

Domstolsverket – lagrummet.se

**System Architecture Document**

**Översikt över lagrummet.se**

Kontaktperson:  
Anders Zakrisson  
telefon 076-102 25 36, anders.zakrisson@sogeti.se

# Sammanfattning

Syftet med det här dokumentet är att vara ett komplement till kodbasen för lagrummet.se för att kunna förklara hur systemet fungerar. Det beskriver den grundläggande systemarkitekturen, den teknik och de komponenter som används samt hur de är integrerade.

# Innehållsförteckning

[1 Sammanfattning 1](#_Toc315769103)

[2 Innehållsförteckning 2](#_Toc315769104)

[3 Bakgrund 3](#_Toc315769105)

[4 Grundläggande systemarkitektur 4](#_Toc315769106)

[5 Grails – Convention over configuration 5](#_Toc315769107)

[6 Content Management System 6](#_Toc315769108)

[7 Sök 7](#_Toc315769109)

[8 Admin 8](#_Toc315769110)

[8.1 Sidor 8](#_Toc315769111)

[8.2 Sidmallar 8](#_Toc315769112)

[8.3 Filter 8](#_Toc315769113)

[8.4 SiteProperties 8](#_Toc315769114)

[8.5 Bilder och filer 9](#_Toc315769115)

[9 Bilagor 10](#_Toc315769116)

# 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”*

Det innebär att 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.

# Grundläggande systemarkitektur

Lagrummet.se är byggt som en standardbaserad webbplats i det Groovy-baserade MVC-ramverket Grails. I grunden är nya lagrummet.se ett skräddarsytt CMS-system anpassat för att sömlöst kunna söka och visa information från Rättsdatalagret (RDL).

Groovy är ett objektorienterat, scriptat språk som körs på Java-plattformen. Det kompileras dynamiskt och använder en syntax baserad kring en naturligare och mindre formell variant av Java. Java går att skriva och använda direkt (vilket gör att Java-utvecklare fort kan sätta sig in i koden), men ren Groovy är ofta mycket effektivare, kompaktare och mer lättöverskådligt.

Grails bygger på konvention över konfiguration, det betyder att MVC-modellen (Model-View-Controller) definieras i en namn- och katalogstruktur och undviker konfigurationsfiler till största möjliga mån. Domänobjekt skapas och lagras i en ORM-modell (Object Relational Mapping) som automatiskt hanterar relationer, mappning och lagring i databasen, det gör att objekten och deras relationer är överskådliga direkt i koden. Det medför att inga objektkartor har skapats eller används då domänobjekten kontinuerligt uppdateras under arbetets gång och förväntas även göra det under förvaltningsarbetet.

Systemet driftsätts som en vanlig Java-webbapplikation (man producerar en .war-fil) som man sedan kör i en applikationsserver, förutom denna server behövs det endast tillgång till en sql-databas. Det gör att driftsmiljön är mycket flexibel och kan placeras på många olika typer av lösningar, allt från dedikerade servrar till virtuella maskiner och olika typer av molnlösningar.

Versioner av ingående komponenter

* 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

# Grails – Convention over configuration

Grails använder sig av ett system där placering och namngivning av filer används i så stor utsträckning som möjligt istället för konfigurationsfiler. Domänobjekt skapas och lagras i en modell vid namn GORM (Grails Object Relational Mapping), mer om hur det fungerar finns att läsa i dokumentationen till Grails 1.3.7. De viktigaste katalogerna för MVC-modellen är

* /grails-app/domain, varje domänobjekt sparas i en egen .groovy-fil
* /grails-app/controllers, controller till varje domänobjekt ligger i DomänobjektController.groovy
* /grails-app/views, varje controller har i sin tur en egen katalog under /views med vyer som matchar till de olika actions som finns i controllern

Det finns fler kataloger av intresse, bland annat under /web-app där bilder, css och JavaScript till vyerna ligger.

Trots den generella uppbyggnaden kring konvention finns det ett par tillfällen när man behöver skriva och definiera konfigueringar explicit och till det finns det ett par filer som ligger under mappen /grails-app/conf. I samtliga dessa filer kan man särskilja på konfigurationer i olika målmiljöer (utveckling, test och produktion).

* UrlMappings.groovy definierar hur url’er mappas mot controllers och actions
* Bootstrap.groovy skapar data till en ren start av applikationen som till exempel sidor och användare
* DataSource.groovy tillhandahåller inställningar för databasen, användarnamn och lösenord för produktion finns dock i en extern konfigurationsfil som inte versionshanteras
* Config.groovy har hand om applikations-generella konfigurationer och specifika inställningar för lagrummet.se . Den externa konfigurationsfilen för produktionsmiljön definieras även här

# Content Management System

I grunden för CMS-systemet finns objekt som kallas Pages (sidor), dessa innehåller data som sedan renderas till en HTML-sida. Förutom dessa sidor finns det Users (användare) och ett objekt som kallas SiteProperties där allmänna inställningar för sajten görs.

