Kostenloses eBook

LERNEN grails

Free unaffiliated eBook created from
Stack Overflow contributors.

#grails

Inhaltsverzeichnis

Über

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Kapitel 1: Erste Schritte mit grails

Bemerkungen

Grails ist ein sehr leistungsfähiges Rapid Application Development (RAD) -Framework für die **Java-Plattform** , **das die Produktivität der Entwickler** durch eine Konvention-über-Konfiguration, vernünftige Standardeinstellungen und meinungsfähige APIs steigern soll. Es lässt sich nahtlos in die JVM und die **Groovy-Sprache integrieren,** sodass Sie **sofort produktiv arbeiten können** und leistungsstarke Funktionen bereitstellen, darunter integriertes objektrelationales Mapping (ORM), domänenspezifische Sprachen (DSL), Laufzeit- und Kompilierzeit-Meta-Programmierung und asynchron Programmierung.

Die Grails-Homepage finden Sie hier

Versionen

Ausführung	Bemerkungen	Veröffentlichungsdatum
2.5.5	neueste Version in 2.x Legacy-Linie	2016-10-27
3.2.2	spätestens am 30.10.2016	2016-06-24
3.2.3	neueste Version in 3.x	2016-11-10

Examples

Grails Installation

Hinweis: Für GRAILS muss auf Ihrem System ein Java-JDK installiert sein (eine Laufzeitumgebung, in der JRE nicht ausreicht), bevor Sie Grails einrichten. Lesen Sie bitte, wie Sie das JDK installieren . Zum jetzigen Zeitpunkt wird empfohlen, das neueste JDK zu installieren.

Für Mac OSX, Linux, Cygwin, Solaris und FreeBSD:

Die einfachste Möglichkeit, Grails-Versionen zu verwalten, ist sdkman . Wenn `sdkman` installiert ist, können Sie jede Version von Grails mit installieren

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

Dadurch werden alle Schritte unternommen, um dies richtig zu machen. Wenn Sie die `version` überspringen, wird die neueste Version von grails installiert. Weitere `sdkman` zur Verwendung von `sdkman` finden Sie auf der Seite zur Verwendung von sdkman .

Für Linux:

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

Für Windows:

1. Laden Sie ein Java-JDK von Oracle herunter und installieren Sie es auf Ihrem Windows-Computer. Beachten Sie den Installationsordner.
2. Laden Sie eine Version von Grails manuell von der Downloadseite herunter .
3. Extrahieren Sie die Grails-Datei an einem beliebigen Ort.
4. **Wichtig:** Sie müssen zwei neue Umgebungsvariablen `JAVA_HOME` und `GRAILS_HOME` (für Windows 10 unter * Systemsteuerung \ System und Sicherheit \ System \ Erweiterte Systemeinstellungen \ Registerkarte Erweitert \ Umgebungsvariablen) * `GRAILS_HOME` , die auf die extrahierten Verzeichnisse verweisen, z

   Name: JAVA_HOME

   Wert: C: \ Programme \ Java \ jdk1.8.0_31

   Name: GRAILS_HOME

   Wert: c: \ grails \ grails-3.2.4

5. **Wichtig:** Sie müssen die Windows `PATH` Variable erweitern, um sowohl JAVA_HOME als auch GRAILS_HOME einzuschließen. Die Pfadvariable befindet sich auch in der Systemsteuerung (siehe 4). Fügen Sie am Ende Folgendes hinzu:

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

5. Um zu überprüfen, ob Ihre Installation korrekt ist, öffnen Sie eine Eingabeaufforderung und `GRAILS -VERSION` . Sie sollten etwas bekommen wie:

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

Anwendung erstellen

Verwenden `grails create-app` zum Erstellen einer Grails-Anwendung den `grails create-app` . Der folgende Befehl erstellt eine Grails-Anwendung mit dem Namen `myapp` im aktuellen Verzeichnis:

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

Das Ausführen ist so einfach wie das Öffnen des neu erstellten Anwendungsverzeichnisses:

```
cd fancy-app
```

und dann

```
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
```

Anwendung testen

Die create- * -Befehle in Grails erstellen automatisch Unit- oder Integrationstests für Sie im Verzeichnis src / test / groovy. Es liegt natürlich an Ihnen, diese Tests mit einer gültigen Testlogik zu bestücken. Informationen dazu finden Sie im Abschnitt zu Unit- und Integrationstests.

