

Domstolsverket – lagrummet.se

**System Maintainance Document**

**lagrummet.se**

Kontaktperson:  
Mabruk Chowdhury  
telefon 072-50 45 222, mabruk.chowdhury@sogeti.se

# Sammanfattning

Syftet med det här dokumentet är att vara ett stöd för utvecklaren för att snabbt kunna komma igång med underhållsarbete av lagrummet.se. Det ska underlätta förståelse av hur applikationens olika delar fungerar tillsammans, tillvägagångssättet för att deploya applikationen samt hur man löser vanligt förekommande buggar som kan uppstå i utvecklingsmiljön.

För information om systemarkitektur och utvecklingsverktyg, hänvisas dokumentet *System Architecture Document. Översikt över lagrummet.se* (författat av Anders Zakirsson. 2013-01-30) som ska vara ett komplement till kodbasen för lagrummet.se som förklarar hur systemet fungerar och beskriver den grundläggande systemarkitekturen, den teknik och de komponenter som används samt hur de är integrerade.

Dokumentet upprättades i inför beta-lanseringen av lagrummet.se

# Innehållsförteckning

[1 Sammanfattning 1](#_Toc353891825)

[2 Innehållsförteckning 2](#_Toc353891826)

[3 Bakgrund 3](#_Toc353891827)

[4 Uppsättning av utvecklingsmiljön 4](#_Toc353891828)

[4.1 Java SDK 4](#_Toc353891829)

[4.2 Grails 5](#_Toc353891830)

[4.3 Git klient 5](#_Toc353891831)

[4.4 ClassNotFoundException: native2ascii 6](#_Toc353891832)

[5 Konfiguration 7](#_Toc353891833)

[5.1 BootStrap.groovy 7](#_Toc353891834)

[5.2 Config.groovy 7](#_Toc353891835)

[5.3 DataSource.groovy 7](#_Toc353891836)

[5.4 UrlMapping.groovy 7](#_Toc353891837)

[5.5 BuildConfig.groovy, Searchable.groovy 7](#_Toc353891838)

[5.6 Konfigurationsfil i produktionsmiljön 8](#_Toc353891839)

[5.7 Teckenuppsättning i applikationsservern på produktionmiljön 8](#_Toc353891840)

[5.8 Mail 8](#_Toc353891841)

[5.9 Google Analytics 9](#_Toc353891842)

[6 Fundamentala sidor och struktur 10](#_Toc353891843)

[6.1 Sidor 10](#_Toc353891844)

[6.2 Struktur 10](#_Toc353891845)

[7 Sökning 11](#_Toc353891846)

[7.1 Automatisk sökning mha Ajax 11](#_Toc353891847)

[7.2 Manuell sökning mha Sök-knappen 11](#_Toc353891848)

[7.3 Flödesschema för sökning 11](#_Toc353891849)

[7.4 Feltolkningen av specifikt sökvillkor bör åtgärdas 12](#_Toc353891850)

[8 Deploy 13](#_Toc353891851)

[8.1 Lokalt 13](#_Toc353891852)

[8.2 Produktionsmiljö 14](#_Toc353891853)

# Bakgrund

Domstolsverket (DsV) har beslutat att skapa en webbplats, lagrummet.se, där medborgare och rättskunniga ska kunna söka i den svenska rättsinformationen. En grundläggande tankegång som genomsyrar hela projektet är att

*”Som medborgare vill jag förstå vilka lagar/rättsinformation som gäller så att jag kan tillvarata mina rättigheter och skyldigheter”*

Webbplatsen ska vara tillgänglig för alla och att den är indexerbar och länkbar så att den kan fungera som ett nav för svensk rättsinformation på nätet. Detta innebär att

* Webbplatsen följer WCAG 2.0
* Information från Rättsdatalagret presenteras integrerat och är länkbara precis som alla andra sidor
* Sidorna är sökmotoroptimerade (SEO)
* Texter ska vara utformade så att vanliga människor förstår och hittar dem
* Administrationsgränssnittet ska vara intuitivt och ändamålsenligt

# Uppsättning av utvecklingsmiljön

I dokumentet *System Architecture Document. Översikt över lagrummet.se* nämnt i Sammanfattnings-avsnittet, specificeras versioner av ingående komponenter i systemarkitekturen

* Grails 1.3.7
* Java 1.6.0.23
* Spring Security Core 1.2.1
* Grails Searchable 0.6.3
* Grails Mail 1.0
* jQuery 1.6.1.1
* TinyMCE 3.4.4

Lämpligtvist bör man installera Java SDK innan Grails eftersom Grails-installationen kommer att leta efter Java-installation i operativsystemet.

