EBook Gratis

APRENDIZAJE grails

Free unaffiliated eBook created from
Stack Overflow contributors.

#grails

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: grails

It is an unofficial and free grails ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official grails.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con los grails

Observaciones

Grails es un marco muy poderoso de desarrollo rápido de aplicaciones (RAD) para la **plataforma Java que** apunta a multiplicar **la productividad de los desarrolladores** gracias a una Convención sobre Configuración, valores predeterminados razonables y API de opinión. Se integra sin problemas con la JVM y el **lenguaje Groovy,** lo que le permite ser **productivo de manera inmediata** al mismo tiempo que proporciona funciones potentes, incluida la asignación de objetos relacional (ORM) integrada, lenguajes específicos del dominio (DSL), tiempo de ejecución y metaprogramación en tiempo de compilación y asíncrono programación.

La página de inicio de Grails se encuentra aquí.

Versiones

Versión	Observaciones	Fecha de lanzamiento
2.5.5	última versión en línea legada 2.x	2016-10-27
3.2.2	último a partir del 30-oct-2016	2016-06-24
3.2.3	última versión en 3.x	2016-11-10

Examples

Instalacion de grails

Nota: GRAILS requiere un JDK de Java instalado (un JRE de entorno de ejecución no es suficiente) en su sistema, antes de configurar Grails. Por favor, consulte, cómo instalar JDK . A partir de este escrito, se recomienda instalar el último JDK.

Para Mac OSX, Linux, Cygwin, Solaris y FreeBSD:

La forma más sencilla de administrar las versiones de Grails es usando sdkman . Si `sdkman` está instalado, puede instalar cualquier versión de Grails usando

```
sdk install grails [version]
```

Esto se hará cargo de todos los pasos para obtener este derecho. Si omite la `version` , se instalará la última versión de grails. Para obtener más información sobre el uso de `sdkman` , consulte la página de uso de sdkman .

Para Linux:

```
GRAILS_HOME=$HOME/bin/grails/current
# abbreviating it using "..." for brevity
PATH=$GRAILS_HOME/bin:$JAVA_HOME/bin: ... :$PATH
```

Para ventanas:

1. Descargue un Java JDK de Oracle e instálelo en su máquina Windows. Tome nota de la carpeta de instalación.
2. Descarga una versión de Grails manualmente desde la página de Descargas .
3. Extrae el archivo de Grails, donde quieras.
4. **Importante:** debe configurar 2 nuevas variables de entorno `JAVA_HOME` y `GRAILS_HOME` (para Windows 10, que se encuentran en * Panel de control \ Sistema y Seguridad \ Sistema \ Configuración avanzada del sistema \ pestaña Avanzada \ Variables de entorno) *, apuntando a los directorios extraídos, por ejemplo

   Nombre: JAVA_HOME

   Valor: C: \ Programs \ Java \ jdk1.8.0_31

   Nombre: GRAILS_HOME

   Valor: c: \ grails \ grails-3.2.4

5. **Importante:** debe extender la variable `PATH` Windows para incluir tanto JAVA_HOME como GRAILS_HOME. La variable de ruta también se encuentra en el panel de control (ver 4), por ejemplo, agregue lo siguiente al final:

   ; C: \ Programs \ Java \ jdk1.8.0_31 \ bin; c: \ grails \ grails-3.2.4; c: \ grails \ grails \ grails-3.2.4 \ bin

5. Para verificar que su instalación es correcta, abra una `GRAILS -VERSION` comando y escriba `GRAILS -VERSION` . Deberías conseguir algo como:

```
| Grails Version: 3.2.4
| Groovy Version: 2.4.6
| JVM Version: 1.8.0_65
```

Creando una aplicación

Para crear una aplicación Grails, use el comando `grails create-app` . El siguiente comando crea una aplicación Grails, llamada `myapp` en el directorio actual:

```
grails create-app fancy-app
```

Ejecutarlo es tan simple como visitar el directorio de aplicaciones recién creado:

```
cd fancy-app
```

y entonces

```
grails run-app
// in order to run the app on a different port, e.g. 8888, use this instead
grails run-app -port 8888
// in order to run the app with a secure communication
grails run-app -https
```

Probando una aplicación

Los comandos create- * en Grails crean automáticamente pruebas de unidad o integración para ti dentro del directorio src / test / groovy. Por supuesto, depende de usted completar estas pruebas con una lógica de prueba válida, información que se puede encontrar en la sección sobre Pruebas de unidad e integración.

Para ejecutar pruebas, ejecute el comando test-app de la siguiente manera:

```
grails test-app
```

Creando un modelo

Un modelo (ver: Patrón Modelo-Vista-Controlador) en Grails está representado por una llamada **Clase de Dominio** . Las clases de dominio pueden definir tanto la persistencia como la presentación de información en gráficos. Las clases de dominio también pueden contener validaciones.

