Gauging the Graphing System

Martin Kamerbeek

Overview

- The WebGUI::Image components.
- Including graphing into your code.
- Writing your own graphing plugins.
- Simple example: SimpleGauge.
- Questions.

The WebGUI::Image subsystem

- WebGUI::Image
- **WebGUI::Image::Graph**
- WebGUI::Image::Color
- WebGUI::Image::Palette
- WebGUI::Image::Font

WebGUI::Image

- Base class for all graphs.
- Handy methods:
  - image()
  - getImageHeight() / getImageWidth()
  - getXOffset() / getYOffset()
  - saveToStorageLocation() / saveToFileSystem()
  - text()

Color handling

- You can either use specific colors or colors from the color manager (system colors).
- Specific colors: use for one off stuff like backgrounds, panes axis etc.
- System colors: use for coloring data.
- System colors are in a palette

System Colors: WebGUI::Image::Color

- A WebGUI::Image::Color consists of a Stroke color and a Fill color.
- Colors are defined in terms of R, G and B values.
- Colors also have an alpha (transparency) component.

WebGUI::Image::Color: Handy methods

- sub getFillColor { }
- sub getFillTriplet { }
- sub getFillAlpha { }
- sub darken { }

WebGUI::Image::Palette

- Collection of WebGUI::Image::Color's
- Allows you to cycle through these colors
- Handy method (the only one you need):
  - sub getNextColor { }

Add graphing to your Asset

- Add a graphing tab to the edit form.
- Process the graph config data when saving the edit form, and store it into your asset.
- Instanciate a graph with the saved config data
- Add data.
- Add labels.
- Draw the graph.

Adding the configuration tab

```
sub getEditForm {
   my $self = shift;
   my $tabform = $self->SUPER::getEditForm( @_ );
   ...
   my $config = JSON::decode_json(
     $self->get('graphConfig')
   );

   $tabform->addTab('graph', 'Graphing');
   $tabform->getTab('graph')->raw(
     WebGUI::Image::Graph->getGraphingTab(
       $self->session, $config
     )
   );
}
```

Adding the configuration tab

```
sub getEditForm {
    my $self = shift;
    my $tabform = $self->SUPER::getEditForm( @_ );
    ...
    my $config = JSON::decode_json(
        $self->get('graphConfig')
    );

    $tabform->addTab('graph', 'Graphing');
    $tabform->getTab('graph')->raw(
        WebGUI::Image::Graph->getGraphingTab(
            $self->session, $config
        )
    );
}
```

Adding the configuration tab

```
sub getEditForm {
   my $self = shift;
   my $tabform = $self->SUPER::getEditForm( @_ );
   ...
   my $config = JSON::decode_json(
     $self->get('graphConfig')
   );

   $tabform->addTab('graph', 'Graphing');
   $tabform->getTab('graph')->raw(
     WebGUI::Image::Graph->getGraphingTab(
       $self->session, $config
     )
   );
}
```

Adding the configuration tab

```
sub getEditForm {
    my $self = shift;
    my $tabform = $self->SUPER::getEditForm( @_ );
    ...
    my $config = JSON::decode_json(
        $self->get('graphConfig')
    );

    $tabform->addTab('graph', 'Graphing');
    $tabform->getTab('graph')->raw(
        WebGUI::Image::Graph->getGraphingTab(
            $self->session, $config
        )
    );
}
```

Adding the configuration tab

```
sub getEditForm {
   my $self = shift;
   my $tabform = $self->SUPER::getEditForm( @_ );
   ...
   my $config = JSON::decode_json(
      $self->get('graphConfig')
   );

   $tabform->addTab('graph', 'Graphing');
   $tabform->getTab('graph')->raw(
      WebGUI::Image::Graph->getGraphingTab(
         $self->session, $config
      )
   );
}
```

Processing the configuration tab

```
sub processPropertiesFromFormPost {
  my $self = shift;
  ...
  my $graph =
    WebGUI::Image::Graph->processConfigurationForm(
        $self->session
    );

  $self->update({
    graphConfig   => JSON::encode_json(
      $graph->getConfiguration
    )
  });
}
```

Processing the configuration tab

```
sub processPropertiesFromFormPost {
  my $self = shift;
  ...
  my $graph =
    WebGUI::Image::Graph->processConfigurationForm(
        $self->session
    );

  $self->update({
    graphConfig   => JSON::encode_json(
      $graph->getConfiguration
    )
  });
}
```

