Servicios de Internet (curso 2021-2022)

Práctica 2. Convocatoria de junio (2 puntos)

1. Objetivo

El objetivo de esta práctica es profundizar en algunas de las tecnologías presentadas en las clases de teoría, y adquirir la habilidad para trabajar con algunas aplicaciones y las APIs que implementan. Para ello, en la práctica se desarrollará un servicio de consulta de información sobre películas de cine.

2. Formato de la información

Para almacenar la información sobre las películas se ha creado el lenguaje MML, una aplicación XML cuyas reglas se describen textualmente en el Apéndice A. La información sobre las películas estará almacenada en un conjunto desconocido de documentos MML. Cada documento MML describe las películas estrenadas en un determinado año, y no habrá dos documentos que correspondan a un mismo año.

Los documentos podrán estar ubicados en URLs de diversos servidores web en distintas máquinas. El único fichero conocido es el del documento inicial, que describe las películas de 2001, y que siempre será válido. Este fichero es:

http://alberto.gil.webs.uvigo.es/SINT/21-22/mml2001.xml

A través de ese documento se accederá a otros documentos MML (que describen las películas de otros años), mediante los URLs de los elementos ***MML***. Los nuevos ficheros que se encuentren podrán ser correctos o erróneos (o bien no son *well-formed,* o bien no son válidos de acuerdo a las reglas de MML). **Un fichero erróneo debe descartarse, como si no existiese. El servicio sólo trabajará con la información de los ficheros válidos.**

3. Tareas de la práctica

Las tareas a realizar en la práctica son:

1. Escribir el XML *schema* que mejor refleje las reglas del lenguaje MML. El fichero con el *schema* se llamará *mml.xsd* y en él **se evitarán las definiciones/declaraciones repetidas**. Las definiciones de nuevos tipos de datos serán globales, disponibles para todos los elementos.
2. Implementar un servicio web de consultas sobre la información de los ficheros MML. El servicio ofrecerá sólo la siguiente consulta (consulta número 2): **obtener la filmografía en un determinado idioma de un actor/actriz.**

La consulta se organizará a través de varias fases, en cada una de las cuales se seleccionan ciertos datos para refinar la consulta. En el documento "*Screens.pdf*" se presentan ejemplos de las pantallas esperadas en el servicio.

La fase solicitada al servidor en una petición se identificará mediante el parámetro *pphase*, que podrá ser:

- **01:** Se pide la pantalla inicial (documento *Screens*, sección 0.1 –S.0.1-), en la que se presentará un mensaje de bienvenida, se ofrecerá la opción de ver la lista de ficheros erróneos, y se ofrecerá un botón para iniciar la consulta. Será el valor por defecto si no hay parámetro *pphase*.
- **02:** Se pide la lista de ficheros erróneos (S.0.2).
- **21:** Se pide la lista de los idiomas en los cuales hay alguna película (S.2.1).
- **22:** Se pide la lista de actores/actrices que tienen películas en un idioma (recibido en el parámetro *plang*). Para cada uno se recibirá su nombre, ID, y contacto, de acuerdo a lo descrito en el Apéndice A (S.2.2).
- **23:** Se pide la filmografía en un idioma (parámetro *plang*) de un actor/actriz (parámetro *pid*). Para cada película se informará de su año, del título, la sinopsis, y los géneros, de acuerdo a lo descrito en el Apéndice A (S.2.3).

Las consultas al servicio podrán realizarse:

- A través del *Firefox* (modo *browser*). La respuesta será código HTML mostrando el resultado, con los elementos necesarios para que el usuario prosiga la consulta, gestionando lo seleccionado en cada fase como un parámetro de las siguientes.
- Directamente desde un programa, **que no forma parte de la práctica** (modo *auto*). Una solicitud en modo auto se identifica porque lleva un parámetro *auto* con valor '*true*' ("*auto=true"*); si no lleva el parámetro *auto*, o si su valor no es '*true*', se entenderá que la consulta es en modo *browser*. En el modo *auto*, lo único que deberá hacer el servicio es enviar como respuesta un fragmento XML con el resultado.

