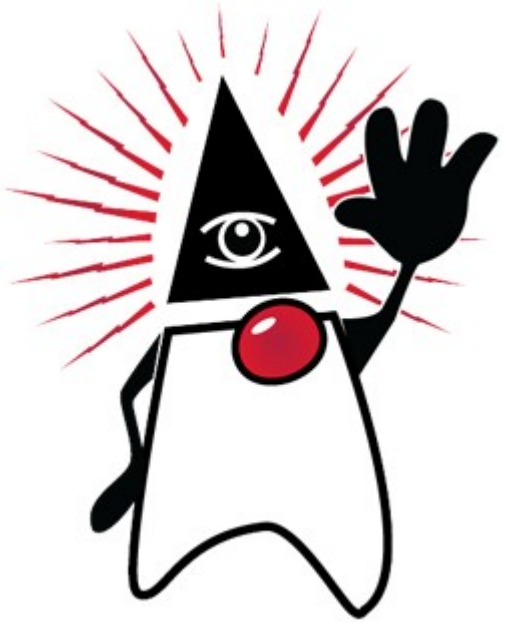
OfficeNet



Lift Co.

Lift from a JEE perspective

Andreas Joseph Krogh

https://github.com/andreak/on-example-rpm

33rd Degree Conference
Conference for Java Masters
19-21 March 2012
Kraków, Poland

OfficeNet

Lift Co.

Who am I

- Andreas Joseph Krogh from OfficeNet (small Norwegian company)
- Lift-committer and co-founder of Lift Co. - the official Lift-support company, www.liftweb.com
- Commercial software-development for 14 years
  - latest 13 with JAVA, and Scala since 2009
- Developed an MVC-framework back in '99/2000, portal-framework, CMS with TopicMaps-integration, CRM/GroupWare/ProjectManagement.
- Lots of contacting-work, back-end, front-end.

Agenda

- What's “wrong” or missing in JEE?
- Scala and Lift added value
- Mixing JEE and Scala+Lift (Not all Java-frameworks are evil)
- Example-application – JSF, JPA, Oval, Spring, Hibernate, Scala, Lift

What's wrong with JEE?

i18n

- Only provides i18n in some sense in the presentation-layer
  - No common way to get i18n-messages in exceptions, services etc.
    => All projects make their own i18n-framework on top of *ResourceBundle* for use in exceptions/services
- I18n is not type-safe => Lots of strings scattered around the application
  - Makes refactoring fragile

What's wrong with JEE?

Error-handling

- No common exception-handling framework for all layers with good i18n-support
- Why still checked Exceptions?
- Strange (IMO) implementation of system and application-exceptions
  - ApplicationException:
    - All checked-exceptions
    - Un-checked and annotated with @ApplicationException
    - Does not rollback unless @ApplicationException(rollback=true)
  - SystemException:
    - java.rmi.RemoteException or RuntimeException and does not carry @ApplicationException
    - Always causes rollback
- EJB-3.1 spec. chapter 14 describing Exception-handling in JEE is 21 pages!
- Throwing checked exceptions behind DynamicProxies that are not declared results in UndeclaredThrowableException. Be aware of this when using frameworks throwing checked-exceptions from Scala, as the Scala-compiler will NOT complain about not declaring them!
- Too general exception-handling in web.xml. All frameworks implement their own exception-handling (JSF's version too complex)

JPA

- No standard for handling lazy-associations.
  - We want to navigate the object-graph retrieved from a repository (which often represents DomainObjects) in all parts of the application without worrying about whether we're in a persistence-context or not.
  - Spring provides some nice mechanisms like OpenEntityManagerInViewFilter and JpaInterceptor to minimize lady-load exceptions but after the connection to DB is closed, nasty things still might happen.
- persistence.xml (http://jcp.org/en/jsr/detail?id=220)
  - Has anyone succeeded in making environment neutral deployment-artifacts with a persistent-unit modularized into multiple jar-files using pure JEE?
  - Maintaining persistence.xml becomes a pain
    - <jar-file>/some/hardcoded/path/to/entities-in-functional-domain1.jar</jar-file>
    - <jar-file>/some/hardcoded/path/to/entities-in-functional-domain2.jar</jar-file>
  - Makes it difficult to make environment-neutral deployment-artifacts

JPA

- Too much boilerplate to make repositores, even when using the *generic DAO* pattern.

JAVA

```
UserRepository.java
public interface UserRepository extends GenericEntityRepository<User> {
}

UserRepositoryJpa.java
@Repository
public class UserRepositoryJpa extends GenericEntityRepositoryJpa<User> implements
UserRepository {
    public UserRepositoryJpa() {
        super(User.class);
    }
}
```

Scala

```
UserRepository.scala
@Repository
class UserRepositoryJpa extends UserRepository // To get constructor
trait UserRepository extends GenericEntityRepository[User]
```

GUI-frameworks

- JSF is the JEE-standard.
  - 2.0 was standardized in 2009. No new version has been standardized.
  - Tries to be a good component-based view-first framework but also suffers from being too complicated (who actually understands the component-model?).
  - Lots of bugs!
    - Try to use the jstl tags and see what breaks in AJAX-forms:-)
    - Some blocks are evaluated (although not shown) even when rendered=”false” is provided on the element.
  - **Allows logic in the view (xhtml-templates)**

What are the alternatives?

- Frameworks like Spring, Guice and Hibernate bring lots of missing peaces
- Struts2 and SpringMVC as alternatives don't solve much
  - AJAX-applications still require too much boilerplate and quickly become unstable
  - No good way to make true reusable components (re-usable AJAX-dialogs with different “on-close”-actions)
  - No server-push support (WebSockets, comet)
  - No type-safe i18n
  - Use of reflection and allowing logic in the view/templates makes applications very fragile and hard to maintain over time.
- Fragile code requires more tests

What has been done?

- Methodologies
  - SCRUM, Kanban etc.
- Guidelines and HOWTOs are supposed to compensate for bad or inadequate tools (or developers).
- Tests – lots of tests

  (how many have tests for all their screens in a web-app?)
- But – we're still struggeling with the same issues!! Results in spending too much time doing none-productive work.

What does Scala/Lift give us?