Varje sida har en permalink och eventuellt en förälder som tillsammans bestämmer den URL just den sidan får, max tre nivåer på sidor kan skapas. En sida har även innehåll (content), rubrik (h1) och title, eventuella barn (children) samt metainformation för att hantera publicering och liknande.

När en sida uppdateras skapas en kopia som sedan kan återställas, dessa kopior innehåller inte några relationer till andra sidor (parent och children) utan ses endast som snapshots av det innehåll en sida innehåll vid en viss tid.

# Sök

En sökning initieras via en förfrågan till URL’en /search (via UrlMappings) som då skickar vidare förfrågan till kontrollern SearchController.groovy. Den kontrollern är inte kopplad till någon domänmodell men använder sig däremot av en service vid namn SearchService som i sin tur fungerar som ett mellanlager för sökningar internt på sidan (som går via localSearchService) och externt mot RDL (rdlSearchService).

RdlSearchService använder sig av ett par hjälpklasser som ligger under src/groovy/se/lagrummet (bland annat SearchResult) för att ordna sökresultatet från RDL i kategorier och ordna till textutdrag.

# Admin

Admin-modulen hanterar dels sidor men även användare, media och objektet SiteProperties som är genensamt för hela applikationen. Där ställer man in information av typen sidfot, länkar i toppmenyn och sajt-titel.

Beroende på typ av användare finns det olika möjligheter i admin-gränssnittet, en redaktör kan bara ändra sidinnehåll och media medan en administratör även kan hantera användare och SiteProperties.

## Sidor

En sida består av innehåll, rubrik, publiceringsstatus, länk och så vidare. Länken måste vara unik, när man hämtar en sida mappar man url mot attributet permalink på en sida, hittas ingen sida får man en 404. En sida kan ha ett eller flera barn, dvs. undersidor. En sida kan också vara av typen ”metasida” vilket betyder att den ligger i trädstrukturen som vilken annan sida som helst (och kan både ha förälder och egna barn) men den visas inte för användare ute på sajten.

När en sida först skapas är den antingen en grundsida (det vill säga den har ingen förälder) eller en undersida. Dess sorteringsordning bestäms utifrån hur den placeras i sidträdet i admin-modulen då parameters sortOrder uppdateras via Ajax (i admin.js).

## Sidmallar

Sidor är definierade till att använda sig av en sidmall. Den pekar på en speficik vy (under views/pages) som sedan används för att visa sidan och göra eventuella tillägg eller modifieringar av data i PageController (i metoden ”show”). Gör man inget aktivt val för sidan en standardmall (”default”) och använder sig då show.gsp som vy.

För att skapa en ny sidmall skapar man en ny vy i views/page som heter någonting på formen ”ny-sidmall.gsp” och sedan lägger man till ett nytt element i den konfiguration som visar sidmallar i adminläget (lagrummet.page.templates i conf/Config.groovy). Det elementet ska då vara ”sidmallsnamn”:”beskrivning” där beskrivning är det som kommer stå i drop-downen i admin-läget för sidan. Det blir till exempel ”ny-sidmall”:”En ny sidmall”. Ett undantag är för sidmallen frontpage där det för en sida som har denna inte visas någon drop-down i redigeringsläget för den sidan.

## Filter

För att hämta information i form av objekt som är gemensamma för alla sidor i Admin-modulen finns två filter definierade, SitePropertiesFilter och PageTreeFilters.

## SiteProperties

Sidfot, sidhuvud, de val som finns i den utfallande menyn och liknande är alla samlade i ett domänobjekt av typen SiteProperties. Genom att ändra den information som finns i detta objekt kan man därigenom dynamiskt ändra hur vissa saker ser ut på sajten under drift.

Den utfallande menyn hanteras i attributet searchCats. Det är en lista med de kategorier som menyn visar, dessa kategorier mappas direkt mot hur sökresultat grupperas i sök. Därför finns det inget enkelt sätt att lägga till fler kategorier, för att de ska få effekt måste man även ändra searchService och Category.groovy som hanterar grupperingen och sedan messages.properties som innehåller namnen och beskrivningen till kategorierna (category.X och category.description.X).

## Bilder och filer

Media-objekt sparas i databasen för att undvika problem vid nya deploys av .war-filen på applikationsservern. I samband med deploy tas alla gamla kataloger bort som tillhör applikationen och då försvinner även de bilder som läggs upp om de ligger i underkataloger (vilket de behöver göra av säkerhetsskäl). Det går att komma runt detta genom att sätta upp en till webservice via en Apache webbserver som tar hand om inkommande och filer men det kräver en lite större insats på servern.