
VINCIT
GRAILS

ANTTI LOPONEN

Moro!

- FM, Tampereen Yliopisto, Tiedonkäsittely
- 2008 Ensikosketus Grailsiin
- 2009-2014 Verkkokauppakehitystä Grailsilla
- 2014 → Passionate Software Developer
  @ Vincit

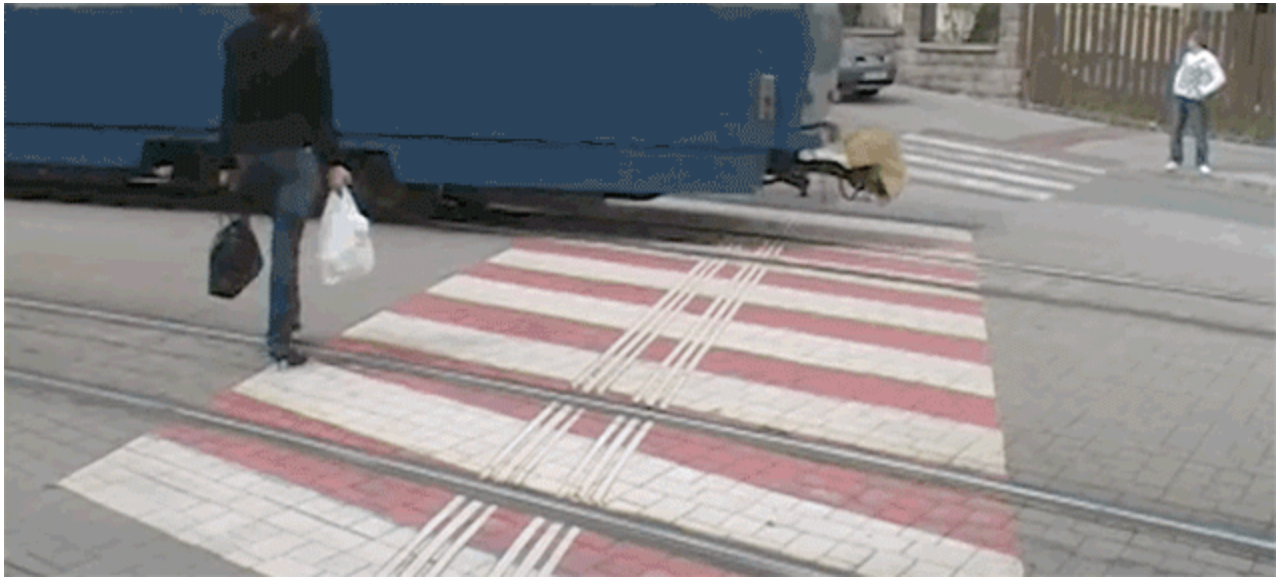
GRAILS

VINCIT

www.vincit.fi

GRAILS

Rails-ajattelu



GRAILS

Demo kertoo enemmän kuin 1000 sanaa

VINCIT

www.vincit.fi

GRAILS

Käsitteet

- Framework
- MVC
- Coding by convention
- JVM
- Pluginit
- Open Source

GRAILS

Komponentit

- Spring
- Groovy
- Hibernate
- Sitemesh

- komentorivityökalu
- gvm

GRAILS

Historia

- 2005 Groovy on Rails
- 2006 → GRAILS
- 2008 v 1.0
- 2008 SpringSource
- 2011 v 2.0
- 2015 OCI
- 3/2015 v 3.0

GRAILS

Tilastojen mukaan

- 7% osuus Spring-sovelluksista: http://zeroturnaround.com/rebellabs/top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/
- Aktiivinen
  http://vitalflux.com/java-top-10-java-based-web-development-frameworks-2014-2015/
- Heuristisen arvion voittaja: http://zeroturnaround.com/rebellabs/the-curious-coders-java-web-frameworks-comparison-spring-mvc-grails-vaadin-gwt-wicket-play-struts-and-jsf/10/

GRAILS

Muuta

- Pääkehittäjä Graeme Roecher

https://twitter.com/graemerocher

- www.grails.org

GROOVY

VINCIT

www.vincit.fi

GROOVY

Perusteet

- Javan skriptikieli
- 2007 v. 1.0
- Javan lisäksi ominaisuuksia esim. Pythonista ja Perlistä

GROOVY

Syntaksi

- Ei puolipistettä
- Myös Java-syntaksi on useimmiten validia
  - paitsi do-while-silmukka

GROOVY

Dynaaminen tyypitys

```
def variable = true
variable = 1
variable = “Antti”
variable = new Product()
variable = [“Grails”, “Groovy”]
```

Suosittelen kuitenkin vahvaa tyypitystä suorituskyvyn vuoksi

GROOVY

Kokoelmien käsittely

- Listaan lisääminen

```
def list = []
list << “Groovy”
```

- Propertyt suoraan käytettävissä

```
if (companies.employees.contains(employee))
...
```

- Each-silmukka

```
company.employees.each({employee -> {
        println employee.name
    }
})
```

- Osajoukot

```
List twoLastEmployees = company.employees[-2..-1]
```

- Haku

```
List administrativeEmployees = company.employees
    .findAll({it.department == Departments.ADMINISTRATION})
```

GROOVY

Operaattorit

- Vältetään NullPointerException
- `person?.address?.zipCode`

- Muutos koskee kaikkia
- `company.employees*.name.toUpperCase()`

GROOVY

Tulostus

- println()
- vastaa javan System.out.println()
- Objekteista on hyvät toString-esitykset

MVC

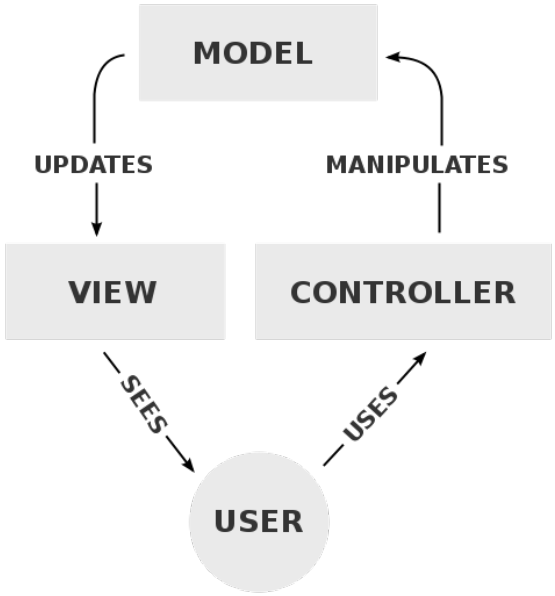
VINCIT

www.vincit.fi

MVC

Model - View - Controller

- Ohjelmistoarkkitehtuurimalli



MVC

Model - View - Controller

- Model
  - Tietomalli
  - Tietokantataulu
  - Business-logiikka
- View
  - Käyttöliittymä
  - Esittää Modelin
- Controller
  - Käyttäjän rajapinta tietomalliin
  - Käsittelee Modelia

MVC

Pros and Cons

+ Modulaarisuus
  + Uudelleenkäytettävyys
  + Testattavuus
+ Bisneslogiikka erotettuna UI:sta
- Tiedon kuvaus jakautuu eri komponentteihin
  - Työläs etenkin pienissä projekteissa
  - Backend ja frontend “liian kaukana toisistaan”

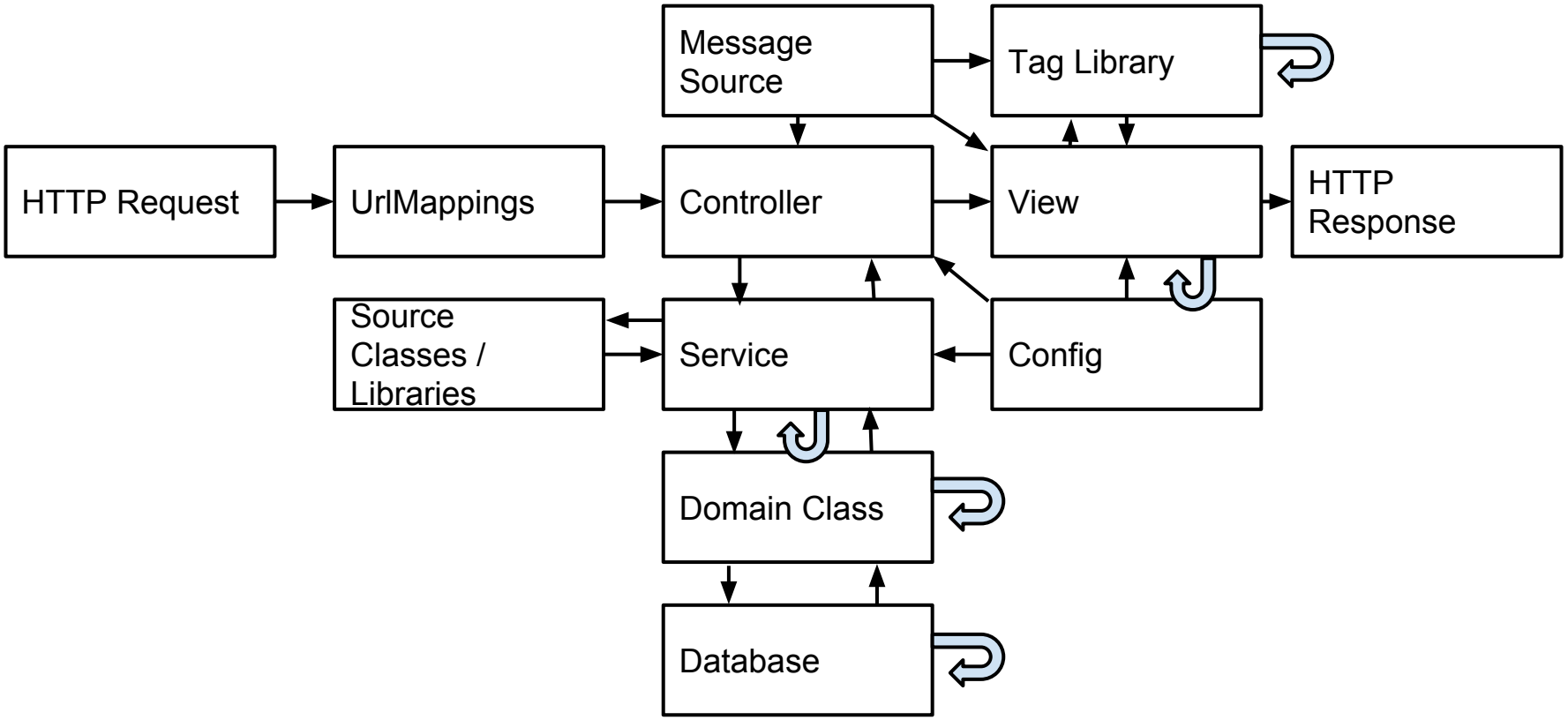
GRAILS

VINCIT

www.vincit.fi

GRAILS

Arkkitehtuurimalli



HARJOITUS

Harjoitussovellus

- Harjoitussovellus: tietokonerekisteri
- Tallennetaan yrityksen tietokoneiden tekniset tiedot sekä käyttäjät

```
grails create-app computer-inventory
```

