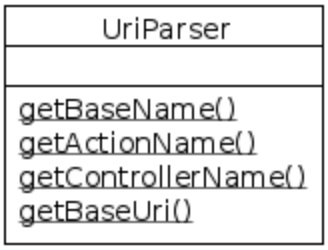
Ejercicios de arquitectura MVC

1. Realizar un “Hola mundo” con un patrón MVC, de manera que el mensaje a mostrar esté en una Base de Datos.
- Mejorarlo para que pueda mostrar una lista de saludos, guardados en la BD
2. Realizar un programa utilizando MVC que muestre un formulario en el que permita introducir un nombre (desde un input tipo text), y al pulsar enviar muestre una pantalla que diga “Hola <nombre>”

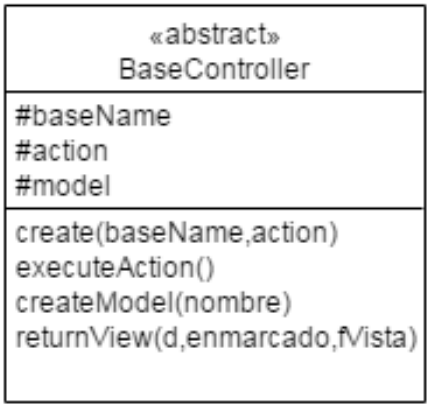
CONSTRUCCIÓN de un FRAMEWORK MVC (I)

3. Crear un carpeta “mvc”y bajo ella, tres directorios: **controller, model** y **view**
4. Crear un fichero **.htaccess** bajo “mvc” para que rescriba cualquier URI a partir de aquí a “index.php”
5. [**controller**] Crear una clase UriParser que ofrezca un cuatro métodos estáticos **getControllerName()**, **getActionName(), getBaseName(), getBaseUri()** que analicen la URI actual para determinar los elementos a cargar, a partir de una URI del estilo: http://server/app/BaseName/action



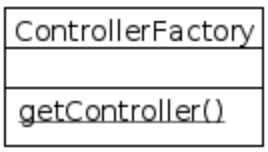
- **getBaseName()**: devolvería “BaseName” en el ejemplo anterior. Si ”BaseName/action” fueran la cadena vacía o la cadena “index.php”, devolvería la cadena “Home”
- **getActionName():** devolvería “action” en el ejemplo anterior. Si no se indicara nada, devolvería “index”
- **getControllerName()**: devolvería “BaseNameController” en el ejemplo anterior.
- **getBaseURI():** devolvería http://server/app/ en el ejemplo anterior.

6. **[controller]** Hacer una clase abstracta **BaseController** común para todos los controladores que creemos que tenga los siguientes elementos.



- **create(baseName,action)**: constructor-setter, que almacena el nombre base del controlador y la acción a realizar.
- **executeAction()** ejecuta la acción "action" invocando a un método de nombre "<action>Get()" o "<action>Post()" que deberán implementar los "hijos" de BaseController. La razón por la que se invocará uno u otro será que se hayan recibido o no datos vía POST
- **createModel(nombre)**: crea el modelo **model/<baseName>Model** por defecto, o bien el modelo **model/<nombre>Model** si se indica otro nombre. En cualquier caso, se almacena el modelo recién creado en el atributo "model"
- **returnView(d, enmarcado, fVista)**
  - **d** son los datos "empaquetados" convenientemente por el controlador, procedentes en su mayoría del modelo.
  - **enmarcado**, es un booleano y servirá para "enmarcar" la vista normal alrededor de una plantilla (cabecera, menú, footer, etc.) si es true (valor por defecto), y no "enmarcarlo" en caso contrario.
  - **fVista**, incluye el fichero cuya ruta es **fVista**, o en el caso de valer la cadena vacía, el fichero **view/BaseName/<action><Get | Post>.php** para desplegar la vista adecuada, dependiendo si los datos se han recibido vía GET ó POST (ej. view/Empleado/crearGet.php)

7. **[controller]** Hacer una clase **ControllerFactory**, que cuyo único método estático, utilice la clase **UriParser** para decidir qué controlador hay que crear, crearlo y devolver una referencia al mismo.

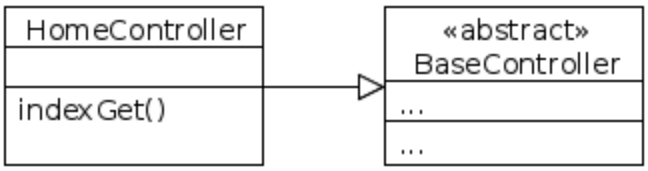


- **getController()** deberá comprobar (en este orden)
  - que exista un fichero llamado **<Base>Controller.class.php** en el directorio "controller"
  - que dentro de este fichero exista una clase llamada <Base>Controller

- que esta clase herede de BaseController
- que esta clase contenga un método llamado <action>Get o <action>Post, en función de si se han recibido datos vía Post, o no..

Funciones útiles a investigar para realizar el ejercicio: file_exists(...), class_exists(...), class_parents(...), in_array(...), method_exists(...)

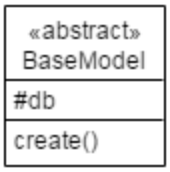
8. **[controller]** Crear una clase **HomeController** así.