Um Tests auszuführen, führen Sie den Befehl test-app wie folgt aus:

```
grails test-app
```

Ein Modell erstellen

Ein Modell (siehe: Model-View-Controller-Muster) in Grails wird durch eine sogenannte **Domain-Klasse dargestellt** . Domänenklassen können sowohl die Persistenz als auch die Darstellung von Informationen in Grails definieren. Domänenklassen können auch Validierungen enthalten.

Um eine Fahrzeugflotte in Ihrer Grails-Anwendung zu verwalten, können Sie eine Domänenklasse zum Beschreiben, Speichern und Darstellen verschiedener Fahrzeuge in Ihrer Flotte definieren.

Um einen Stub für eine Domänenklasse zu erstellen, führen Sie den folgenden Befehl in Ihrem Anwendungsordner aus:

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

Öffnen Sie als Nächstes die generierte car.groovy-Datei und bearbeiten Sie Ihre Domänenklasse wie folgt:

```
package org.fleetmanager

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

Generieren Sie schließlich einen Controller für Ihre Autodomäne und eine Ansicht mit dem folgenden Grails-Befehl:

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

Jetzt können Sie Ihre Anwendungen ausführen, die Fahrzeugsteuerung auswählen und Ihre Flotte verwalten.

Erste Schritte mit grails online lesen: https://riptutorial.com/de/grails/topic/1435/erste-schritte-mit-grails

Kapitel 2: Domänenklassen als REST-Ressourcen

Einführung

Die einfachste Möglichkeit zum Erstellen einer RESTful-API in Grails besteht darin, eine Domänenklasse als REST-Ressource bereitzustellen. Dies kann durch Hinzufügen der Grails.rest.Resource-Transformation zu einer beliebigen Domänenklasse erfolgen.

Examples

Einfache REST-API mit Grails

```
import grails.rest.*

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

    String title

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

Durch Hinzufügen der Ressourcenumwandlung und Festlegen eines URI wird Ihre Domänenklasse automatisch als REST-Ressource im XML- oder JSON-Format verfügbar. Die Umwandlung registriert automatisch die erforderliche RESTful-URL-Zuordnung und erstellt einen Controller namens BookController.

Sie können es ausprobieren, indem Sie einige Testdaten zu BootStrap.groovy hinzufügen:

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

Und dann die URL `http://localhost:8080/books/1` , die die Antwort wie folgt rendert:

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

Wenn Sie die URL in `http://localhost:8080/books/1.json` Sie eine JSON-Antwort, beispielsweise:

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

Wenn Sie den Standardwert so ändern möchten, dass JSON anstelle von XML zurückgegeben wird, können Sie dies tun, indem Sie das Formatattribut der Resource-Umwandlung festlegen:

```
import grails.rest.*

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

Zuordnung zu REST-Ressourcen

Wenn Sie die Deklaration der URL-Zuordnung lieber in Ihrer UrlMappings.groovy-Datei behalten möchten, müssen Sie einfach das uri-Attribut der Resource-Umwandlung entfernen und die folgende Zeile zu `UrlMappings.groovy` hinzufügen.

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

Wenn Sie Ihre API erweitern, um mehr Endpunkte einzubeziehen, wird dies einfach:

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

Das obige Beispiel zeigt den URI `/books/1/publisher`.

Fügen Sie HTTPS zu Grails Server hinzu

SSL-Zertifikate verwenden eine sogenannte Public-Key-Kryptografie. Wir müssen anstelle von Http Https verwenden, da die Daten zwischen Servern geschützt sind und das Vertrauen der Kunden verbessert wird. Um diese Option in Grails zu aktivieren, müssen wir unsere App anders ausführen. Der Befehl unten:

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

Domänenklassen als REST-Ressourcen online lesen:
https://riptutorial.com/de/grails/topic/8944/domanenklassen-als-rest-ressourcen

Kapitel 3: Einsatz

Examples

Ausführbares Glas

Eine der einfachsten Methoden zum Bereitstellen von Grails 3.x ist das Erstellen einer ausführbaren JAR-Datei, die einen Servlet-Container (Tomcat, Undertow usw.) in die Anwendung einbettet.

`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')
}
```

Erstellen Sie mit `./gradlew assemble`

Ergebnis jar ist jetzt eine vollständig ausführbare App, die Sie starten können:

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

Sie können es wie gewohnt für jede Befehlszeilen-App starten:

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

Es verhält sich auch wie ein Init-Dienst:

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

Ausführliche Dokumentation finden Sie unter den Spring-Boot-Dokumenten:
http://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html

Kriegsdateierstellung

Wenn wir eine Webanwendung in Grails schreiben, benötigen wir zur Bereitstellung der Anwendung eine "war" -Datei, die in den Servlet-Container (Tomcat usw.) gestellt werden muss.