- Scala – the obvious stuff:
  - More consice syntax and structure, focus on the business-logic
  - Error-handling
    - Only un-checked exceptions! Be aware of frameworks throwing checked exceptions if using dynamic-proxies, will result in *UndeclaredThrowableException.*
    - We can pattern-match exceptions on anything (ie. traits), not just classes extending Throwable.
  - Better handling of “not set”, Option vs. null.
  - Functions are objects too, which can be partially applied before passed on
  - Ritcher type-system; Abstract types, type-projections, structural types etc.
  - implicits
  - Manifest – no need to pass MyClass.class around anymore in DAOs etc.
  - case classes, immutable per default (make up good DTOs and builders)
- Lift
  - The rest of this presentation

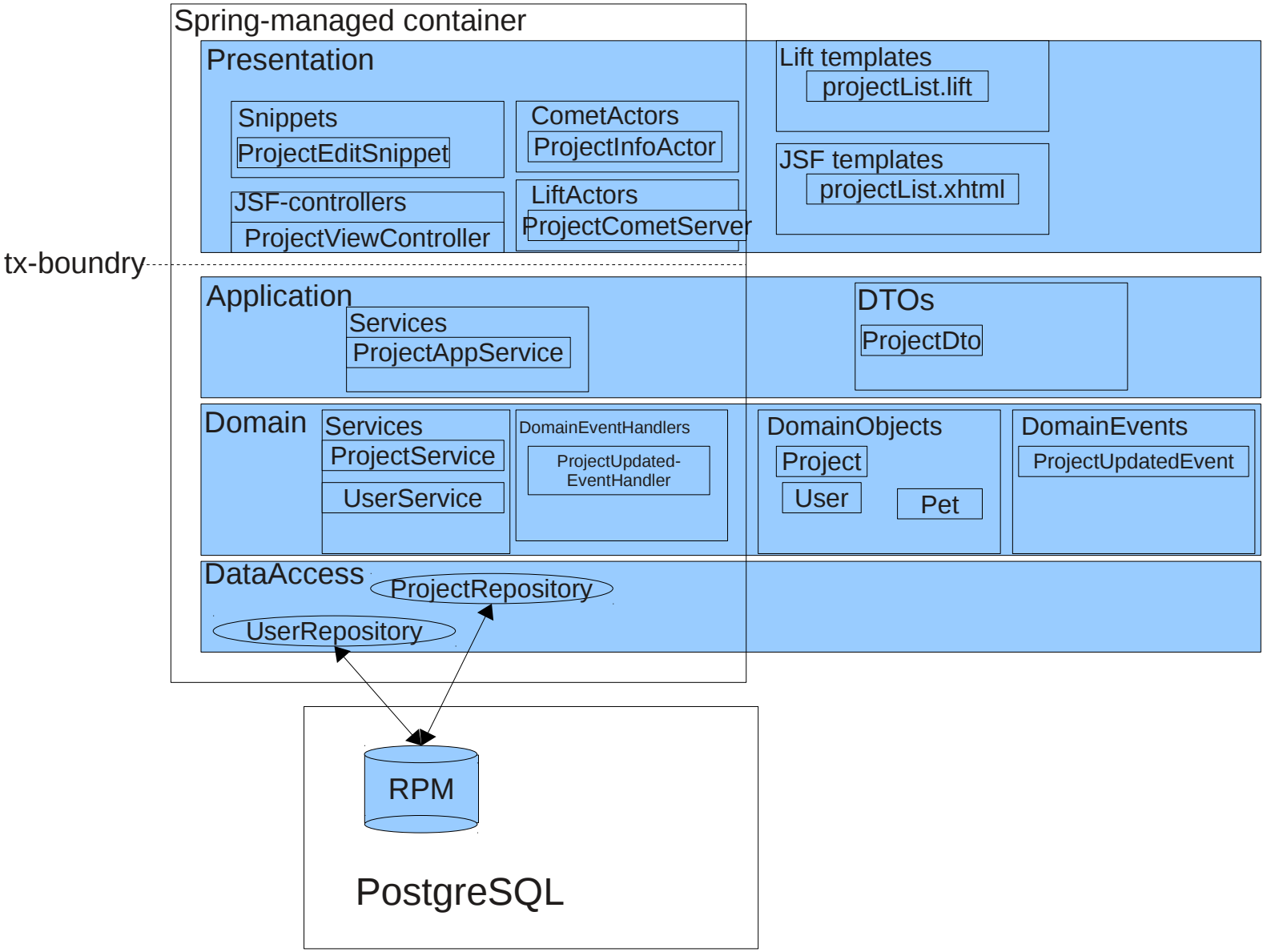
OfficeNet

RPM – Rolf's Project Management

Example application mixing JEE, Scala and Lift

- Spring
  - DI, AOP, tx-management, domain-event handling and “after successful commit callbacks”
- JPA
  - scala-jpa with Hibernate and Spring-ORM
  - LazyInitAspect – No more LazyInitializationException
- Spring Security for authentication and authorization (also in services)
- Using Scala's Option and Enumeration with JPA
- Type-safe i18n
- Type-safe and advanced form-fields with in-place AJAX-validation (onblur/onchange), automatic “required”-support, length-constraints and formatting
- Oval as validation-framework, also show validation on Option-types
- Simple yet robust exception-handling
- Server-push using Lift's comet-support
- JSF – Just to show it's possible, showcasing some AJAX and comet-stuff using Lift from a JSF-page

RPM – Architecture



RPM – Configuration – web.xml

In web.xml

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    classpath*:spring/*-datasource.xml
    classpath*:spring/*-context.xml
    /WEB-INF/spring-security-context.xml
  </param-value>
</context-param>


<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.xhtml</url-pattern>
</servlet-mapping>
```

RPM – Configuration – web.xml

In web.xml

```
<filter>
    <description>The Proxy that intercepts lift calls</description>
    <display-name>Lift proxy Filter</display-name>
    <filter-name>LiftProxyFilter</filter-name>
    <filter-class>no.officenet.example.rpm.web.filter.RegexpMappingFilter</filter-class>
    <init-param>
        <param-name>filterClass</param-name>
        <param-value>net.liftweb.http.LiftFilter</param-value>
    </init-param>
    <init-param>
        <!-- Match URIs not ending with .xhtml -->
        <param-name>matchPattern</param-name>
        <param-value>^(?!.*\.xhtml$).+$</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>LiftProxyFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

RPM – Security

- Use Spring Security for login (with remember-me) and controlling access to projectAppService.update
- Use standard jdbc-user-service
- Use jsr-250 for securing domain-services

RPM – Configuration - Spring

```
<context:property-placeholder location="classpath*:spring/props/*.properties"/>

<aop:aspectj-autoproxy/>
<context:spring-configured />
<context:load-time-weaver/>

<security:global-method-security jsr250-annotations="enabled" order="6"/>

<!-- Maps the following CDI scope annotations to matching Spring-scope
javax.enterprise.context.RequestScoped
javax.enterprise.context.SessionScoped
javax.enterprise.context.ApplicationScoped
 -->
<context:component-scan base-package="no.officenet.example.rpm"
  scope-resolver=
"no.officenet.example.rpm.support.infrastructure.spring.CdiScopeMetadataResolver"/>


<tx:annotation-driven order="5"/>
```

RPM – Validation

- Oval – http://oval.sf.net - provides more flexible validation than JSR-303
  - Validation-rules are business-rules and hence shall be a part of the domain-layer. Expressing validation-constraints as domain-constraints in the domain-objects prevents duplication of validation-rules in the presentation-layer.
- Full i18n support
- Custom validators for supporting Option-types
- Validation in WritableRepository.save() as safty-net
- In-place field-validation in forms

OfficeNet

RPM – Configuration - Oval

Lift Co.