DOMAIN-LUOKAT

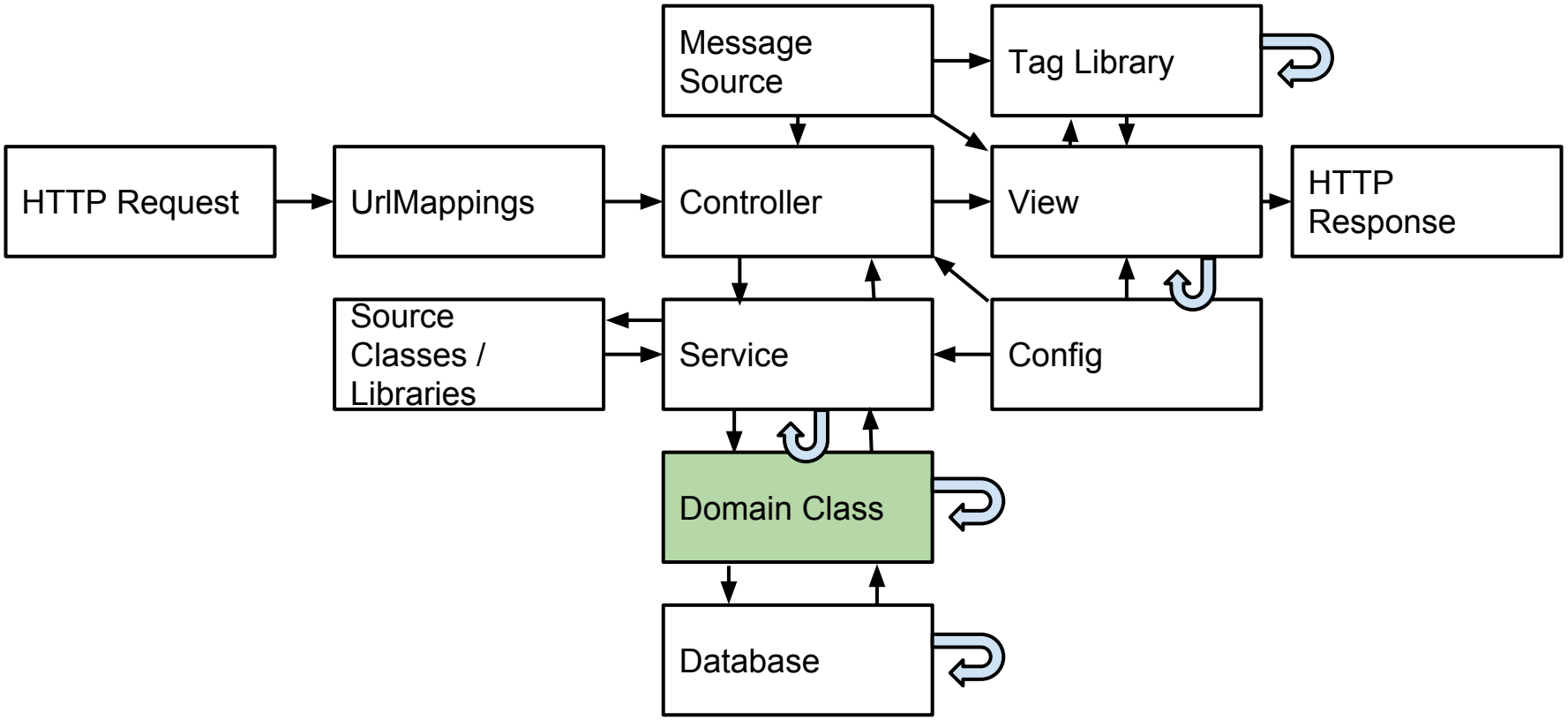
MVC-Model

VINCIT

www.vincit.fi

GRAILS

MVC Model: Domain Class



DOMAIN CLASS

Perusteet

- Objektimallinnus
- Tietokantataulun malli
- Rajapinta tietokantaan (“Entity”, “Repository”)
  - GORM-metodit
- Rajapinta näkymään (“DTO”)
- Vahva tietotyypitys
- Taulut suoraan tietokantaan
  - Nimi tulee classin nimestä
    - camelCase -> underscore_case
  - Id on automaattinen

DOMAIN CLASS

Konstruktori

- Automaattinen konstruktori ottaa Mapin parametrina

```
Map values = [name: “Antti”, company: “Vincit”]
Person person = new Person(values)
```

DOMAIN CLASS

Aksessorit

- Arvoja voi asettaa suoraan ilman settereitä

```
person.age = 31
```

- Arvoja voi lukea suoraan ilman gettereitä

```
println person.age
```

- Aksessorit voi luoda, jos niistä on hyötyä

```
void setAge(int age) {
     isUnderAge = age < 18
     this.age = age
}

int getBonus() {
     worksFullHours ? bonus : bonus * 0.5
}
```

DOMAIN CLASS

Liitos toiseen domainiin

- Lisätään toinen domain tietotyyppinä

```
class Company {
    String name
    Person owner
    ...
}
```

company-tauluun tulee owner_id-kenttä viittauksena person-tauluun

DOMAIN CLASS

Kokoelmaliitokset - associations

- Tietokantaan automaattiset viittaukset
  - 1:N-tapauksessa id-viittaus
  - N:M-tapauksessa liitostaulu

Office-luokka:

```
static hasMany = [rooms: Room]
```

Room-luokka:

```
static belongsTo = [office: Office]
```

Omistussuhde → toimiston poistaminen poistaa kannasta myös huoneet

HARJOITUS

Domain-luokat

Tee domain-luokat yrityksen tietokonerekisteriä varten. Tietokoneista talletettavia tietoja:

- Valmistaja
- Malli
- Tyyppi (kannettava, pöytäkone)
- Käyttöjärjestelmätyyppi(Windows, OS X, Linux)
- Käyttöjärjestelmän versio
- Prosessorin kellotaajuus GHz
- Muistin määrä GB
- Sarjanumero

HARJOITUS

Domain-luokat

- Lisätään käyttäjärekisteri tietokoneista.
- Työntekijöistä riittää tallentaa käyttäjänimi, etunimi, sukunimi, hetu ja osasto (Administration, Production tai Research).
- Rekisteriin merkitään, milloin käyttäjä otti tietokoneen käyttöön ja milloin lopetti sen käytön

DOMAIN CLASS

Transientit arvot

- Kaikkia kenttiä ei tarvitse tallentaa tietokantaan
  - yhdistelmä kahta kenttää
  - muunnelma kentästä
- Getteri ja lisäys transients-listaan

```
String getFullName() {
    return firstName + " " + lastName
}
static transients = ["fullName"]
Person person = new Person(firstName: "Antti", lastName: "Loponen")
println person.fullName
```

HARJOITUS

Transientit

Tee tietokone-luokalle transient-muuttuja nimeltä *information*, joka sisältää sen merkin ja mallin, sekä käyttöjärjestelmän tyypin ja version.

Esim. Apple MacBookPro, OSX Mavericks

DOMAIN CLASS

Rajoitteet = constraints

- Validoidaan objektin tietosisältö
  - Saako kenttä olla null?
  - Saako numero olla negatiivinen?
  - Saako merkkijono olla tyhjä?
  - Saako sama arvo olla kahdella objektilla?
- Rajoitteet suoraan tietokantaan

```
static constraints = {
    name (nullable: false, blank: false)
    position (nullable: false, blank: true)
    age (min: 0)
    employeeId (unique: true)
}
```

http://grails.github.io/grails-doc/2.5.0/ref/Constraints/blank.html

HARJOITUS

Constraintit

Luonnostele tietokoneelle järkeviä constraintteja
Mikä arvo ei saa olla tyhjä?
Mikä arvo ei saa toistua?

DOMAIN CLASS

Custom Validator

- Validointi voi olla myös tarkemmin määriteltyä
  - validator-closure jonka palautusarvo määrittää tuloksen

```
static constraints = {
    zipCode(validator: {val ->
        return val ==~ /\d{5}/
    })
}
```

http://grails.github.io/grails-doc/2.5.0/ref/Constraints/blank.html

HARJOITUS

Constraintit

Tee henkilötunnukselle custom-validator, joka tarkistaa että

- kuusi ensimmäistä on numeroita
- loput viisi mitä tahansa merkkiä
- (jos ehdit, niin määrittele tarkemmin loput viisi)

BOOTSTRAP

VINCIT

www.vincit.fi

BOOTSTRAP

- Käynnistyksen yhteydessä alustettava data
- Kätevä kehittämiseen
- Bootstrap.groovy
  - init
  - destroy

HARJOITUS

- Luo tietokone ja käyttäjä ja aseta käyttäjä käyttämään tietokonetta
- Tulosta tietokoneen transientti informaatio
- Tulosta myös käyttäjä

SERVICET

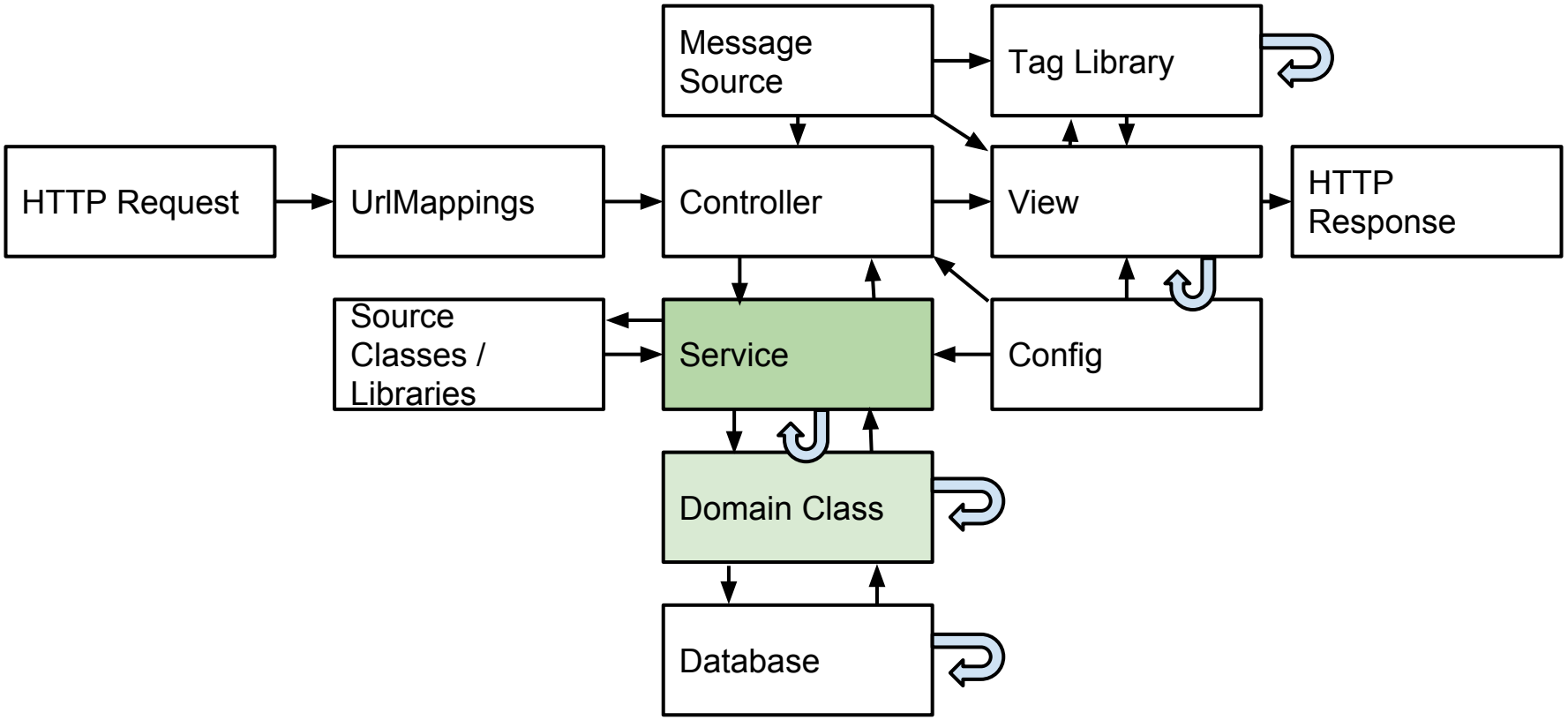
MVC-MODEL

VINCIT

www.vincit.fi

GRAILS

MVC Model: Service



SERVICE

Perusteet

- Business-logiikkaa suorittava metodi
  - Käsittelee domain-objekteja
  - Kutsuu ulkoisia rajapintoja
- Erillään Grails-kehyksestä
- “Rajapinta” Controllerin ja Domain Classin välillä
- “Rajapinta” tietokantaan
- Logiikka erillään Controllereista, miksi?
- Service voi käyttää toista serviceä
  - HUOM: ei ristikkäin

SERVICE

Dependency injection

- Servicen injektointi controlleriin tai toiseen serviceen on helppoa:

```
def productService
```

SERVICE

Transaktionaalisuus

- transactional-servicemetodi suoritetaan tietokantatransaktion puitteissa
  - Poikkeuksen sattuessa transaktio perutaan
- Oletuksena true
- Voi myös asettaa metodikohtaisesti annotaatiolla @Transactional
- Voi myös käyttää lohkoa withTransaction metodin sisällä

```
static transactional = false
```

SERVICE

Scope

- kuinka monta instanssia servicestä tarvitaan?
- Oletuksena singleton, eli yksi per sovellus
  - Servicen muuttujat ylikirjoitetaan joka suorituskerralla
- Scope voi olla myös esim. request- tai sessiokohtainen, jolloin sen muuttujiin voi tallentaa tietoja

```
static scope = “session”
```

HARJOITUS

Servicet

Millaisia service-luokkia esimerkkisovelluksessa tarvitaan? Millaisia metodeja niissä tarvitaan?

