

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

Nya lagrummet.se är i grunden ett Content Management System (CMS) med en nära integrering mot Rättsdatalagret (RDL). Det redaktionella innehållet ligger som sidor i lagrummet.se’s databas medans rättsinformation hämtas dynamiskt från RDL och sedan införlivas direkt på en sida.

## Pages

I grunden för CMS-systemet finns objekt som kallas *Pages* (sidor), dessa är organiserade i en trädstruktur genom relationerna children och parent, andra relationer är mediaobjekt (*Media*) och puffar (*Puff*) som också kan vara kopplade till en sida.

När en sida uppdateras skapas en kopia (via domänobjektsfunktionen *Page.backup()*) som sedan kan återställas (via relationen *autoSaves/masterRevision*), dessa kopior innehåller inte några relationer till andra sidor eller objekt utan ses endast som ett snapshot av innehållet på en sida för en viss revision. När en tidigare revision återställs (via metoden *restore()* i /grails-app/controllers/se/lagrummet/PageController.gsp) kopieras det innehållet till den sida som för närvarande är den aktuella versionen för att behålla samtliga kopplingar.

Varje sida har ett antal egenskaper, från sidordning till rubrik, innehåll, title och datum när de skapades, senast uppdaterades och som hanterar dess publicering. Varje sida måste även ha en unik *permalink*, när en URL efterfrågas används metoden *show()* i controllern till sidorna för att se om det finns någon sida med permalinken och om den finns kolla om den är publicerad för tillfället och i sådant fall hämta rätt vy (i /grails-app/views/page) för den sidmallen som den sidan använder sig av.

## Meta-sidor

Det finns även en property (*metaPage*) för att ställa in att sidan ska vara en så kallad meta-sida som inte är publicerad (eller ens har något innehåll) men som fungerar som en menyavdelare både i admin-gränssnittet och på den externa webbsajten.

## Sidmallar

Ett koncept med sidmallar (propertyn *template*) används för att anpassa hur en sida renderas. Normalt används vyn show.gsp enligt den konvention och MVC-modell som Grails bygger på, men istället kan alltså en annan sidmall skapas som en separat vy (i /grails-app/views/page) som sedan laddas istället. Systemet bygger fortfarande på konvention, controllern söker upp den vy som heter template.gsp och vill man lägga till nya sidmallar räcker det med att man kopierar en gammal, döper den rätt och sedan lägger till ett nytt villkor i *show()*-metoden i *PageController*.

# 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, puffar, synonymer och objektet SiteProperties där genensam information för hela applikationen ställs in.

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.

## 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.

# Front-end och webbläsare

Applikationen använder sig av HTML5 och CSS3 enligt metoden *graceful degradation*, det betyder att nyare webbläsare får ta full del av den utveckling som har skett i webbstacken under andra halvan av 00-talet samtidigt som den användarmässigt fungerar lika bra även i äldre browsers (från Internet Explorer 7).

Det har lagts särskild vikt vid att få HTML-koden semantisk, sidorna snabba att ladda, hela sajten sökmotoroptimerad samt tillgänglig enligt WCAG 2.0 och E-delegationens Vägledning för webbutveckling.

## CSS

En huvudfil för css-styling har använts (/web-app/css/main.css) för att hålla nere antalet anrop. För Internet Explorer tidigare än version 9 laddas det in en extra css-fil(/web-app/css/ie.css) för att lösa kompatibilitetsproblem och en js-fil (html5shiv) som skapar grunden för de nya HTML5-elementen så att dessa även fungerar.

## Kompatibilitet

Under utvecklingsarbetet har sajten testats och anpassats för

* Internet Explorer 7, 8 och 9
* Firefox 10 och 11
* Google Chrome 17

Senare versioner av dessa browsers samt övriga browsers som följer W3C’s rekommendationer ska fungera bra.

## Responsiv design

Sajten är förberedd för att sömlöst kunna anpassas även till mobila enheter genom tekniken *Responsive design*. Den går ut på att man med hjälp av specifik styling för olika skärmstorlekar och upplösningar anpassar sidans layout, styling och interaktion till att passa andra typer av enheter än persondatorer.

Då implementationen nedprioriterades under utvecklingsprojektet är dock det faktiska genomförandet bortkommenterat i koden, det är klart till största delen men är otestat. Med relativt små medel kan det slås igång genom att

* I layoutmallarna (/grails-app/views/\*.gsp) ta bort html-kommentarerna kring länken med id mobileNavLink
* I /web-app/css/main.css ta bort kommentaren kring sektionen med media-query