SEAM IN ACTION

DAN ALLEN

JBoss, A DIVISION OF **RED HAT**

Part 1

Flyover

Seam Brings Java EE Together

JSF EJB 3

Seam's Mission

"To provide a fully integrated development platform for building rich Internet applications based upon the Java EE environment." – Gavin King

http://in.relation.to/Bloggers/Seam3

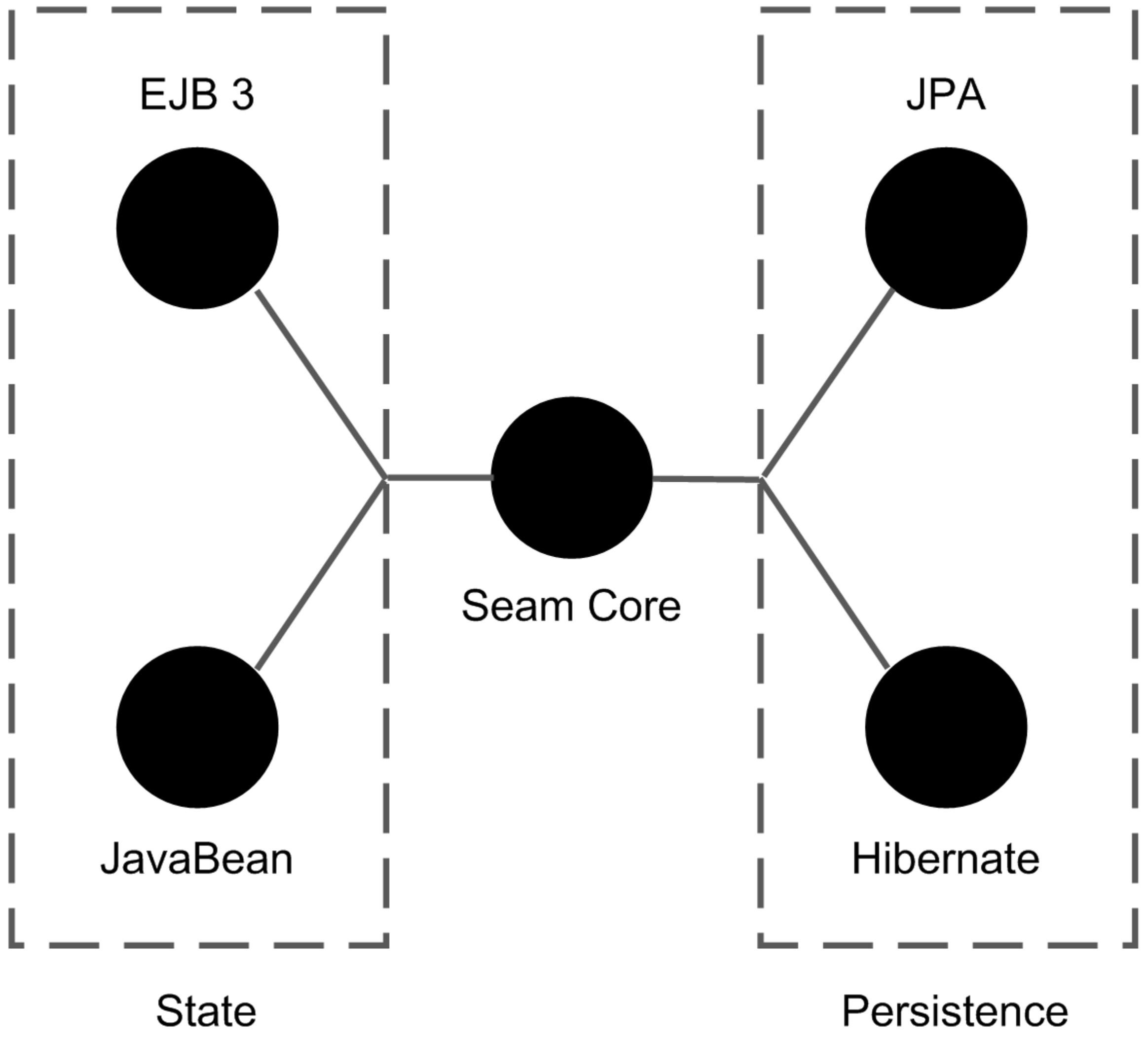
DEFINE "FRAMEWORK"

- The "glue"
  - Contextual component model
  - Dependency injection framework
  - Enhanced EL
- Simplifies trouble spots in Java EE
  - Expands declarative programming model
  - Manages Java persistence correctly
- Integrates third-party libraries
- Delivers rapid development
  - Project generator, CRUD framework, hot deploy, Java EE test environment

