Spring

Sesión 8: Desarrollo rápido de aplicaciones con Roo

Indice

- ¿Qué es Roo?
- *Demo*
- Capa de acceso a datos
  - Active Record
  - Finders
  - Pruebas
- Capa web
  - Scaffolding
  - REST

Qué es Roo

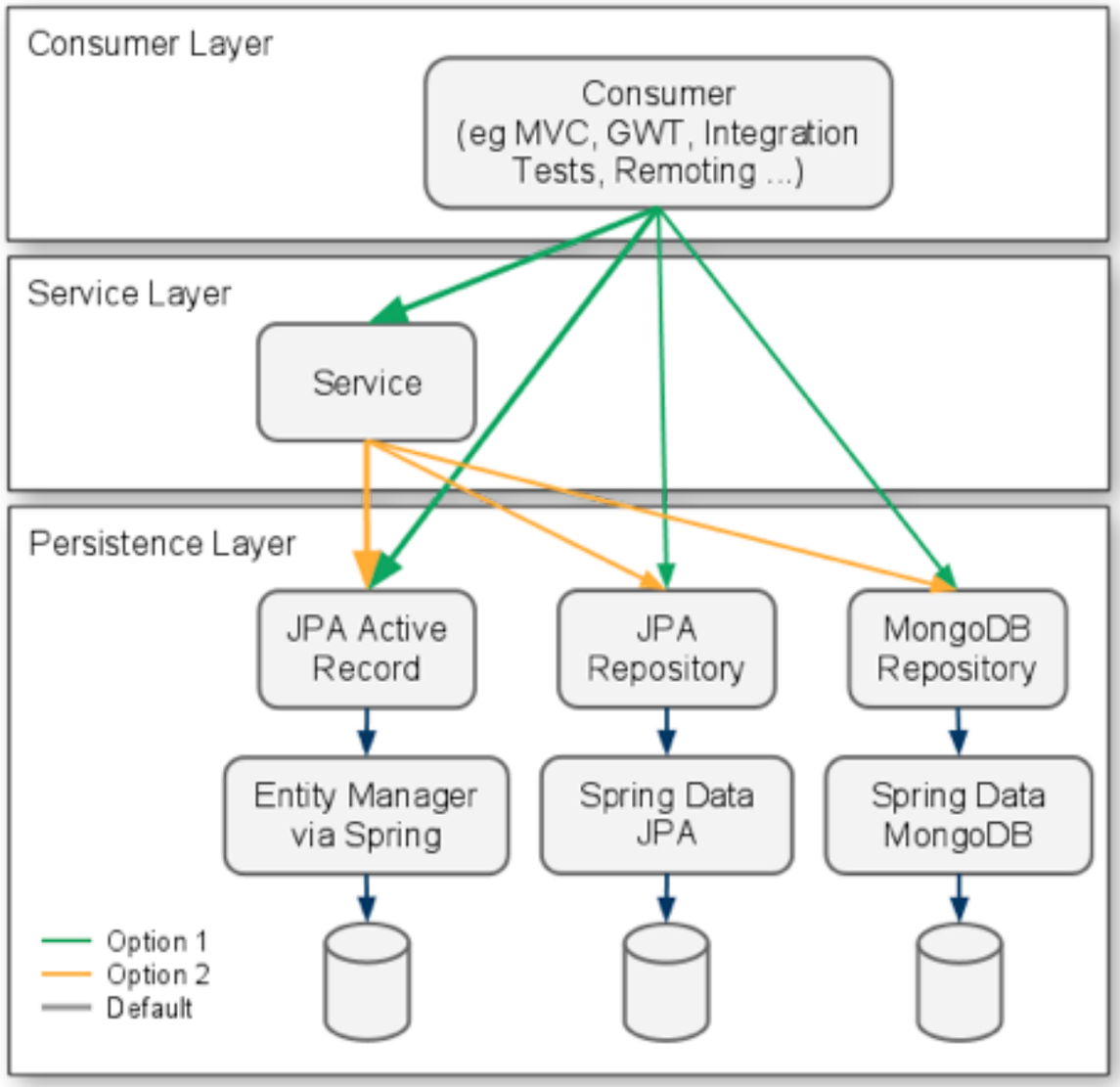
- Motivación
  - Empezar un proyecto de cero es tedioso (dependencias, configuración,...)
  - Hay mucho código repetitivo en las aplicaciones (p.ej. CRUD)
- Spring Roo
  - Herramienta de generación de código

