DP2 2024

Acme Software Factory

Repositorio: https://github.com/DP2-2024-C1-029/Acme-Software-Factory.git

Miembro:

- Jaime Varas Cáceres (jaivarcac@alum.us.es)

Tutor: José González Enríquez

GRUPO C1.029

25/05/2024

Versión 1.0

Índice

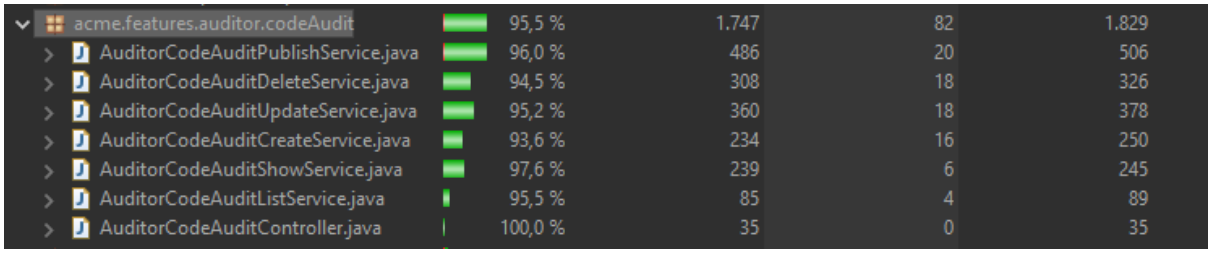
Historial de versiones

Fecha	Versión	Descripción	Entrega
27/05/2024	V1.0	Inicio del documento	D04

Capítulo 1 – Pruebas funcionales

Code Audit

Tras ejecutar todos los test, se puede observar que para las codeAudit se cubre el 95.5%, valor que está por encima de la recomendación mínima del 90% que debería cubrir al menos todos los test.



acme.features.auditor.codeAudit	95,5 %	1.747	82	1.829
AuditorCodeAuditPublishService.java	96,0 %	486	20	506
AuditorCodeAuditDeleteService.java	94,5 %	308	18	326
AuditorCodeAuditUpdateService.java	95,2 %	360	18	378
AuditorCodeAuditCreateService.java	93,6 %	234	16	250
AuditorCodeAuditShowService.java	97,6 %	239	6	245
AuditorCodeAuditListService.java	95,5 %	85	4	89
AuditorCodeAuditController.java	100,0 %	35	0	35

En primer lugar, para no repetirlo durante todo el documento, se va a comentar que las líneas que los “assert” siempre aparecen en amarillo, y que el “status” tampoco se puede poner en verde, porque hay un caso que nunca se puede probar.

```
        codeAuditId = super.getRequest().getData("id", int.class);
        codeAudit = this.repository.findOneCodeAuditById(codeAuditId);
        auditor = codeAudit == null ? null : codeAudit.getAuditor();
        status = codeAudit != null && codeAudit.isDraftMode() && super.getRequest().getPrincipal().hasRole(auditor);

        super.getResponse().setAuthorised(status);
    }
```

Voy a empezar hablando por del Publish, en el que tenemos un condicional que no ha podido ser probado en todos los casos, esto aparece en todos los servicios en los que se calculan la moda de la notas:

```
    for (Map.Entry<Mark, Integer> entry : modeMap.entrySet())
        if (entry.getValue() == maxFrequency) { // si empata a la frecuencia máx
            if (mode != null && mode.ordinal() < entry.getKey().ordinal()) // nos quedamos con la de menor nota
                mode = entry.getKey();
        } else if (entry.getValue() > maxFrequency) { // si la frecuencia es mayor nos quedamos con esa
```

