UNIVERSIDAD DE ALCALÁ

Departamento de Automática

Grado en Ingeniería Informática
Grado en Ingeniería de Computadores
Grado en Sistemas de Información

Práctica 2: Aplicaciones web de monitorización con patrón MVC: Desarrollo de un controlador PHP

Técnicas de diseño de sistemas de supervisión y entrenamiento remoto

Índice

1. Competencias asociadas a la práctica

1. Ser capaz de comprender la utilidad de los patrones de diseño en el desarrollo de aplicaciones web.
2. Ser capaz de aplicar el patrón de diseño MVC (Modelo-Vista-Controlador) en el diseño de una aplicación web.
3. Ser capaz de aplicar la sintaxis básica del lenguaje PHP para el desarrollo de aplicaciones web.
4. Ser capaz de comprender la comunicación MVC en una aplicación web específica.
5. Ser capaz de comprender la necesidad de instalar las herramientas y lenguajes necesarios para el desarrollo de una aplicación Web con patrón MVC.
6. Ser capaz de comprender la utilidad de seleccionar un lenguaje de intercambio de datos sencillo entre las diferentes partes de una aplicación Web con patrón MVC.
7. Ser capaz de instalar y configurar las herramientas necesarias para desarrollar una aplicación web con patrón MVC.
8. Ser capaz de desarrollar aplicaciones modularizadas.
9. Ser capaz de trabajar en equipo en el desarrollo de aplicaciones.

2. Introducción

Una vez que se ha aprendido en la práctica 1 a diseñar un ejemplo de aplicación web de supervisión, como es una aplicación de monitorización del aprendizaje del alumno, en la presente práctica, y como continuación, enfocaremos el interés en el desarrollo de aplicaciones Web con patrón Modelo-Vista-Controlador (MVC), frecuentemente utilizado en este tipo de aplicaciones.

Por lo expuesto anteriormente, uno de los objetivos claves de esta práctica es que el alumno sea capaz de implementar adecuadamente un Controlador de acuerdo a las peticiones realizadas por el usuario así como la comunicación con el Modelo de datos de la aplicación. La implementación de la Vista y su comunicación con el resto de la aplicación queda fuera del ámbito de esta asignatura por lo que se planteará como ejercicio optativo únicamente la realización representativa de alguna interfaz.

Otro objetivo importante de la práctica es que el alumno aprenda a usar lenguajes de programación de amplia difusión en el desarrollo de aplicaciones web tales como PHP y JSON (lenguaje de intercambio de datos).[1].

Un aspecto novedoso en la realización de esta práctica es el uso del Modelo basado en ontologías y su tecnología asociada. Dado que esta tecnología ofrece muchos

[1] Alternativamente, es muy habitual utilizar para el desarrollo de estas aplicaciones frameworks que permiten al programador centrarse en la lógica de la aplicación tales como, por ejemplo, los frameworks Zend y Spring.

beneficios que están ya siendo utilizados frecuentemente en el desarrollo actual de aplicaciones web (aplicaciones de la Web Semántica) y en especial en las aplicaciones web de supervisión y entrenamiento remoto, aquí se presenta un ejemplo de su uso real en el desarrollo de la aplicación de monitorización del aprendizaje del alumno ya diseñada por el alumno previamente.

Las ontologías, ventajas, características y tecnología asociada, se estudiarán en el tema siguiente de la asignatura, por lo que en esta práctica se proporcionará al alumno el Modelo desarrollado con esa tecnología para el desarrollo de la práctica. En cualquier caso, para mayor comprensión, se proporciona a continuación, como preámbulo, una de las definiciones de ontología existentes[2]:

Ontología Es una herramienta conceptual que define un vocabulario común para quien necesita compartir información dentro de un determinado dominio. Esto incluye definiciones de los conceptos básicos del dominio, así como sus relaciones, que tienen que ser interpretables por las máquinas. *(Noy & McGuinness, 2001)*

Asimismo, un sub-objetivo importante dentro de la práctica es acostumbrar al alumno a trabajar en equipo, una cualidad esencial de cara al mundo laboral.

