BRANTECT

Architectural　Design Conventions

2013/5/31

Daisuke Tsujino

The content below is a summary for policies during the development of BRANTECT. It is aimed to increase system quality, and maintainability by decisions related to conventions. Although, mainly the design of architecture will be made depending on content of this agreement, deployment of emerging technologies, system performance improvement etc., issues are requires review of the design in future.

1. **Basic policy**
2. **Entry Point**

Entry point to whole system is simply through accessing “r.php”. “r.php” has the functionalities shown below.

1. Include of files that is in use through whole system.
2. Acquisition and verification of the transition destination.
3. Include of transition destination
4. Storage of access log
5. Argument validation (only argument validation of numerical assumption)

After accessing “r.php”, each designated module gets included. However, if there is no need to leave log for access of resources such as images, access can be done directly to specific URL.

1. **Structure within the Module**

In BRANTECT major functions for each category are modularized. The structure in modules is as in MVC structure. By dividing into model, view, controller, it is intended to have good structure outlook. And, entry point in module has one unified structure (app.php). Related to flow of process with the structure of one unified entry point and MVC;

1. The label (t) that is specified in argument at “app.php” includes the file that determines the controller to be processed. Module includes the file that will be used in common.
2. The process of logic occurs in controller. In regards to requirements, corresponding view for acquisition of data, and storage gets included by model class.
3. Acquisition and storage of logical data that has been collected occurs in model. Since there is no controller, in logic model class itself holds the targeted data.
4. View, generates the HTML that presents user. In doing so, whenever it is necessary it presents the information by taking over the model class that has been generated at controller.
5. **File Structure**

1. System-wide Structure

As the structure of first level of the system below, “r.php” as an entry point and directory of each module, consists with required resources.

-r.php

-ad/

-api/

-css/ （common resources）

-de/

-dn/

-faq/ （covers the structure r.php uses）

-img/ （common resources）

-inc/ （common include file）

-js/　（common resources）

-lib/　（common include file）

-mo/

-nr/

-st/

-sub/ (system-wide common sub-window)

-tm/

1. Structure in Module

The directory structure of the module that acts as second level of the system is configured as below.

-de/

+action/

+js/

+lib/

+model/

+view/

+app.php

The controller, model, and view files are placed in the directories: [action], [model], [view]. Into the [lib], [js] directories, place commonly used functions such as JS. As the files in this module are used only for this level, if you want to use system-wide it is better to add it under level 1.

1. File Name

The files in the [action], [view] directories are named to summarizes function name and screen name. The controller of both directories, are named with a same file name that aims explains interaction with view. However, (Name convention that can be separated by only file name) there is a possibility that view may have the same name that is in use by another controller. In [model] directory, for each file there is a separate model class. Class name and file name represents the model (For example; table name).

1. **Switching between Languages ( English – Japanese )**

In BRANTECT, there is a button at header to allow switching of text between Japanese and English. When user clicks the button, the embedded form that holds all the POST value is sent after adding the language information. System saves the language information into session and re-displays it. The procedure of language settings is as follows;

1. Description for Master File of Language Settings

\_\_others/tools/Language\_Output\_File.xls lets both English and Japanese adding of the text key. With buttons [output of lang\_jp.inc] and [output of lang\_en.inc], it is possible to extract separated files for Japanese (lang\_jp.inc) and English (lang\_eng.inc) into same directory.

1. Copy of Language File

lang\_jp.inc and lang\_en.inc are copied under lib/lang/ .

1. How to Use Language Settings

The description key for the language file for display place of HTML is as follows;

<?php echo getString(‘KEY’);?>

1. **Corresponding to Invalid Argument**

As a countermeasure for SQL injection, it is necessary to check passed argument before executing SQL. As a way of executing SQL, usage of placeholder for *PreparedStatement* can be used.

However, related to inputting of non-numeric value into the place for inputting numeric value, when there is addition of numerical variable, it is necessary to add a variable name for checkIllegalData() function of r.php.

(Scheme)

**.** Describe the variable name that holds corresponding variable name of numeric value at lib/numeric\_variable.php.

**.** By using prepare of PEAR::MDB2, type of each placeholder is specified. Or you can use PDO and set value to placeholder at bindParam for specifying explicit type at the same time.

1. **Module Class**

At the Module Class, mainly database tables are created.

1. Properties

The Module Class holds the column of table related to properties. For example, you can use application that returns instance is used as a reference to return value of method, by setting the information of fetched row related to each property. (However, related to JOIN etc., since there may not be correspond, there is no necessity for return value type.)

1. Execution of SQL

As long as there is no special restriction, *PreparedStatement* is used.

1. Error

SQL runtime error check definitely occurs. When error occurs appropriate exception is thrown as well as rollback.

1. Transaction

If it is desired to use transaction to have effect across multiple tables (multiple model class methods), caller gets connected to database. However, the structure is that if connection fails, only method can connect to database.

When using transaction, handling of error is as described before in the whole SQL runtime error handling.