Para el update service podemos observar que todo está en verde, menos una rama extra del cálculo de la moda:

```
    @Override
    public void validate(final CodeAudit object) {
        assert object != null;

        if (!super.getBuffer().getErrors().hasErrors("code")) {
            CodeAudit existing;

            existing = this.repository.findOneCodeAuditByCode(object.getCode());
            super.state(existing == null || existing.equals(object), "code", "auditor.codeAudit.form.error.duplicated");
        }

        if (!super.getBuffer().getErrors().hasErrors("executionDate")) {
            Date minimunMoment = MomentHelper.parse("2000/01/01 00:00", "yyyy/MM/dd HH:mm");

            super.state(MomentHelper.isPresentOrPast(object.getExecutionDate()), "executionDate", "auditor.codeAudit.form.error.too-close");
            super.state(MomentHelper.isAfterOrEqual(object.getExecutionDate(), minimunMoment), "executionDate", "auditor.codeAudit.form.erro
        }

        if (!super.getBuffer().getErrors().hasErrors("project"))
            super.state(!object.getProject().isDraftMode(), "project", "auditor.codeAudit.form.error.drafted-project");
    }
```

```
        types = SelectChoices.from(AuditType.class, object.getType());

        projects = this.repository.findManyPublishedProjects();
        choices = SelectChoices.from(projects, "title", object.getProject());

        // CÁLCULO DE LA MARK MEDIANTE LA MODA

        Collection<AuditRecord> auditRecords;
        auditRecords = this.repository.findAllAuditRecordsByCodeAuditId(object.getId());

        EnumMap<Mark, Integer> modeMap = new EnumMap<>(Mark.class);

        for (AuditRecord record : auditRecords) {
            Mark mode = record.getMark();
            modeMap.put(mode, modeMap.getOrDefault(mode, 0) + 1);
        }

        Mark mode = null;
        int maxFrequency = 0;
        for (Map.Entry<Mark, Integer> entry : modeMap.entrySet())
            if (entry.getValue() == maxFrequency) { // si empata a la frecuencia máx
                if (mode != null && mode.ordinal() < entry.getKey().ordinal()) // nos quedamos con la de menor nota
                    mode = entry.getKey();
            } else if (entry.getValue() > maxFrequency) { // si la frecuencia es mayor nos quedamos con esa
                maxFrequency = entry.getValue();
                mode = entry.getKey();
            }
```

En el ListService tenemos una línea en amarillo que corresponde a la internacionalización del valor del modo borrador:

```
    Dataset dataset;

    dataset = super.unbind(object, "code", "executionDate", "type");
    if (super.getRequest().getLocale().getLanguage().equals("es"))
        dataset.put("published", object.isDraftMode() ? "No" : "Si");
    else if (super.getRequest().getLocale().getLanguage().equals("en"))
        dataset.put("published", object.isDraftMode() ? "No" : "Yes");

    super.getResponse().addData(dataset);
}
```

En el controlador, todo está verde:

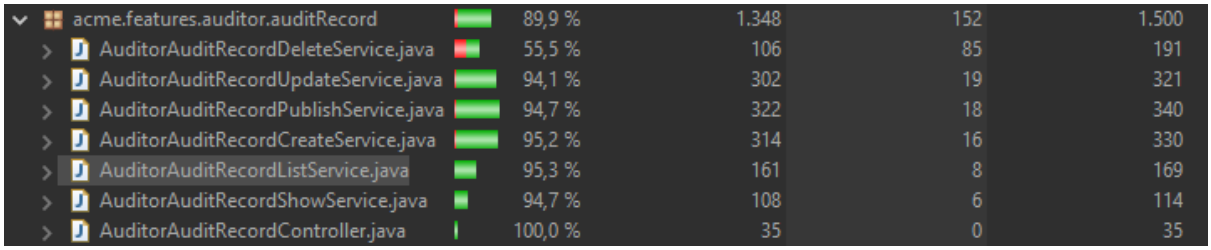
```
@PostConstruct
protected void initialise() {
    super.addBasicCommand("list", this.listService);
    super.addBasicCommand("show", this.showService);
    super.addBasicCommand("create", this.createService);
    super.addBasicCommand("update", this.updateService);
    super.addBasicCommand("delete", this.deleteService);

    super.addCustomCommand("publish", "update", this.publishService);
}
```

El resto de los servicios que no han sido nombrados es porque están cubiertas o las faltas de coberturas ya han sido mencionadas en otros servicios.