En el documento "*Screens.pdf*" pueden encontrarse los formatos de todas las solicitudes, y ejemplos de las respuestas en ambos modos, **incluyendo el orden de los resultados de cada fase, el mismo en ambos modos.**

4. Presentación en el caso de acceso desde un navegador

El interfaz de usuario en el modo *browser* consistirá en una secuencia de pantallas donde se presenten las distintas fases de forma sucesiva. Al seleccionar un resultado de la página actual se llamará automáticamente a la página de la fase siguiente. Cada página debe informar de las selecciones previas que han conducido a ella, y debe incluir sendos botones con enlaces hacia la página inicial y hacia la página anterior.

Cada página irá firmada al pie con el nombre del autor.

5. Entorno de desarrollo y ejecución del servicio

El servicio de esta práctica será proporcionado por un servidor Apache TOMCAT (**versión 10**[1]), que ejecutará **un único** *servlet* desarrollado por el alumno. El documento "*Writing Servlets.pdf*" de MOOVI presenta información sobre el desarrollo de *servlets* y un sencillo ejemplo.

Cada alumno deberá configurar y ejecutar un TOMCAT propio en su cuenta. La instalación y uso del TOMCAT se describe en el documento "*TOMCAT.pdf*". Ese TOMCAT será el empleado en el examen práctico.

En la configuración del TOMCAT se definirá un contexto llamado *sintX* (donde *X* es el número de la cuenta del alumno), que será el ámbito en el que se desarrollarán todos los servicios del alumno. Ese contexto estará enlazado con la carpeta "*public_html/webapps*" de la cuenta del alumno.

Todas las solicitudes al servicio deberán llevar un parámetro *'p'* cuyo valor será una contraseña de diez caracteres (letras y números, con al menos uno de cada). Esta contraseña deberá ser comunicada a los profesores en el fichero '`readme.txt`' que se menciona en la sección 8 de este documento.

El acceso inicial al servicio mediante el *Firefox* será accediendo al siguiente URL (suponiendo que el *Firefox* se está ejecutando en la misma máquina que el TOMCAT), y donde *port* es el resultado de sumar 7000 y *X*:

http://localhost:port/sintX/P2M

Ese acceso al servicio provocará la ejecución directa de un *servlet* llamado **SintXP2** (su fichero fuente será *SintXP2.java*), que será el encargado de crear y enviar las respuestas. El primer método de la clase que implementa ese *servlet* será el *init*, y el segundo el *doGet* (sólo para facilitar la corrección, y simplificar su búsqueda).

Cada selección del usuario provocará **siempre** el envío de datos al *servlet*, que identificará la consulta, buscará el resultado, construirá la respuesta, y la devolverá al solicitante.

Aunque el servicio será desarrollado y probado por cada alumno en su TOMCAT, la corrección se realizará con otro TOMCAT, gestionado por los profesores, y que tendrá configurado el contexto del alumno de la forma arriba indicada. Por ello, **es extremadamente importante** que se respeten los nombres mencionados.

6. Tecnologías a emplear por el servlet: DOM y JAXP

El *servlet* debe analizar los ficheros MML mediante un *parser* DOM de la librería JAXP de Java. **Sólo se puede crear un parser una única vez (por tanto, fuera de cualquier bucle)**. El resultado de la lectura de cada fichero será un árbol DOM en donde el *servlet* buscará la información necesaria para resolver la consulta. **Los ficheros se leerán una sola vez**, y el tiempo que dure este proceso debe ser razonable.

Al leer un fichero para generar su árbol DOM debe comprobarse si el documento contiene errores, implementando un gestor de errores (interfaz *DefaultHandler* de la librería JAXP). La especificación del XML *Schema* contra el cual se validarán los ficheros se realizará en el servlet, indicando el *schema* desarrollado por el alumno.

El documento "*Using JAXP.pdf*" presenta información sobre el uso de la librería JAXP y un sencillo ejemplo.