    Separa físicamente el código generado del creado manualmente por el desarrollador, gracias al uso de AOP
  - Subproyecto de Spring que usa sus tecnologías y otras tecnologías estándar JavaEE o estándar *de facto*

    Maven, JPA, Tiles, JSF, GWT,...
  - No incorpora componentes adicionales en tiempo de ejecución
- Sería ingenuo creer que Roo nos va a escribir la aplicación entera...

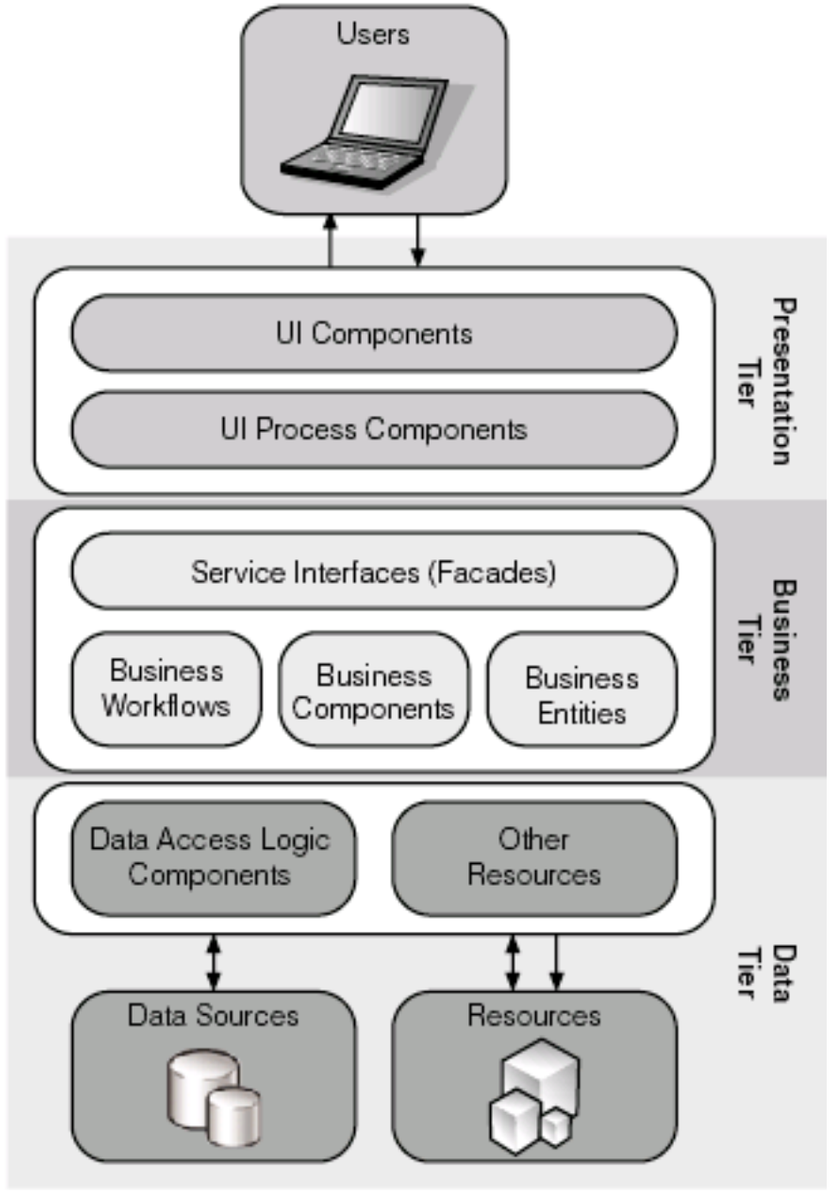
Arquitectura apps Roo vs “estándar”

http://blog.springsource.org/2011/09/14/new-application-layering-and-persistence-choices-in-spring-roo/



Roo

http://i.msdn.microsoft.com/ms998418.j2ee_interop_c6_fig6-1(en-us,MSDN.10).gif



“Estándar”