## Java SDK

Ladda ner Java version 1.6.0.23 från *Java SE 6 Downloads*, på URL <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jdk-6u23-oth-JPR>

Beskrivning av installation av Java i Linux miljö finns att läsa på *Ubuntu documentation*, på URL <https://help.ubuntu.com/community/Java> .

Utdrag ur avsnittet *Oracle (Sun) Java 6* i dokumentationen  
(ange rätt versionsnummer nedan)

1. Installera Java JDK

$ chmod u+x jdk-6u34-linux-i586.bin  
$ ./jdk-6u34-linux-i586.bin  
$ sudo mkdir -p /usr/lib/jvm  
$ sudo mv jdk1.6.0\_34 /usr/lib/jvm/  
$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jdk1.6.0\_34/bin/java" 1  
$ sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/lib/jvm/jdk1.6.0\_34/bin/javac" 1  
$ sudo update-alternatives --install "/usr/lib/mozilla/plugins/libjavaplugin.so" "mozilla-javaplugin.so" "/usr/lib/jvm/jdk1.6.0\_34/jre/lib/i386/libnpjp2.so" 1  
$ sudo update-alternatives --install "/usr/bin/javaws" "javaws" "/usr/lib/jvm/jre1.6.0\_34/bin/javaws" 1

$ sudo update-alternatives --config java  
$ sudo update-alternatives --config javac  
$ sudo update-alternatives --config mozilla-javaplugin.so  
$ sudo update-alternatives --config javaws

1. Ange default Java installation

Om Java fanns förinstallerat på operativet, välj rätt version.

$ sudo update-alternatives --config java

## Grails

På sidan *Grails on Ubuntu*, på URL <http://grails.org/download/ubuntu> finns information om hur man laddar ner önskat version av Grails.

Utdrag ur sidan, (ange rätt versionsnummer nedan)

sudo add-apt-repository ppa:groovy-dev/grails  
sudo apt-get update  
sudo apt-get install grails-ppa

#to add grails 1.3.9  
sudo apt-get install grails-1.3.9

#switch between versions  
sudo update-alternatives --config grails

Ett vanligt förekommande fel som kan uppstå är att Grails-installaltionen inte hittar miljövariabeln JAVA\_HOME och skriver ut följande felmeddelande

$ grails JAVA\_HOME not set correctly; cannot execute /usr/lib/jvm/jdk1.6.0\_23/bin/java/bin/java

Undersök först om JAVA\_HOME variabeln existerar och pekar mot korrekt sökväg. Om den existerar men felet kvarstår undersök konfigurationsfilen i Grails installationen. I den definieras var Grails försöker hitta Java installationen och skriver ut felmeddelandet om den misslyckas. Om sökvägen i Grails konfigurationsfil inte överenstämmer med var Java SDK är installerat, ändra i så fall i konfigurationsfilen.

## Git klient

Källkoden ligger som projekt lagrummet.se i GitHub för användaren rinfo. För att enkelt ladda ner, checka-in och checka-ut ändringar i källkoden kan ett grafiskt klient användas.

Ett enkelt sådant verktyg är SmartGitHg från Syntevo. Det finns som Zip-fil att ladda ner från <http://www.syntevo.com/smartgithg/index.html>. Packa upp filen och kör programmet från installtionskatalogen/bin/smartgithg.sh.

## ClassNotFoundException: native2ascii

Ett vanligt fel som kan uppstå när man bygger är java.lang.ClassNotFoundException: sun.tools.native2ascii.Main

Det uppstår om miljövariablen JAVA\_HOME pekar mot JRE och inte JDK. Det åtgärdas genom att peka om variablen till JDK dvs installationskatalogen för Java.

Om problemet kvarstår, kopiera tools.jar från från lib-katalogen under Java SDK installationen till projektets lib-katalog.

# Konfiguration

Under katalogen conf finns konfigurationsfiler som används av Grails och applikationen för att applikationen ska kunna köras.

## BootStrap.groovy

Uppdelad i en utvecklingsmiljö och en icke-utvecklingsmiljö (produktionsdel) och innehåller default värden för att fylla ut komponenter och sidor och definitionen av dessa.

