phpGroupWare admin/config.php A brief introduction to writing hooks and templates for any application to use this admin interface, by Miles Lott <milosch@phpgroupware.org> Dec 22, 2001. Files config.tpl (required) In your application/templates/default directory, create a new template file named ’config.tpl’. This will be included by config.php and used to draw the page. This template should include a POST method form. The following template tags may be used: {action_url} - A phpgw->link to config.php will be inserted. {title} - This will be parsed to display ’Site Configuration’. {th_bg},{th_text},{row_on},{row_off} - Replaced with the current theme colors. and the following special types: {lang_XXX} - Filled with lang(’XXX’). {value_XXX} - Filled with the current value of config item ’XXX’. {selected_XXX} - set to ”, or ’ selected’ if an option value is current. {hook_XXX} - Calls a function named XXX (will be discussed later). {checked_XXX+YYY} - Handles checkbox/radio values set to YYY or ” (will be discussed later). Following is an example from the addressbook application: <form method="POST" action="{action_url}"> <table border="0" align="center"> <tr bgcolor="{th_bg}"> <td colspan="2"> {title}</td> </tr> <tr bgcolor="{th_err}"> <td colspan="2"> {error}</td> </tr> <!-- END header --> <!-- BEGIN body --> <tr bgcolor="{row_on}"> <td colspan="2"> </td> </tr> <tr bgcolor="{row_off}"> <td colspan="2"> {lang_Addressbook}/{lang_Contact_Settings} </td> </tr> <tr bgcolor="{row_on}"> <td>{lang_Contact_application}:</td> <td><input name="newsettings[contact_application]" value="{value_contact_application}"></td> </tr> ... Note the fieldname, newsettings[contact_application]. This array name must be used for the form values. Next, note the value setting for this form element, {value_contact_application}. This indicates that we want the current value of the config setting, ’contact_application’, to be set and displayed on the form. Lastly, look at the template element, {lang_Contact_application}. Here, the value from the lang db table will be inserted if available. Let’s take a look at part of the preferences/default/config.tpl: <tr bgcolor="{row_on}"> <td>{lang_Country_Selection} ({lang_Text_Entry}/{lang_SelectBox}):</td> <td> <select name="newsettings[countrylist]"> {hook_country_set} </select> </td> </tr> Here, we are adding a new element, {hook_country_set}. This brings up the next file we will need to parse this value, hook_config.inc.php. But first, let’s look at the last template type, ’checked’: <input name="newsettings[enable_remote_addressbook]" type="radio" value=""{checked_enable_remote_addressbook+}>{lang_no} <input name="newsettings[enable_remote_addressbook]" type="radio" value="True"{checked_enable_remote_addressbook+True}>{lang_yes} We want to check the value of the setting ’enable_remote_addressbook’. The value could be ” or ’True’. We use the ’+’ character to isolate the config name from the check value. If the value is empty or unset in the phpgw_config table, {checked_enable_remote_addressbook+} is replaced with ’ checked’. If the value is ’True’,{checked_enable_remote_addressbook+True} is replaced with ’ checked’. Note that the part after the ’+’ character matches what is in the value=”XXX” part in the html for this form element. hook_config.inc.php (optional) At each invocation of config.php, a call to the common class function hook_single() is made. It attempts to include a file, hook_config.inc.php as a set of code for config.php to use. In the case of the preferences example above, using hook_country_set, here is the corresponding function in preferences/inc/hook_config.inc.php: function country_set($config) { $country = array(’user_choice’ => ’Users Choice’, ’force_select’ => ’Force Selectbox’); while (list ($key, $value) = each ($country)) { if ($config[’countrylist’] == $key) { $selected = ’ selected’; } else { $selected = ”; } $descr = lang($value); $out .= ’<option value="’ . $key . ’"’ . $selected . ’>’ . $descr . ’</option>’ . "\n"; } return $out; } Note again the template value we used earlier, {hook_country_set}. This causes config.php to look for a function named country_set(). Since we included the file with this function via the hook_single() call, this function is executed. It’s return is a string, and the function prints nothing itself. hook_config_validate.inc.php (optional) Once the admin clicks the submit button to post the form, we can optionally validate their input using one or many different functions. This is done by first making another call to hook_single() in the API common class. This time, the name config_validate is used, so common tries to include ’application/inc/hook_config_validate.inc.php’. If this file exists, it sets a var to tell config.php it was found. Following then are functions named after each config we want to validate. The following example is for addressbook: $GLOBALS[’phpgw_info’][’server’][’found_validation_hook’] = True; /* Check a specific setting. Name must match the setting. */ function ldap_contact_context($value=”) { if($value == $GLOBALS[’phpgw_info’][’server’][’ldap_context’]) { $GLOBALS[’config_error’] = ’Contact context for ldap must be different from the context used for accounts’; } elseif($value == $GLOBALS[’phpgw_info’][’server’][’ldap_group_context’]) { $GLOBALS[’config_error’] = ’Contact context for ldap must be different from the context used for groups’; } else { $GLOBALS[’config_error’] = ”; } } Here we created a function to check the entered value for the config item, ldap_contact_context. We want to make sure the admin did not set this value to one which would conflict with another config item, used for accounts or groups in phpGroupWare. config.php calls this function, sending it the POSTed value. config.php continues, adding all other config items from the POSTed values. The variable $GLOBALS[’config_error’] is parsed through lang(), then appended to the local variable, $error. If this has any value after the POSTed variables are checked, the form then has its {error} tag filled with this result. The form is displayed again, with the error. If $error has no value, config.php redirects to admin/index.php. However, there is one more function that may be included in hook_config_validate.inc.php: /* Check all settings to validate input. Name must be ’final_validation’ */ function final_validation($value=”) { if($value[’contact_repository’] == ’ldap’ && !$value[’ldap_contact_dn’]) { $GLOBALS[’config_error’] = ’Contact dn must be set’; } elseif($value[’contact_repository’] == ’ldap’ && !$value[’ldap_contact_context’]) { $GLOBALS[’config_error’] = ’Contact context must be set’; } else { $GLOBALS[’config_error’] = ”; } } config.php checks for the existence of the function ’final_validation()’. This function can be used to check all form values at once. It gets sent the entire $newsettings array POSTed from the form. As with the other functions in this file, final_validation() should set $GLOBALS[’config_error’] if there is a problem.