Para administrar una flota de autos en su aplicación Grails, puede definir una clase de dominio para describir, almacenar y representar varios autos en su flota.

Para crear un código auxiliar para una clase de dominio, ejecute el siguiente comando dentro de la carpeta de su aplicación:

```
grails create-domain-class org.fleetmanager.Car
```

A continuación, abra el archivo car.groovy generado y edite su clase de dominio de la siguiente manera:

```
package org.fleetmanager

class Car {
   String      manufacturer
   String      model
   String      color
   Integer     year
   Date        acquisitionDate
   Boolean     isElectric
}
```

Finalmente, genere un controlador para el dominio de su automóvil y una vista usando el siguiente comando de Grails:

```
grails generate-all org.fleetmanager.Car
```

Ahora, puede ejecutar sus aplicaciones, seleccionar el controlador de automóvil y administrar su flota.

Lea Empezando con los grails en línea: https://riptutorial.com/es/grails/topic/1435/empezando-con-los-grails

Capítulo 2: Clases de dominio como recursos REST

Introducción

La forma más fácil de crear una API RESTful en Grails es exponer una clase de dominio como un recurso REST. Esto se puede hacer agregando la transformación grails.rest.Resource a cualquier clase de dominio.

Examples

API REST simple con grails

```
import grails.rest.*

@Resource(uri='/books')
class Book {

    String title

    static constraints = {
        title blank:false
    }
}
```

Simplemente agregando la transformación de recursos y especificando un URI, su clase de dominio estará disponible automáticamente como un recurso REST en formatos XML o JSON. La transformación registrará automáticamente la asignación de URL RESTful necesaria y creará un controlador llamado BookController.

Puede probarlo agregando algunos datos de prueba a BootStrap.groovy:

```
def init = { servletContext ->
    new Book(title:"The Stand").save()
    new Book(title:"The Shining").save()
}
```

Y luego presionando la URL `http://localhost:8080/books/1` , que mostrará la respuesta como:

```
<?xml version="1.0" encoding="UTF-8"?>
<book id="1">
    <title>The Stand</title>
</book>
```

Si cambia la URL a `http://localhost:8080/books/1.json` obtendrá una respuesta JSON como:

```
{"id":1,"title":"The Stand"}
```

Si desea cambiar el valor predeterminado para devolver JSON en lugar de XML, puede hacerlo configurando el atributo de formatos de la transformación de recursos:

```
import grails.rest.*

@Resource(uri='/books', formats=['json', 'xml'])
class Book {
    ...
}
```

Mapeo a recursos REST

Si prefiere mantener la declaración de la asignación de URL en su archivo UrlMappings.groovy, basta con eliminar el atributo uri de la transformación de recursos y agregar la siguiente línea a `UrlMappings.groovy` :

```
"/books"(resources:"book")
```

Extender su API para incluir más puntos finales se vuelve trivial:

```
"/books"(resources:"book") {
    "/publisher"(controller:"publisher", method:"GET")
}
```

El ejemplo anterior expondrá el URI `/books/1/publisher`.

Añadir HTTPS a Grails Server

Los certificados SSL usan algo llamado criptografía de clave pública. Necesitamos usar HTTP en lugar de HTTP debido a que los datos se mantienen seguros entre los servidores y a la confianza del cliente. Para habilitar esta opción en Grails, tenemos que ejecutar nuestra aplicación de manera diferente. El siguiente comando:

```
grails run-app -https
```

Lea Clases de dominio como recursos REST en línea:
https://riptutorial.com/es/grails/topic/8944/clases-de-dominio-como-recursos-rest

Capítulo 3: Despliegue

Examples

Jar ejecutable

Una de las maneras más fáciles de implementar Grails 3.x es compilar un archivo jar ejecutable que incruste un contenedor de servlets (Tomcat, Undertow, etc.) con la aplicación.

Modificar `build.gradle` :

```
// Remove or comment out the war plugin:
// apply plugin:"war"

// Enable the executable jar:
springBoot {
    executable = true
}

// Optional: Customize the jar properties:
//  https://docs.gradle.org/current/dsl/org.gradle.api.tasks.bundling.Jar.html
jar {
    archiveName('myapp.jar')
}
```

Construir usando `./gradlew assemble`

El jar resultante es ahora una aplicación completamente ejecutable que puedes iniciar:

```
$ head build/libs/myapp.jar
#!/bin/bash
#
#     .   ____          _            __ _ _
#    /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
#   ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
#    \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
#     '  |____| .__|_| |_|_| |_\__, | / / / /
#    =========|_|==============|___/=/_/_/_/
#    :: Spring Boot Startup Script ::
#
```

Puede iniciarlo como lo haría normalmente para cualquier aplicación de línea de comandos:

```
$ ./build/libs/myapp.jar
Grails application running at http://localhost:8080 in environment: production
```

También se comporta como un servicio de iniciación:

```
$ ln -s /opt/myapp/myapp.jar /etc/init.d/myapp
$ service myapp [start|stop|status|restart]
```

La documentación detallada se encuentra en los documentos de arranque de primavera: http://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html

Creación de archivos de guerra

Cuando escribimos una aplicación web en Grails, para implementar la aplicación necesitamos un archivo "war" que se debe colocar en el contenedor de servlets (Tomcat, etc.).

