

# Drupal NodeAPI



# Alapfogalmak

- node object:
  - {node} tábla
  - hookok által adott plussz dolgok
- node id:
  - egyedi azonosítója a node-nak
  - nid néven szerepel az adatbázistáblákban

# Alapműveletek

- betöltés: `node_load($nid)`
- mentés: `node_save($node)`
- törlés: `node_delete($nid)`
- megjelenítés (HTML-t ad vissza):  
`node_view($node)`

# {node} tábla

Column	Type	Modifiers
nid	integer	not null default nextval('node_nid_seq'::regclass)
vid	int_unsigned	not null default 0
type	character varying(32)	not null default ''::character varying
language	character varying(12)	not null default ''::character varying
title	character varying(255)	not null default ''::character varying
uid	integer	not null default 0
status	integer	not null default 1
created	integer	not null default 0
changed	integer	not null default 0
comment	integer	not null default 0
promote	integer	not null default 0
moderate	integer	not null default 0
sticky	integer	not null default 0
tnid	int_unsigned	not null default 0
translate	integer	not null default 0

# Felosztás

- Meglévő tartalomtípus módosítása
- Új tartalomtípus hozzáadása

# Új tartalomtípus definiálása

# Kötelező hookok

- `hook_node_info()`
- `hook_perm()`
- `hook_access()` \*
- `hook_form()` \*

# Opcionális hookok

- `hook_insert()` \*
- `hook_update()` \*
- `hook_delete()` \*
- `hook_validate()` \*
- `hook_view()` \*
- `hook_load()` \*

# hook\_node\_info()

```
hook_node_info() {  
    return array(  
        'foobar' => $tulajdonsagok;  
    );  
}
```

# Kötelező elemek

```
array(  
  'name' => t('Könnyen olvasható név'),  
  'module' => 'content_type_neve',  
  'description' => t('hosszas leírás...'),  
);
```

# Opcionális elemek

```
array(  
  'help' => t('help szöveg...'),  
  'has_title' => TRUE,  
  'title_label' => t('Title'),  
  'has_body' => TRUE,  
  'body_label' => t('Body'),  
  'min_word_count' => 0,  
  'locked' => TRUE,  
);
```

# hook\_perm()

```
function hook_perm() {  
    return array(  
        'create example content',  
        'delete own example content',  
        'delete any example content',  
        'edit own example content',  
        'edit any example content',  
    );  
}
```

# hook\_access()

```
function hook_access($op, $node, $account) {
  if ($op == 'create') {
    return user_access('create example content', $account);
  }
  if ($op == 'update') {
    if (user_access('edit any example content', $account) || (user_access('edit own example content', $account) && ($account->uid == $node->uid))) {
      return TRUE;
    }
  }
  if ($op == 'delete') {
    if (user_access('delete any example content', $account) || (user_access('delete own example content', $account) && ($account->uid == $node->uid))) {
      return TRUE;
    }
  }
  if ($op == 'view') {
    return NULL;
  }
}
```

# hook\_access()

- Visszatérési érték:
  - TRUE – engedély megadva
  - FALSE – engedély megtagadva
  - NULL – nem rendelkezik, így majd a node\_access tábla dönt, vagy valamelyik más access control modul

# hook\_form()

```
function hook_form(&$node, $form_state) {  
    $form = array();  
    $type = node_get_types('type', $node);  
    if($type->has_title) {  
        $form['title'] = array(  
            '#type' => 'textfield',  
            '#title' => check_plain($type->title_label),  
            '#required' => TRUE,  
            '#default_value' => $node->title,  
            '#weight' => -5,  
        );  
    }  
}
```

# hook\_form()

```
if($type->has_body) {
    $form['body_field'] = node_body_field($node, $type->body_label, $type->min_word_count);
}
$form['color'] = array(
    '#type' => 'textfield',
    '#title' => t('Color'),
    '#default_value' => isset($node->color) ? $node->color : "",
);
$form['quantity'] = array(
    '#type' => 'textfield',
    '#title' => t('Quantity'),
    '#default_value' => isset($node->quantity) ? $node->quantity : 0,
    '#size' => 10,
    '#maxlength' => 10,
);
return $form;
}
```