I delen för utvecklingsmiljö definieras bl a en användare till administrationsportalen och dess inloggningsuppgifter.

## Config.groovy

Basfilen för applikationens grundläggande konfiguration. Vissa av konfigurationerna är per utvecklings-, test- och produktionsmiljö.

Här definieras bl a

* Den externa konfigurationsfilen för produktionsmiljön finns i produktionsmiljön.
* De olika URL:en till applikationen i de olika driftsmiljöerna definieras här. Om produktionsmiljön skulle byta server och/eller få ny IP-address, ska det pekas om här.
* Mail-server, Google Analytics, RDL service URL:er definieras också här.

## DataSource.groovy

Här definieras datakälla och inloggningsuppgifter per olika driftsmiljö. Information gällande datakälla i produktionmiljö finns i en egen konfigurationsfil, som specificeras i Config.groovy; dessa används istället för uppgifterna i DataSource.groovy.

## UrlMapping.groovy

Mappar URL anrop mot olika controller och action.

## BuildConfig.groovy, Searchable.groovy

Konfigurationer som vanligtvist inte behövs ändras.

* BuildConfig.groovy innehåller standard inställningar, är modifieringar, för hur Grails applikation byggs.
* Searchable.groovy är konfigurationsfil för Grails Searchable Plugin.

## Konfigurationsfil i produktionsmiljön

I Config.groovy anges filen lagrummet.se-config.groovy som den konfigurationsfil som används i produktionsmiljön och var den filen finns. Den kompletterar och/eller används istället för de produktions-avsnitten i Config.groovy. I lagrummet.se-config.groovy definieras bl a databasinställningar.

## Teckenuppsättning i applikationsservern på produktionmiljön

I konfigurationsfilen web.xml för Tomcat applikationsserver som lagrummet.se driftsätts på i produktionsmiljön, hanteras alla POST-request till servern som med teckenuppsättningen ISO-8859-1, vilket inte stödjer det svenska tecknen ”åäö”.

I web.xml finns nedan avnitt bortkommenterat under rubriken ”Built In Filter Definitions”, ”A filter that sets character encoding that is used to decode parameters in a POST request”

<filter>  
 <filter-name>setCharacterEncodingFilter</filter-name>  
 <filter-class>org.apache.catalina.filters.SetCharacterEncodingFilter</filter-class>  
 <init-param>  
 <param-name>encoding</param-name>  
 <param-value>UTF-8</param-value>  
 </init-param>  
 <async-supported>true</async-supported>  
</filter>

Ta bort kommentaren runt detta och starta om server.

## Mail

Konfiguration för mail-servern definieras i Config.groovy och den körs mot rätt server men när man skickar iväg ett meddelande från lagrummet.se, oavsett produktions eller utvecklingsmiljön, får man följande felmeddelande i logfilen

Error 500: Executing action [contact] of controller [se.lagrummet.PageController] caused exception: Mail server connection failed; nested exception is javax.mail.MessagingException: Could not connect to SMTP host: Mailgw1.dom.se, port: 25; nested exception is: java.net.ConnectException: Connection timed out. Failed messages: javax.mail.MessagingException: Could not connect to SMTP host: Mailgw1.dom.se, port: 25; nested exception is: java.net.ConnectException: Connection timed out

Meddelandet tyder på att lagrummet.se misslyckas med att etablera en anslutning mot mailservern. Man bör undersöka om det är brandvägg eller annan form av skydd hos domstolsverket som förhindrar lagrummet.se att kommunicera med den.

## Google Analytics

Tjänsten Google Analytics kräver att man definierat en sk Web Property ID. Motsvarande nyckel webPropertyId är definierat i Config.groovy men med ett tomt värde. Det riktiga värdet står bortkommenterat och bör sättas inför en skarp lansering av lagrummet.se för att undvika registrera besök under utvecklingsfasen.

# Fundamentala sidor och struktur

Applikationen använder sig av vyer (sidor) som antingen bäddas in i olika sidormallar eller visas på egen sida med egen struktur. De sidor och sidomallar som används definieras i respektive BootStrap.groovy och Config.groovy.

## Sidor

För att underlätta underhåll av lagrummet.se listas nedan de fundamentala sidorna i applikationen och deras syfte.