Rpm-validation-context.xml

```
<bean id="rpmOvalExceptionTranslator"
class="no.officenet.example.rpm.support.infrastructure.errorhandling.RpmOvalExceptionTranslator
"/>

<bean id="ovalMessageResolver"
   class="no.officenet.example.rpm.support.infrastructure.validation.OvalMessageResolver"
   factory-method="getInstance">
  <property name="resourceBundles">
    <set>
<value>no.officenet.example.rpm.support.infrastructure.validation.oval.Messages</value>
<value>no.officenet.example.rpm.support.infrastructure.validation.oval.customValidationMessages
</value>
    </set>
  </property>
</bean>

<bean id="ovalValidator"
class="no.officenet.example.rpm.support.infrastructure.validation.OvalValidator">
  <constructor-arg index="0">
    <set>
      <bean
class="no.officenet.example.rpm.support.infrastructure.validation.RpmAnnotationsConfigurer"/>
    </set>
  </constructor-arg>
  <property name="exceptionTranslator" ref="rpmOvalExceptionTranslator"/>
</bean>
```

RPM – Configuration - Oval

```
RpmAnnotationsConfigurer
 /**
  * The only difference between this and
  * {@link net.sf.oval.configuration.annotation.AnnotationsConfigurer}
  * is that this implementation assues IsInvariant=true for all getter-based annotations
  */
```

```
OvalValidator
```

- Extends Oval's Validator but use our locale-selection strategy; Spring's LocaleContextHolder
- Don't do validation if not Hibernate.isInitialized()

RPM – JPA/Hibernate

- No XML. Only pure Spring

```
<bean id="RPM"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="rpmDataSource"/>
    <property name="persistenceUnitName" value="RPM"/>
    <property name="packagesToScan" value="no.officenet.example.rpm"/>
    <property name="jpaVendorAdapter" ref="jpaVendorAdapter"/>
    <property name="jpaDialect" ref="jpaDialect"/>
</bean>
```

The new *packagesToScan* property introduced in Spring 3.1 makes it way easier to split persistence-units in multiple modules (jars). No need for the old *MergePersistentUnitManager*

OfficeNet

Lift Co.

RPM – JPA/Hibernate

- Using Scala's Option-type as property in JPA-entities
  - OptionUserType
    - Convert the value to Some or None, never null
  - Write custom Oval-checks to deal with optional-values

```
class LongOptionUserType extends
     OptionUserType[java.lang.Long] {def nullableType =
StandardBasicTypes.LONG}
```

```
@Column(name = "budget")
@OptionalMax(value = 999999.0)
@org.hibernate.annotations.Type(`type` = CustomJpaType.LongOptionUserType)
var budget: Option[Long] = None
```

OfficeNet

Lift Co.

RPM – JPA/Hibernate

- Scala enums

```
class ProjectUserType extends EnumUserType(ProjectType)

object ProjectType extends
        EnumWithDescriptionAndObject[ProjectTexts.D.ExtendedValue] {
    val scrum = Value(ProjectTexts.D.type_scrum)
    val sales = Value(ProjectTexts.D.type_sales)
}

@Column(name = "project_type", nullable = false)
@org.hibernate.annotations.Type(`type` = CustomJpaType.ProjectUserType)
@net.sf.oval.constraint.NotNull
var projectType: ProjectType.ExtendedValue = null
```

Showing this field's i18n-value:

```
".projectType *" #> L(project.projectType.wrapped)
```

OfficeNet

Lift Co.

RPM – JPA - scala-jpa

Scala-JPA

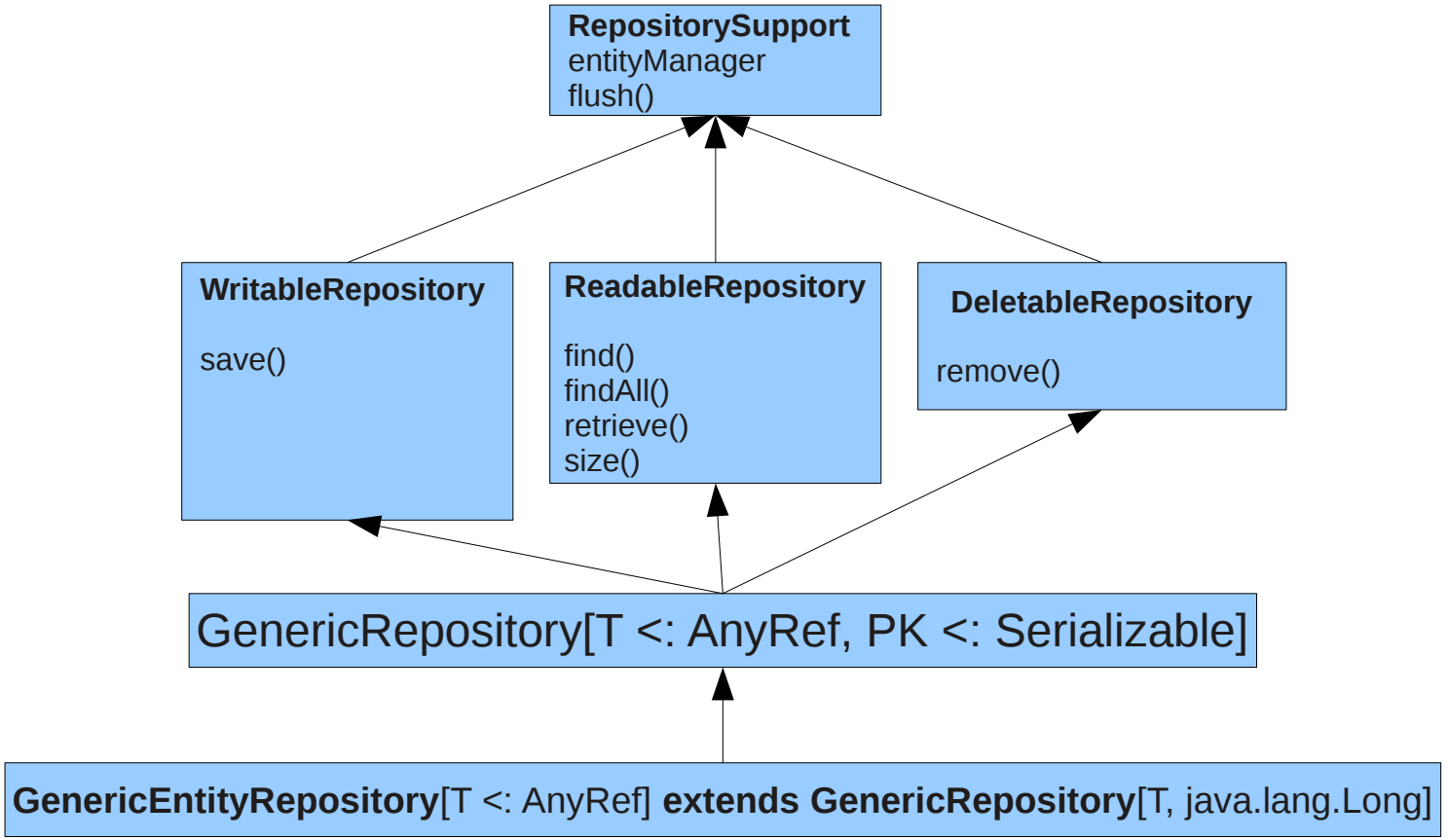
- We use our own custom implementation of EntityManager to expose certain JPA2-features as scala-jpa is not updated to support JPA-2

```
class ExtScalaEntityManager(owner: ScalaEMFactory, underlying: EntityManager)
      extends ScalaEntityManager {
    def em = underlying
    val factory = owner
    def getCriteriaBuilder = underlying.getCriteriaBuilder
    def createQuery[T <: AnyRef](criteriaQuery: CriteriaQuery[T]) =
            underlying.createQuery[T](criteriaQuery)
}
```