TIETOKANTA

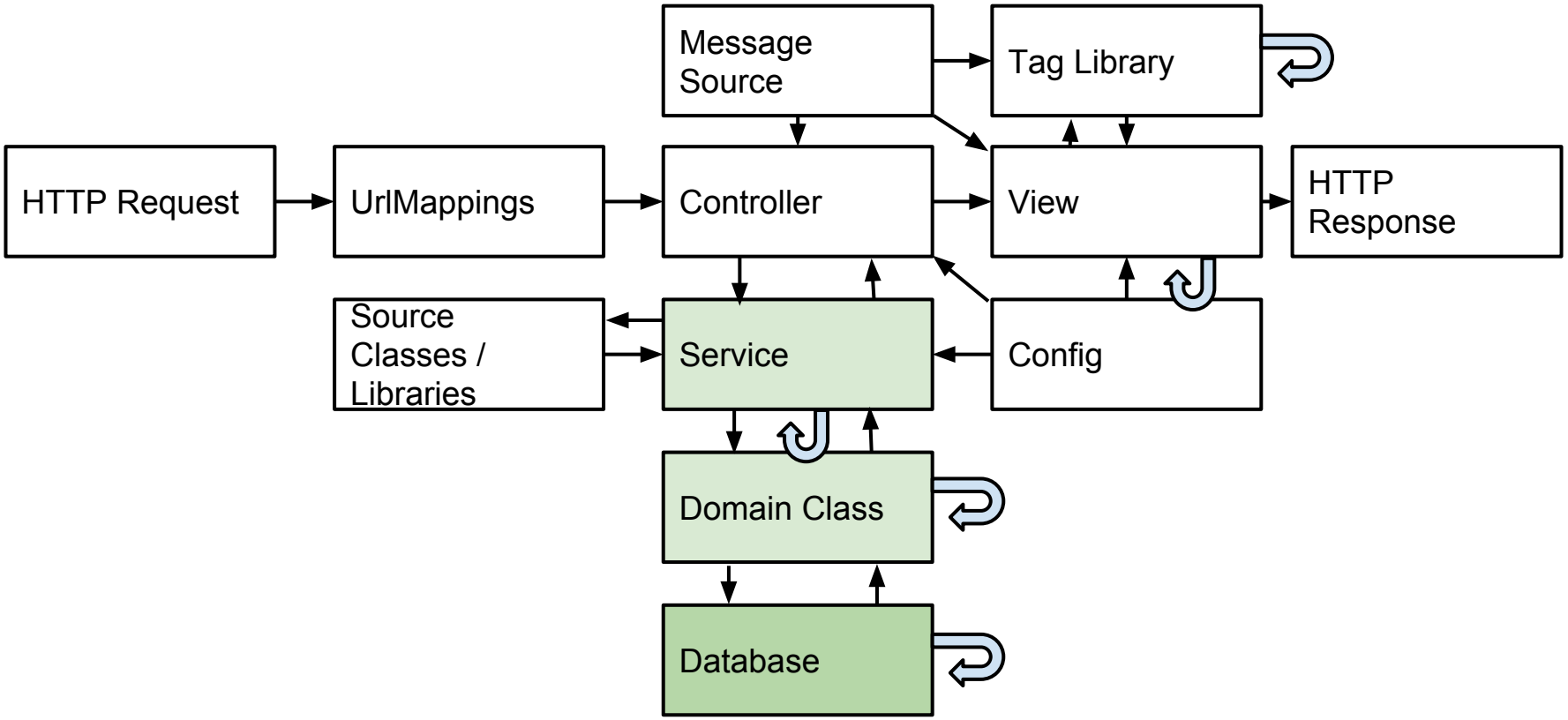
MVC-MODEL

VINCIT

www.vincit.fi

GRAILS

MVC Model: Database



GRAILS GORM

Perusteet

- Yksinkertainen rajapinta tietokantaan
- Hakumetodit suoraan domain-luokalla

```
Product.get(id)
Product.list()
Product.findByName(name)
Product.findAllByIsOnline()
Product.findByNameAndIsOnline(name)
```

TALLENNUS

- Tallennusmetodit suoraan instansseilla
- `validate()` kertoo, täyttääkö olio rajoitteiden asettamat vaatimukset
- `save()` epäonnistuu, jos olio ei täytä rajoitteiden asettamia vaatimuksia

Täten oliota voidaan tallentaa näin:

```
if (!product.save()) {
    println("Error saving product");
    ...
}
```

POISTAMINEN

```
product.delete()
```

Aiheuttaa poikkeuksen, jos tallennus ei onnistu esim. omistussuhteen vuoksi

FLUSH

- Hibernate suorittaa operaatiot kantaan vasta, kun sen on pakko.
- Antamalla parametri `[flush: true]` save- tai delete-metodille, suoritetaan operaatio välittömästi

PERSISTENCY LISTENERS

- Tapahtumia ennen tai jälkeen tietokantaoperaation
- beforeInsert, beforeUpdate, beforeDelete, beforeValidate, afterInsert, afterUpdate, afterDelete, onLoad

```
def afterInsert() {
    println("Saved succesfully with id ${id}")
}
```

https://grails.github.io/grails-doc/latest/guide/GORM.html#eventsAutoTimestamping

AUTOMATIC TIMESTAMPING

Automaattisesti päivittyvät päivämääräkentät

```
Date dateCreated
Date lastUpdated
```

https://grails.github.io/grails-doc/latest/guide/GORM.html#eventsAutoTimestamping

HIBERNATE CRITERIA BUILDER

- Helppo tapa tehdä tietokantakyselyjä

```
List results = Product.createCriteria().list {
    and {
        between("price", priceMin, priceMax)
        like("name", "%" + query + "%")
    }
    eq("productType", selectedProductType)
}
```

HUOM: älä käytä muuttujissa samoja nimiä kuin propertyjen nimissä
Myös muita varattuja sanoja (version, user, id, jne)

HIBERNATE CRITERIA BUILDER

Sivutus

- Jos tuloksia on paljon, kannattaa käyttää sivutusta

```
List results = Product.createCriteria().list (max: paramsMax, offset:
offsetMax) {
    and {
        between(“price”, priceMin, priceMax)
        like(“name”, “%” + query + “%”)
    }
    eq(“productType”, selectedProductType)
}
```

HIBERNATE CRITERIA BUILDER

Taulujen yhdistäminen

- Liitokset tulevat mukaan kyselyyn helposti

```
List results = Product.createCriteria().listDistinct {

    components {
        eq("name", componentName)
    }

}
```

HUOM: käytä listDistinctiä joinien kanssa
HUOM: listDistinct ei toimi sivutuksen kanssa
→ jos tarvitaan molempia, kannattaa tehdä kysely HQL:llä

HIBERNATE CRITERIA BUILDER

Projektiot

- Laskettuja arvoja

```
List results = Product.createCriteria().listDistinct {
    projections {
        max("price")
    }

}
```

- Kysely voi palauttaa vain liitetyt objektit

```
List results = Product.createCriteria().listDistinct {

    components {
        projections {
            property("componentCode")
        }
    }
}
```

https://grails.github.io/grails-doc/latest/ref/Domain%20Classes/createCriteria.html

HIBERNATE CRITERIA BUILDER

Järjestäminen

- Järjestäminen kentän mukaan

```
List results = Product.createCriteria().listDistinct {
    order("name", "asc")

}
```

- Tulosten määrän rajaus

```
List results = Product.createCriteria().listDistinct {

    order("name", "asc")
    maxResults(1)
}
```

https://grails.github.io/grails-doc/latest/ref/Domain%20Classes/createCriteria.html

HIBERNATE CRITERIA BUILDER

Yhden olion hakeminen

- Get-metodi palauttaa suoraan yhden olion
  - Kyselyn on palautettava vain 1
  - list-kysely ja maxResults(1) palauttaisi listan yhdellä alkiolla

```
Product newest = Product.createCriteria().get {
    order("dateCreated", "desc")
    maxResults(1)
}
```

https://grails.github.io/grails-doc/latest/ref/Domain%20Classes/createCriteria.html

HARJOITUS

Tietokannat

Tee serviceen seuraavat kyselyt GORMilla tai Criterialla. Testaa BootStrapin avulla.

- Hae kaikki IBM-merkkiset tietokoneet
- Hae kaikki OS X -tietokoneet
- Hae kaikki RESEARCH-osaston tietokoneiden sarjanumerot
- Hae viimeksi käyttöönotettu tietokone
- Hae tietokoneet, jotka eivät ole juuri nyt käytössä

HQL

Hibernate Query Language

- SQL-variaatio
- Domain-luokan kautta käytettävissä
  - findAll()
  - executeQuery() (huom. tulosten esitysmuoto)
  - executeUpdate()

```
List products = Product.findAll("from Product as p where p.quantity > 0")

List results = Product.executeQuery("select name from Product where
quantity > 0")
println results[0][0]

Product.executeUpdate("update Product set quantity = 100")
```

HQL

Hibernate Query Language

- Parametrien välittäminen

```
List products = Product.findAll("from Product as p where p.name = :name",
[name: "MacBook Pro"])

Product.executeUpdate("update Product set status = :status, [status:
params.status])
```