Processing the configuration tab

```
sub processPropertiesFromFormPost {
  my $self = shift;
  ...
  my $graph =
    WebGUI::Image::Graph->processConfigurationForm(
        $self->session
    );

  $self->update({
    graphConfig   => JSON::encode_json(
      $graph->getConfiguration
    )
  });
}
```

Processing the configuration tab

```
sub processPropertiesFromFormPost {
  my $self = shift;
  ...
  my $graph =
    WebGUI::Image::Graph->processConfigurationForm(
        $self->session
    );

  $self->update({
    graphConfig   => JSON::encode_json(
      $graph->getConfiguration
    )
  });
}
```

Draw a graph

```
my $config = JSON::decode_json(
  $self->get('graphConfig')
);

if ($config) {
  my $graph =
    WebGUI::Image::Graph->loadByConfiguration(
      $self->session, $config
    );
  $graph->addDataset( \@dataset );
  $graph->setLabels( \@labels );
  $graph->draw;

  ... (write the graph to disk, discussed later)
}
```

Draw a graph

```
my $config = JSON::decode_json(
  $self->get('graphConfig')
);

if ($config) {
  my $graph =
    WebGUI::Image::Graph->loadByConfiguration(
      $self->session, $config
    );
  $graph->addDataset( \@dataset );
  $graph->setLabels( \@labels );
  $graph->draw;

  ... (write the graph to disk, discussed later)
}
```

Draw a graph

```
my $config = JSON::decode_json(
  $self->get('graphConfig')
);

if ($config) {
  my $graph =
    WebGUI::Image::Graph->loadByConfiguration(
      $self->session, $config
    );
  $graph->addDataset( \@dataset );
  $graph->setLabels( \@labels );
  $graph->draw;

  ... (write the graph to disk, discussed later)
}
```

Draw a graph

```
my $config = JSON::decode_json(
  $self->get('graphConfig')
);

if ($config) {
  my $graph =
    WebGUI::Image::Graph->loadByConfiguration(
      $self->session, $config
    );
  $graph->addDataset( \@dataset );
  $graph->setLabels( \@labels );
  $graph->draw;

  ... (write the graph to disk, discussed later)
}
```

Draw a graph

```
my $config = JSON::decode_json(
  $self->get('graphConfig')
);

if ($config) {
  my $graph =
    WebGUI::Image::Graph->loadByConfiguration(
      $self->session, $config
    );
  $graph->addDataset( \@dataset );
  $graph->setLabels( \@labels );
  $graph->draw;

  ... (write the graph to disk, discussed later)
}
```

Draw a graph

```
my $config = JSON::decode_json(
  $self->get('graphConfig')
);

if ($config) {
  my $graph =
    WebGUI::Image::Graph->loadByConfiguration(
      $self->session, $config
    );
  $graph->addDataset( \@dataset );
  $graph->setLabels( \@labels );
  $graph->draw;

  ... (write the graph to disk, discussed later)
}
```

Draw a graph

```
my $config = JSON::decode_json(
  $self->get('graphConfig')
);

if ($config) {
  my $graph =
    WebGUI::Image::Graph->loadByConfiguration(
      $self->session, $config
    );
  $graph->addDataset( \@dataset );
  $graph->setLabels( \@labels );
  $graph->draw;

  ... (write the graph to disk, discussed later)
}
```

Write the image to disk

```
    ...(see previous slides)
    $graph->draw;

    my $storage = WebGUI::Storage::Image->createTemp(
      $self->session
    );
    $graph->saveToStorageLocation(
      $storage, 'filename.png'
    );
}
```

Write the image to disk

```
    ...(see previous slides)
    $graph->draw;

    my $storage = WebGUI::Storage::Image->createTemp(
      $self->session
    );
    $graph->saveToStorageLocation(
      $storage, 'filename.png'
    );
}
```

Write the image to disk

```
    ...(see previous slides)
    $graph->draw;

    my $storage = WebGUI::Storage::Image->createTemp(
      $self->session
    );
    $graph->saveToStorageLocation(
      $storage, 'filename.png'
    );
}
```

Required methods

- sub configurationForm { }
- sub formNamespace { }
- sub getConfiguration { }
- sub setConfiguration { }
- sub draw { }

Example: SimpleGauge

```
package WebGUI::Image::Graph::SimpleGauge;

use strict;

use base qw{ WebGUI::Image::Graph  };
use constant pi => 3.14159265358979;
use WebGUI::HTMLForm;

code here

1;
```

sub configurationForm { }

- Provides the form for the properties of the graph.
- Returns a hash of printRowsOnly'd HTMLForm objects.
- Use namespace with :: replaced with _ as a hash key.