RPM – JPA - scala-jpa

- Use generic-dao pattern

```
@Repository
class ProjectRepositoryJpa extends ProjectRepository with PersistenceUnits.PersistenceUnitRPM
trait ProjectRepository extends GenericEntityRepository[Project]
```



RPM – JPA - scala-jpa

- Finders return Option[T] or Buffer[T]
- retrieve() throws ObjectNotFoundByPrimaryKeyException if not found
- Use scala's implicit Manifest[T] to avoid having to pass classOf[T] as parameter

```
trait ReadableRepository[T <: AnyRef, PK <: Serializable] extends RepositorySupport {
// ...
def retrieve(id: PK)(implicit m: Manifest[T]) =
        entityManager.find[T](m.erasure.asInstanceOf[Class[T]], id) match {
    case Some(e) => e
    case _ =>
        throw new ObjectNotFoundByPrimaryKeyException(m.erasure.getSimpleName,
                                                      id.toString)
}
}
```

```
trait UserRepository extends GenericEntityRepository[User] {

    def findByUserName(userName: String): Option[User] = {
        entityManager.createQuery[User]("SELECT u FROM User u WHERE u.userName = :userName")
            .setParams("userName" -> userName)
            .findOne
    }
```

RPM – Configuration - Lift

bootstrap.liftweb.Boot.scala

```
@Configurable // For Spring + AspectJ to inject the executor-service
class Boot {
    // Use custom executor-service (wired up in Spring) to be able to monitor
    // it using JMX. Lift's default is private so we need to install our own
    @Resource(name = "liftSchedulerExecutor")
    val liftSchedulerExecutor: ExecutorService = null

    def boot() {
        // Do nothing. We don't want Lift to try to mess up our logging.
        // Having log4j.xml in classpath is sufficient
        Logger.setup = Full(() => ())

        LiftRules.htmlProperties.default.set((r: Req) => new
                    XHtmlInHtml5OutProperties(r.userAgent))

        LiftRules.templateSuffixes = "lift" :: LiftRules.templateSuffixes

        LiftRules.addToPackages("no.officenet.example.rpm.web")

        // Reset i18n-cache on start of each request if dev-mode
        LiftRules.onBeginServicing.append(req => {
            Props.mode match {
                case Props.RunModes.Development =>
                    ResourceBundleHelper.resetCachedFormats()
                case _ =>
            }
        })
```

OfficeNet

RPM – Configuration - Lift

Lift Co.

Configure locale-handling

```
class Boot {
    def boot() {
…
…
        // Ensure the LocalContextHolder is reset on request-end
        LiftRules.onEndServicing.append((req, liftResponse) => {
            LocaleContextHolder.resetLocaleContext()
        })

        // Install our URI-based locale-calculator
        LiftRules.localeCalculator = UrlLocalizer.calcLocale

        SiteMap.enforceUniqueLinks = false
        LiftRules.setSiteMapFunc(() => SiteMap(RpmMenu.menu:_*))
```

RPM – Configuration - Lift

Configure exception-handling

Normal HTTP-requests

```
class Boot {
    def boot() {
         ExceptionHandlerDelegate.setUpLiftExceptionHandler()


 def setUpLiftExceptionHandler() {
 LiftRules.exceptionHandler.prepend {case (runMode, req, ex) =>
      ex match {
           case c: RpmConstraintsViolatedException =>
                if (req.acceptsJavaScript_? && req.section == LiftRules.ajaxPath) {
                     JavaScriptResponse(createValidationErrorDialog(c).open)
                } else {
                     XhtmlResponse(createValidationErrorPage(req.uri, ex, c),
                          S.htmlProperties.docType,
                          List("Content-Type" -> "text/html; charset=utf-8"), Nil, 500, S.ieMode)
                }
           case _ =>
                val localizableEx = handleException(log, ex)
                if (req.acceptsJavaScript_? && req.section == LiftRules.ajaxPath) {
                     JavaScriptResponse(createErrorDialog(localizableEx).open)
                } else {
                     XhtmlResponse(createHtmlErrorPage(req.uri, ex, localizableEx),
                          S.htmlProperties.docType,
                          List("Content-Type" -> "text/html; charset=utf-8"), Nil, 500, S.ieMode)
                }
      }
 }
 }
```

RPM – Error-handling

In comet-actors

```
override protected def exceptionHandler = {
    case c: RpmConstraintsViolatedException =>
        partialUpdate(ExceptionHandlerDelegate.createValidationErrorDialog(c).open)

    case ex =>
        val localizableEx = ExceptionHandlerDelegate.handleException(log, ex)
        partialUpdate(ExceptionHandlerDelegate.createErrorDialog(localizableEx).open)
}
```

In other actors

```
override protected def exceptionHandler = {
    case ex => ExceptionHandlerDelegate.handleException(log, ex)
}
```