Audit Record

En esta ocasión, no superamos el umbral del 90% por un 0.1%, pero se a unas líneas en el deleteService correspondientes al unbind que no se usan nunca y que se han dejado por seguir el estilo de AcmeJobs.



acme.features.auditor.auditRecord	89,9 %	1.348	152	1.500
AuditorAuditRecordDeleteService.java	55,5 %	106	85	191
AuditorAuditRecordUpdateService.java	94,1 %	302	19	321
AuditorAuditRecordPublishService.java	94,7 %	322	18	340
AuditorAuditRecordCreateService.java	95,2 %	314	16	330
AuditorAuditRecordListService.java	95,3 %	161	8	169
AuditorAuditRecordShowService.java	94,7 %	108	6	114
AuditorAuditRecordController.java	100,0 %	35	0	35

A continuación, el unbind del delete:

```
@Override
public void unbind(final AuditRecord object) {
    assert object != null;

    SelectChoices marks;
    Dataset dataset;

    marks = SelectChoices.from(Mark.class, object.getMark());

    dataset = super.unbind(object, "code", "startPeriod", "endPeriod", "link", "draftMode");
    dataset.put("mark", marks.getSelected().getKey());
    dataset.put("marks", marks);

    dataset.put("masterId", object.getCodeAudit().getId());
    dataset.put("draftMode", object.getCodeAudit().isDraftMode());

    super.getResponse().addData(dataset);
}
```

El update, a excepción de las dos líneas del authorise que están en amarillo siempre, esta todo en verde:

```
    super.bind(object, "code", "startPeriod", "endPeriod", "mark", "link");
}

@Override
public void validate(final AuditRecord object) {
    assert object != null;

    if (!super.getBuffer().getErrors().hasErrors("code")) {
        AuditRecord existing;

        existing = this.repository.findOneAuditRecordByCode(object.getCode());
        super.state(existing == null || existing.equals(object), "code", "auditor.auditRecord.form.error.duplicated");
    }

    if (!super.getBuffer().getErrors().hasErrors("startPeriod")) {
        Date minimunMoment = MomentHelper.parse("2000/01/01 00:00", "yyyy/MM/dd HH:mm");

        super.state(MomentHelper.isPresentOrPast(object.getStartPeriod()), "startPeriod", "auditor.auditRecord.form.error.not-past");
        super.state(MomentHelper.isAfterOrEqual(object.getStartPeriod(), minimunMoment), "startPeriod", "auditor.auditRecord.form.error.too-early");
        super.state(MomentHelper.isAfterOrEqual(object.getStartPeriod(), object.getCodeAudit().getExecutionDate()), "startPeriod", "auditor.auditRecord.form.error.before-audit");
    }

    if (!super.getBuffer().getErrors().hasErrors("endPeriod")) {
        Date minimunMoment = MomentHelper.parse("2000/01/01 00:00", "yyyy/MM/dd HH:mm");

        super.state(MomentHelper.isPresentOrPast(object.getEndPeriod()), "endPeriod", "auditor.auditRecord.form.error.not-past");
        super.state(MomentHelper.isAfterOrEqual(object.getEndPeriod(), minimunMoment), "endPeriod", "auditor.auditRecord.form.error.too-early");
    }

    if (!super.getBuffer().getErrors().hasErrors("startPeriod") && !super.getBuffer().getErrors().hasErrors("endPeriod"))
        super.state(MomentHelper.isAfter(object.getEndPeriod(), object.getStartPeriod()), "endPeriod", "auditor.auditRecord.form.error.end-after-start");

    if (!super.getBuffer().getErrors().hasErrors("startPeriod") && !super.getBuffer().getErrors().hasErrors("endPeriod"))
        super.state(MomentHelper.isLongEnough(object.getStartPeriod(), object.getEndPeriod(), 1, ChronoUnit.HOURS), "endPeriod", "auditor.auditRecord.form.error.too-short");
}
```