CONTROLLERIT

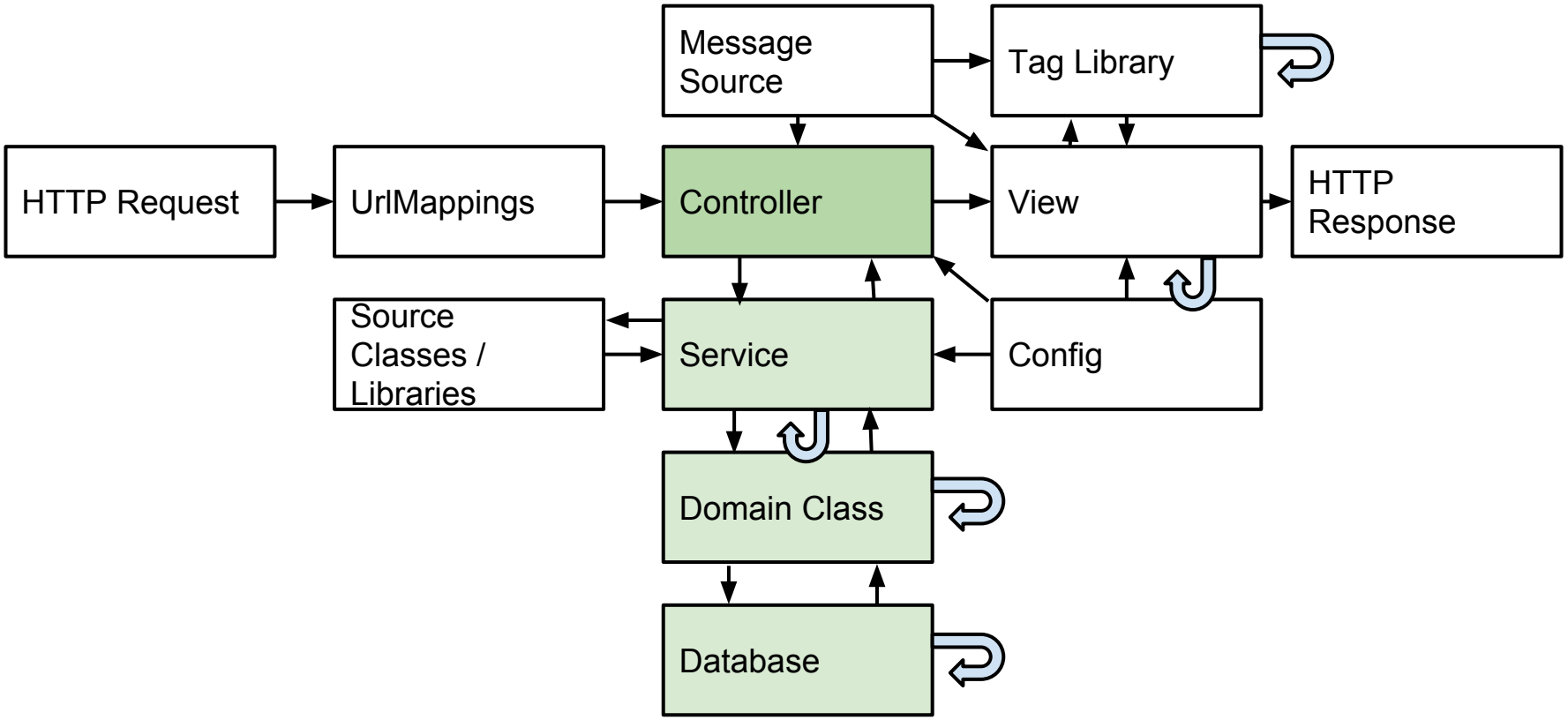
MVC-CONTROLLER

VINCIT

www.vincit.fi

GRAILS

MVC Controller: Controller



CONTROLLER

Perusteet

- Ohjaa pyynnöt oikeisiin toimintoihin
- Renderöi oikean näkymän
- Lisää viestit käyttäjälle
  - Onnistuiko tallennus?
  - Löytyikö kannasta tuloksia?
- Business-logiikka muualla
- Controller-logiikka liittyy näkymän valintaan

CONTROLLER

Perusteet

- Koostuu actioneista

```
def list() {
    ….
}
def save() {
    ….
}
```

RENDERÖINTI

- Kutsutaan actionin lopuksi
- render()
  - Parametreiksi näkymän nimi view ja tietomalli model
  - Mahdollista renderöidä myös tiedostoja, tekstiä, JSONia ja XML:ää
- respond()

UUDELLEENOHJAUS

- redirect()
  - Ohjaa toiseen controlleriin HTTP redirectillä
- forward()
  - Ohjaa toiseen controlleriin ilman HTTP redirectiä

PARAMS

- params-niminen Map controllereissa
- HTTP-parametrit
  - sekä GET että POST
- Domain-luokka voidaan alustaa suoraan antamalla konstruktorille params
  → ei kannata, miksi?

```
println params.name
```

FLASH

- Map-muuttuja
- Käytössä controllerissa
- Säilyy requestin ja seuraavan requestin ajan
- Täten voidaan siirtää dataa redirectin jälkimmäiseen requestiin
- Tyypillisin käyttö erilaiset viestit, kuten

```
flash.message = “Saved succesfully product ${product.id}”
```

REQUEST

- HttpServletRequest on käytettävissä controllerissa muuttujassa request
- Siihen voi myös tallentaa dataa requestin ajaksi

```
println request.forwardURI
```

SESSION

- HttpSession on käytössä muuttujassa session
- Myös voidaan lisätä dataa session ajaksi

```
session.searchSettings = params.searchSettings
```

HARJOITUS

Luonnostele controller-metodit tietokonekäyttäjärekisterin käyttöön:

- Listaa henkilöt
- Listaa tietokoneet
- Näytä tietokoneen tiedot
- Näytä käyttäjän tiedot (ja käytössä olleet tietokoneet)

Lisää puuttuvat service-metodit

HARJOITUS

Luonnostele controller-metodit tietokonekäyttäjärekisterin tiedonsyöttöön

- Lisää henkilö
- Lisää tietokone
- Ota tietokone käyttöön henkilölle
- Poista tietokone käyttöön henkilöltä

Lisää puuttuvat service-metodit

URLMAPPINGS

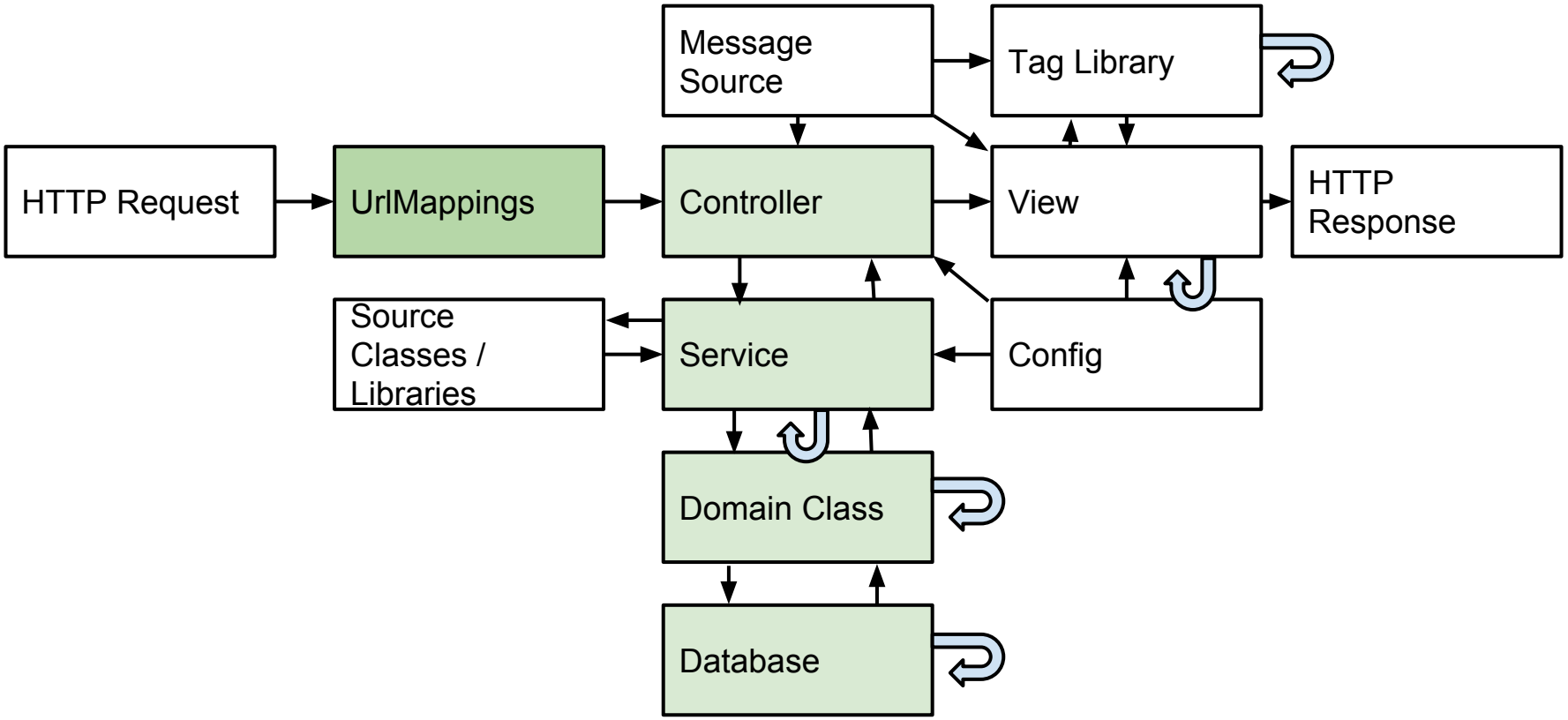
MVC-CONTROLLER

VINCIT

www.vincit.fi

GRAILS

MVC Controller: UrlMappings



URLMAPPINGS

- Valitaan controller-toiminto url-osoitteelle
- Välitetään parametreja käyttöliittymästä
- Scaffolding luo automaattiset CRUD-osoitteet
  - suoraan pääsy controlleriin
  - ei kannata käyttää oikeassa projektissa

```
"/product"(controller: "product", action: "list")
"/product/${id}"(controller: "product", action: "show")
```

URLMAPPINGS

Automaattinen RESTful-tapa

```
"/products"(resources:"product")
```

Metodi	**URI**	**Controller Action**
GET	/products	index
GET	/products/create	create
POST	/products	save
GET	/products/${id}	show
GET	/products/${id}/edit	edit
PUT	/products/${id}	update
DELETE	/products/${id}	delete

ERROR-SIVU

...koska kaikki ei aina mene suunnitelmien mukaan

- Stacktracen näyttäminen on tietoturvariski
- Sisäisen virheen sattuessa käyttäjää pitää kohdella silkkihansikkain

```
“500” (view: “error.gsp”)
```

HARJOITUS

- Lisää URLMappings-määrittelyt tietokonerekisterin controller-metodeille
- Valitse etusivuksi käyttäjälista
- Lisää myös virhetilannekäsittely
  - 404:n tapauksessa ohjataan etusivulle

MESSAGE SOURCE

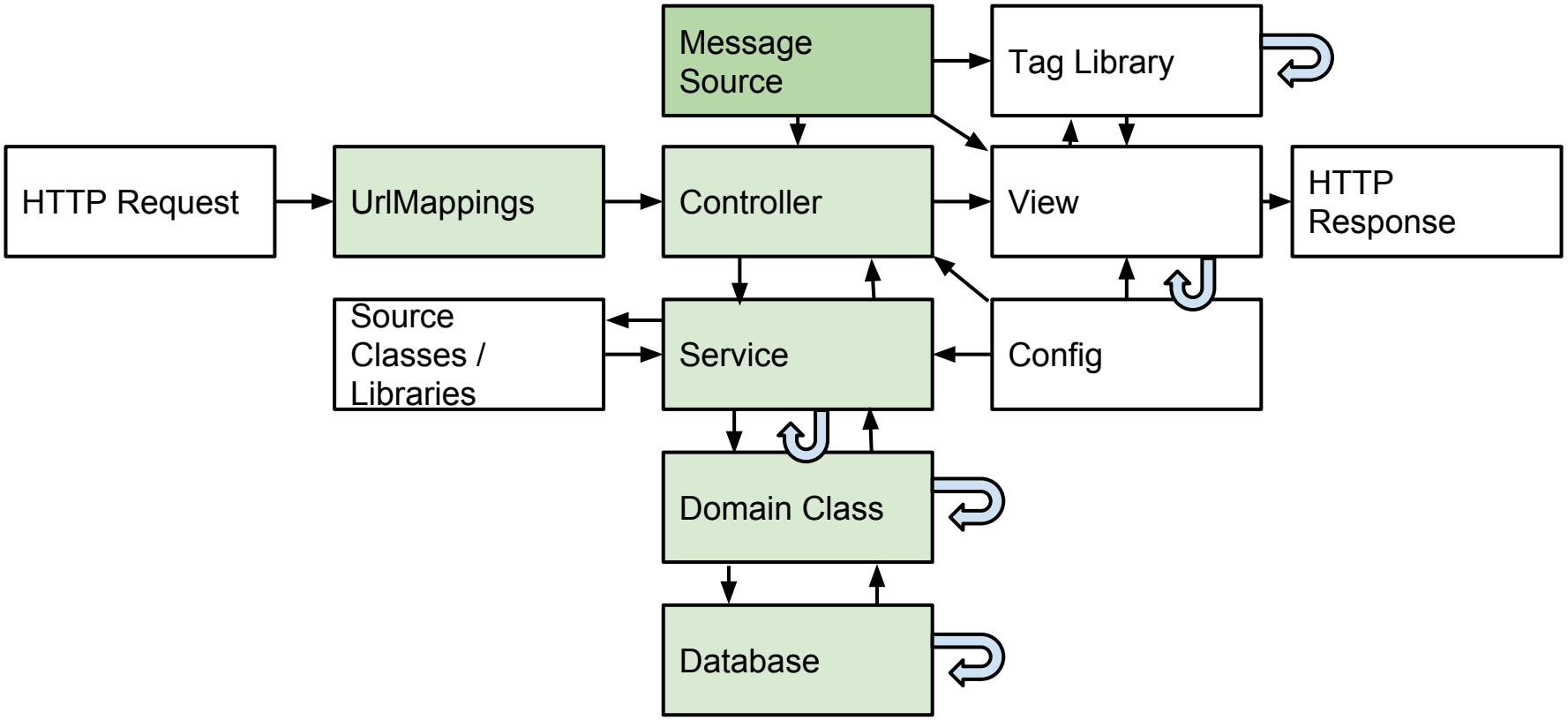
Käännöstiedostot

VINCIT

www.vincit.fi

MESSAGE SOURCE

Käännöstiedostot



MESSAGE SOURCE

Käännöstiedosto

- Projektin näkymien tekstejä varten
- Älä kovakoodaa yhdelle kielelle
  - Varaudu tulevaan
- Tulosta käännösmuuttuja
  - muuttuja käännetään valitulle kielelle
- Modulaarisuus
  - Sama teksti monessa paikassa
    - Muutettavissa kerralla