RPM – Error-handling

Exception-handling

- Provide 2 main-traits

  trait **ApplicationException**
  trait **SystemException**

  self: Throwable =>

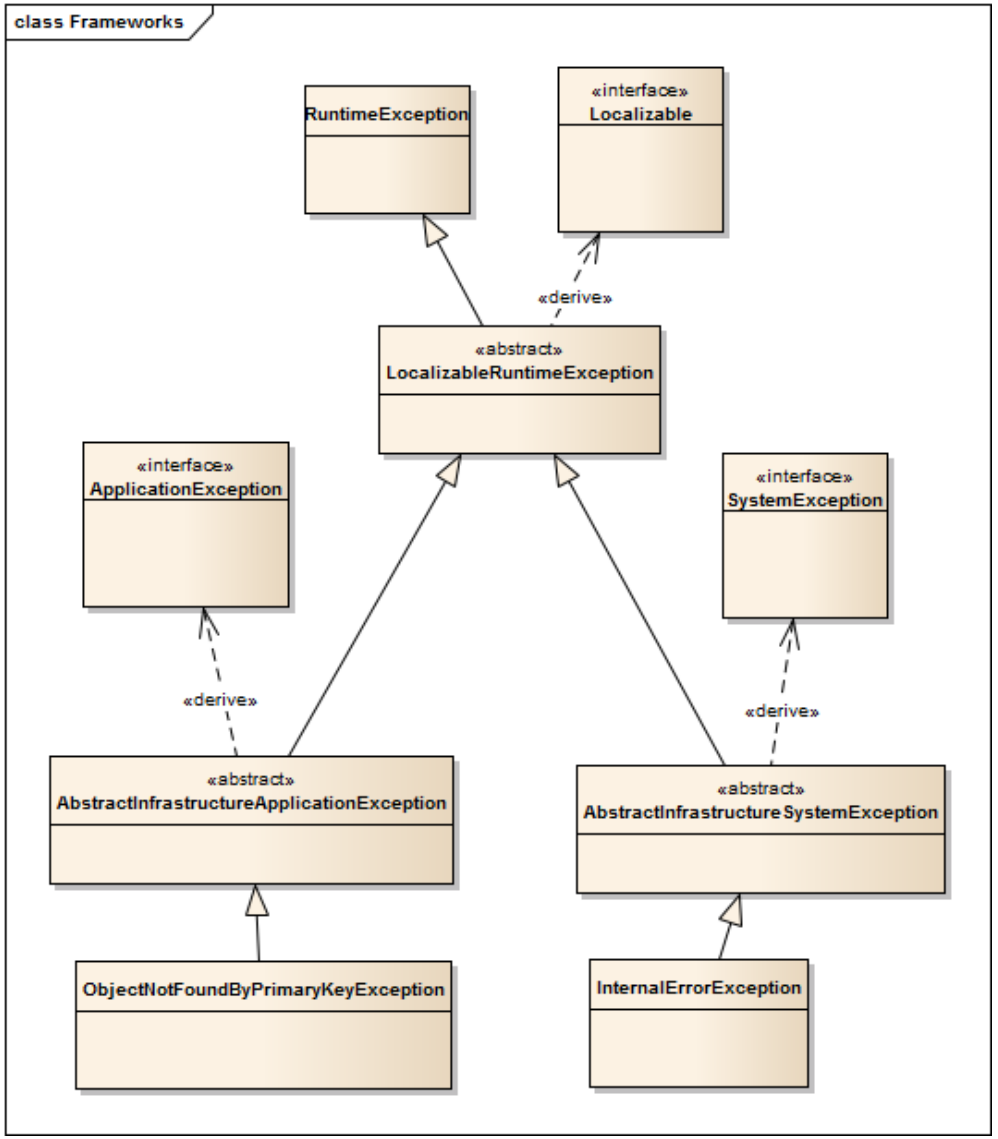
- Must support 3rd-party exception-hierarchies, like Spring's DataAccessException

```
public interface PersistenceExceptionTranslator {
  DataAccessException translateExceptionIfPossible(RuntimeException ex);
}
```

We see Spring expects `DataAccessException` in return so we must use traits for marking our exceptions as Application or System-Exceptions

```
class RpmDataIntegrityViolationException(val constraintName: String, cause: Throwable)
    extends DataIntegrityViolationException(constraintName, cause)
    with ApplicationException with Localizable
```

RPM – Error-handling



RPM – i18n

- Why is Lift's i18n-support inadequate for RPM
  - The other modules in the application don't have a dependency to Lift
  - Domain-services, exceptions etc. also need i18n
  - Doesn't play well with JSF (or other frameworks)
    - Uses String.format instead of standard java.text.MessageFormat, different rules
    - All texts are merged, no way to specify resource-bundle
  - Doesn't support “choice-format” or markup in text

```
Showing {0, choice, 0#{0, number} activities | 1#{0, number} activity | 1<{0, number} activities} for project <strong>{1}</strong>
```

  - Not type-safe, lots of strings are spread across the application

OfficeNet

RPM – I18n using Scala-enums

Lift Co.

Advantages:

- Type-safe
- Easy to test for missing keys and refactor

```
object Bundle extends ResourceBundleNameEnum {

    // The global resource-bundle. Holds texts for general use. Used in GlobalTexts
    val GLOBAL = BundleName("no.officenet.example.rpm.resources.globalResources")

    // Holds texts for Project (D) domain object fields. Used in ProjectTexts.D
    val PROJECT_D = BundleName("no.officenet.example.rpm.resources.projectDomainResources")

    // Holds texts for Project (V) views (project-related pages). Used in ProjectTexts.V
    val PROJECT_V = BundleName("no.officenet.example.rpm.resources.projectViewResources")
}
```

RPM – I18n using Scala-enums


```
object GlobalTexts extends ResourceBundleEnum {
    val
    logged_in_user,
    dateformat_fullDateTimeSeconds,
    button_edit
    = BundleEnum(Bundle.GLOBAL)
}
```

The key in the properties-file is the name of the enum:

```
/no/officenet/example/rpm/resources/globalResources_en.properties
dateformat_fullDateTimeSeconds=MM.dd.yyyy HH:mm:ss
```

RPM – I18n in snippets


```
import no.officenet.example.rpm.support.domain.i18n.Localizer._
```

```
<span class="activitiesForProjectHeader">
    Showing activities for project <strong>HEY</strong>
</span>
```

```
".activitiesForProjectHeader" #> L_!(ProjectTexts.V.header_activitiesForProject_text,
                                     project.activityList.size, project.name)
```

```
header_activitiesForProject_text=Showing {0, choice, 0#{0, number} activities
| 1#{0, number} activity | 1<{0, number} activities} for project
<strong>{1}</strong>
```

Output:
Showing 2 activities for project **SomeProject**
Showing 1 activity for project **SomeOtherProject**

RPM – I18n in templates


```
object ProjectTexts {
    object V extends ResourceBundleEnum {
        val
        label_chosenColor,
        label_niceColor,
        label_badColor

        = BundleEnum(Bundle.PROJECT_V)
    }
}
```

```
<div>
  <div class="lift:i18n?bundle=PROJECT_V;key=label_chosenColor">
        This text will be replaced
    </div>
  <span class="nice_color_id lift:i18n.i?bundle=PROJECT_V"
          style="display: none">label_niceColor</span>
  <span class="bad_color_id lift:i18n.i?bundle=PROJECT_V"
          style="display: none">label_badColor</span>
</div>
```

Output:

```
<div>
  The color you've chosen is a:
  <span style="display: none">Nice color</span>
  <span style="display: none">Bad color</span>
</div>
```

Don't use i18n in templates; It's not type-safe!!