El list, tiene un par de líneas de las que no se han podido cubrir todas las ramas, una es el idioma y la otra es una comprobación para mostrar un botón:

```
        Dataset dataset;

        dataset = super.unbind(object, "code", "mark", "startPeriod", "endPeriod");
        if (super.getRequest().getLocale().getLanguage().equals("es"))
            dataset.put("published", object.isDraftMode() ? "No" : "Si");
        else if (super.getRequest().getLocale().getLanguage().equals("en"))
            dataset.put("published", object.isDraftMode() ? "No" : "Yes");

        super.getResponse().addData(dataset);
    }

    @Override
    public void unbind(final Collection<AuditRecord> objects) {
        assert objects != null;

        int masterId;
        CodeAudit codeAudit;
        final boolean showCreate;

        masterId = super.getRequest().getData("masterId", int.class);
        codeAudit = this.repository.findOneCodeAuditById(masterId);
        showCreate = codeAudit.isDraftMode() && super.getRequest().getPrincipal().hasRole(codeAudit.getAuditor());

        super.getResponse().addGlobal("masterId", masterId);
        super.getResponse().addGlobal("showCreate", showCreate);
    }
```

Por último, el controlador y el resto de servicios no tienen nada que destacar ya que están cubiertos en su gran mayoría.

```
    @PostConstruct
    protected void initialise() {
        super.addBasicCommand("list", this.listService);
        super.addBasicCommand("show", this.showService);
        super.addBasicCommand("create", this.createService);
        super.addBasicCommand("update", this.updateService);
        super.addBasicCommand("delete", this.deleteService);

        super.addCustomCommand("publish", "update", this.publishService);
    }
```

Como conclusión se puede sacar que todo el “CodeAudit” ha sido probado de manera muy exhaustiva, probando todas las validaciones posibles. Al final de este capítulo se muestran capturas de un bloc de notas usado para la realización de los tests.

```
pasar un proyecto no publicado
usar el show (calcular moda con auditrecords)
usar el delete
usar el update
usar el publish

show de auditores ya creados
probar los show con usuarios no validos y roles no validos


SAFE

login con auditor1
listar 3 primeros, 2 intermedios, 2 finales
probar publish de algunos vacios, otros sin nota suficiente, otros con nota pero insuficientes
probar delete
probar create



SAFE RECORD

login auditor1
list code audits
chf001
update sin project a ESE codeAudit con algunos records dentro
update con project a ESE codeAudit con algunos records dentro
ver sus records
publicar el q tiene
borrar el chf001 después de publicar (salta la validación) -delete a un codeAudit con algunos records publicados dentro

crear auditRecord c/code usado:
 -form vacio
update auditRecord
delete
crear auditRecord
show auditRecord
update auditRecord campo a campo
publish

publish chf001

HACK CODEAUDIT

asociar con proyecto no publicado (create, update, publish)
show audit nulo y entrar con un user de otro rol

create: proyect no publicado

update, delete, publish de otros usuarios y de codeAudits ya publicados y con proyectos no públicos (+ probar cambio idioma) 61=publicado 64=no publicado ambos de auditor2


HACK AUDITRECORD

empatar frecuencias al calcular la moda (ejecutar en el show, update, publish y delete) creando nuevos auditRecords

list: idioma y listar de un codeAudit q no es tuyo
create: hijos de otros padres, draftmode false,
delete: records de otros, records publicados: 94 es de auditor1 no publicado
update: records de otros, records publicados: 94 es de auditor1 no publicado
publish: create en codeAudits tuyos publicados, records de otros, records publicados: 94 es de auditor1 no publicado

SAFE LIST CODEAUDIT
list: auditor2: tamaño pagina, paginas, cambiar idioma
show: auditor2: publicado, no publicado, sin hijos
create:
        -ref: espacio, patron incorrecto, muchos caracteres, caracteres raros: Переводчик
ПереводчикПереводчикПереводчикПереводчикПереводчикПереводчикПереводчикПереводчикПереводчикПереводчи
        -
delete: sin hijos, con un hijo, con hijos con empate de notas, con algun hijo publicado
publish: sin hijos, con hijos con empate de notas, con algun hijo publicado
```

