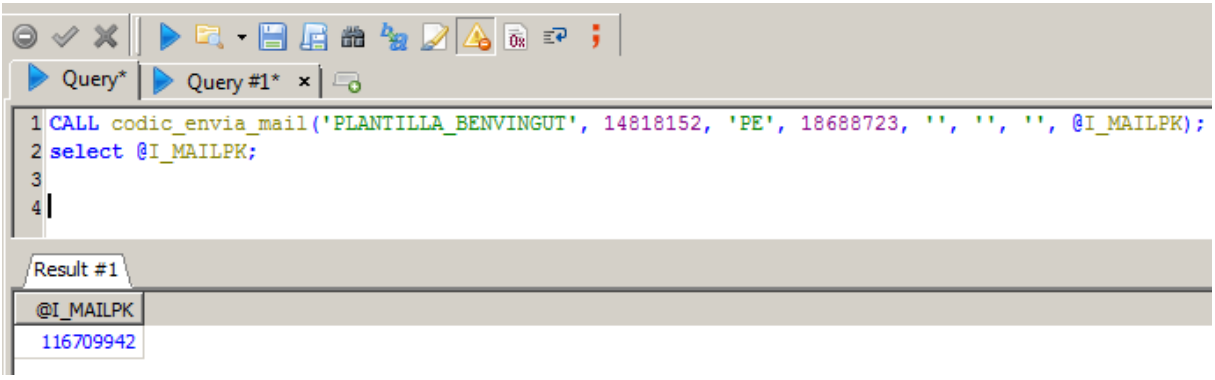
03/08/10 - Crear un procediment emmagatzemat en MySql que retorni paràmetres

Els procediments emmagatzemats (stored procedures) poden tenir paràmetres d'entrada IN i paràmetres de sortida OUT.



Per a definir aquest procediment empraríem un codi SQL com ara aquest:

```
CREATE PROCEDURE `codic_envia_mail`(IN `S_PLANTILLA`
VARCHAR(50), IN `I_USUARI` INT, IN `S_TIPUS` VARCHAR(5), IN
`I_CODI` INT, IN `S_PARAM1` VARCHAR(50), IN `S_PARAM2`
VARCHAR(50), IN `S_PARAM3` VARCHAR(50), OUT `I_MAILPK`
INT)ArrayLANGUAGE SQLArrayNOT DETERMINISTICArrayCONTAINS
SQLArraySQL SECURITY DEFINERArrayCOMMENT ''ArrayBEGIN
```

Els paràmetres d'entrada són les dades que li passem al procediment, i els paràmetres de sortida són els que ens permeten recuperar amb una altra comanda SELECT les dades que ha preparat el procediment.

En la imatge superior es passen 7 paràmetres d'entrada i un de sortida @I_MAILPK.

A nivell de codi PHP i ZendDb cridaríem el procediment i recuperaríem la variable de sortida amb un codi similar a aquest:

```
Array$s_variable_de_sortida = '@I_MAILPK';Arraytry {Array
$o_sel=dbHandler::getAdapter()Array    ->query("CALL
codic_envia_plantilla('{$s_plantilla}', {$i_usuari},
'{$s_tipus}', {$i_codi}, '{$s_parametre1}', '{$s_parametre2}',
```

```
'{$s_parametre3}', {$s_variable_de_sortida})");Array    //
Recuperar la variable de sortida. Pe: @I_MAILPKArray
$o_sel2=dbHandler::getAdapter()Array        ->query("SELECT
{$s_variable_de_sortida}");Array    // Nota: Encara millor si
feu servir la substitució amb ? ja que llavors és ZendDb qui
filtra possibles caracters problemàticsArray    $o_sql = "CALL
codic_envia_plantilla(?, ?, ?, ?, ?, ?, ?,
{$s_variable_de_sortida})";Array
$o_sel=dbHandler::getAdapter()Array -->query($s_sql,
array($s_plantilla, $i_usuari, $s_tipus, $i_codi,
$s_parametre1, $s_parametre2, $s_parametre3));ArrayArray    //
Recuperar la variable de sortida. Pe: @I_MAILPKArray
$o_sel2=dbHandler::getAdapter()Array        ->query("SELECT
{$s_variable_de_sortida}");ArrayArray    if
($o_sel2->rowCount() > 0){Array        $st_dades =
$o_sel2->fetch();Array        $i_resultat =
$st_dades[$s_variable_de_sortida];Array        return true;
// S'ha executat bé, sortim de la funció retornant trueArray
 }Array    elseArray    {Array        // El procediment no ha
retornat cap valor dins la variableArray        // així que no
ha anat béArray        return false;Array    }Array} catch
(Exception $e) {Array    // Hi ha hagut una excepció
(Exception)Array    throw new Exception('Excepció enviant el
correu: '.$e->getMessage());Array}
```

Els prefixes $s_ $i_ ens indiquen quin tipus de variable és:

$s_ string, cadena de caracters
$i_ integer, número sencer
$b_ boolean, valor booleà true/false >0 o 0/null/blanc
$st_ structure, estructura, com ara un array
$o_ object, objecte per exemple de base de dades

La variable de sortida a MySql porta l'ensaïmada o arrova davant.
Podem utilitzar altres notacions com $r_ per als arrays, etc.. però és important de cara a la llegibilitat del nostre codi sempre emprar la mateixa notació.
Emprar aquests prefixes ens estalviarà errors de programació, per exemple sumar cadenes amb números.

Un exemple de programació sense ZendDb i dos paràmetres de sortida:

```
mysql_query = "call sp_GetUsers('ProjectName',@ErrorNumber,
@ErrorMessage)";Array$result = mysql_query('select
@ErrorNumber, @ErrorMessage');Array$row =
mysql_fetch_assoc($result);Arrayprint_r($row);
```

Com veieu sense indicar els prefixes resulta força menys intuitiu.

Per a més exemple de crides IN/OUT sense ZendDb podeu veure:

http://www.herongyang.com/jdbc/MySQL-CallableStatement-Procedure-Parameters.html

http://www.tek-tips.com/viewthread.cfm?qid=1441148&page=10

Traduir a l'Anglès. Translate to English	Compartir: ||