3. Patrones de diseño

Los patrones de diseño se propusieron por Christopher Alexander en 1979. Inicialmente, los patrones se utilizaron para mejorar la calidad en la construcción de edificios reduciendo tiempos y costes. Según este autor, cada patrón describe un problema que ocurre muchas veces en nuestro entorno, y proporciona una solución al mismo. De este modo, se podrá utilizar esta solución muchas veces en el futuro sin necesidad de volver a pensar en ella.

Más adelante, los patrones de diseño se utilizaron en la búsqueda de soluciones a problemas comunes en el desarrollo de software y además, en el contexto relacionado con el diseño de interfaces. Se construyeron 23 patrones para el diseño de software definidos en 1995 por Gamma, E. y colegas en su libro *Design Patterns: elements of reusable object-oriented software*.

Un patrón de diseño debe tener las siguientes características:

- Proporcionar catálogos de elementos *reusables* en el diseño de sistemas software.
- Evitar repetir la búsqueda de soluciones a problemas ya conocidos y solucionados previamente.
- Estandarizar el modo en se que realiza el diseño.
- Formalizar un vocabulario común entre diseñadores.
- Facilitar el aprendizaje de nuevas generaciones de diseñadores aunando conocimiento ya existente.

[2]Para más información sobre las definiciones y características véase http://zarza.usal.es/~fgarcia/doctorado/iuce/WSemantica.pdf

Los patrones abstraen el comportamiento de un cierto problema. En función del nivel de abstracción, los patrones se dividen en:

- *Patrones de arquitectura*. Son patrones de alto nivel de abstracción y se utilizan para definir la estructura y organización de un sistema software.
- *Patrones de diseño*. Son patrones de nivel medio de abstracción y se utilizan para ayudar a definir mejor los componentes de un sistema software.
- *Patrones de interacción*. Se utilizan para el diseño de las interfaces de usuario. Se aplican frecuentemente en el diseño de las interfaces de aplicaciones web.

Existen más tipos de patrones, tal y como se describe en el tema 2. Aquí nos centramos exclusivamente en la fase de diseño software.

4. Patrones de arquitectura

El uso de patrones arquitectónicos en el desarrollo de software conlleva numerosas ventajas; los sistemas son más robustos, de mayor calidad, con mayor facilidad de mantenimiento y por ello con un desarrollo más rápido. Como contrapartida, el uso de patrones hace que el diseño del software sea más complejo. Por esta razón, requiere abstraer el comportamiento del conjunto de componentes del diseño para que pueda ser utilizado un componente independientemente del resto. En definitiva, en esto consiste la reusabilidad considerada cada vez más como un aspecto esencial para reducir tiempos de desarrollo y coste del software lo que conduce a una mayor calidad del mismo. Una vez que ya están todos los componentes, la clave está en integrarlos de forma coherente.

Existen una gran variedad de patrones arquitectónicos adecuados en distintos dominios: para sistemas distribuidos, para sistemas en capas, para sistemas basados en componentes, etc. El objetivo de todos ellos es garantizar la resolución de un determinado problema utilizando una configuración específica de los componentes para ese dominio.

4.1. Patrón Modelo-Vista-Controlador (MVC)

El patrón MVC es uno de los patrones de arquitectura que se utilizan más frecuentemente para facilitar el desarrollo de aplicaciones Web. Este patrón se utilizó para construir las interfaces de usuario en el lenguaje Smalltalk-80. La característica esencial de MVC es la separación de los componentes de la aplicación relacionados con los datos de los componentes de la interfaz de usuario. Desde el punto de vista del desarrollo, esta división permite obtener un código más claro, reusable y flexible.

El patrón MVC es un tipo de patrón multicapa. En el desarrollo de un patrón de esta clase, se tiende a que cada capa encapsule componentes que comparten unas determinadas características, es decir, para una determinada función.

En concreto, el patrón Modelo-Vista-Controlador descompone la aplicación en capas permitiendo tener una separación entre la lógica de negocio de la propia aplicación, la representación y la persistencia.