Vyer/Sidor

* adminMain.gsp är startsidan för administrationssidan.
* extendedSearch.gsp är sökformulär för utökad sökning.
* main.gsp är startsidan.
* searchForm.gsp visar sökresultat för manuell sökning.
* searchResultByCategory.gsp visar sökresultat på en ”Visa fler” sida.
* show.gsp öppnar innehållet i en sökträff man klickat på.

Kontoller

* SearchController.groovy hanterar sökning.
* RinfoController.groovy hanterar hämtning av en sökträff när man klickat på den.

Service

* SearchService.groovy gör HTTP-request för att hämta JSON feed med sökresultat.
* RinfoService.groovy gör HTTP-request för att hämta JSON feed med sökträff man klickat på.

## Struktur

Den grundläggande HTML struktur som används på lagrummet.se finns på sidan main.gsp. Bortsett från ett par få sidor som har egen HTML struktur inkludras övriga sidor/vyer in i main.gsp:s olika sektioner. Strukturen är följande

<body><div id=logo></div><header id=siteHeader></header><div id=content></div><nav id=primaryNavigation></nav></body>

Den mesta dynamiska informationen inkluderas i content-sektionen. Övriga innehåller statisk information och länkar.

# Sökning

En sökning initieras antingen genom att man endast fyllt i sökfältet varvid det görs en Ajax-anrop, eller om man fyllt i sökfältet och tryckt på Sök-knappen, i de båda fallen anropas kontrollern SearchController.groovy. Mappningen för förfrågans URL /search definieras i UrlMappings.groovy.

Hanteringen av resultaten i de olika sätten att initiera en sökning, automatisk och manuellt, görs på respektive sätt. Därför är det viktigt att komma ihåg att en ändring i HTML-koden eller det svar som skickas tillbaka från servern, för presentation av resultatet måste göras på båda ställena.

## Automatisk sökning mha Ajax

När man fyllt i fler än två tecken initieras en automatisk sökning efter 350 millisekunder genom att det sker en Ajax-anrop till SearchController.groovy. Allt eftersom man skriver i sökfältet skickas en ny sökfråga efter 350 millisekunder. Funktionaliteten är definierad i funktionen instantSearch i javascriptfilen application.js. Därefter skapar funktionen all HTML-kod och bäddar in resultat i det och visar det på sidan.

För att kunna navigera tillbaka till sidan för sökträffarna efter att man klickat på en sökträff, lägger funktionen instantSearch till sökparametrarna till URL:en. På det viset får sidan en unik URL som webbläsaren lägger till i sin historik och möjliggör navigering till sidan från en annan sida och att kunna bokmärka sidan.

## Manuell sökning mha Sök-knappen

När man fyllt i sökfältet initieras en automatisk sökning men man kan också välja att trycka på Sök-knappen. Sökningen initieras, sidan laddas om och resultatet presenteras i den HTML-mall som finns på sidan.

Sidan får en unik URL, och man kan därmed navgiera till den från en annan sida och dessutom bokmärka sidan.

## Flödesschema för sökning

En sökning tas emot av kontrollern SearchController. Den paketerar om sökfrågan och anropar tjänsten SearchService som ansvarar för att delegera sökningen till en lokalsök på redaktionellt material och en externsök mot rättsdatalagret RDL. Den slår även samman resultaten från de olika sökningarna och returnerar det till SearchController som i sin tur skickar det resultat till korrekt sida för presentation.

Klassen RdlSearchService gör HTTP-request mot rättsdatalagret. För att debugga exakt vad som skickas iväg och tas emot i kommunikationen mellan lagrummet.se och RDL är funktionen searchWithQuery rätt plats att sätta brytpunkter eller skriva logutskrifter.

Enkelt flödesschema för sökning

1. Ett sökord fylls i och sökning initieras (automatisk eller manuell).
2. Sökningen tas emot av funktionen index i SearchController.
3. SearchController paketerar om sökordet och anropar tjänsten SearchService.
4. SearchService initierar lokalsökning mha tjänsten LocalSearchService och externsökning mot rättsdatalagret RDL mha tjänsten RdlSearchService.
5. Funktionerna plainTextSearch i RdlSearchService gör HTTP-request mot RDL.
6. Resultat från lokal och extern sökning slås samman av SearchService och returneras till SearchController.
7. SearchController returnerar till resultat till korrekt sida (om det är en automatisk sökning visas resultatet på samma sida, annars visas det på en annan sida), se avsnitten 6.1 och 6.2.