Capa de acceso a datos

- Tecnologías de persistencia: JPA y MongoDB
  - En JPA podemos seleccionar el proveedor de persistencia (implementación) y la base de datos que deseamos
- Opciones de arquitectura
  - **Active Record (por defecto)**: los propios objetos del dominio se "responsabilizan" de la persistencia. Idea usada en muchos frameworks RAD web: Rails, Grails, Django, ...
  - **Repositories**: al estilo de los DAOs. Implementados con Spring Data (subproyecto de Spring)
- Funcionalidades adicionales
  - Finders: métodos de búsqueda de datos generados automáticamente
  - Validación JSR303
  - Podemos generar automáticamente las entidades haciendo ingeniería inversa de la base de datos
  - Pruebas

Active Record

- Automáticamente se generan métodos CRUD

```
//Create
Tarea t = new Tarea();
t.setLimite(new Date());
t.setPrioridad(1);
t.setTitulo("Probando");
t.persist();
//Read
for (Tarea t_i : Tarea.findAllTareas()) {
    System.out.println(t_i.getTitulo());
}
//Update
Tarea t2 = Tarea.findTarea(1L);
t2.setPrioridad(1);
t2.merge();
//Delete
Tarea t3 = Tarea.findTarea(2L);
t3.remove();
```

Finders

- Métodos para buscar por uno o más campos que cumplan una serie de condiciones
- A partir del nombre del finder, y siguiendo una serie de convenciones simples Roo generará automáticamente el código

```
findTareasByTituloEquals(String titulo)
findTareasByTituloNotEquals(String titulo)
findTareasByTituloLike(String titulo)
findTareasByPrioridad(int prioridad)
findTareasByLimiteLessThan(Date limite)
findTareasByLimiteBetween(Date minLimite, Date
maxLimite)
```

```
~.domain.Tarea roo> finder add --finderName findTareasByPrioridad
```

Pruebas

- Podemos generarlas automáticamente

```
roo> test integration --entity es.ua.jtech.domain.Proyecto
```

- Para ejecutar pruebas es posible que necesitemos entidades con valores válidos pero al azar (clase `XXXDataOnDemand`)

```
@Test
public void testDeEjemplo() {
    ProyectoDataOnDemand pdod = new ProyectoDataOnDemand();
    Proyecto p = pdod.getNewTransientProyecto(1);
    p.persist();
    Proyecto p2 = Proyecto.findProyecto(p.getId());
    assertEquals(p.toString(),p2.toString());
}
```

Relaciones entre entidades

- Podemos hacerlo desde línea de comandos o editar directamente el .java
- Ejemplo: relación entre Proyecto y Tareas pertenecientes a él

```
roo> focus --class es.ua.jtech.domain.Proyecto
~.domain.Proyecto roo> field set --fieldName tareas --type Tarea
                                 --cardinality ONE_TO_MANY --mappedBy proyecto
~.domain.Proyecto roo> focus --class es.ua.jtech.domain.Tarea
~.domain.Tarea roo> field reference --fieldName proyecto --type Proyecto
                                    --cardinality MANY_TO_ONE
```

La capa web

- Comenzar con

```
roo> web mvc setup
```

- *Scaffolding*: generar automáticamente CRUD para la capa web: controllers y vistas (JSP + Tiles)

```
roo> web mvc all --package es.ua.jtech.web
```

```
roo> focus es.ua.jtech.domain.Proyecto
~.domain.Proyecto roo> web mvc scaffold --class es.ua.jtech.web.ProyectoController
```

La capa web (II)