El patrón identifica las siguientes tres capas fundamentales para cualquier aplicación:

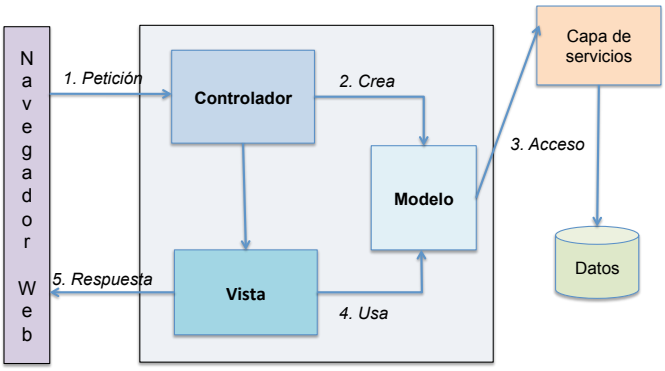

Figura 1: Arquitectura Modelo-Vista-Controlador

- *Modelo*. Capa que encapsula los datos de la aplicación y la lógica para interactuar con ellos.
- *Vista*. Capa que maneja la interacción con el usuario y la representación del modelo.
- *Controlador*. Capa intermediaria entre el Modelo y la Vista, una vez generadas las peticiones por el cliente en la Vista. El controlador se encarga de seleccionar el modelo solicitado por el usuario y la Vista adecuada para representarlo.

En la figura 1 se muestra el esquema de una arquitectura básica para una aplicación Web utilizando el patrón MVC. En la figura se puede observar cómo se aplica el patrón MVC para interactuar con otros componentes ya definidos: el navegador Web y Bases de Datos.

Al separar la presentación, los datos y la lógica de negocio se produce menor acoplamiento. Si se necesita modificar algo, se modifican sólo las partes involucradas en la modificación, y de forma transparente para las demás. Además, el uso de MVC proporciona la capacidad de crear interfaces personalizadas, es decir, poder representar la misma información de diversas maneras sin necesidad de modificar la fuente. Un modelo puede tener varias vistas que representen los datos en diferentes formatos: un diagrama de barras, un diagrama de sectores, una tabla, etc.

El alto nivel de abstracción del patrón MVC ha permitido desarrollar con gran éxito aplicaciones Web complejas. Para el desarrollo de aplicaciones Web no resulta muy evidente la aplicación de este patrón puesto que la interfaz por defecto de una aplicación Web es el navegador Web. En ocasiones, la tarea de diseño puede resultar muy compleja debido a que entre el cliente y servidor puede existir una gran cantidad de componentes. Suponiendo además que la aplicación posee una capa de persistencia de los datos muy compleja y que requiere de varios servidores en diferentes partes de la organización, muchas veces en estos casos conviene tener otro subsistema encargado sólo de la persistencia y que cada componente pueda tener varios componentes internos a su vez.

5. Lenguaje de desarrollo de aplicaciones web: PHP

PHP es un acrónimo que significa *PHP Hipertext Preprocessor* (inicialmente *PHP Tools* o, *Personal Home Page Tools*). El lenguaje PHP fue creado originalmente por Rasmus Lerdorf en 1995 y es un lenguaje interpretado con una sintaxis similar a la de C++ o JAVA.

PHP es un lenguaje de uso general de *scripts* del lado del servidor que se puede utilizar para realizar cualquier tipo de programa pero se ha hecho popular en la generación dinámica de páginas web. En concreto, suele incluirse en páginas HTML (o XHTML), y es el servidor web el encargado de ejecutarlo. Además, puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas.

A continuación se muestran algunas de sus principales características:

- Es un lenguaje que forma parte del software libre. Puede descargarse de `http://www.php.net`.
- Está disponible para muchos sistemas operativos: GNU/Linux, Windows, UNIX, etc.
- Tiene una extensa documentación oficial disponible libremente en varios idiomas (en `http://www.php.net`).
- Existen multitud de extensiones: para conexión con bases de datos, para manejo de sockets, para generar dinámicamente páginas en Flash, para generar documentos PDF, etc.
- Los programas PHP, al ejecutarse en el servidor, lo pueden usar todo tipo de máquinas con todo tipo de sistemas operativos.

La sintaxis básica de este lenguaje, necesaria para la realización de esta práctica, se irá aprendiendo por el alumno mediante la ejecución y análisis de ejemplos comentados[3].