MESSAGE SOURCE

Käännöstiedosto

properties-formaatti

```
#User Stuff
user.username=Käyttäjätunnus
user.address.zip=Postinumero
...
```

kutsuminen gsp:stä

```
<g:message code="user.username"/>
g.message(code: "user.username")
```

Muuttujat argumentteina:

```
user.hello=Hello {0} {1}!

<h2><g:message code="user.hello" args=[user.firstName, user.lastName]</h2>
```

MESSAGE SOURCE

Käännöstiedosto

Tiedostot tallennetaan kielen nimen mukaan:

- messages.fi.properties
- messages.sv.properties
- jne

messages.properties on oletus

Kieltä voidaan vaihtaa suorituksen aikana:

```
/?lang=sv
```

MESSAGE SOURCE

Käännöstiedosto

Enumien lokalisointi

```
enum Departments {
     ADMINISTRATION,
     PRODUCTION,
     RESEARCH
     ...
}

Departments.ADMINISTRATION=Hallinto
Departments.PRODUCTION=Tuotanto
Departments.RESEARCH=Tutkimus

<g:message code="Departments.${employee.department}"/>
```

MESSAGE SOURCE

Käännöstiedosto

Constrainttien lokalisointi

- Validointivirhe tuottaa virheviestin
  - luokka.kenttä.constraint

```
user.username.blank=Käyttäjänimi ei saa olla tyhjä
user.email.nullable=Sähköposti on syötettävä
```

http://grails.github.io/grails-doc/2.5.0/ref/Constraints/blank.html

MESSAGE SOURCE

Käännöstiedosto

Huomioi merkistöt, IDE saattaa tallentaa tiedoston oletuksena väärällä merkistöllä

UTF-8 on aina varma valinta :)

JAVASCRIPTIN LOKALISOINTI

https://sergiosmind.wordpress.com/2013/07/25/getting-all-i18n-messages-in-javascript/

NÄKYMÄT

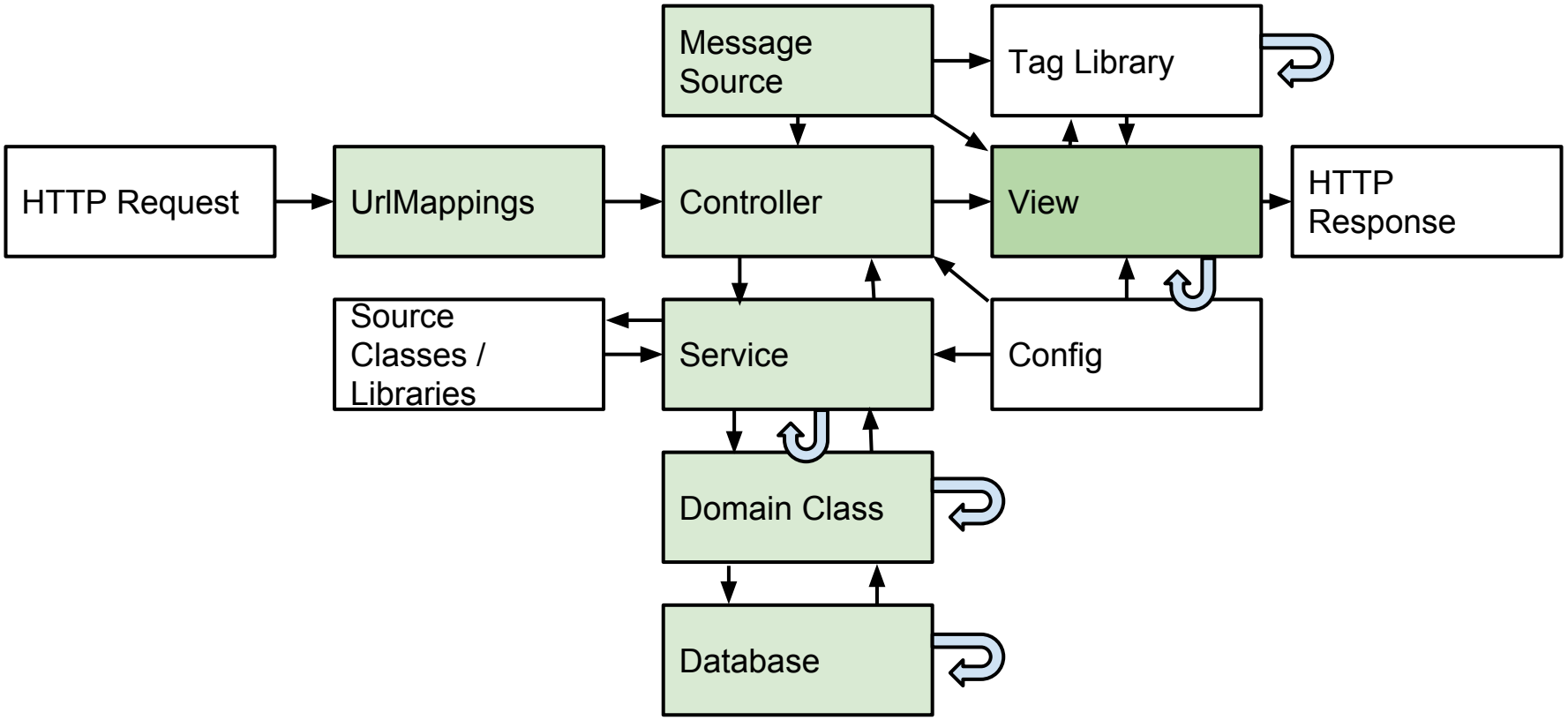
MVC-VIEW

VINCIT

www.vincit.fi

GRAILS

MVC View: View



GSP

Grails Server Pages

- Grailsin versio Java Server Pagesista
- HTML-koodin sisään asetetaan sovelluslogiikkaa tageilla
  - g-namespace esim `<g:if>`
  - tageja ilman bodya voidaan käyttää myös `${g.createLink(...)}`
  - custom-tageille voi tehdä oman namespacen ja käyttää samoja taginimiä

```
<vincit:createLink ...>
<vincit:form>...</vincit:form>
```

GSP-MUUTTUJAT

- Controllerin antama model
- Käytössä suoraan `${var}`-syntaksilla
- Uusia muuttujia voi myös asettaa set-tagilla

```
<g:set var="name" value="Antti"/>
<g:set var="nameUpper" value="${name.toUpperCase()}"/>
```

INLINE-KOODI

- JSP:n tavoin GSP:ssä voi kirjoittaa myös koodirivejä

```
<h1><g:message code=”title”/></h1>
<%
    def name = “Antti”
    def nameUpper = name.toUpperCase()
%>
<p><g:message code=”hello” args=[nameUpper]/></p>
```

- Kuitenkin, jos mahdollista, pidä Groovy-koodi controller/service-puolella

GSP IMPORT

Esim. Configin tai muun apuluokan importointia varten:

```
<%@ page import="utils.HelperClass" %>
```

GSP-EHTOLAUSEET

<g:if>

- testin parametrina test
- environment-kohtainen if parametrilla env
- if-tagin jälkeen voi tulla else-tagi
- ketjutusta voi tehdä elseif-tagia käyttäen

```
<g:if test="${product.isOnline()}">
    <g:message code="product.quantity" args="[product.quantity]"/>
</g:if>
<g:else>
    <g:message code="product.out.of.stock"/>
</g:else>
```

GSP-SILMUKAT

<g:each>

- Kokoelma parametrilla in
- Oletuksena muuttuja on it, tai parametrilla var
- Laskuri parametrilla status

```
<ul>
<g:each in="${results}" var="result" status="i">
    <li> ${i}. : ${result.name} ${result.time}</li>
</g:each>
</ul>
```

LINKIT

<g:link>

- Ulkoiset linkit voi tehdä perinteisesti a-tagilla
- Sisäiset linkit tehdään link-tagilla hyödyntäen urlmappingsia

```
<g:link controller=”product” action=”list”><g:message code=”product.list”
/></g:link>

<g:link controller=”product” action=”show” id=”${product.id}><g:message
code=”show.product” args=”[product.name]”/></g:link>

<g:link controller=”person” action=”show” params=”[id: person.id,
showDetails: true]”><g:message code=”person.details”/></g:link>
```

LINKIT

<g:createLinkTo>

- Generoi pelkän sisäisen urlin ilman linkkitagia
- Kätevä, jos haluaa lisätä muita attribuutteja

```
<a href="${g.createLinkTo(controller: "product", action: "show", id: id)}"
class="product-show-link" id="product-${id}">${product.name}</a>
```

HARJOITUS

Näkymät

- Tee näkymä käyttäjälistalle. Käyttäjästä näytetään id, etunimi ja sukunimi, osasto
- Tee näkymä tietokonelistalle. Koneesta näytetään id, sarjanumero, merkki, malli, kannettava/pöytäkone ja käyttöjärjestelmätyyppi sekä versio
- Muista käännöstiedostot!