- Acceso web a los finders
  - Anotar el controlador con `@RooFinder`, o bien

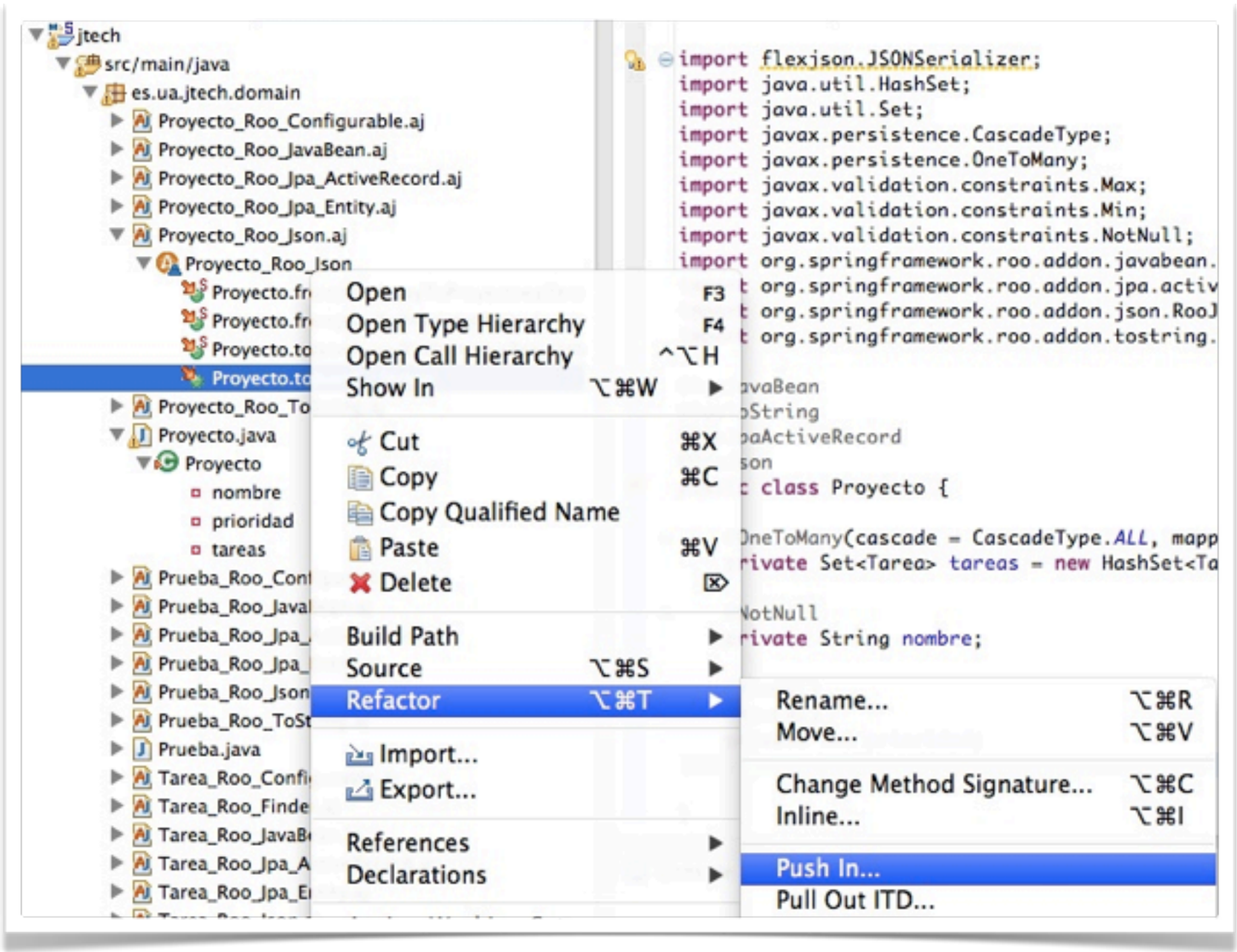
```
roo> web mvc finder all
```

- Capa REST
  - Las URL generadas automáticamente por el scaffolding son RESTful, pero no devuelven JSON
  - Para generar controladores CRUD que devuelven JSON

```
roo> json all #da soporte JSON a todas las entidades
roo> web json setup #solo necesario si no hemos hecho ya web mvc setup
roo> web mvc json all #genera los controllers CRUD para json
```

Refactorizar código Roo

- Caso más típico: *push-in*, pasar código de un .aj a Java para poder editarlo y modificarlo según nuestras necesidades
  - Por ejemplo modificar un finder para cambiar la ordenación de los resultados



Roo vs. Grails

- Hay ciertas semejanzas aparentes
  - Herramientas en línea de comandos
  - Usan Spring “por debajo”
  - Active Record
  - Finders
  - Ambos “son” de SpringSource
- **Diferencia**: Roo genera código Java puro. En Grails se usa **Groovy** (dinámico)

¿Preguntas...?