Primero ve al directorio del proyecto:

```
cd to_project_directory
```

1. Creación de archivos de guerra desde el símbolo del sistema:

```
grails war
```

2. Siempre es recomendable que limpies tu aplicación antes de crear la guerra.

Aplicación de limpieza desde el símbolo del sistema:

```
grails clean
```

La combinación de los dos pasos anteriores en uno resultará en

```
grails clean && grails war
```

También puede especificar el entorno en el que desea crear el archivo war.

```
grails [environment] war
```

Donde `[environment]` puede tomar los siguientes valores: `dev` , `prod` o `test` por ejemplo.

A diferencia de otros comandos, el comando war se ejecuta en el entorno de producción por defecto en lugar de desarrollo.

Lea Despliegue en línea: https://riptutorial.com/es/grails/topic/3701/despliegue

Capítulo 4: SGP

Parámetros

Variables y alcances	Detalles
solicitud	Instancia de ServletContext
aplicaciónContexto	Instancia de Spring ApplicationContext
destello	El objeto flash
aplicación aplicada	Instancia de aplicación Grails
afuera	Escritor de respuestas para escribir en el flujo de salida.
params	objeto params para recuperar los parámetros de solicitud
solicitud	Instancia de HttpServletRequest
respuesta	Instancia de HttpServletResponse
sesión	Instancia HttpSession
solicitud de web	Instancia de GrailsWebRequest

Examples

Lo esencial

GSP admite el uso de `<% %>` %% `<% %>` bloques de scriptlet para incrustar código Groovy (esto no se recomienda):

```
<html>
   <body>
     <% out << "Hello GSP!" %>
   </body>
</html>
```

También puede usar la sintaxis `<%= %>` para generar valores, como en JSP:

```
<html>
   <body>
     <%="Hello GSP!" %>
   </body>
</html>
```

GSP también admite comentarios del lado del servidor de estilo JSP:

```
<html>
   <body>
     <%-- This is my comment --%>
     <%="Hello GSP!" %>
   </body>
</html>
```

Expresiones

En GSP, la sintaxis `<%= %>` rara vez se usa debido al soporte para **expresiones GSP** .

Una expresión GSP es similar a una expresión **JSP EL** o **Groovy GString** y toma la forma `${expr}` :

```
<html>
  <body>
    Hello ${params.name}
  </body>
</html>
```

Sin embargo, a diferencia de JSP EL, puede tener cualquier expresión Groovy dentro del bloque `${..}` .

Cualquier expresión Groovy se puede interpolar en todos los literales de cadena, aparte de las cadenas entre comillas simples y triples. La interpolación es el acto de reemplazar un marcador de posición en la cadena con su valor tras la evaluación de la cadena. Las expresiones de marcador de posición están rodeadas por $ {} o prefijadas con $ para expresiones punteadas. El valor de la expresión dentro del marcador de posición se evalúa en su representación de cadena cuando GString se pasa a un método que toma una cadena como argumento al llamar a toString () en esa expresión.

Etiquetas GSP

Hay una variedad de etiquetas gsp disponibles que se pueden usar para crear formularios, campos de texto, botones de opción, casillas de verificación, si-else, para cada uno, etc.

<g: si>

```
<g:if test="${session.role == 'admin'}">
   <%-- show administrative functions --%>
</g:if>
<g:else>
   <%-- show basic functions --%>
</g:else>
```

<g: cada>

```
<g:each in="${[1,2,3]}" var="num">
  <p>Number ${num}</p>
```

```
</g:each>
```

formar

```
<g:form name="myForm" url="[controller:'book',action:'list']">...</g:form>
```

campo de texto

```
<g:textField name="myField" value="${myValue}" />
```

radio

```
<g:radio name="myGroup" value="1"/>
```

Siga este enlace para obtener más información:
http://docs.grails.org/latest/guide/theWebLayer.html#tags

Lea SGP en línea: https://riptutorial.com/es/grails/topic/4531/sgp

Creditos

S. No	Capítulos	Contributors
1	Empezando con los grails	Abdelsalam A. Shahlol, Adeel Ansari, Andrea Zago, bronoman, Burt Beckwith, Community, Farshid Zaker, Graeme Rocher, Jason Bourne, Jesús Iglesias, rahul, saw303
2	Clases de dominio como recursos REST	Jason Bourne
3	Despliegue	erichelgeson, NachoB, Prakash Thete, saw303
4	SGP	Adeel Ansari, Anshul, audittxl