PAGINATION

<g:paginate>

- Sivutus
- Parametrit
  - offset - mistä kohtaa tuloksia aloitetaan
  - max - paljonko tuloksia näytetään
- Paginate-tagilla helppo rakentaa

```
<g:paginate controller=”product” action=”list” total=”${productCount}” />
```

HARJOITUS

Pagination

- Tee paginaatio tietokonelistaan
- Lisää käännöstiedostoon paginaten käännökset
  - default.paginate.prev
  - default.paginate.next

LOMAKKEET

<g:form>

- Form-tagi käyttää myös UrlMapping-muotoista osoitesyntaksia
- Mahdollisuus lähettää tietyt muuttujat automaattisesti id- ja params-attribuutilla

```
<g:form controller="product" action="save" id="${product.
id} params="[saveDetails: true]">
...
</g:form>
```

HARJOITUS

Lomakkeet

- Tee lomakke käyttäjän syöttämistä varten
- Tee lomake, jossa valitaan käyttäjä ja käyttöönotettava tietokone
- Lisää käyttäjätietoihin linkki, jolla voidaan vapauttaa käyttäjän tietokone käytöstä

SISÄKKÄISET GSP:t

<g:render>

- Mahdollistaa näkymien modularisoinnin ja uudelleenkäytön
- Samaan tapaan kuin controllerista palauttaessa:
  - Yksittäiselle oliolle:
    ```
    <g:render template=”technicalDetails” bean=”product” />
    ```
  - Kokoelmalle :
  - ```
    <g:render template=”productListItem” collection=” products”/>
    ```
    ```

SISÄKKÄISET GSP:t

<g:include>

- Include-tagilla voidaan sijoittaa kokonaisen Grails HTTP requestin tulos näkymän sisälle

```
<g:include controller="product" action="viewDetails"
id="${product.id}"/>
```

Includella voi myös sisällyttää näkymiä.

Miksi Includen käyttöä kannattaa välttää?

LAYOUT

- Layout on sivun näkymän runko, jota useat sivut käyttävät
- Layout voi sisältää esim headerin, footerin ja valikon
- Layoutiin määritellään sivukohtaisesti muuttuva body-alue
- Layout määritellään head-osan tagilla:
  `<meta name="layout" content="main"/>`
- Layouteja voi olla monta, esim: main, wide, admin

LAYOUT

Esimerkki

```
<html>
     <head>
          <title>My great app 0.1 - <g:layoutTitle/></title>
          <!-- SEO etc. meta stuff, CSS and JS here -->
          <g:layoutHead/>
     <head>
<body>
     <header>
          <h1>My great app</h1>
     </header>
     <sidebar>
          <g:printSidebarMenu/>
     </sidebar>
     <article>
          <g:layoutBody/>
     </article>
     <footer>
          Created by Vincit 2015
     </footer>
</body>
</html>
```

Layoutin sisään voi syöttää sivun otsikon, headin ja bodyn

HARJOITUS

Layout

Tee kaksi layouttia tietokonerekisterisovellukseen.

- Main: Tulostaa headeriin sovelluksen nimen ja sivupalkkiin valikon
- Plain: Tulostaa sisällön lisäksi pelkän linkin, joka vie etusivulle

Lisää Main-layout listasivuille ja Plain-layout lomakesivuille

FRONTEND-RESURSSIT

CSS, JavaScript, kuvat, jne

Staattisen sisällön linkitys external-tagilla

```
<g:external dir=”css” file=”main.css”/>
<g:external dir=”js” file=”hacks.js”/>
<g:external dir=”img” file=”cat.gif”/>
```

FRONTEND-RESURSSIT

LESS, SASS, CoffeeScript jne

Grails Asset-Pipeline

```
<asset:stylesheet href="styles.css"/>
<asset:javascript src="application.js"/>
```

Minifiointi → Configiin

```
grails.assets.minifyJs = false
```

Assets-kansio WEB-APP-kansion sijasta

HARJOITUS

Resurssit

Lisää main-layoutin headeriin kuva. Sama kuva toimii myös plain-layoutin etusivu-linkkinä. Ilmaisia ikoneja tietokoneista esim: http://bit.ly/1HENPk8

Lisää CSS-määritykset, joilla saat Plain-layoutin etusivu-linkin tekstin isommaksi sekä Main-layoutin valikon vasempaan reunaan

TAG LIBRARYT

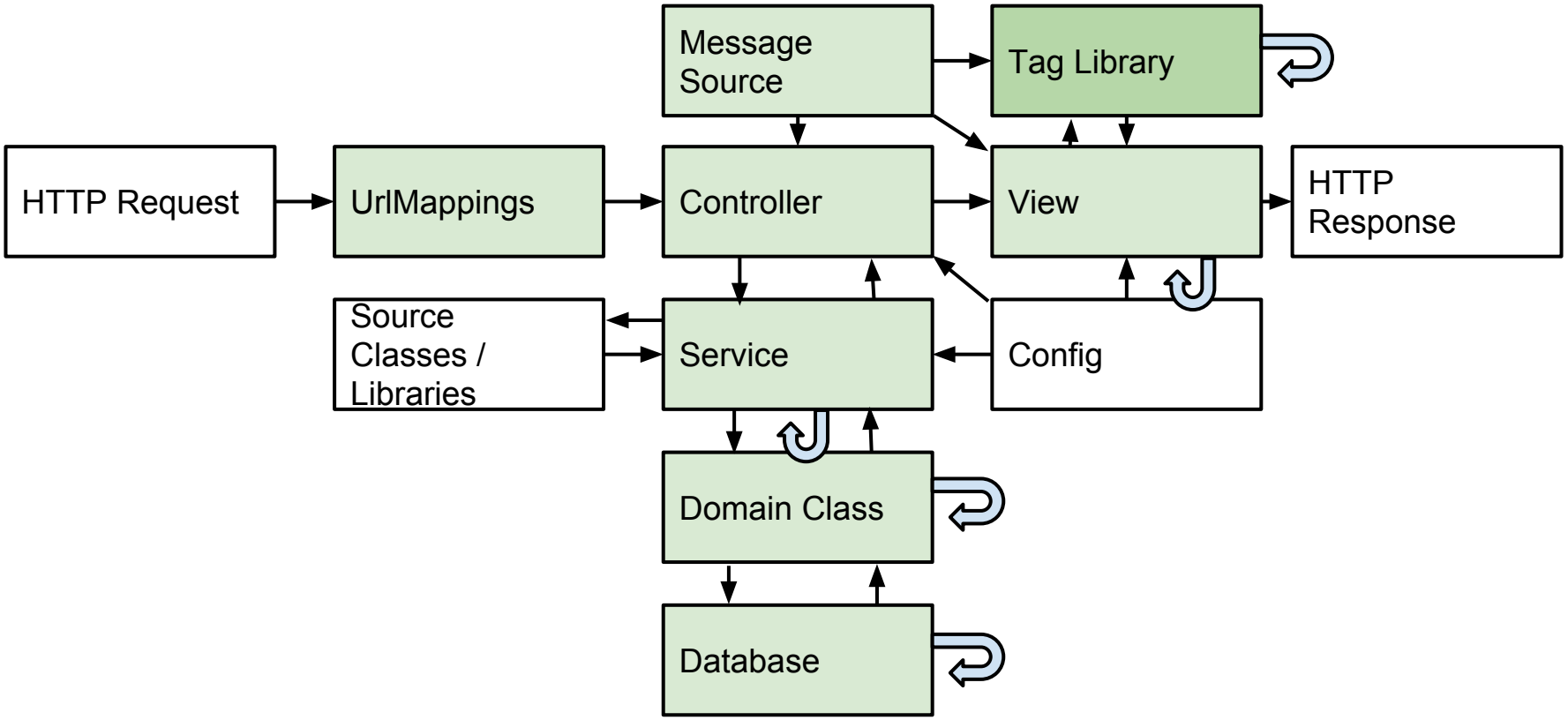
MVC-VIEW

VINCIT

www.vincit.fi

GRAILS

MVC View: View



TAG LIBRARYT

- Modulaarisia frontendin komponentteja
- Voi muokata vain dataa tai tehdä HTML-koodia

```
class ProductTagLib {
     def printProductName = { attrs, body ->
          def product = attrs.product
          out << "<b>${product.manufacturer.name}</b> ${product.modelName}
     }
     def printProductPrice = { attrs, body ->
          out << attrs.price / 100
     }
}

<g:printProductName product="${product}"/>
<g:printProductPrice price="${product.price}"/>
```

HARJOITUS

Tee tagi, joka tulostaa tietokoneen tiedot HTML-muotoiltuna seuraavankaltaisesti:

Apple MacBookPro
OS X Mavericks

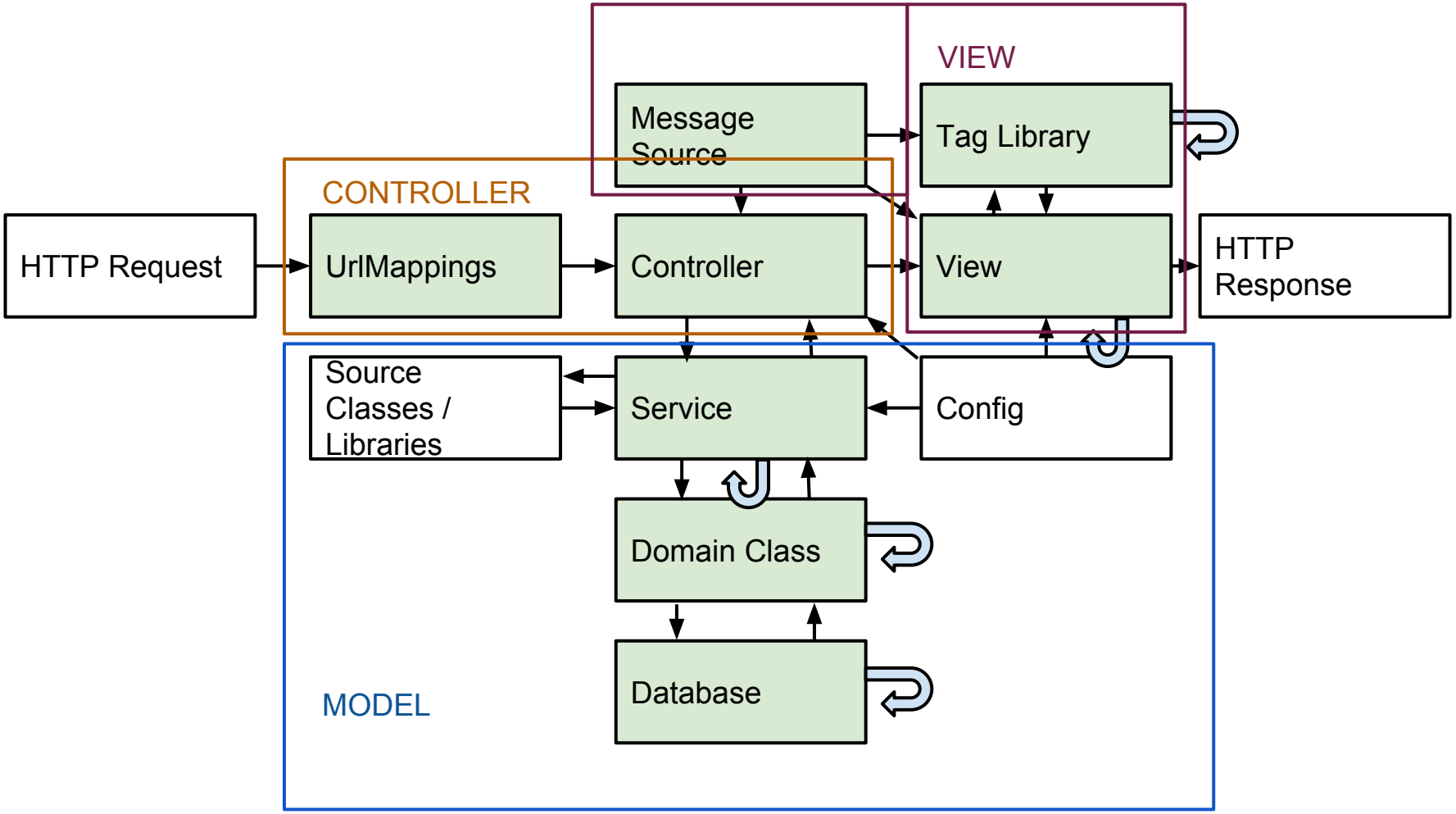
MVC-KERTAUS

VINCIT

www.vincit.fi

GRAILS

MVC-kertaus



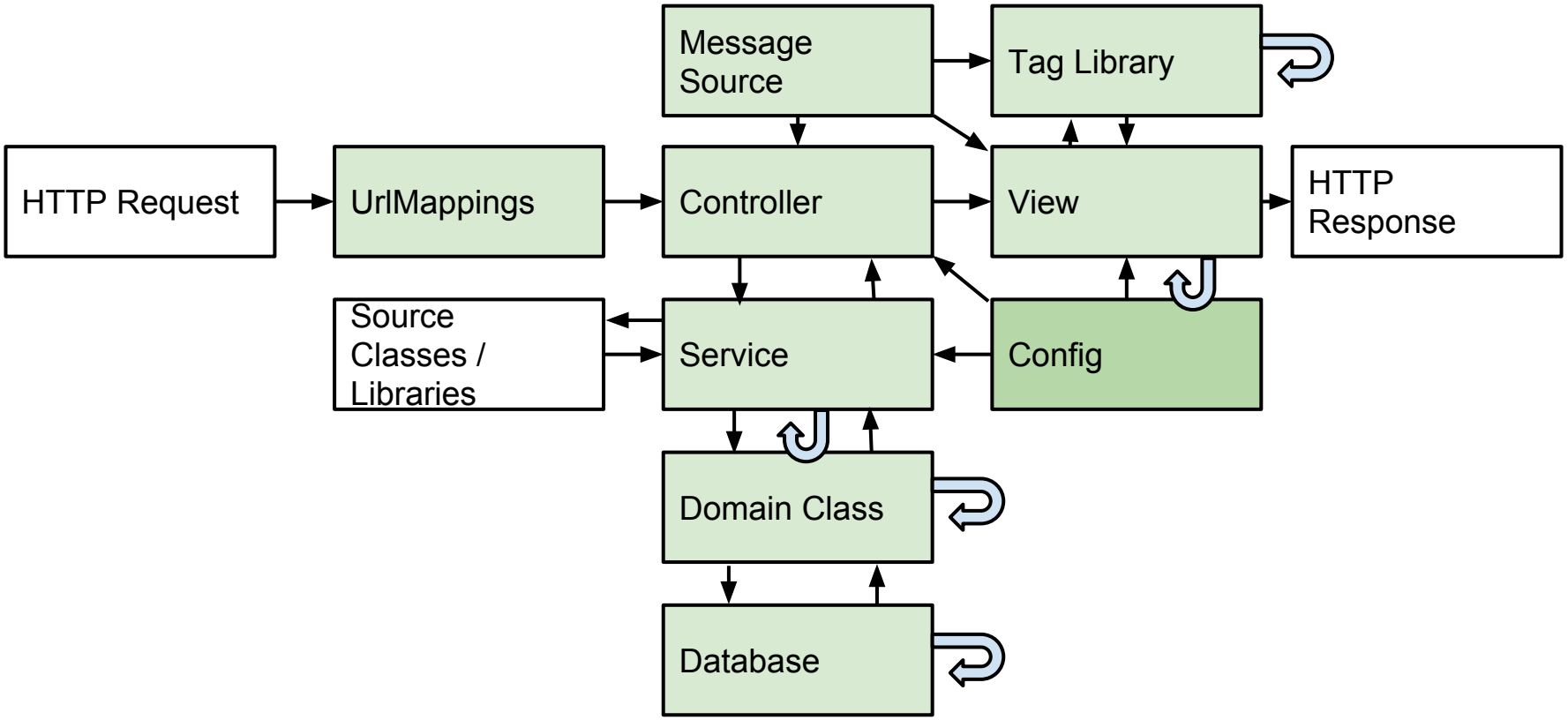
CONFIG

Asetukset

VINCIT

www.vincit.fi

CONFIG


Message Source
Tag Library
HTTP Request
UrlMappings
Controller
View
HTTP Response
Source Classes / Libraries
Service
Config
Domain Class
Database

CONFIG

Perusteet

- Nimensä mukaisesti Configiin tallennetaan erilaisia vakioita ja muita asetuksia
- Configissa voi määritellä eri asetuksia ympäristökohtaisesti
- Sijaitsee Config.groovyssä
- HUOM: Grails 3.0:ssa muuttunut

CONFIG

Formaatti

- Config käyttää Groovyn ConfigSlurper-formaattia
  - Pistenotaatio
  - Blokit