Para la realización de la práctica no se podrá emplear sin permiso ninguna librería o tecnología que no esté instalada en los ordenadores del laboratorio.

Un servlet nunca debe realizar la llamada 'System.exit()'.

[1] En la versión 10 del TOMCAT, los paquetes "javax.servlet.*" han cambiado su nombre, comenzando ahora por la palabra "jakarta" en lugar de "javax" (por ejemplo, ahora es "jakarta.servlet.http.HttpServlet"

7. Implementación del servicio

La práctica se realizará de forma estructurada, sangrando apropiadamente el código para facilitar su lectura. Las variables tendrán nombres significativos.

Cada alumno escribirá una serie de métodos (ubicados en un fichero llamado *DataModel.java*) que **deben recibir los parámetros indicados y simplemente devolver (ya ordenados) los datos de la respuesta a cada fase de la consulta**, independientemente de que ésta sea en modo *browser* o *auto*. Los métodos para la consulta 2 son:

- *ArrayList<String> getQ2Langs ()*
- *ArrayList<Cast> getQ2Cast (String lang)*
- *ArrayList<Movie> getQ2Movies (String lang, String pid)*

Los nombres y parámetros son los indicados, todos obligatorios, y no se permite ninguno adicional.

Los tipos de datos *Movie y Cast* serán clases definidas en la práctica para agrupar toda la información de cada película y actor/actriz, que se usará posteriormente para crear la respuesta según el modo solicitado**.**

La llamada a estos métodos aparecerá una única vez en todo el código, y sus resultados serán posteriormente empleados para generar la respuesta en modo *auto* o *browser*, según se haya solicitado.

El servicio debe funcionar independientemente de su ubicación, es decir, cambiando el directorio *webapps* a otro ordenador y ruta. Para ello, **todos los URLs deben ser relativos, no puede haber en los ficheros fuente ningún URL o nombre de fichero con ruta absoluta, ni mención explícita a las carpetas anteriores a *webapps*.**

Existirá un fichero *FrontEnd.java* que se encargará **exclusivamente** de proporcionar los métodos para generar cada respuesta (HTML o XML, un método por cada solicitud y tipo de respuesta). Esos métodos no sabrán nada del modelo de datos (recibirán la lista de resultados como un parámetro), y sus respuestas deben generarse de forma continua dentro de cada método, no se permite escribir un fragmento de la misma, y llamar a otro método para que realice parte de esa tarea.

Los ficheros de la práctica 2 serán exclusivos de la misma, no compartidos con ninguna otra práctica. Para ello, para la ejecución de la práctica, los ficheros que se lean estarán ubicados dentro del directorio *webapps/p2*, mientras que las clases a ejecutar estarán en el directorio *webapps/WEB-INF/classes/p2* (formarán el paquete Java *p2*).

Se recomienda la realización de la práctica en dos fases:

1. En una primera fase, se implementará la estructura de pantallas que requiere la práctica para el modo *browser*, y las respuestas en modo *auto*, aunque ofreciendo datos ficticios en ambos casos (para ello, los métodos arriba indicados se limitarán a inventarse los datos que devuelvan, que podrán ser siempre los mismos).
2. En la segunda fase se completará la parte inicial con el *schema*, el análisis de los ficheros MML y la integración de los datos reales en la estructura de consultas de la primera fase. En esta segunda fase, se modificarán los citados métodos para que no se inventen los datos, sino que los obtengan de los ficheros MML.

8. Entrega de la práctica

Para notificar la conclusión de la práctica, es obligatorio entregar la práctica (**entrega no preliminar**) en la correspondiente tarea de MOOVI. Lo único a subir será un fichero `"readme.txt"` que incluya la información relevante para corregir la práctica (al menos, el identificador `sintX` del usuario y la contraseña del servicio).

Tras la entrega de la práctica se podrán corregir todos los errores identificados, momento en el que se obtendrán los puntos de la práctica. Sin embargo, tal como se describe en la guía docente de la asignatura, los incumplimientos reiterados de la especificación podrán dar lugar a una penalización en la nota final de la asignatura.