Zuerst gehen Sie zum Projektverzeichnis:

```
cd to_project_directory
```

1. Kriegsdatei-Erstellung über die Eingabeaufforderung:

```
grails war
```

2.Ist es immer empfehlenswert, dass Sie Ihre Anwendung vor der Erstellung des Krieges bereinigen

Anwendung von der Eingabeaufforderung bereinigen:

```
grails clean
```

Das Kombinieren der beiden obigen Schritte in einem führt zu

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

Sie können auch die Umgebung angeben, in der Sie die Kriegsdatei erstellen möchten.

```
grails [environment] war
```

Wobei `[environment]` die folgenden Werte annehmen kann: `dev` , `prod` oder `test` zum Beispiel.

Im Gegensatz zu anderen Befehlen wird der Kriegsbefehl standardmäßig in der Produktionsumgebung statt in der Entwicklung ausgeführt.

Einsatz online lesen: https://riptutorial.com/de/grails/topic/3701/einsatz

Kapitel 4: GSP

Parameter

Variablen und Gültigkeitsbereiche	Einzelheiten
Anwendung	ServletContext- Instanz
applicationContext	Spring ApplicationContext- Instanz
Blitz	Das Flashobjekt
grailsAnwendung	GrailsApplication- Instanz
aus	Antwortschreiber zum Schreiben in den Ausgabestrom
Params	Params- Objekt zum Abrufen von Anforderungsparametern
anfordern	HttpServletRequest- Instanz
Antwort	HttpServletResponse- Instanz
Session	HttpSession- Instanz
webRequest	GrailsWebRequest- Instanz

Examples

Grundlagen

GSP unterstützt die Verwendung von `<% %>` %% `<% %>` Scriptletblöcken zum Einbetten von Groovy-Code (dies wird nicht empfohlen)

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

Sie können auch die `<%= %>` -Syntax verwenden, um Werte wie in JSP auszugeben:

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

```
</html>
```

GSP unterstützt auch serverseitige Kommentare im JSP-Stil:

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

Ausdrücke

In GSP wird die `<%= %>` -Syntax selten verwendet, da **GSP-Ausdrücke unterstützt werden** .

Ein GSP-Ausdruck ähnelt einem **JSP-EL-** Ausdruck oder einem **Groovy-GString** und hat die Form `${expr}` :

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

Im Gegensatz zu JSP EL können Sie jedoch einen beliebigen Groovy-Ausdruck im `${..}` -Block verwenden.

Jeder Groovy-Ausdruck kann in allen String-Literalen interpoliert werden, mit Ausnahme von ein- und dreifach zitierten Einzelstrings. Bei der Interpolation wird ein Platzhalter in der Zeichenfolge durch seinen Wert bei der Auswertung der Zeichenfolge ersetzt. Die Platzhalterausdrücke sind von $ {} umgeben oder mit einem vorangestellten $ für punktierte Ausdrücke versehen. Der Ausdruckswert innerhalb des Platzhalters wird für seine Zeichenfolgendarstellung ausgewertet, wenn der GString an eine Methode übergeben wird, die einen String als Argument verwendet, indem toString () für diesen Ausdruck aufgerufen wird.

APS-Tags

Es gibt verschiedene gsp-Tags, mit denen Formulare, Textfelder, Optionsfelder, Kontrollkästchen, if-else usw. erstellt werden können.

<g: if>

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

<g: each>

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

bilden

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

Textfeld

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

Radio

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

Folgen Sie diesem Link für weitere Informationen:
http://docs.grails.org/latest/guide/theWebLayer.html#tags

GSP online lesen: https://riptutorial.com/de/grails/topic/4531/gsp

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit grails	Abdelsalam A. Shahlol, Adeel Ansari, Andrea Zago, bronoman, Burt Beckwith, Community, Farshid Zaker, Graeme Rocher, Jason Bourne, Jesús Iglesias, rahul, saw303
2	Domänenklassen als REST-Ressourcen	Jason Bourne
3	Einsatz	erichelgeson, NachoB, Prakash Thete, saw303
4	GSP	Adeel Ansari, Anshul, audittxl