Universidad de Sevilla

Ingeniería Informática. Ingeniería del Software

DO5

R3 – Architecture of a WIS

Fecha	Versión
21/11/2022	0.3

Enlace al repositorio: *https://github.com/dansuaper/Acme-Courses*

Grupo de prácticas	Diciembre #1
Autores	***Roles***
Raúl Montalbán Martín *raumonmar1@alum.us.es*	*Developer, Tester and Operator*
Daniel Suárez Perea *dansuaper@alum.us.es*	*Manager, Developer and Tester*

Tabla de contenido

1. Resumen ejecutivo

El objetivo de este documento es presentar y detallar los conocimientos obtenidos sobre la arquitectura WIS.

2. Tabla de versiones

Versión	Fecha	Descripción
0.1	21/11/2022	Creación del documento
0.2	21/11/2022	Edición del documento
0.3	21/11/2022	Edición final del documento

3. Introducción

En este documento vamos a detallar los conocimientos que hemos obtenido durante el transcurso de la asignatura en las clases teóricas y prácticas sobre la arquitectura WIS.

4. Contenido

En primer lugar, vamos a explicar el modelo de datos aprendido. Un modelo de datos permite representar requisitos de información fácilmente entendibles y manipulables para los desarrolladores.

Las entidades son objetos que tienen que tener la etiqueta @Entity guardados en la base de datos con sus atributos incluyendo sus tipos (se indicarán si son derivados o no). A su vez implementa la clase "*AbstractEntity*" ofreciendo una base proporcionando id, número de versión, un atributo *transient* con sus métodos equals(), hashcode() y toString().

Los atributos de las entidades se les pueden poner restricciones que no permiten introducir datos erróneos en la base de datos. Hay diferentes tipos de restricciones, en objetos, en números, en texto y en momentos. Para probar la eficiencia de los validadores hay que crear datos de ejemplo.

En cuanto a las relaciones entre entidades permite conseguir navegabilidad pudiendo obtener atributos de otras entidades. Hay dos tipos de navegabilidad, *single-way navigation* (unidireccional) y *two-way relations* (bidireccional). La multiplicidad de estas puede variar siendo *OneToOne, ManyToOne, OneToMany, ManyToMany*, conglomerados y agregaciones.

Las entidades son clasificadas jerárquicamente con la taxonomía. A su vez estas también deben ordenarse topológicamente de menos a más dependencias.

Con *JPQL* hemos aprendido a realizar consultas sobre atributos de las entidades. Según la relación entre entidades las consultas de sus datos se harán de distintas maneras. También se pueden hacer *subqueries* para probar restricciones.

En segundo lugar, vamos a explicar la arquitectura *WIS* aprendida. Está compuesta de la capa de navegador donde se hacen peticiones *HTTP* y renderiza los datos devueltos, la capa de aplicación recibe peticiones *HTTP* y las pasa mediante los controladores y la capa de bases de datos que recibe peticiones *SQL* y devuelve los datos.

Las peticiones *HTTP* antes mencionadas deben realizarse a través de las *URLs* donde usaremos localhost con el puerto 8080 con el nombre del proyecto y el camino serán las peticiones con la ruta del controlador.

Una cookie es un pequeño archivo que es creado por un servidor cuando trata una solicitud. Las cookies son enviadas por el servidor en la primera petición y devueltas por el navegador en cada petición posterior hasta que expiran o se borran. Las 3 aprendidas son Language, *JSESSION* y *Remember*.

Los controladores reciben peticiones sobre una función concreta. Estos extienden de "*AbstractController*" donde hay que indicar el rol del usuario y el tipo de objeto al que se le hace y este proporciona métodos que permite realizar las peticiones.

Los servicios proporcionan las funcionalidades que se realizan sobre una entidad. Para cada funcionalidad debe realizarse un servicio distinto que extienda de los distintos "*AbstractService*" que ofrece el framework pasándole el rol del usuario y el tipo de objeto. Cada tipo de "*AbstractService*" implementa distintos métodos que son necesarios para cada tipo funcionalidad.

Los repositorios hacen las peticiones a las bases de datos para la obtención de los requisitos de información.

Las vistas proporcionan interfaces de usuario para características de la aplicación. Se hacen en formato JSP siendo un conjunto de HTML+CSS+JS.

5. Conclusión

En conclusión, durante la asignatura hemos adquirido conocimientos sobre cómo implementar entidades, repositorios, servicios y controladores en una arquitectura WIS. A pesar de no partir con un gran nivel en lo que respecta a este tipo de arquitecturas, la curva de aprendizaje no nos ha resultado muy pronunciada; lo que nos ha permitido progresar adecuadamente en el proyecto a lo largo de todo el cuatrimestre.

Tras finalizar el proyecto, gracias a las técnicas utilizadas, hemos ganado conocimiento y experiencia valiosos que resultan indispensables de cara a nuestro inminente futuro profesional.

6. Bibliografía

Intencionalmente en blanco.