Taller Web 1 Evaluación de Aprendizaje Nro 3

Alumno: Marcos Cabral.

ENVIO	Jueves 18/6/2020	Fecha de Envío de EV1 a alumnos
ENTREGA	Jueves 25/6/2020	Fecha de Entrega
DEVOLUCION RDOS	Jueves 2/7/2020	Fecha Máxima de Devolución de Resultados
REENTREGA	Jueves 9/7/2020	Fecha de Reentrega

Pautas de resolución

1. Entregar a través de la plataforma MIEL
2. El formato de la entrega debe ser PDF obligatoriamente
3. Agregar el nombre del alumno en el encabezado del documento
4. No incluir capturas de pantallas
5. El texto correspondiente a código fuente debe estar en tipografía `Curier` o `Consolas`

Resolver

1-Pegar código del producto que se está desarrollando en equipo donde se utilice mocks, puede ser de un Controller o de un Servicio, explicar las partes más importantes.

2- Por qué no utilizaría mocks cuando se hace una prueba de un repositorio?

1.@Test

```
	public void testMostrarRestriccionesDeUsuario() {

		Usuario user=mock(Usuario.class);

		HttpServletRequest request=mock(HttpServletRequest.class);

		HttpSession sesion=mock(HttpSession.class);


	    ServicioRestriccion
			servicioRestriccion=mock(ServicioRestriccion.class);

		ServicioUsuario servicioUsuario=mock(ServicioUsuario.class);

		ServicioComida servicioComida=mock(ServicioComida.class);


		ControladorRestriccion controlador=new
	ControladorRestriccion(servicioRestriccion,servicioUsuario,servicioComida
	);

		List<Restriccion> restricciones=mock(List.class);


		when(request.getSession()).thenReturn(sesion);


		when(request.getSession().getAttribute("usuario")).thenReturn(user)
	;


		when(servicioRestriccion.listarRestriccionesDeUsuario(user)).thenRe
	turn(restricciones);


		ModelAndView model=controlador.x(request);



		assertThat(model.getViewName()).isEqualTo("usuarioConRestricciones"
	);

	}
```

Este test es en base del controlador Restricciones.

Para que funcione necesitamos de todos los objetos y clases que participen en el controlador, servicio y método que vayamos a testear (resaltados en **negrita**).

Para que funcione el test utilizamos mock(.class). Esto nos permite tener un mock de una clase, que permite tener una "copia" del tipo de objeto. O más bien tener un objeto que se comporta de la misma manera que otro.

Una vez manejado todos estos objetos, vamos a ordenar que nos devuelva un dato preciso al momento en que se ordene una acción concreta.

Esto se ve en las sentencias when(amarillo) y thenReturn(verde). Se traduce como, "cuando se invoque el metodo request.getSession(), entonces retórname la sesión". Esta sesión es una sesión de mock.

Finalmente llamamos al método del controlador, y guardamos el ModelAndView que nos retorna en una variable del mismo tipo.

Y para probar nuestro test nos hace falta el método assertThat, donde vamos a comparar un objeto con otro mediante el método isEqualTo.

2. No utilizaría mock para la prueba de repositorios ya que estarías reemplazando lo que se desea testear por el mock. Es decir, por ejemplo con Junit podemos testear un dato de salida de un método en concreto porque le asignamos datos reales, pero en mock los objetos que se pasan son copias, y no poseen dichos datos. Entonces en resumen, estarías haciendo un test falso, cualquier mock en el repositorio no sería testeable.