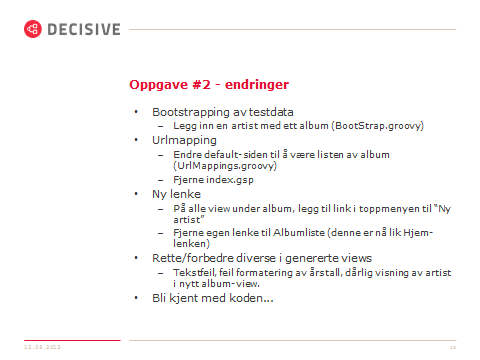
Oppgaver

Grails workshop

****

****

**Artist.groovy**

**package** budapest

**class** Artist {

String navn

**static** *hasMany* = [album:Album]

**static** *constraints* = {

}

}

**Album.groovy**

**package** budapest

**class** Album {

String navn

Integer aar

Artist artist

**static** *constraints* = {

*aar* nullable:**true**

}

}

**Bootstrap.config**

**import** budapest.Album

**import** budapest.Artist

**class** BootStrap {

**def** init = { servletContext ->

Artist artist = **new** Artist(navn: "Adele").save();

Album album = **new** Album(navn: "21", artist: artist, aar: 2011).save()

}

**def** destroy = {

}

}

**UrlMappings.config**

**class** UrlMappings {

**static** *mappings* = {

"/$controller*/$*action?/$id?"{

constraints {

// apply constraints here

}

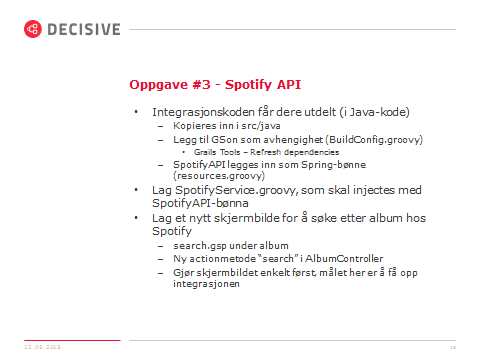
}

"/"(controller:"album")

"500"(view:'/error')

}

}



**Build.config**

…

grails.project.dependency.resolution = {

…

dependencies {

…

compile group:'com.google.code.gson', name:'gson', version:'2.2.2'

…

}

…

}

**Resource.groovy**

**import** org.rakvag.spotifyapi.HTTPBufferedReader

**import** org.rakvag.spotifyapi.SpotifyAPIImpl

// Place your Spring DSL code here

beans = {

spotifyAPI(SpotifyAPIImpl){

httpReader = **new** HTTPBufferedReader()

}

}

**AlbumController.groovy**

**class** AlbumController {

**static** allowedMethods = [save: "POST", update: "POST", delete: "POST"]

SpotifyService spotifyService

…

def search() {

**if**(params.search && (params.artist || params.album)){

def liste = spotifyService.sokEtterAlbum(params.artist, params.album)

**if**(liste.size() < 1) {

flash.message = "Fant ikke no gitt"

}

[albumliste: liste]

} **else** **if**(params.search){

flash.message = "Må jo skrive inn no da..."

}

}

}

**SpotifyService.groovy**

**package** budapest

**import** java.util.List;

**import** org.rakvag.spotifyapi.SpotifyAPI;

**import** org.rakvag.spotifyapi.entity.SpotifyAlbum;

**class** SpotifyService {

SpotifyAPI spotifyAPI

List<Album> sokEtterAlbum(String artistNavn, String albumNavn) {

**def** spotifyAlbum = spotifyAPI.soekEtterAlbum(artistNavn, albumNavn, 1)

List<Album> albumer = []

spotifyAlbum.*each* { sa ->

Album nyttAlbum = **new** Album(navn: sa.getName(), aar: sa.getReleased(), artist: **new** Artist(navn: sa.getArtists().iterator().next().getName()))

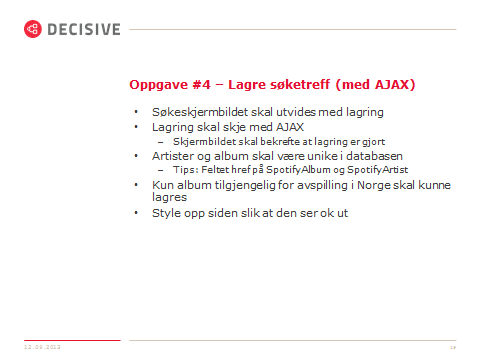
albumer.add(nyttAlbum)

}

albumer

}

}



**AlbumController.groovy**

**package** budapest

**import** org.springframework.dao.DataIntegrityViolationException

**class** AlbumController {

…

def add() {

def uri = params.spotifyURI

Album album = spotifyService.hentAlbum(uri)

album.artist.save()

album.save()

render "Albumet \"" + album.artist.navn + " - " + album.navn + "\" har blitt lagt til"

}

}

**Album.groovy**

**package** budapest

**class** Album {

String navn

Integer aar

Artist artist

String spotifyURI

**static** constraints = {

aar nullable:**true**

spotifyURI unique:**true**, nullable: **true**

}

@Override

**public** String toString() {

navn

}

}

**Artist.groovy**

**package** budapest

**class** Artist {

String navn

String spotifyURI

**static** hasMany = [album:Album]

**static** constraints = {

spotifyURI unique:**true**, nullable: **true**

}

@Override

**public** String toString() {

navn

}

}

**SpotifyService.groovy**

**package** budapest

**import** java.util.List;

**import** org.rakvag.spotifyapi.SpotifyAPI;

**import** org.rakvag.spotifyapi.entity.SpotifyAlbum;

**class** SpotifyService {

SpotifyAPI spotifyAPI

List<Album> sokEtterAlbum(String artistNavn, String albumNavn) {

**def** spotifyAlbum = spotifyAPI.soekEtterAlbum(artistNavn, albumNavn, 1)

List<Album> albumer = []

spotifyAlbum.*each* { sa ->

**if**(sa.erTilgjengeligINorge() && !Album.*findBySpotifyURI*(sa.getHref())){

**def** nyttAlbum = mappAlbum(sa)

albumer.add(nyttAlbum)

}

}

albumer

}

Album hentAlbum(String spotifyURI) {

Collection<SpotifyAlbum> albumColl = spotifyAPI.hentAlbumPaaSpotifyURIer([spotifyURI], 3)

**def** spotAlbum = albumColl.iterator().next()

**def** album = mappAlbum(spotAlbum);

**return** album

}

**private** Album mappAlbum(SpotifyAlbum sa) {

**def** nyttAlbum = Album.*findBySpotifyURI*(sa.getHref())

**if** (nyttAlbum)

**return** nyttAlbum

nyttAlbum = **new** Album()

nyttAlbum.navn = sa.getName()

nyttAlbum.spotifyURI = sa.getHref()

nyttAlbum.aar = sa.getReleased()

nyttAlbum.artist = mappArtist(sa)

**return** nyttAlbum

}

**private** Artist mappArtist(SpotifyAlbum spotifyAlbum) {

**def** artistURI = spotifyAlbum.getArtistid()

artistURI = artistURI ?: spotifyAlbum.getArtists()?.iterator()?.next()?.getHref()

**def** nyArtist = Artist.*findBySpotifyURI*(artistURI);

**if** (nyArtist)

**return** nyArtist;

**def** artistNavn = spotifyAlbum.getArtist();

artistNavn = artistNavn ?: spotifyAlbum.getArtists()?.iterator()?.next()?.getName()

**return** **new** Artist(navn: artistNavn, spotifyURI: artistURI)

}

}



