



Geeks México

BLOG DE PROGRAMACIÓN EN ESPAÑOL SOBRE JAVA,
FRAMEWORKS, BASES DE DATOS, CÓMPUTO EN LA NUBE, ETC.
EN ESPAÑOL Y EN INGLÉS.

[TUTORIALES EN ESPAÑOL](#)

[TUTORIALS IN ENGLISH](#)

[ABOUT](#)

[CONTACT](#)

Anuncios

5 Comandos, una aplicación web

completa con JHipster en español

📅 OCTUBRE 4, 2016 💬 DEJA UN COMENTARIO

En el post anterior se explicó la configuración del entorno de trabajo con JHipster, para verlo ir a [Crea una aplicación Web+REST \(Spring boot + AngularJS\) en minutos utilizando JHipster](https://geeksjavamexico.wordpress.com/2016/09/29/crea-una-aplicacion-webrest-spring-boot-angularjs-en-minutos-utilizando-jhipster/) (<https://geeksjavamexico.wordpress.com/2016/09/29/crea-una-aplicacion-webrest-spring-boot-angularjs-en-minutos-utilizando-jhipster/>). En este post se explica paso a paso como crear una aplicación completa haciendo uso de comandos simples con JHipster.

Creación de entidades

En JHipster no es necesario escribir una sola línea de código para crear una aplicación web debido a que es una herramienta scaffolding lo cual permite crear la aplicación en base a comandos.

Definiendo aplicación a crear

La aplicación a crear será una aplicación para registrar departamentos, empleados y habilidades de una empresa, la idea de este post es crear la aplicación completa sin escribir una sola línea de código. Para conseguir esto se deben crear las siguientes entidades:

- Departamento

- Empleado
- Habilidad

Creando entidad *department*

Lo primero será crear la entidad departamento utilizando el siguiente comando:

```
1 | yo jhipster:entity department
```

Una vez que se ejecuta el comando, JHipster preguntará la siguiente información sobre la entidad que se desea construir:

- Desea agregar un campo a la entidad?
 - Nombre del campo
 - Tipo del campo
 - Reglas de validación del campo

Para la entidad ***department*** se crearán el siguiente campo:

- name
 - String

Una vez hecho esto JHipster preguntará si se desea agregar una relación a la entidad que se generó, en este caso se seleccionará que si se desea y se proporcionará la siguiente información:

- Nombre de la otra entidad : Employee
- Nombre de la relación :employee
- Tipo de relación : one-to-many
- Nombre de la relación en la otra entidad: department

Cuando se termine JHipster preguntará lo siguiente:

- Si desea crear otra relación: Seleccionaremos en este ejemplo que no.
- Si se desea utilizar DTO para los endpoints que se generen. es importante mencionar que dichos DTO's son creados utilizando MapStruct, para más información sobre esto ver el post [Mapeo de Beans con MapStruct](#) : Seleccionaremos en este ejemplo que no.
- Si se desea que el servicio REST acceda directamente al repositorio de Spring data o si se desea que se genere un servicio separado para esto: Seleccionaremos en este ejemplo *Yes, generate a separate service class*.
- Si se desea agregar paginación a la entidad consultada: Seleccionaremos en este ejemplo no.

Una vez hecho esto JHipster notificará los archivos a sobre escribir, seleccionaremos que si deseamos que lo haga y la entidad se generará de forma completa.

Creando entidad *employee*

En la definición de la entidad ***department*** se definió que existirá una relación entre ella y employee, pero falta definir la entidad employee con el fin de definir los campos que contendrá y las reglas de cada uno. Para esto ejecutaremos el siguiente comando:

```
1 | yo jhipster:entity employee
```

Una vez que se ejecuta el comando, agregaremos los siguientes campos a la entidad:

- name
 - String
- age
 - Integer
- salary

- Double

El siguiente paso es definir las relaciones que tendrá la entidad **employee**, para este caso serán 2 la que se registrarán una con la entidad ya creada **department** y una con la entidad **Skills** que contendrá las habilidades del empleado, para hacerlo se responderá lo siguiente:

- Desea agregar una relación a otra entidad: Si
 - Nombre de la otra entidad: Department
 - Nombre de la relación: department
 - Tipo de relación: many-to-one
 - Cuando se muestre esta relación en la aplicación, que campo de Department desea utilizar : id
 - Desea agregar validaciones a la relación: No
- Desea agregar una relación a otra entidad : Si
 - Nombre de la otra entidad : Skill
 - Nombre de la relación :skill
 - Tipo de relación : one-to-many
 - Nombre de la relación en la otra entidad: employee

Creando entidad *skill*

La última entidad será skill la cuál contendrá información sobre las habilidades del empleado. Para generarla ejecutaremos lo siguiente:

```
1 | yo jhipster:entity skill
```

Una vez que se ejecuta el comando, agregaremos los siguientes campos a la entidad:

- name
 - String
- yearsExperience
 - Integer

Una vez definidos los campos de la entidad **skill** se definirá la relación que tiene con la entidad empleado:

- Desea agregar una relación a otra entidad: Si
 - Nombre de la otra entidad: Employee
 - Nombre de la relación: employee
 - Tipo de relación: many-to-one
 - Cuando se muestre esta relación en la aplicación, que campo de Employee desea utilizar : id
 - Desea agregar validaciones a la relación: No

Probando todo junto

Con estos comandos ya se tiene lista la aplicación completa de JHipster lo único que resta es ejecutar los siguientes dos comandos:

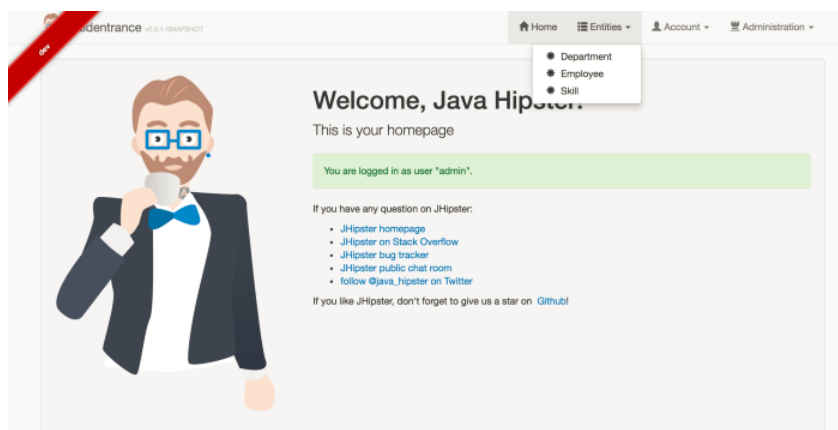
```
1 | mvn clean install
```

Construirá la aplicación de nuevo con los nuevos cambios generados con Spring boot.

```
1 | mvn spring-boot:run
```

Ejecutará la aplicación y la pondrá disponible en la siguiente URL <http://localhost:8080/> (<http://localhost:8080/>).

A continuación se muestra como se ve la aplicación:



Como se puede observar el menu Entities ahora contiene las 3 entidades que se generaron:

- *Department*
- *Employee*
- *Skill*

Department

La primera pantalla de entidad que se mostrará será department ya que no depende de ninguna otra:



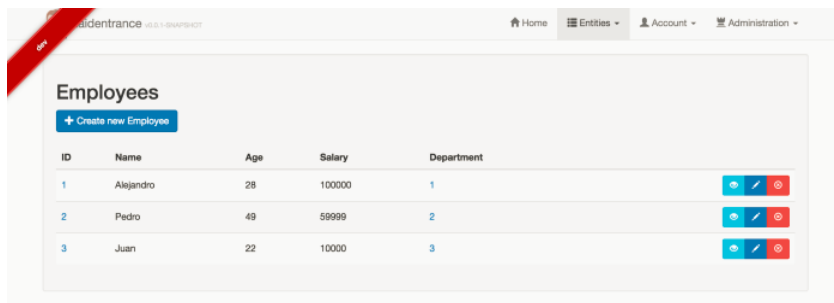
ID	Name	
1	Administration	  
2	Accounting	  
3	Marketing and Advertising	  

Como se puede observar la entidad Department tiene la siguiente funcionalidad:

- *Crear nuevos departamentos*
- *Ver departamentos existentes*
- *Editar departamentos existentes*
- *Borrar departamentos*

Employee

La pantalla employee se ve como se muestra a continuación



ID	Name	Age	Salary	Department
1	Alejandro	28	100000	1
2	Pedro	49	59999	2
3	Juan	22	10000	3

