through and collect ages that exist in the array to map them to the two-dimensional array that Highcharts needs. Now that we have our data in the correct format, let's render our chart.

Download highcharts/affiliates.html

```
options.series = [{
  type: "column",
  name: "Customer Ages",
  data: age_data
}];

var chart = new Highcharts.Chart(options);
```

Now with our final chart rendered, we can easily glance at the chart and see the most occurring ages for our customers.

Further Exploration

Highcharts is a powerful JavaScript library. In this recipe, we built simple to complex charts that only begin to take advantage of the number of options that are available. The Highcharts reference[9] is a great way to learn about just how much Highcharts is capable of. We recommend taking a look at the documentation and considering what options you would like to use on future projects. Also, the documentation includes a link to an example of most of the available options on JSFiddle.net.[10]

Also See

- Recipe 18, *Accessing Cross-site Data with JSONP*, on page 134
- Recipe 15, *Adding an Inline Google Map*, on page 112
- Recipe 9, *Interacting with Web Pages Using Keyboard Shortcuts*, on page 59
- Recipe 23, *Mobile Drag and Drop*, on page 163

9. http://highcharts.com/ref
10. A JavaScript-sharing site: http://jsfiddle.net