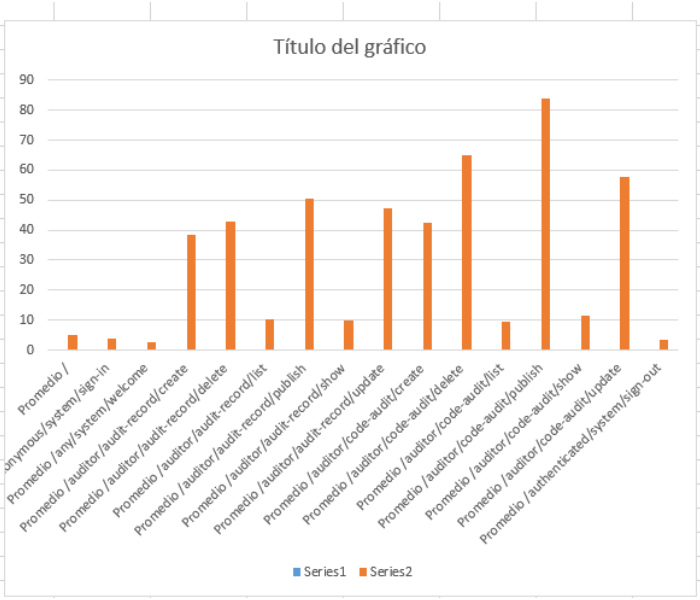
Para terminar este capítulo me gustaría comentar que he usado el Excel de SampleData para la realización de los tests .safe de los create, update, y publishl

Capítulo 2 – Pruebas de desempeño

El desarrollo del software se ha ejecutado durante todo el cuatrimestre en mi PC personal. Obteniendo los resultados de ejecutar el replayer en eclipse, nos genera una batería de datos, los cuales, analizándolos mediante las técnicas enseñadas en clase, hemos podido obtener resultados claros.

Vamos a empezar por los promedios de los resultados de búsqueda.

Promedio /	4,91299
Promedio /anonymous/system/sign-in	3,9497
Promedio /any/system/welcome	2,598694
Promedio /auditor/audit-record/create	38,58867
Promedio /auditor/audit-record/delete	43,0233
Promedio /auditor/audit-record/list	10,12158
Promedio /auditor/audit-record/publish	50,64746
Promedio /auditor/audit-record/show	9,884022
Promedio /auditor/audit-record/update	47,41088
Promedio /auditor/code-audit/create	42,66395
Promedio /auditor/code-audit/delete	64,82476
Promedio /auditor/code-audit/list	9,528978
Promedio /auditor/code-audit/publish	83,82191
Promedio /auditor/code-audit/show	11,65851
Promedio /auditor/code-audit/update	57,60908
Promedio /authenticated/system/sign-out	3,5311
Promedio general	18,01173



Como se puede observar, hay alguna ruta que tiene peticiones que han tardado más, lo que es un resultado muy bueno, porque nos indica que las búsquedas se realizan de manera rápida. En este apartado tengo que comentar que este replayer ha sido realizado tras implementar índices y optimizar funciones ya que ya realicé análisis de los tests antes de este test, pero tuvieron que repetirse por un cambio en la base de datos. Estas optimizaciones eran principalmente streams que filtraban colecciones que podían ser filtradas en las queries del repositorio.