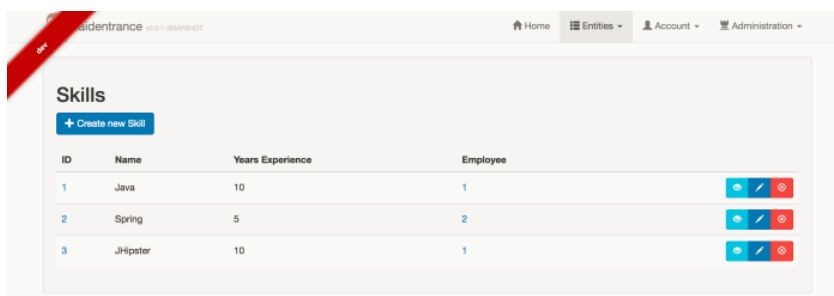
Como se puede observar la entidad Employee tiene la siguiente funcionalidad:

- Crear nuevos empleados
- Ver empleados existentes
- Editar departamentos existentes
- Borrar departamentos existentes

Nota : Como se puede observar en la columna Department aparece un número el cual es el id del departamento con un link a la pantalla del departamento, la razón por la que aparece el id es porque se seleccionó "Cuando se muestre esta relación en la aplicación, que campo de Department desea utilizar"

Skill

La pantalla employee se ve como se muestra a continuación



ID	Name	Years Experience	Employee
1	Java	10	1
2	Spring	5	2
3	JHipster	10	1

Otros recursos creados

Además de las pantallas creadas JHipster crea lo siguiente de forma automática:

Web services REST y documentación de los mismos:

department-resource : Department Resource				Show/Hide	List Operations	Expand Operations
GET	/api/departments					getAllDepartments
POST	/api/departments					createDepartment
PUT	/api/departments					updateDepartment
DELETE	/api/departments/{id}					deleteDepartment
GET	/api/departments/{id}					getDepartment
employee-resource : Employee Resource				Show/Hide	List Operations	Expand Operations
GET	/api/employees					getAllEmployees
POST	/api/employees					createEmployee
PUT	/api/employees					updateEmployee
DELETE	/api/employees/{id}					deleteEmployee
GET	/api/employees/{id}					getEmployee
profile-info-resource : Profile Info Resource				Show/Hide	List Operations	Expand Operations
skill-resource : Skill Resource				Show/Hide	List Operations	Expand Operations
GET	/api/skills					getAllSkills
POST	/api/skills					createSkill
PUT	/api/skills					updateSkill

Estadísticas de uso de los servicios web:

Services statistics (time in millisecond)									
Service name	Count	Mean	Min	p50	p75	p95	p99	Max	
com.raidentrance.jhipster.web.rest.AccountResource.activateAccount	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.changePassword	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.finishPasswordReset	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.getAccount	1	33	33	33	33	33	33	33	
com.raidentrance.jhipster.web.rest.AccountResource.getCurrentSessions	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.invalidateSession	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.isAuthenticated	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.registerAccount	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.requestPasswordReset	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.AccountResource.saveAccount	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.DepartmentResource.createDepartment	3	9	5	5	19	19	19	19	
com.raidentrance.jhipster.web.rest.DepartmentResource.deleteDepartment	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.DepartmentResource.getAllDepartments	9	5	4	5	5	5	41	41	
com.raidentrance.jhipster.web.rest.DepartmentResource.getDepartment	1	4	4	4	4	4	4	4	
com.raidentrance.jhipster.web.rest.DepartmentResource.updateDepartment	0	0	0	0	0	0	0	0	
com.raidentrance.jhipster.web.rest.EmployeeResource.createEmployee	3	6	4	6	10	10	10	10	

Código siguiendo buenas prácticas

Es importante saber que si se desea modificar la aplicación para cambiar la apariencia o mostrar otro tipo de información es posible hacerlo modificando el código, ya que el código generado por JHipster sigue buenas prácticas de desarrollo lo cual lo hace muy fácil de modificar.

Autor: Alejandro Agapito Bautista

Twitter: [@raidentrance](#)

(<https://geeksjavamexico.wordpress.com/mentions/raidentrance/>)

Contacto:raidentrance@gmail.com

Anuncios