RPM – I18n Testing

Testing

```
class ProjectTextsTest extends TextsTest {
    @Test
    def testDomain() {
        assetResourceBundleEnumFound(ProjectTexts.D)
    }
    @Test
    def testView() {
        assetResourceBundleEnumFound(ProjectTexts.V)
    }
}
```

java.lang.AssertionError: problems found:
ArrayBuffer(
(PROJECT_V, en, color_black),
(PROJECT_V, en, color_red),
(PROJECT_V, no_NO, label_badColor)
)

RPM – ExecutionContext

- Need to know the “current locale”
  - Use Lift's locale-calculator
  - Need other way of propagating locale to services, use Spring's LocaleContextHolder
- Need to know the “current user”
  - Spring Security's SecurityContextHolder.getContext.getAuthentication

RPM – ExecutionContext

- Need to set locale at start of HTTP-threads

http://localhost:8080/rpm/**no**/**project**/**100**

```
object ProjectLoc extends Loc[ProjectParam] with LocalizableMenu {

…
  override val rewrite: LocRewrite = Full(NamedPF("Project rewrite") {
    case RewriteRequest(ParsePath(UrlLocalizer(locale) :: "project" :: AsLong(projectId) ::
Nil,_,_,_),_,_) => {
      (RewriteResponse("lift" :: "project" :: "projectView" :: Nil, Map("id" ->
projectId.toString), true), ProjectViewParam(projectId))
    }
  })
}
```

```
UrlLocalizer
    def unapply(in: String): Option[Locale] = {
        val locale = locales.get(in)
        locale.foreach{l =>
            LocaleContextHolder.setLocale(l)
            currentLocale.set(l)
        }
        locale
    }
```

RPM – ExecutionContext

- Make S.locale (which delegates to this function) work in comet-requests

```
object currentLocale extends RequestVar(Locale.getDefault)

def calcLocale(in: Box[HTTPRequest]): Locale =
    if (LocaleContextHolder.getLocaleContext != null) {
        // Set by LoanWrapper in Comet-requests
        val locale = LocaleContextHolder.getLocale
        locale
    } else if (currentLocale.set_?) {
        // Set by the UrlLocalizer.unapply extractor in Locs
        val locale = currentLocale.get
        LocaleContextHolder.setLocale(locale)
        locale
    }
    else {
        // Use the browser's locale or system's default
        in.flatMap(r => r.locale).openOr(Locale.getDefault)
    }
```

RPM – ExecutionContext

- Convention for Comet-actor's name

  name=<locale>:<id>

```
class ProjectPageSnippet(projectParam: ProjectParam)

def render(in: NodeSeq) = {
    // The locale too has to be a part of the comet's name.
    // Else switching locale doesn't affect the actor
    val cometName = List(S.locale, projectParam.id)
    <div class={"lift:comet?type=ProjectInfoActor;name="+cometName.mkString(":")}
         style="display: inline;">
        {in}
    </div>
}
```

RPM – ExecutionContext

- Initialize locale and user for the comet-actor

```
trait RpmCometActor extends CometActor with CometListener

// First part of name is always the locale
lazy val nameParts = name.open_!.split(":")
 // Get from 1st part of the actor's name
lazy val locale = UrlLocalizer.locales.get(nameParts(0))
var authentication: Box[Authentication] = Empty

override protected def localSetup() {
    super.localSetup()
    authentication = S.session.flatMap(ls => {
        ls.httpSession.flatMap(session => {
            tryo{session.attribute(HttpSessionSecurityContextRepository
                .SPRING_SECURITY_CONTEXT_KEY)
                .asInstanceOf[SecurityContext].getAuthentication}
        })
    })
}
```

RPM – ExecutionContext

- Need to set locale and current user at start of Comet-threads

```
trait RpmCometActor extends CometActor with CometListener

lazy val locale: Option[Locale] = UrlLocalizer.locales.get(nameParts(0))

override protected def aroundLoans: List[CommonLoanWrapper] = {
  val lw = LoanWrapperHelper.getLoanWrapper(() => locale)
  val cometLW = new LoanWrapper  {
    def apply[T](f: => T) : T =  {
      authentication.foreach(auth =>
        SecurityContextHolder.getContext.setAuthentication(auth))
        try f
        finally SecurityContextHolder.clearContext()
    }
  }
  cometLW :: lw :: Nil
}
```

Must set Auth-context on the thread for jsr250-based ACL to work. Or else we wouldn't be able to update a project from a comet-related AJAX-req.

```
@RolesAllowed(Array("PROJECT_MANAGER"))
def update(projectDto: ProjectDto): ProjectDto
```

RPM – Lift and forms


```
<form class="lift:form.ajax">
 <div class="lift:project.ProjectSnippet.list">
  <div class="firstName"></div>
 </div>
</form>
```

- Shtml.text etc.
  - Raw building-blocks for forms
    `".firstName *" #> Shtml.text(firstName, s => firstName = s)`
- Lift provides LiftScreen and Wizard, but..
  - None of them provide dynamic creation of form-elements.
  - No built-in mechanism for hooking in Oval or other external validation frameworks

OfficeNet

RPM – Lift and forms

Lift Co.

ValidatableScreen

- Composable form-field generation
- Type-safe validation
- Know how to render in “non-edit”-mode
- Has “disabled”-mode for all types
- Supports date-picker with localized format
- Select-box with ajax-callback and assignment-callback on submit
- Ability to render label + input-field with different type of containers (TD, DIV or other custom stuff)

RPM – Lift - validation

- Using external validation-libraries with Lift
  - Mix-in the **ValidatableScreen** trait which provide methods for generating input-fields which are validation-aware. May provide validation-functions for each field to perform validation.
  - Mix-in the **JpaFormFields**-trait to get type-safe field-names AND external validation (Oval here).

```
JpaTextField(project, Project.name, project.name, (v:String) => project.name = v)
```

Project.name is defined in Project's companion-object and represents the field-name in a type-safe way.

```
object Project {
     object name extends StringField[Project]
     object description extends StringField[Project]
     object budget extends LongField[Project]
     object estimatedStartDate extends DateTimeField[Project]
}
```

```
Project.name serves two purposes here:
1.
Provide the external validation-library information about which property on the
object is to be validated in a type-safe way.
2.
Dynamic forms; Servers as a unique key, together with the reference (as in
identityHashCode) of the validated object itself (project), to be able to group
field-errors for each input-field. Makes it support editing multiple instances of a
class with the same field (ie. 2 projects) in the same form.
```

RPM – Lift - validation

- Template defining markup for editing a project in an AJAX-dialog

web/src/main/webapp/lift/project/_projectEdit.lift

```
<table>
  <tbody>
  <tr class="projectName"></tr>
  <tr class="projectDescription"></tr>
  <tr class="projectType"></tr>
  <tr class="budget"></tr>
  <tr class="estimatedStart"></tr>
  </tbody>
</table>
```

RPM – Lift - validation

Example of how a snippet binds form-fields to a template for editing a Project-object