```
site.url = 'http://www.site.com'
smtp = {
    mail {
        host = 'smtp.site.com'
    }
}
productsPerPage = 10
showAds = true
```

CONFIG

Käyttö

Lisätään dependency-injektio:

```
def grailsApplication
```

Config löytyy tämän objektin .config:sta:

```
int productsPerPage = grailsApplication.config.productsPerPage
String host = grailsApplication.config.smtp.mail.host
```

YMPÄRISTÖKOHTAINEN CONFIG

Config-arvot voivat olla ympäristökohtaisia

```
environments {
     development {
          site.url = 'localhost:8080/myapp'
          errorMail = {
               send = true
               address = 'antti@vincit.fi'
          }
     }
     test {
          site.url = 'localhost:8080/myapp'
          errorMail = {
               send = false
          }
     }
     production {
          site.url = 'www.myapp.com'
               errorMail = {
               send = true
               address = 'errors@myapp.com'
          }
     }
}
```

YMPÄRISTÖT

- development = kehittäjän kone
- test = automaattista testausta varten
- production = tuotantopalvelin
  - monistettavissa ja nimet muutettavissa
  - esim prod1, prod2 etc.
  - ympäristön nimi annetaan war-paketin luonnin yhteydessä parametrina

GLOBAALI CONFIG

- Environments-lohkon ulkopuolinen config on globaali, yhteinen kaikille ympäristöille
- Globaalit muuttujat voi ylikirjoittaa ympäristökohtaisesti

HARJOITUS

Config

- Lisää Configiin paginate.MAX-vakio, jonka arvo voi olla vaikkapa 10
- Muuta paginate-listat käyttämään kyseistä vakiota

DATA SOURCE

Tietokannan oma config

- DataSource.groovy
- Tietokannan ajuri, osoite, tunnukset ja muut asetukset

```
dataSource {
     pooled = false
     driverClassName = "org.h2.Driver"
     username = “grails”
     password = “secret”
     dbCreate = “update”
}
```

DATA SOURCE

Tietokannan oma config

- Myös ympäristökohtaisesti säädettävissä

```
dataSource {
    pooled = false
    driverClassName = "org.h2.Driver"
}
environments {
     development {
          url = “jdbc:j2:mem:devDb”
     }
     test {
          ...
     }
     ...
}
```

TIETOKANNAN ALUSTAMINEN

- Testatessa BootStrapin kanssa kannan voi aina tyhjätä
  - DataSource: dbCreate = “create-drop”
- HUOM: Tuotannossa aina dbCreate=”update” :)

BUILD CONFIG

BuildConfig.groovy

- Komentorivikomentojen asetukset
- JDK yms. versiot
- Tarvittavat pluginit versioineen
- Muistiasetukset
- Hakemistorakenteet käännetyille tiedoistoille

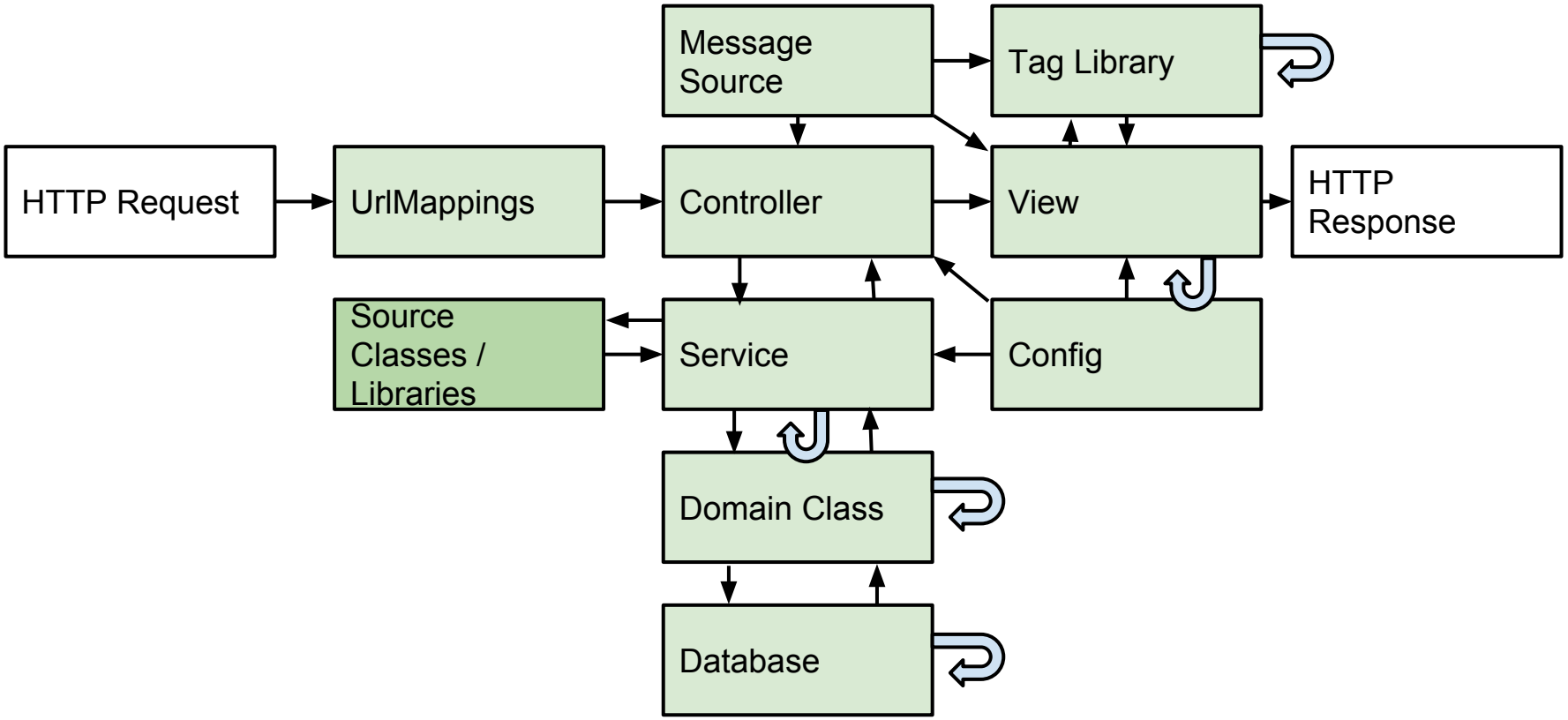
SOURCE CLASSES & LIBRARIES

Apukoodit ja -kirjastot

VINCIT

www.vincit.fi

SOURCE CLASSES & LIBRARIES



SOURCE CLASSES

- Domain-kokonaisuuden ulkopuolinen apuluokka
  - ei kuitenkaan ei-transaktionaalinen Service?
- src-hakemisto
- Groovy tai Java
- Enumit

LIBRARIES

- Valmiit apukirjastot
- jar
- lib-kansio
- Esim. Maven-repository http://search.maven.org/

TESTAUS

VINCIT

www.vincit.fi

MIKSI AUTOMAATTISET TESTIT?

- Varmistetaan koodin toimivuus
  - kaikilla syötteillä
  - myös virhetilanteissa
- Koodi toimii myös kun sitä muutetaan
  - Muutetun toiminnallisuuden osalta myös testit muutettava
- Helppo varmistaa tuotantoversiota päivittäessä että kaikki toimii odotetusti

UNIT-TESTIT

- Testataan yksiköittäin
  - Metodi kerrallaan
  - Dependencyt mockataan

    → eivät vaikuta testiin
- Annetulle syötteelle oletettu palaute
  - Oletetuissa kohdissa poikkeukset

TDD

Test Driven Development

- Kirjoitetaan ensin testit

  → Määritellään businesslogiikka ja toiminta virhetilanteissa
- Sitten vasta kirjoitetaan koodi täyttämään testien määritelmät
  + Modulaarinen ajattelu kehittyy
  + Testit eivät ainakaan unohdu
  - Kehittämisessä alkuun pääseminen hidastuu
  - Joskus toiminnallisuus valkenee kehittäjälle tehdessä

TESTAUS GRAILSISSA

- Tärkeintä on testata Model-alue
  - Service-metodit
  - Domain-luokkien metodit
  - Transient-muuttujat
- Myös Controllereita voi testata
  - Millainen näkymä näytetään tietyllä syötteellä?
- Myös Constraintteja voi testata
  - Validoituuko tietyillä arvoilla luotu olio?

TESTATTAVA KOODI

- “Spagettikoodia” on hankala testata
- “Yksi metodi tekee yhden asian”-periaate
- Järkevät palautusarvot

TESTATTAVA KOODI

EI NÄIN :(