6. Lenguaje de intercambio de datos en aplicaciones web: JSON

JSON (*J*ava*S*cript *O*bject *N*otation)[4] es un formato ligero de intercambio de datos propuesto por *Douglas Crockford*, un experto ingeniero software. Tiene forma de texto plano, de simple lectura y escritura y de fácil interpretación y generación por las máquinas. Está basado en un subconjunto del Lenguaje de Programación JavaScript, específicamente en literales de matrices y objetos.

JSON es un formato de texto completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidas por los programadores de la familia de lenguajes C (C, C++, C#, Java, JavaScript, Perl, Python, etc). Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

Como usa la sintaxis *JavaScript*, las definiciones JSON pueden incluirse dentro de archivos *JavaScript* y acceder a ellas sin ningún análisis adicional como los necesarios

[3] Una fuente interesante sobre la sintaxis de PHP es `http://www.php.net/manual/es/langref.php`

[4] `http://www.json.org/json-es.html`

con lenguajes basados en XML. En definitiva, el beneficio de JSON no es que sea ”más pequeño” a la hora de transmitir que XML, sino que representa mejor la estructura de los datos y requiere menos codificación y procesamiento.

JSON está constituido por dos estructuras:

1. **Objeto**: conjunto desordenado de pares nombre/valor. Un objeto comienza con una llave de apertura (símbolo {) y termina con una llave de cierre (símbolo }). El nombre y el valor están separados por el símbolo : (dos puntos) y los pares nombre/valor están separados por el símbolo , (coma).

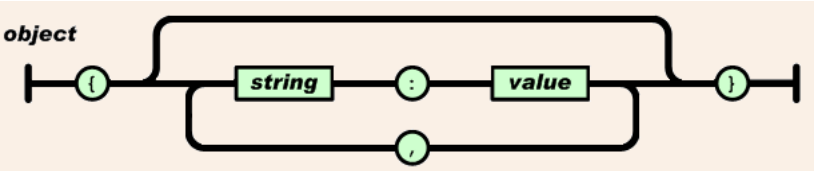

Figura 2: Objeto JSON

Ejemplo: `{’’nombre’’:’’Juan’’, ’’edad":23, ’’hijos’’:true}`

2. **Array**: colección ordenada de valores. Un arreglo comienza con un corchete izquieredo (`[`) y termina con un corchete derecho (`]`). Los valores se separan por , (coma).

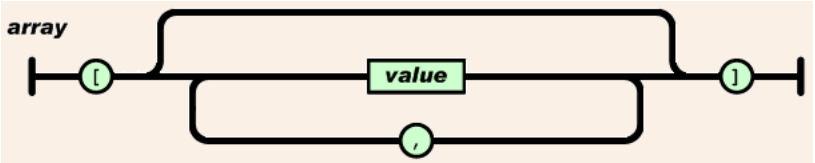

Figura 3: Array JSON

Ejemplo: `[’’Jose’’, ’’Juan’’, ’’Alfonso’’]`

Un valor puede ser uno de los siguientes elementos:

- Una cadena de caracteres entre comillas dobles.
- Un número.
- `true`, `false` o `null`.
- Un objeto.
- Un array.

Las estructuras previas pueden anidarse.

Una cadena de caracteres (*string*) es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles y donde los caracteres de escape utilizan la barra invertida (es parecida a una cadena de caracteres en C o Java). Un carácter está representado por una cadena de caracteres de un único carácter. Un número es similar a un

número C o Java, excepto que no se usan los formatos octales y hexadecimales. Puede representar `Integer`, `Real` o `scientific`. Los espacios en blanco pueden insertarse entre cualquier par de símbolos.

La sintaxis de JSON realmente es una combinación de literales de objeto y arrays para almacenar datos. JSON representa solamente datos; no incluye el concepto de variables, asignaciones o igualdades.

Un ejemplo más complejo:

```
[{''nombre'':''Juan'', ''edad'':30, ''hijos'':[''Jose'',"Juan'',''Alfonso'']},
{''nombre'':''Laura'', ''edad'':37, ''hijos'':[''Hijo1 Laura'',''Hijo2 Laura'']}]
```

6.1. Librerías JSON para PHP

Desde Javascript podemos procesar directamente cualquier objeto JSON y existen librerías para la mayoría de los lenguajes de programación que tienen funciones para interpretar este formato. Por ello se ha adoptado universalmente[5]. Por ejemplo, con JSON podemos comunicar datos fácilmente entre *scripts* Javascript y *scripts* PHP.