A continuación, vamos a observar el intervalo de confianza del PC personal frente al del PC 2 obtenido de una simulación con función random:

time	time2						
118,981	128	*Before*			*After*		
4,2176	4,4204						
5,3789	5,5568	Media	18,01173466		Media	18,90199475	
2,7016	2,9224	Error típico	0,800853773		Error típico	0,843978306	
5,6944	6,2135	Mediana	8,4451		Mediana	8,882994905	
2,813	3,0785	Moda	1,6443		Moda	#N/D	
3,8537	3,8993	Desviación estándar	23,08626825		Desviación estándar	24,32942222	
2,6418	2,6693	Varianza de la muestra	532,9757817		Varianza de la muestra	591,9207856	
3,8901	4,0155	Curtosis	6,064819444		Curtosis	6,65967996	
2,8518	3,0145	Coeficiente de asimetría	2,227677693		Coeficiente de asimetría	2,285082191	
4,4202	4,5072	Rango	176,9036		Rango	193,5139574	
2,002	2,1692	Mínimo	1,4693		Mínimo	1,501044814	
3,2454	3,4588	Máximo	178,3729		Máximo	195,0150022	
2,2707	2,3879	Suma	14967,7515		Suma	15707,55764	
3,1305	3,3808	Cuenta	831		Cuenta	831	
2,2492	2,4616	Nivel de confianza(95,0%)	1,571936802		Nivel de confianza(95,0%)	1,656582769	
5,2131	5,5099						
2,2196	2,3162	Interval ms			Interval ms		
3,4333	3,5102	16,43979785	19,58367146		17,24541198	20,55857752	
2,6386	2,6561						

Con estos datos se ha realizado un Z-Test, el cual se muestra a continuación.

Prueba z para medias de dos muestras		
	118,981	*127,5729791*
Media	17,89008494	18,79126619
Varianza (conocida)	532,9757817	591,9207856
Observaciones	830	830
Diferencia hipotética de las medias	0	
z	-0,774096135	
P(Z<=z) una cola	0,219436971	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,438873943	
Valor crítico de z (dos colas)	1,959963985	

Podemos observar que Alpha es 0.05, y que el p-value es 0.438… por lo que podemos decir que los cambios **no** dieron como resultado ninguna mejora significativa; los tiempos de muestreo son diferentes, pero son globalmente iguales.

<u>También se ha replicado estas pruebas en otro ordenador</u> (PC2 – características similares) y he obtenido los siguientes resultados:

Before	
Media	17,676054
Error típico	0,889129834
Mediana	8,2571
Moda	1,8587
Desviación estándar	22,01382046
Varianza de la muestra	484,6082912
Curtosis	5,013810694
Coeficiente de asimetría	2,151641913
Rango	129,7616
Mínimo	1,4517
Máximo	131,2133
Suma	10835,4211
Cuenta	613
Nivel de confianza(95,0%)	1,746115661

Interval ()ms:	
19,42216966	15,92993834

After	
Media	16,1123892
Error típico	0,86213183
Mediana	7,77585
Moda	2,2533
Desviación estándar	20,8700019
Varianza de la muestra	435,556978
Curtosis	6,79789118
Coeficiente de asimetría	2,40979951
Rango	133,031
Mínimo	1,4817
Máximo	134,5127
Suma	9441,8601
Cuenta	586
Nivel de confianza(95,0%)	1,69325055

Interval ()ms:	
17,8056398	14,4191387

Prueba z para medias de dos muestras		
	108,5408	*134,5127*
Media	17,52758219	15,90999556
Varianza (conocida)	484,6082912	435,5569784
Observaciones	612	585
Diferencia hipotética de las medias	0	
z	1,305020867	
P(Z<=z) una cola	0,095942871	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	**0,191885742**	
Valor crítico de z (dos colas)	**1,959963985**	

Podemos observar que Alpha es 0.05, y que el p-value es 0.19… por lo que podemos decir que los cambios no dieron como resultado ninguna mejora significativa; los tiempos de muestreo son diferentes, pero son globalmente iguales.

Como **conclusión**, ninguno de los PCs muestra una diferencia significativa en el rendimiento (antes y después) a un nivel de variación del 5%. Por lo tanto, las diferencias observadas en las medias no son estadísticamente significativas, lo que sugiere que ninguno de los PCs es concluyentemente más rápido o lento que el otro según los datos proporcionados.

Bibliografía

Diapositivas de Diseño y Pruebas 2 – Universidad de Sevilla.