# hook\_insert()

```
function hook_insert($node) {  
    db_query('INSERT INTO {node_foo}(nid, vid, extra)  
VALUES(%d, %d, \'%s\')', $node->nid, $node->vid,  
$node->extra);  
}
```

# hook\_update()

```
function hook_update($node) {  
    if($node->revision) {  
        hook_insert($node);  
    } else {  
        db_query('UPDATE {node_foo} SET extra = \'%s\'  
WHERE nid = %d', $node->extra, $node->nid);  
    }  
}
```

# hook\_delete()

```
function hook_delete($node) {  
    db_query('DELETE FROM {node_foo} WHERE nid  
    = %d', $node->nid);  
}
```

# hook\_validate()

```
function hook_validate($node, &$form) {  
    if(something_is_wrong_with($form)) {  
        form_set_error(...);  
    }  
}
```

# hook\_view()

```
function hook_view($node, $teaser = FALSE, $page = FALSE) {  
    $node = node_prepare($node, $teaser);  
    $node->content['myfield'] = array(  
        '#value' => theme('node_example_order_info', $node),  
        '#weight' => 1,  
    );  
  
    return $node;  
}
```

# hook\_load()

```
function hook_load($node) {  
    $additions = db_fetch_object(db_query('SELECT *  
    FROM {node_foo} WHERE nid = %d AND vid =  
    %d', $node->nid, $node->vid));  
    return $additions;  
}
```

# Theme függvény

```
function theme_node_example_order_info($node) {  
  $output = '<div class="node_example_order_info">';  
  $output .= t('The order is for %quantity %color items.',  
array('%quantity' => check_plain($node->quantity),  
'%color' => check_plain($node->color)));  
  $output .= '</div>';  
  return $output;  
}
```

# Példa

- Blog modul (könnyen olvasható, rövid)
- Book modul

# Meglévő tartalomtípus módosítása

# hook\_nodeapi()

- Az egyik legcsúnyább ős-Drupal függvény, ami benne maradt a Drupal 6-ban
- Szignatúra

```
function hook_nodeapi(&$node, $op, $a3 = NULL,  
$a4 = NULL)
```

# \$op lista

- alter: \$node->content lerenderelődött, a body és a teaser HTML-t tartalmaz. Ezt az \$op-ot nyers szövegműveletekre (cserélés, szűrés) szabad használni
- delete: a node törlésénél hajtódik végre
- delete revision: node revision törlése (erre az egy \$op-ra saját tartalomtípusnál is lehet szükség, mivel csak így lehet revision-t törölni)

# \$op lista

- insert: a node létrejön
- load: a node betöltődik. Itt lehet hozzáadni dolgokat a node objecthez.
- prepare: a node megjelenni készül egy add/edit formon
- prepare translation: a node fordításhoz való klónozásakor fut le
- print: a node előkészül nyomtatásra (példa: book modulban)

# \$op lista

- rss item: RSS generálódik. Lásd `comment_nodeapi()` függvény
- search result: a node keresési eredményként lesz megjelenítve. Csak plussz információ átadására lehet ezt használni.
- presave: a node már validálódott, de még nincs mentve

# \$op lista

- update: node frissül
- update index: a node indexelődik. Arra való, hogy extra információ is indexelődjön (amit nem ad át a hook\_view())
- validate: ugyanaz, mint a hook\_validate()
- view: közvetlen a hook\_view() után lesz meghívva a node megjelenítésekor

# \$a3

- view: \$teaser
- validate: \$form

\$a4

- view: \$page

# Visszatérési értékek

- A "presave", "insert", "update", "delete", "print" és "view" \$op esetén nincs.
- "load" \$op esetén egy asszociatív tömb(!), ami hozzáadódik a node objecthez

# Kérdések