sub configurationForm { }

```
sub configurationForm {
    my $self    = shift;

    my $f = WebGUI::HTMLForm->new($self->session);
    $f->trClass( 'Graph_SimpleGauge' );
    $f->color(
        name    => 'simplegauge_axisColor',
        label   => 'Axis color',
        value   => $self->getAxisColor,
    );

    my $cf = $self->SUPER::configurationForm;
    $cf->{ 'graph_simplegauge'  } =
      $f->printRowsOnly;

    return $configurationForms;
}
```

sub formNamespace { }

- Returns the namespace with :: replaced with _.

```
sub formNamespace {
    my $self = shift;

    return
      $self->SUPER::formNamespace.'_SimpleGauge';
}
```

sub getConfiguration { }

- Returns a hashref with the configuration options of the graph.
- Make sure keys are named just like those in the configurationForm generated form.

sub getConfiguration { }

```
sub getConfiguration {
    my $self = shift;

    my $config = $self->SUPER::getConfiguration;

    return {
        %{ $config },
        simplegauge_axisColor   =>
          $self->getAxisColor,
    };
}
```

sub setConfiguration { }

- Takes a hashref containing configuration data and applies it to the graph.
- The hashref should have the same form as that generated by getConfiguration.

sub setConfiguration { }

```
sub setConfiguration {
    my $self    = shift;
    my $config  = shift;

    $self->SUPER::setConfiguration( $config );

    $self->setAxisColor(
      $config->{ simplegauge_axisColor }
    );
}
```

sub draw { }

- Performs all graphical manipulations necesarry.
- Does not save an image to disk!

sub draw { }

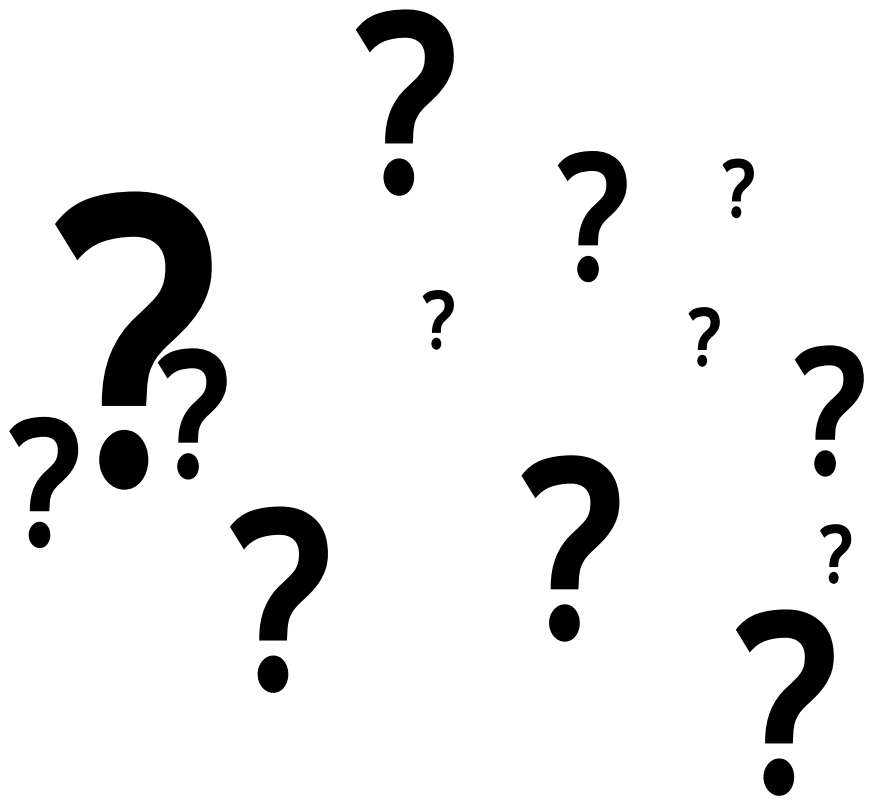
```
sub draw {
    my $self = shift;

    $self->drawBackground;
    $self->drawAxes;
    $self->drawLabels;

    my $palette = $self->getPalette;

    foreach my $set ( @{ $self->getDataset } ) {
        foreach my $element ( @{ $set } ) {
            $self->drawNeedle(
                  $element,
                  $palette->getNextColor );
        }
    }
}
```

Questions?



oqapi