blob: 55fcc6ff23847b80e81c7bfeab21014af05e76c2 (plain)

```php
<?php

/**
 * Implements hook_tribune_after_post().
 */
function tribune_weather_tribune_after_post($node, $post) {
  $enabled = variable_get('tribune_weather_enabled_' . $node->nid, FALSE);

  if ($enabled) {
    $message = $post['text'];
    if (strpos($message, t('/weather')) === 0) {
      $place = trim(str_replace(t('/weather'), '', $message));

      $units = variable_get('tribune_weather_units_' . $node->nid, 'metric');

      $weather = tribune_weather_get_condition($place, $units);

      $clock = tribune_format_time($node, $post['timestamp']);
      tribune_post($node, $clock . ' ' . $weather, user_load(0), 'Meteo');
    } elseif (strpos($message, t('/forecast')) === 0) {
      $place = trim(str_replace(t('/forecast'), '', $message));

      $units = variable_get('tribune_weather_units_' . $node->nid, 'metric');

      $weather = tribune_weather_get_forecast($place, $units);

      $clock = tribune_format_time($node, $post['timestamp']);
      tribune_post($node, $clock . ' ' . $weather, user_load(0), 'Meteo');
    }
  }
}

function tribune_weather_get_condition($place, $units) {
  $url = 'http://api.openweathermap.org/data/2.1/find/name?q=' . urlencode($place);

  switch ($units) {
    case 'imperial':
      $url .= '&units=imperial';
      $degree_unit = t('°F');
      $wind_unit = t('mph');
      break;
    case 'si':
      $url .= '&units=si';
      $degree_unit = t('°K');
      $wind_unit = t('km/h');
      break;
    case 'metric':
    default:
      $url .= '&units=metric';
      $degree_unit = t('°C');
      $wind_unit = t('km/h');
  }

  $result = drupal_http_request($url);

  if (isset($result->error) or !($data = drupal_json_decode($result->data)) or !$data['count']) {
    return t('Unable to retrieve data for !place', array('!place' => $place));
  } else {
    foreach ($data['list'] as $city) {
      $name = $city['name'];
      $temp = round($city['main']['temp']);
      $humidity = round($city['main']['humidity']);
      $wind = round($city['wind']['speed']);
      $condition = $city['weather'][0]['main'];

      return t('Weather in <b>!city</b>: !condition, !degrees!degreeunit, !humidity% humidity, !wind !windunit', array(
        '!city' => $name,
        '!condition' => $condition,
        '!degrees' => $temp,
        '!degreeunit' => $degree_unit,
        '!humidity' => $humidity,
        '!wind' => $wind,
        '!windunit' => $wind_unit,
      ));
    }
  }
}

function tribune_weather_get_forecast($place, $units) {
  $code_url = 'http://api.openweathermap.org/data/2.1/find/name?q=' . urlencode($place);
  $url = 'http://api.openweathermap.org/data/2.2/forecast/city/%code?mode=daily_compact';

  switch ($units) {
    case 'imperial':
      $url .= '&units=imperial';
      $degree_unit = t('°F');
      $wind_unit = t('mph');
      $date_format = 'm/d';
      break;
    case 'si':
      $url .= '&units=si';
      $degree_unit = t('°K');
      $wind_unit = t('km/h');
      $date_format = 'Y-m-d';
      break;
      break;
    case 'metric':
    default:
      $url .= '&units=metric';
      $degree_unit = t('°C');
      $wind_unit = t('km/h');
      $date_format = 'd/m';
  }

  $result = drupal_http_request($code_url);

  if (isset($result->error) or !($data = drupal_json_decode($result->data)) or !$data['count']) {
    return t('Unable to retrieve data for !place', array('!place' => $place));
  } else {
    $code = $data['list'][0]['id'];
    $result = drupal_http_request(str_replace('%code', $code, $url));
    $data = drupal_json_decode($result->data);

    $name = $data['city']['name'];
    $string = t('Forecast for <b>!city</b> — ', array('!city' => $name));
    $days = array();
    foreach ($data['list'] as $point) {
      $date = date($date_format, $point['dt']);

      if (isset($days[$date]) or count($days) > 4) {
        continue;
      }

      $temp_min = round($point['morn']);
      $temp_max = round($point['temp']);
      $humidity = round($point['main']['humidity']);
      $wind = round($point['wind']['speed']);
      $condition = $point['weather'][0]['description'];

      $days[$date] = t('!date: !condition !degrees_min»!degrees_max!degreeunit', array(
        '!date' => $date,
        '!condition' => $condition,
        '!degrees_min' => $temp_min,
        '!degrees_max' => $temp_max,
        '!degreeunit' => $degree_unit,
      ));
    }

    $string .= join(', ', $days);

    return $string;
  }
}

/**
 * Implements hook_tribune_form().
 */
function tribune_weather_tribune_form($node, $form_state) {
  $default_value = 'NULL';
  $default_value_units = 'metric';

  if (!empty($node->nid)) {
    $default_value = variable_get('tribune_weather_enabled_' . $node->nid, FALSE);
    $default_value_units = variable_get('tribune_weather_units_' . $node->nid, 'metric');
  }

  return array(
    'tribune_weather_details' => array(
      '#weight' => 10,
      '#type' => 'fieldset',
      '#title' => t('Weather'),
      '#description' => t('Weather condition and forecast is taken from http://openweathermap.org.'),

      'tribune_weather_enabled' => array(
        '#type' => 'checkbox',
        '#title' => t('Enable weather commands'),
        '#default_value' => $default_value,
        '#description' => t('Weather condition for a place will be available by typing <code>/weather &lt;city&gt;</code>, forecast with <code>/forecast &lt;city&gt;</code>.'),
      ),
      'tribune_weather_units' => array(
        '#type' => 'select',
        '#title' => t('Unit system use'),
        '#default_value' => $default_value_units,
        '#options' => array(
          'metric' => t('Metric'),
          'imperial' => t('Imperial'),
          'si' => t('SI'),
        ),
      ),
    ),
  );
}

/**
 * Implements hook_node_update().
 */
function tribune_weather_node_update($node) {
  if ($node->type == 'tribune' and isset($node->tribune_weather_enabled)) {
    variable_set('tribune_weather_enabled_' . $node->nid, $node->tribune_weather_enabled);
    variable_set('tribune_weather_units_' . $node->nid, $node->tribune_weather_units);
  }
}

/**
 * Implements hook_node_insert().
 */
function tribune_weather_node_insert($node) {
  if ($node->type == 'tribune' and isset($node->tribune_weather_enabled)) {
    variable_set('tribune_weather_enabled_' . $node->nid, $node->tribune_weather_enabled);
    variable_set('tribune_weather_units_' . $node->nid, $node->tribune_weather_units);
  }
}

/**
 * Implements hook_node_delete().
 */
function tribune_weather_node_delete($node) {
  if ($node->type == 'tribune') {
    variable_del('tribune_weather_enabled_' . $node->nid);
    variable_del('tribune_weather_units_' . $node->nid);
  }
}
```