## Feltolkningen av specifikt sökvillkor bör åtgärdas

Allt som skrivs i sökfältet skickas iväg till server för sökning. Men om man skriver ett sökord innehåller kolon som t ex ”1998:144” hanteras det av rättsdatalagret genom att separera sökordet på kolon-tecknet, ock resultera i en sökning på både ”1998” och/eller ”144”, vilket är en feltolkning av sökordet.

# Deploy

För att köra applikationen måste man först bygga den för rätt målmiljö, antingen för development eller production. Beroende på målmiljön används under bygget respektive konfigurationsfil som finns definerade i Config.groovy. Om man inte bygger för rätt målmiljö kommer applikationen inte kunna köras och man kan få fel som t ex SQLNestedException: Cannot create JDBC driver of class 'org.hsqldb.jdbcDriver' for connect URL 'jdbc:mysql:

Målmiljö sätts genom att markera projektet, välja Properties, och Active Grails Environment och där efter Development, Production eller Test.

Under utveckling kan man vanligtvist välja köra applikationen direkt från sin integrerade utvecklingsmiljö (IDE) om den är Eclipse eller Netbeans. I samband med att man laddar ner någon av utav dessa kan man välja om applikationsserver ska ingå i installtionspaketet, i vilket fall det är GlassFish eller Tomcat som installeras. I produktionsmiljön körs applikationen i en Tomcat applikationsserver.

Under bygget skrivs det ut i konsolen vilken målmiljö som är valt.

## Lokalt

Ifall man vill köra applikationen i en Tomcat applikationsserver för att efterlikna den på produktionsmiljön men Tomcat inte finns installerad i såfall installera den först, instruktioner finns på *Ubuntu documentation*, på URL <https://help.ubuntu.com/10.04/serverguide/tomcat.html>

För att installera Tomcat på Ubuntu, kör kommandot i ett terminalfönster och följ anvisningarna

sudo apt-get install tomcat6

För att köra applikationen följ anvisningarna (ange rätt sökväg till din Tomcat installation)

1. Ange målmiljö Active Grails Environment som Development.
2. Bygg projektet varvid en war fil produceras.
3. Kör igång Tomcat applikationsserver
   1. cd /etc/init.d
   2. sudo ./tomcat6 start
4. Starta utskrift av logfilen genom för att följa installationsförloppet
   1. $ tail –f /var/log/tomcat6/catalina.out –n 100
5. Kopiera filen lagrummet.se-x.x.x.war från lagrummet.se/target/ till /var/lib/tomcat6/webapps/

Öppna en webbläsare och ange URL:en till applikationen, definierad i Config.groovy, <http://localhost:8080/lagrummet.se>

## Produktionsmiljö

I Config.groovy anges den lokala konfigurationsfil som läses in från produktionsmiljön.

Viktiga filer och katalog i produktionsmiljön

* /etc/lagrummet.se/lagrummet.se-config.groovy innehåller applikationens konfigurationer för produktionsmiljön.
* /opt/tomcat/webapps är deploy katalogen för produktion
* /opt/tomcat/logs/catalina.out är logfilen för serverns utskrifter, innehåller händelseförloppet för applikationen

Följ anvisningarna för att deploya lagrummet.se

1. Ange målmiljö Active Grails Environment som Production
2. Bygg projektet, war fil produceras
3. Kopiera war filen till produktionsmiljön beta.lagrummet.se.. Observera ”kolon och punkt” efter domännamnet, för att kopiera till användarens hemkatalog.
   1. scp lagrummet.se/target/lagrummet.se-x.x.x.war [rinfo@beta.lagrummet.se](mailto:rinfo@beta.lagrummet.se):.
   2. Ange lösenordet
4. Logga in på beta.lagrummet.se
   1. ssh [rinfo@beta.lagrummet.se](mailto:rinfo@beta.lagrummet.se)
   2. Ange lösenordet
5. Döp om tidigare ROOT.war (om sådan finns) till ROOT.war.bak# (löpnummer) och sedna döp om filen lagrummet.se-x.x.x.war till ROOT.war.
6. Starta utskrift av logfilen genom för att följa installationsförloppet
   1. $ tail –f /opt/tomcat/logs/catalina.out –n 100
7. Kopiera ROOT.war till /opt/tomcat/webapps/

Öppna en webbläsare och ange URL:en till applikationen, <http://beta.lagrummet.se> .