6.2. Cómo producir JSON desde PHP

Para crear una cadena para expresar un objeto u otro tipo de variable en JSON desde PHP, se dispone de una función llamada `json_encode()`, que recibe lo que deseamos convertir en notación JSON y devuelve una cadena de texto con el JSON producido. Se puede convertir en JSON cualquier objeto que necesitemos como: una cadena, una variable numérica, un array -normal o asociativo-, y objetos con todo tipo de datos dentro.

Ejemplo 1:

```
$mivariable = ''Hola'';
print_r(json_encode($mivariable));
```

Devuelve: `''Hola''`

Ejemplo 2: Conversión de un array sencillo

```
$miArray = array(1,2,3,4,5,7,8.1,9);
print_r(json_encode($miArray));
```

Devuelve: `[1,2,3,4,5,6,7,8.1,9]`

Ejemplo 3: Conversión de un array asociativo:

```
$miArray = array(''ficha1''=>''verde'', ''ficha2''=>''amarilla'', ''ficha3''=>''roja'',
''ficha4''=>''azul'');
print_r(json_encode($miArray));
```

Devuelve:

[5] http://www.json.org/

```
{''ficha1'':''verde'', ''ficha2'':''amarilla'', ''ficha3'':''roja'', ''ficha4'':''azul'');
```

Desde PHP podemos convertir cualquier objeto que tengamos y generar el JSON asociado para, por ejemplo, enviarlo a un componente *Javascript* o en cualquier otro lenguaje que necesitemos que lo procese. El mecanismo es exactamente igual, pero primero es necesario crear una clase, instanciar un objeto y luego convertirlo a JSON con `json_encode()`[6].

6.3. Cómo consumir JSON en PHP

PHP puede interpretar datos procedentes de una cadena con notación JSON y que se puedan guardar en variables para luego utilizarlas en los *scripts* del lado del servidor.

Para mostrar un ejemplo, vamos a crear una variable PHP con la cadena para hacer este objeto JSON y lo cargaremos en una variable PHP interpretando el JSON. Para ello, PHP dispone de una función llamada `json_decode()` que recibe la cadena con notación JSON y devuelve un objeto, o cualquier otro tipo de variable, que estuviera representada en el dato JSON.

Ejemplo:

```
<?php
$str_obj_json = '{
   ''elemento1'': ''valor1'',
   ''elemento2'': 12,
   ''elemento3'': null,
   ''masCosas'': {
      ''estoy'': ''aqui'',
      ''saludo'': ''hola'',
   }
}';
$obj_php = json_decode($str_obj_json);
?>
```

Se puede mostrar el contenido de esa variable con la función de PHP `print_r()`, que muestra el contenido de variables que puede entenderse por un humano.

```
stdClass Object
{
   [elemento1] => valor1
   [elemento2] => 12
   [elemento3] =>
   [masCosas] => stdClass Object
   {
      [estoy] => aqui
      [saludo] => hola
   }
}
```

[6] http://developea.me/1085/convertir-un-objeto-php-a-json.html

También podemos intentar acceder a una de las propiedades de este objeto, de esta manera:

```
echo ''Una propiedad cualquiera:'' . $obj_php->elemento1;
```

Recuerde que en la cadena JSON tanto el nombre de una propiedad como su valor deben estar entre comillas (y comillas dobles no comillas simples), menos los valores numéricos, o palabras como `null`, que pueden estar sin comillas. Véase más ejemplos interesantes en http://php.net/manual/es/function.json-decode.php.

7. Ejercicio

Como aplicación del patrón MVC, el alumno debe desarrollar un Controlador para la aplicación de la práctica 1 (''Monitorización del aprendizaje del alumno'').

El Controlador se realizará utilizando el lenguaje de programación PHP cuyo aprendizaje se realizará mediante ejemplos comentados que se proporcionarán al alumno para su ejecución y análisis. A través de la página de la asignatura se irán dando las guías para su correcto aprendizaje.