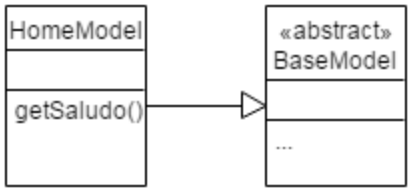
   - **indexGet()**, lo único que hace es invocar al método returnView(). En este ejemplo sencillo no necesitamos ninguna información proveniente del modelo.
9. **[view]** Crear una vista para la home page en un script llamado **view/Home/indexGet.php**, que Muestre un mensaje estático. Este script debería ser invocado automáticamente, si los pasos anteriores se han hecho adecuadamente (especialmente el método “returnView()” de BaseController)
10. Probar la vista anterior creando un controlador frontal en **index.php,** que haga siempre lo siguiente:

```
require_once('controller/ControllerFactory.class.php');
$controlador = ControllerFactory::getController();
$controlador -> executeAction();
```

11. **[controller, view]** Modificar **indexGet()** de HomeController, para crear datos ficticios procedentes de un supuesto modelo (por ejemplo, la cadena “DATOS DESDE EL MODELO”). “Pasárselos” a su vista para que ésta los muestre, modificando el HTML necesario de dicha vista.
12. **[model]** Crear una clase abstracta **BaseModel**, que servirá de plantilla para cualquier clase modelo que creemos. Tan sólo contendrá un constructor que me conecte con la base de datos con la que estemos trabajando y guarde la referencia a la misma en un atributo “db” protegido.



13. **[model]** Crear una clase **HomeModel** que herede de BaseModel e implemente un método getSaludo(), que obtendrá el saludo a mostrar de una Base de Datos **test**, que contenga una única tabla **saludos**, con una única columna **saludo**, y una única fila que contenga el saludo que más te guste.



CONSTRUCCIÓN de una APLICACIÓN MVC (I)

Para el siguiente esquema de tablas:
DEPARTAMENTOS (IDDEP, NOMBRE)
EMPLEADOS (IDEMP**,** NOMBRE; APELLIDO, FNAC, IDDEP)
FK = { IDDEP → DEPARTAMENTOS (IDDEP) }

14. Implementar, utilizando el framework MVC anterior, el caso de uso “Listar departamentos”
15. Implementar, utilizando el framework MVC anterior, el caso de uso “Crear Departamento”

CONSTRUCCIÓN de un FRAMEWORK MVC (II)

16. Crear un marco común para desplegar las vistas consistente en una cabecera (header), una barra de navegación con menús (nav), nuestro cuerpo central (section#central), y un footer. Crear una plantilla CSS asociada a estos elementos para que tengan una apariencia uniforme en toda la aplicación. A la hora de desplegar la vista:
    - El controlador deberá potencialmente "empaquetar" datos para cada uno de estos elementos.
    - El método return_view será responsable de desplegarlos en el orden adecuado (sólo si el "tipo" de despliegue es "enmarcado")
    - **[head]** Se desplegará para cualquier página un "head" común cuyo único dato variable podría ser el 'título' de la página.
    - **[header]** Se desplegará un header que contendrá el logo (o el título) de la aplicación, y una parte variable que informará del usuario actualmente autenticado, o bien el mensaje "No has hecho login")
    - **[nav]** La barra de navegación recibirá un array del estilo
      [['menu1','accion1'] , ['menu1','accion2'], ['menu3','', [['menu 3-1','accion 3-1'],['menu 3-2','accion 3-2']]]]
    - **[nav]** La barra de navegación sabrá dibujar los menús incluyéndolos en una jerarquía similar <ul> … ... , en el que cada elemento del menú <li> incluirá un vínculo <a href=""> asociado a la acción indicada. Si el menú es un "menú de menús" el href estará vacío. Los "menús de menús" también contendrán otro bloque <ul>.... en el que esté toda la información de los submenús. La barra de menús principal (troncal) deberá estar etiquetada con un id="navigation" para que CSS la pueda distinguir de las demás.
    - **[article]** La vista de cada acción la desplegaremos aquí.
    - **[footer]** En el footer habrá un dato variable 'mensaje' en donde desplegaremos nuestros mensajes de error.

CONSTRUCCIÓN de una APLICACIÓN MVC (II)

Para el siguiente esquema de tablas:
DEPARTAMENTOS (IDDEP, NOMBRE)
EMPLEADOS (IDEMP**,** NOMBRE, APELLIDO, FNAC, IDDEP)

17. Implementar, utilizando el framework MVC anterior, el caso de uso "Listar empleados". Para cada empleado se indicará el nombre de departamento a que pertenece.
18. Mejorar la vista anterior para incluir un filtro por departamentos con un "SELECT". Por defecto estará seleccionada la opción "todos". Si se selecciona un departamento concreto, sólo se mostrarán los empleados de dicho departamento.
19. Implementar el caso de uso "Nuevo empleado". En el formulario de entrada se mostrará el nombre del departamento al que debe pertenecer mediante un "SELECT". De esta manera impediremos que se introduzca un nombre de departamento erróneo.
20. En el caso de que no se hubiera hecho ya, mostrar los mensajes de confirmación de "crear departamento" y "crear empleado" como una respuesta AJAX que se visualiza en el "footer" de la página.