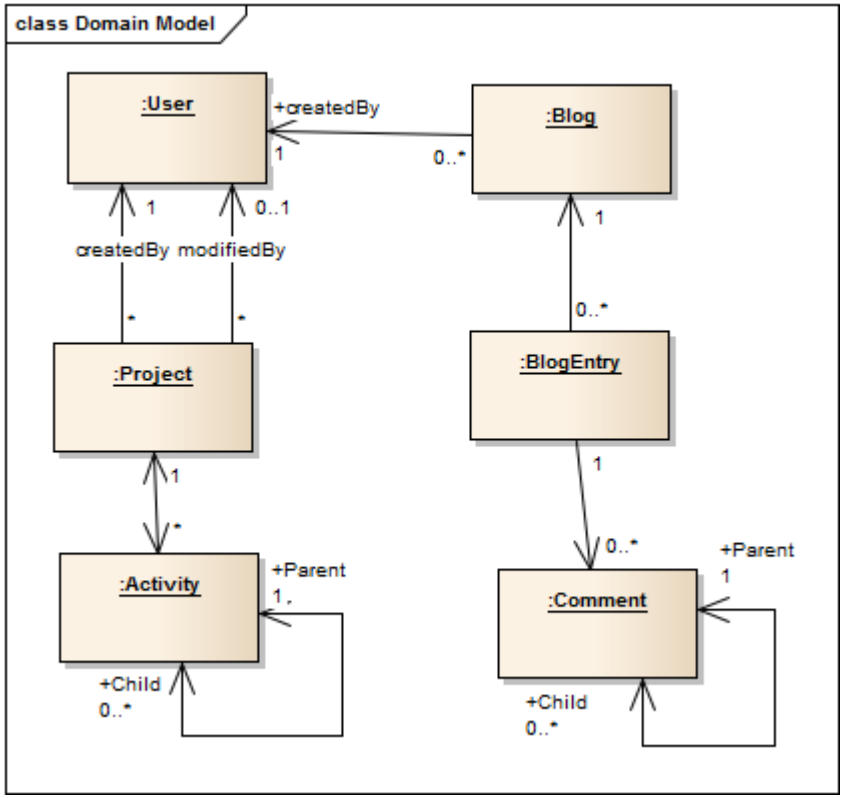
```
".projectName *" #> JpaTextField(project, Project.name, project.name,
  (v:String) => project.name = v).
  withContainer(TdInputContainer(L(ProjectTexts.D.name))) &
".projectDescription *" #> JpaTextAreaField(project, Project.description,
  project.description.getOrElse(""), (v: Option[String]) => project.description = v).
  withContainer(TdInputContainer(L(ProjectTexts.D.description))) &
".projectType *" #> JpaSelectField(project, Project.projectType,
  projectTypes.toList, project.projectType,
  (pt: ProjectType.ExtendedValue) => project.projectType = pt,
  (pt: ProjectType.ExtendedValue, idx) => L(pt.wrapped)).
  withContainer(TdInputContainer(L(ProjectTexts.D.projectType))) &

".budget *" #> JpaTextField(project, Project.budget,
    project.budget.map(d => d.toString).getOrElse(""),
    (v: Option[Long]) => project.budget = v).
    withContainer(TdInputContainer(
    L(ProjectTexts.V.projectDialog_details_label_budget))).
    withInputMask(NaturalNumberMask()) &
".estimatedStart *" #> JpaDateField(project, Project.estimatedStartDate,
    Localizer.formatDateTime(L(GlobalTexts.dateformat_fullDate),
        Option(project.estimatedStartDate)).getOrElse(""),
    (v: DateTime) => project.estimatedStartDate = v).
    withContainer(TdInputContainer(
    L(ProjectTexts.V.projectDialog_details_label_estimatedStartDate) +
        "(%s)".format(L(GlobalTexts.dateformat_fullDate)))) &
```

RPM – Validation

Benefits

- Automatically extract validation-rules from Oval-annotations
- Provides in-place validation with fully i18n messages on form-fields (“onblur” for text-fields and “onchange” for selects)
  - Optionally turn off in-place validation per field
- Automatically extract “required” information for use of rendering a * next to required fields
- Automatically extract max-length information for input-fields
- Automatically format numbers and date-inputs according to locale
- Automatically apply input-masks to enforce only legal characters (ie. Numbers), using the jQuery autoNumber-plugin
- Conversion-errors
- Having Scala in our toolbox enables us to pass in an optional validation-function and error-message function to each form-field
- Type-safe field-validation

RPM – Domain Model



RPM – Live Demo

- Use URI-based locale-selection for Lift-pages.
- Project-list page - Lift
- Project-details page - Lift
- Use Lift-backed AJAX-dialog with Oval validation – Lift
- Show project-list and detail in JSF
  - Shortly explain JSF-configuration here
- Show usage of the same Lift-based project-edit dialog in JSF-page
- Use Lift's Comet-support for updating project-details page, both on Lift and JSF version
  - Updating the GUI happens whenever someone saves a project using the domain-service ProjectService.update(project) and it successfully commits.

JSF

- Configuration

```
<application>
  <!-- This makes JSF resolve el-expressions to Spring-beans,
         enabling spring-beans to act as spring-managed JSF-controllers -->
  <el-resolver>org.springframework.web.jsf.el.SpringBeanFacesELResolver</el-resolver>
  <locale-config>
    <default-locale>en</default-locale>
    <supported-locale>no</supported-locale>
    <supported-locale>en</supported-locale>
  </locale-config>
  <!-- Message properties components-->
  <resource-bundle>
     <base-name>no.officenet.example.rpm.resources.globalResources</base-name>
     <var>GLOBAL</var>
  </resource-bundle>
  <resource-bundle>
     <base-name>no.officenet.example.rpm.resources.projectDomainResources</base-name>
     <var>PROJECT_D</var>
  </resource-bundle>
  <resource-bundle>
     <base-name>no.officenet.example.rpm.resources.projectViewResources</base-name>
     <var>PROJECT_V</var>
  </resource-bundle>
</application>
```

JSF

- Spring-managed controllers

```
@Controller
@Scope("view")
class ProjectViewController @Autowired() (projectAppService:
ProjectAppService) {
    case class ProjectBean(projectDto: ProjectDto) {
        val project = projectDto.project
        @BeanProperty
        var id = project.id
        @BeanProperty
        var name = project.name
        @BeanProperty
        var description = project.description.orNull
        @BeanProperty
        var created = project.created
        @BeanProperty
        var createdBy = project.createdBy.displayName
    }
    @BeanProperty
    var id: java.lang.Long = null
    private var project: ProjectBean = null
    def getProject = {
        if (project == null) {
            project = ProjectBean(projectAppService.retrieve(id))
        }
        project
    }
}
```

JSF

- Hidden iframe trick for Comet

```
<iframe id="project_view_jsf_iframe" name="project_view_jsf_iframe"
    src="${facesContext.externalContext.requestContextPath}/$
{configController.locale}/wrapper/project/projectViewWrapperForJSF?
id=#{projectViewController.id}"
    height="0"
    width="0"
    frameborder="0"
  ><!-- --></iframe>
```

Use a `ProjectViewWrapperLoc` handling this URL:

```
ParsePath(UrlLocalizer(locale) :: "wrapper" :: "project" :: "projectViewWrapperForJSF" :: Nil
RewriteResponse("lift" :: "project" :: "projectViewWrapperForJSF"
```

Template in:
/src/main/webapp//lift/project/projectViewWrapperForJSF.lift

```
<div class="lift:ProjectJSFHelperSnippet?eager_eval=true">
    <div class="lift:embed?what=lift/project/_projectDetailViewComponent"></div>
</div>
```

This **`ProjectJSFHelperSnippet`** creates a ProjectJsfActor which will send JS-events to the iframe's parent
Embeds same template as /lift/project/projectView.lift (the Lift-version) does

OfficeNet

Lift Co.