```
def saveProduct(Map params) {
     def product = Product.get(params.id)
     if (product) {
          product = new Product(.....)
     }
     product.name = params.name
     product.manufacturer = Manufacturer.findByName(params.manufacturerName)
     ...
     product.price = params.price / 100
     product.discountedPrice = params.price / 100 * params.discount
     ...
     if (product.save()) {
          existingProductCategories = ProductCategory.findByProduct(product)
          existingProductCategories.each({category ->
               ...
           })
          return product
     } else {...}
}
```

TESTATTAVA KOODI

Mieluummin näin :)

```
def saveProduct(Map params) {
     def product = findOrCreateProduct(params.id)
     product = bindValuesToProduct(params, product)
     product = priceService.determineProductPrice(params, product)
     if (product.save()) {
          productCategoryService.determineProductCategories(product, params)
          return product
     } else {
          ...
     }
}
```

HARJOITUS

Varmista, että kirjoittamasi harjoituskoodi on testattavaa. Muokkaa tarvittaessa sitä lyhyempiin metodeihin

SPOCK

Testiframework

VINCIT

www.vincit.fi

SPOCK

- Grailsin valinta
- Toimii myös Javalla
- Testit ajetaan JUnitilla

SPOCK

Testin rakenne

- def-tyyppinen muuttuja
  - nimi on String-literaali
    - kuvaa mitä testin pitäisi tehdä
- setup
  - asetetaan alkuehtoja
  - alustetaan muuttujia
- when
  - suoritetaan testattava metodi
- then
  - tarkistetaan oletetut arvot

SPOCK

```
def “should attach company to employee”() {
    given:
        Company company = new Company(name: “Vincit”)
        Employee employee = new Employee(firstName: “Antti, lastName:
“Loponen”)
    when:
        company.addEmployee(employee)
    then:
        employee.company == company
}
```

SPOCK

Vaihtoehtoinen rakenne

- When ja then -osiot voidaan korvata expect-lohkolla, joka sisältää operaation ja oletetun tuloksen

SPOCK

```
def “should print name in correct form”() {
    given:
        User user = new User(firstName: “Antti”, lastName: “Loponen”)
    expect:
        User.getFullName().equals(“Antti Loponen”)
}
```

SPOCK

GORMin mockaus

- GORM-metodeja ei saa käyttää sellaisinaan testeissä
- Ne täytyy mockata @Mock-annotaatiolla

```
@TestFor(User)
@Mock([User, Project])
def UserSpec extends Specification {
...
```

SPOCK

Stub

- Toinen metodi, jota testattava metodi käyttää
- Metodin on palautettava jotain
- Oletetaan että metodi palauttaa tässä testissä jotain
- Voidaan testata useilla arvoilla, mutta varsinaista koodia ei suoriteta

SPOCK

Stub

```
def “should return empty list”() {
    given:
        ProductService productService = Stub()
        productService.getOnlineProducts() >> {
            return []
        }
        ProductSEOService productSEOService = new ProductSEOService()
        Product comparedProduct = new Product(....)
    when:
        List similarProducts = productSEOService
        .findSimilarOnlineProducts(comparedProduct)
    then:
        similarProducts.isEmpty() == true

}
```

SPOCK

Stub

```
def “should return the similar product”() {
    given:
        ProductService productService = Stub()
        Product similarProduct = new Product(...)
        Product differentProduct = new Product(...)
        productService.getOnlineProducts() >> {
            return [similarProduct, differentProduct]
        }
        ProductSEOService productSEOService = new ProductSEOService()
        Product comparedProduct = new Product(....)
    when:
        List similarProducts = productSEOService
        .findSimilarOnlineProducts(comparedProduct)
    then:
        similarProducts.isEmpty() == false
        similarProducts.first().equals(similarProduct)

}
```

VALIDOINNIN TESTAUS

```
// firstName ja LastName ovat pakollisia ei-tyhjiä kenttiä...

def “should not validate”() {
    given:
        User user = new User(firstName: “Antti”, lastName: “”)
    expect:
        user.validate() == false
}
def “should validate”() {
    given:
        User user = new User(firstName: “Antti”, lastName: “Loponen”)
    expect:
        user.validate() == true
}
```

CONTROLLERIN TESTAUS

```
def “Controller should respond as expected”(){
    ProductService productService = Stub()
    Product product = new Product(...)
    setup:
        controller.productService = productService
        productService.getProduct(_) >> {
            return product
        }
    when:
        controller.show()
    then:
        model.product == product
        view == “product/show”
}
```

HARJOITUS

Kirjoita spock-testejä tietokonerekisterin luokille

Tee ainakin yksi validointitesti, yksi controller-testi ja yksi service-testi

MUITA TESTEJÄ

VINCIT

www.vincit.fi

INTEGRAATIOTESTIT

Mahdollisuus testata järjestelmää käyttäen kaikkia luokkia ja Grails-kehikkoa

https://grails.github.io/grails-doc/latest/guide/testing.html#integrationTesting

FUNKTIONAALISET TESTIT

Lisätään Integraatiotesteihin HTTP-requestit

https://grails.github.io/grails-doc/latest/guide/testing.html#functionalTesting

PLUGINIT

VINCIT

www.vincit.fi

PLUGINIT

- Usein open sourcena kehitettyjä apukirjastoja
- Esim.
  - Spring Security
  - MongoDB GORM
  - Cache Plugin
  - Quartz
  - Spock
  - Searchable
  - Spring Web Flow

PLUGINIT

Asentaminen

- Komentorivillä

```
grails install-plugin searchable
grails uninstall-plugin searchable
```

Versiopäivitys onnistuu ajamalla install-plugin uusiksi

BuildConfig.groovyssa voi hallita versioita

OMAN PLUGININ TEKEMINEN

- Modulaarisuuden lisääminen
- Komponenttien jakaminen projektin välille
- Hyvän pluginin voi julkaista!

OMAN PLUGININ TEKEMINEN

- Komentorivillä

```
grails create-plugin awesome-plugin
```

- Grails-projektin näköinen rakenne
- UrlMappings, controllerit, näkymät, domainit jne. tulevat projektin käyttöön
- Hierarkinen dependecy-mahdollisuus

SPRING SECURITY

VINCIT

www.vincit.fi

SPRING SECURITY

- Yleisin plugin käyttöoikeuksien rajoittamiseen
- Käyttäjille asetetaan roolit
  - Role-oliot
  - Esim. ROLE_USER, ROLE_ADMIN jne.

SPRING SECURITY

Käyttöoikeusrajoitteet

- Config.groovy
- Oletuksena kaikki sivut vaativat ROLE_ADMIN-roolin, vain poikkeukset toimivat alemmilla tasoilla:

```
grails.plugin.springsecurity.rejectIfNoRule = true
grails.plugin.springsecurity.fii.rejectPublicInvocations = false
```

Onnistuu myös päinvastaisilla asetuksilla

```
grails.plugin.springsecurity.rejectIfNoRule = false
grails.plugin.springsecurity.fii.rejectPublicInvocations = true
```

SPRING SECURITY

Poikkeukset

Etusivu, fronttiresurssit yms. on oltava saatavissa ilman kirjautumista

```
grails.plugin.springsecurity.controllerAnnotations.staticRules = [
   '/':                ['permitAll'],
   '/index':           ['permitAll'],
   '/index.gsp':       ['permitAll'],
   '/assets/**':       ['permitAll'],
   '/**/js/**':        ['permitAll'],
   '/**/css/**':       ['permitAll'],
   '/**/images/**':    ['permitAll'],
   '/**/favicon.ico':  ['permitAll']
   '/login/**':        ['permitAll'],
   '/logout/**':       ['permitAll']

]
```

SPRING SECURITY

Roolit

Lisätään pari rooli-URL-määrittelyä

```
grails.plugin.springsecurity.controllerAnnotations.staticRules = [
    ...
   '/user/**':        [‘ROLE_USER’],
   '/admin/**':       ['ROLE_ADMIN']


]
```

USER

- User-luokan pakolliset kentät

```
String username
String password
boolean enabled
hasMany = [“authorities”: UserRole]
```

Usernamen tulee olla unique

SECURITYTAGLIB

- Rajoitetaan näkymän osia roolin mukaan

```
<sec:ifAnyGranted=”ROLE_ADMIN”>
    <!-- ADMIN ONLY STUFF -->
</sec:ifAnyGranted>
<sec:ifNotGranted=”ROLE_ADMIN”>
    <!-- VISIBLE FOR OTHER THAN ADMINS -->
</sec:ifNotGranted>
```

- Kirjautuneen käyttäjän tiedot

```
<sec:loggedInUserInfo field=”username/>
```

QUARTZ

VINCIT

www.vincit.fi

QUARTZ

- Plugin Cron-ajoja varten
- Eräajot laitetaan jobs-kansion alle
- Jobilla voi kutsua service-metodeja

HARJOITUS

Croniajo

- Tee Croniajo, joka tulostaa vapaana olevat tietokoneet logiin
- Croniajo ajetaan kerran minuutissa

TIETOTURVA

VINCIT

www.vincit.fi

SQL-INJEKTIO

- Käyttäjä manipuloi tietokantaa parametrilla

```
SELECT * FROM PRODUCT WHERE NAME =’$name;
$name = “‘; UPDATE USER SET ROLE=’ADMIN’ WHERE ID=’123’;”;
```

- Grails torjuu suurimman osan
- HUOM! HQL: käytä named parameter-mappia

```
Book.find("from Book as b where b.title = :title", [title: params.title])
// EI NÄIN: Book.find("from Book as b where b.title ='${params.title}'")
// EI NÄIN: Book.find("from Book as b where b.title ='" + params.title + "'")
```

CROSS SITE SCRIPTING

- Sellaisenaan tulostettava kenttä
  - käyttäjä kirjoittaa scriptin
  - scripti ajetaan
- Grails escapoi oletuksena kaiken
- Poikkeuksia voi tehdä raw-tagilla

CROSS SITE REQUEST FORGERY

- Session hyödyntäminen esim. salasanan vaihtamiseen käyttäjän tietämättä
- Estetään käyttämällä tokenia lomakkeille

```
<g:form useToken="true">...
```

Tokenin vastaanottaminen controllerissa:

```
withForm {
    // oikea token
}.invalidToken {
    // väärä token
}
```

Samalla estää lomakkeen tuplalähetyksen

LOGITUS

VINCIT

www.vincit.fi

LOGITUS

Groovyn print-metodi

```
println “Hello World”
```

LOG4J

- log-objekti
- Logitus tasoittain
  - info, debug, warn, error, fatal jne

```
if (product.save()) {
    log.info("Product ${product.id} was saved. ")
} else {
    log.error("Product ${product.id} could not be saved")
    product.errors.each({error -> log.error(error)})
}
```

LOG4J

- Logitus luokittain (Config.groovy)
  - Määriteltyä tasoa alemmat jätetään pois

```
log4j = {

    info “grails.app”
    error “grails.app.controllers”
    debug “grails.app.services.ProductService”

}
```

LOGITUS TIEDOSTOON

- Appenders (Config.groovy)

```
log4j {
    appenders {
        rollingFile  name: “basicLogAppender”,
                     maxFileSize: 1024,
    }                file: “tmp/logs/debug.log”
}
```

HARJOITUS

- Lisää harjoitussovellukseen LOG4J-tulostukset onnistuneille ja epäonnistuneille tallennuksille

TUOTANTOKÄYTTÖ

VINCIT

www.vincit.fi

PAKETIN TEKEMINEN

```
grails -Dgrails.env=production war
```

Muistiasetukset BuildConfig.groovy

```
grails.tomcat.jvmArgs = ["-Xms256m", "-Xmx1024m"]
```

WAR-paketti voidaan siirtää Tomcat-palvelimelle ja suorittaa

OPTIMOINTIVINKKEJÄ

- Vahva tyypitys
- for-silmukka > each-silmukka
- HQL > Criteria
- Yksi iso kysely > Monta pientä
  - Filtteröi ohjelmallisesti
- Include-tagin kanssa varovaisuutta
- Isoissa eräajoissa toistuvat kirjastokutsut kannattaa cachettaa Mappiin


VINCIT
KIITOS

KURSSIPALAUTE

- http://goo.gl/forms/Z2WduH5q0k