Una copia de todos los ficheros fuente (*java*, *xsd, Makefile…*) debe quedar en el directorio *public_html/p2* de la cuenta del alumno, junto con el fichero `"readme.txt"`. En ese directorio debe estar todo el código desarrollado (sin subdirectorios), y los ficheros deben estar preferiblemente limpios, sin que haya en su interior código anterior comentado que ya no se use. Este será el código que se revisará por parte de los profesores, para corregir o para ayudar en la solución de algún problema, por lo que es **muy importante** que siempre esté actualizado.

La situación de los ficheros fuente y del contexto del *TOMCAT* que incluye los *servlets* debe ser la indicada en este documento. También los nombres de los ficheros deben ser los especificados, incluidas mayúsculas y minúsculas.

Para la corrección de la práctica, los ficheros y carpetas de la cuenta deben tener los siguientes permisos:

- La raíz de la cuenta tendrá permisos 750.
- La carpeta *public_html* y sus descendientes tendrán permisos 755.
- Los ficheros descendientes de *public_html* tendrán permisos 644.

Apéndice A: Reglas del lenguaje MML

- El elemento raíz, que se llamará ***Movies***, almacenará: el año al que pertenecen las películas (en un elemento ***Year***); y la información sobre de una o más películas (la información de cada película se almacenará en un elemento ***Movie***).
- El contenido del elemento ***Year*** será un número de cuatro dígitos (entre 1900 y 2021).
- El elemento ***Movie*** almacenará: el título de la película (en un elemento ***Title*** que contendrá texto); opcionalmente, su duración en minutos (en un elemento ***Duration***); los posibles géneros en los que se encuadra (cada género en un elemento ***Genre***); y la información de cero o más miembros del reparto (cada miembro descrito en un elemento ***Cast***). En medio de esos elementos, podrá aparecer un resumen o sinopsis de la película, un fragmento de texto sin formato predefinido.
- El elemento ***Movie*** `podrá` llevar un atributo llamado *langs* que almacenará la lista de idiomas en los que está disponible la banda sonora de la película. De existir, será un será una cadena de caracteres compuesta por códigos de dos letras separados por espacios.
- El contenido del elemento ***Duration*** será un número entero, entre 60 y 300.
- El contenido del elemento ***Genre*** podrá ser: *Comedy*, *Drama* o *Action*.
- El contenido del elemento ***Cast*** será: el nombre del participante (en un elemento ***Name*** que contendrá texto); su papel en la película (en un elemento ***Role***); su contacto, o bien su teléfono en un elemento ***Phone***, o su correo en un elemento ***Email***; y cero o más elementos ***MML***, que contiene el URL de otro fichero MML donde hay una película donde aparece esa persona.
- El elemento ***Cast*** `deberá` llevar un atributo llamado *id* que asociará cada persona con un identificador, cuyo formato será de tres letras seguidas de tres números.
- El contenido del elemento ***Role*** puede ser *Main*, *Supporting* o *Extra*.
- El contenido del elemento ***Phone*** será un número de nueve cifras.
- El contenido del elemento ***Email*** será el de una dirección de correo electrónico.

Aclaraciones

- Si no se especifica lo contrario, el orden de los elementos debe ser el indicado en los puntos anteriores.
- No habrá dos películas con el mismo título en el mismo año.
- Los miembros del reparto se identifican por su *id*. Puede haber dos personas con el mismo nombre, pero no el mismo *id*.
- El elemento ***MML*** sirve únicamente para encontrar más documentos.
- El URL contenido en un elemento ***MML*** puede ser absoluto (comienza por 'http://') o relativo. Si es relativo, su URL absoluto se construirá añadiéndole como prefijo el mismo del documento en el que aparece ese elemento ***MML***.
- Los únicos errores que van a contener los documentos MML empleados para las pruebas son aquellos detectables por el *parser* de acuerdo a lo descrito en el *schema*. Si un error no es detectable por el *parser*, ese error nunca va a ocurrir en los documentos de pruebas.