JSF

Making Lift AJAX-dialogs work on JSF-pages

1. Load Lift's liftAjax.js
2. Register some JS-variables and functions Lift uses
3. Define a JS funciton which loads a Lift-managed URL using AJAX
4. Make a custom Loc in Lift to handle this URL
5. Invoke a Snippet in the template the Loc rewrites to for setting correct value of the “lift_page” JS-var and also pupulate RequestVars if needed

OfficeNet

JSF

Lift Co.

Making Lift AJAX-dialogs work on JSF-pages

1. Register liftAjax.js

```
<script src="$
{facesContext.externalContext.requestContextPath}/#{configController.liftAjaxPath}/liftAjax.js"
type="text/javascript"></script>
```

2. Register some JS-variables and functions Lift uses

```
<script type="text/javascript">
  //<![CDATA[
  jQuery(document).ready(function() {liftAjax.lift_successRegisterGC();});
  var lift_page = "NO_PAGE";
  //]]>
```

3. funciton which loads a URL using AJAX

```
function openLiftPopup(template, params) {
// ...
}
</script>

<a href="javascript:void(0)"
   onclick="openLiftPopup('project/projectEditDialogWrapper',
           {id: #{projectViewController.project.id}}); return false">
    EDIT (Lift-popup)
</a>
```

JSF

Making Lift AJAX-dialogs work on JSF-pages

4. Make a custom Loc in Lift to handle this URI

```
ProjectEditDialogWrapperLoc:

ParsePath(UrlLocalizer(locale) :: "wrapper" :: "project" :: "projectEditDialogWrapper"
RewriteResponse("lift" :: "project" :: "projectEditDialogWrapperForJSF"
```

OfficeNet

JSF

Lift Co.

Making Lift AJAX-dialogs work on JSF-pages

5. Invoke a Snippet in the template the Loc rewrites to for setting correct value of the “lift_page” JS-var and also pupulate RequestVars if needed

projectEditDialogWrapperForJSF.lift

```
<div class="lift:project.ProjectEditDialogWrapperSnippet.render">
  <span class="liftPageSetter"/>
  <div class="lift:embed?what=lift/project/_projectEdit"></div>
</div>
```

```
@Configurable
class ProjectEditDialogWrapperSnippet {
  @Resource
  val projectAppService: ProjectAppService = null
  lazy val projectId: Box[Long] = asLong(S.param("id"))

  def render(ns: NodeSeq): NodeSeq = {
    // Set the projectDto in a RequestVar
    projectId.foreach(id => ContextVars.projectVar.set(projectAppService.retrieve(id)))
    // Hook up JavaScript to set the “lift_page” JS-var correctly
    (".liftPageSetter" #> Script(SetExp(JsVar("lift_page"), S.renderVersion))).apply(ns)

  }
}
```

RPM – Domain Events for Comet

Use-case:
We want to update GUI after successful save/update (transaction-commit)

Soluiton:
Use Domain Events and implement an event-handler in the web-module for doing the comet-updates

Important:
Use IMMUTABLE data-structures in messages sent to actors. Trying to use Hibernate-managed entities will cause all kinds of trouble, especially when using LazyInitAspect.

RPM – Domain Events for Comet

In the [**on-example-rpm-projectmgmt-domain**] artifact, which doesn't see the web-artifact

```
override def update(project: Project) = {
     val updated = super.update(project)
     DomainEventDispatcher.raiseEvent(new ProjectUpdatedEvent(updated, OperationType.UPDATE))
     updated
}


@Component
class ProjectUpdatedEventHandler extends DomainEventHandler[ProjectUpdatedEvent] {
     DomainEventDispatcher.registerEventHandler(classOf[ProjectUpdatedEvent], this)
     def handleEvent(event: ProjectUpdatedEvent) {
          AfterCommitEventDispatcher.registerAfterCommitEvent(event)
     }
}
```

In the [**on-example-rpm-web**] artifact

```
@Component
class ProjectUpdatedForCometEventHandler extends DomainEventHandler[ProjectUpdatedEvent] {

     AfterCommitEventDispatcher.registerEventHandler(classOf[ProjectUpdatedEvent], this)

     def handleEvent(event: ProjectUpdatedEvent) {
          if (OperationType.UPDATE == event.operationType) {
               // Send actor a message for comet-updates
               ProjectCometMasterServer.findProjectCometServerFor(event.project.id)
                   .foreach(_ ! ProjectCometDto(event.project))
          }
     }
}
```

OfficeNet

RPM – Domain Events

Lift Co.

```
class ProjectCometServer(id: Long) extends LiftActor with ListenerManager with Loggable {

     override def lowPriority = {
          case project: ProjectCometDto =>
               cachedProject = project
               updateListeners(project) // Update all subscribed comet-actors
     }

}
```

?

Github:
git@github.com:andreak/on-example-rpm.git
git@github.com:andreak/33degree-2012.git

RPM – JPA/Hibernate

- LazyInitAspect
  - Prevents LazyInitializationException and lets you navigate the object-graph in a natural way without resorting to unwanted calls to repositories or services

META-INF/aop.xml

```
<aspectj>
  <weaver options="-verbose -showWeaveInfo">
          <include within="@org.springframework.beans.factory.annotation.Configurable no.officenet.example.rpm..*"/>
          <include within="@org.springframework.beans.factory.annotation.Configurable bootstrap.liftweb.Boot"/>
          <include within="@javax.persistence.Entity *"/>
          <include within="@javax.persistence.MappedSuperclass *"/>
          <include within="no.officenet.example.rpm.support.infrastructure.spring.aop.LazyInitAspect"/><!-- For aspectOf -->
  </weaver>
  <aspects>
    <aspect name="no.officenet.example.rpm.support.infrastructure.jpa.LazyInitAspect"/>
  </aspects>
</aspectj>
```

```
@Aspect
public class SystemArchitectureAspect {

  @Pointcut("(@within(javax.persistence.Entity) || @within(javax.persistence.MappedSuperclass))" +
       " && (" +
       "@annotation(javax.persistence.ManyToOne)" +
       " || @annotation(javax.persistence.ManyToMany)" +
       " || @annotation(javax.persistence.OneToMany)" +
       " || @annotation(javax.persistence.OneToOne)" +
       ")")
  public void lazyLoadableJpaProperties() {
  }
```