Para la realización de esta práctica se proporcionará al alumno un Modelo basado en tecnología de Ontologías (Ingeniería Ontológica). El Modelo consta de varias ontologías interrelacionadas junto con las consultas necesarias implementadas con SPARQL (lenguaje de consultas de ontologías) y mediante el uso del framework *Jena* (cuyo estudio y aplicación se realizará en la parte 3). Por lo tanto, el objetivo en esta práctica **no** es la implementación del Modelo sino del Controlador. El siguiente objetivo, ya en la práctica siguiente (práctica 3), será aprender a desarrollar una ontología y a utilizarla para su uso en una aplicación como la que está siendo desarrollada.

Para realizar este ejercicio, se recomienda seguir los siguientes pasos en orden:

1. Instalar *Apache* y *PHP*.
2. Configurar servidor *Apache*.
3. Instalar el *framework Jena*. Este *framework* se utiliza para manejar las ontologías que componen el Modelo de la aplicación. En definitiva, las ontologías han sido instanciadas para almacenar los datos asociados al modelo conceptual que el alumno realizó en la práctica 1. Asimismo, el modelo incluye las consultas realizadas a las ontologías para obtener los datos solicitados por alumnos/profesores durante la monitorización. Dado que las ontologías serán estudiadas en el tema 3, el alumno sólo debe ubicar las ontologías y el modelo implementado con *Jena* en los directorios que se indiquen en su momento.
4. Instalar la biblioteca *gson* (una libreria Java para convertir JSON a Objetos Java y viceversa). Esta biblioteca ha sido necesaria para invocar en el Modelo a las funciones de JSON y devolver al controlador el resultado de la monitorización (que serán los resultados de consultas en lenguaje *SPARQL* sobre las ontologías del Modelo). El Controlador recibirá así los resultados en formato JSON de acuerdo a las peticiones de un alumno/profesor.

5. Realizar el controlador en *PHP*. **Sólo** se llevará a cabo una de las funcionalidades del controlador para la aplicación de ”Monitorización del aprendizaje del alumno” siguiendo la estructura del pseudocódigo que se facilitará al alumno en su momento.
    - Invocar en el Controlador al Modelo mediante la orden que se facilite en su momento.
6. Compruebe que el Controlador funciona. Para ello, realice la siguientes pruebas:
    - Invocar al controlador desde su navegador.
    - Pedir a un compañero que invoque a su Controlador desde otra máquina utilizando la orden `curl`.
7. La implementación de la Vista de la aplicación queda fuera del alcance de esta asignatura. Se deja como *ejercicio optativo* al alumno crear una página en PHP en la que se muestren los resultados del Modelo enviados en formato JSON. *Sugerencia*: hay que darse cuenta que los datos se devuelven en formato JSON. Si se desea realizar una página *HTML* y utilizando lenguaje PHP, es necesario, inmediatamente después del retorno de los datos del Modelo al Controlador, convertir los resultados devueltos por el Modelo de JSON a PHP (¿Cómo se realizará?: ¿mediante la función descrita en la sección 6.2 o, por el contrario, con la función descrita en la sección 6.3?). Una vez obtenidos los datos en PHP, maneje la sintaxis de este lenguaje para devolver al usuario una página *HTML* donde se muestre la información devuelta, como el alumno crea más adecuado (inclusive, puede tenerse en cuenta la correspondiente Vista que el alumno creó en la fase de *diseño de la presentación* de esta aplicación en la práctica 1[7].

Todos los pasos se llevarán a cabo de forma guiada. Sobre ellos se irá proporcionando al alumno la información necesaria para su adecuada realización.

8. Duración de la práctica

La práctica se desarrollará en aproximadamente 8 horas (práctica y aprendizaje de la sintaxis básica de PHP). De 4 a 6 horas de trabajo en casa, y de 2 a 4 horas en el grupo pequeño correspondiente. Debe realizarse necesariamente en grupos de dos alumnos.

La entrega se realizará a través del enlace correspondiente que se habilitará en la página web de la asignatura.

[7] Otras posibilidades para implementar Vistas muy adecuadas para este tipo de aplicaciones en las que se muestran gráficas de diferentes tipos son frameworks tales como *Sencha* (`http://www.sencha.com/products/extjs`)