SEAM TODAY

- JavaServer Faces (JSF)
- Facelets (standardized in JSF 2)
- Java Persistence API (JPA)
- Enterprise JavaBeans 3 (EJB 3)
- Java Transaction API (JTA)
- RESTeasy (JAX-RS)
- jBPM
- JavaMail
- iText (PDF)
- JExcelApi (Excel)
- YARFRAW (RSS)
- Java Authentication and Authorization Service (JAAS)
- Hibernate Search
- Drools
- JavaScript / Ajax
- Quartz
- Wicket
- GWT
- Groovy (interpreted)
- Spring framework
- JBoss Cache
- ...and more

Seam's Degrees of Freedom



EJB Functionality

WITHOUT EJBs

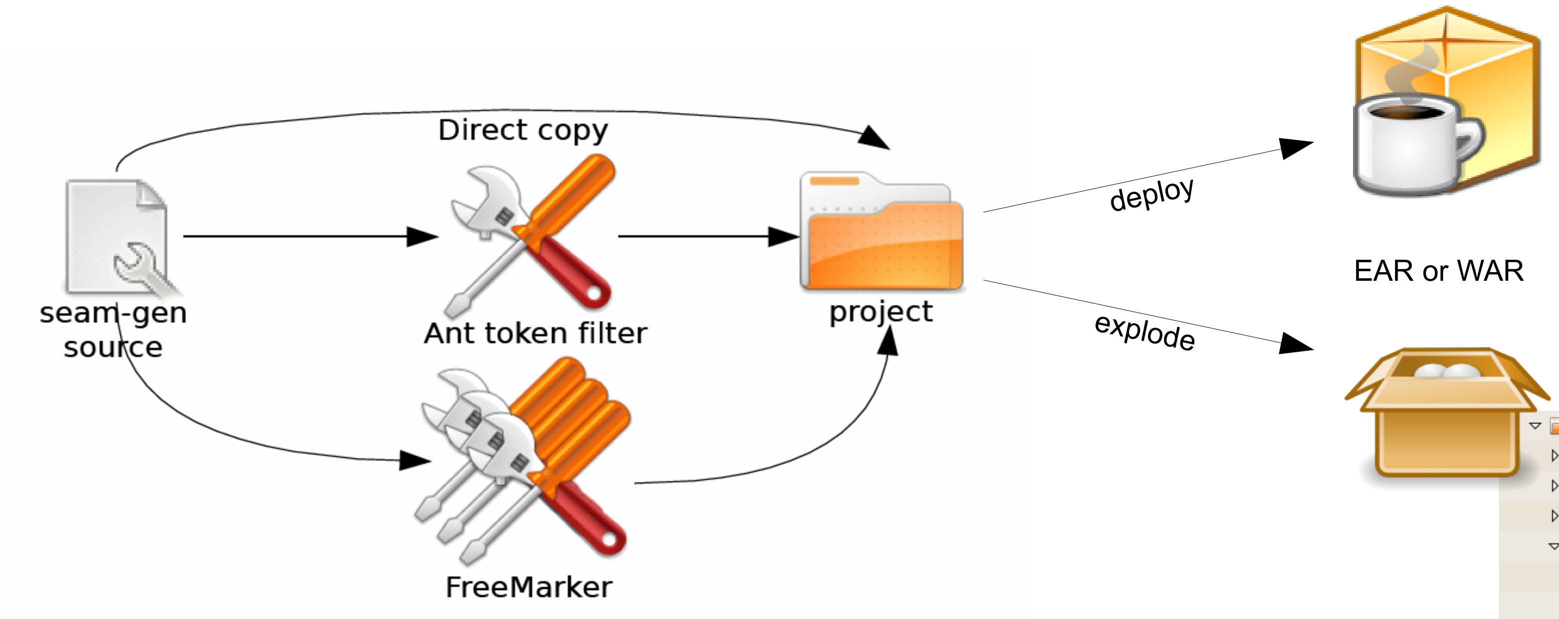
KEY INNOVATIONS IN SEAM

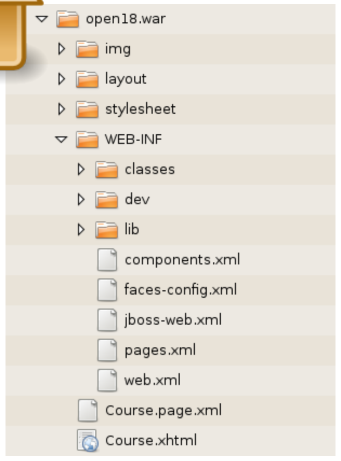
- Conversations and workspaces
  - Solves challenge of state management
- Business process integration
- Component events
  - Can be deferred until transaction completion
  - Can be asynchronous
- XHTML → PDF, Excel, RSS, Charts, ...
  - via Facelets compositions
- Hot deploy classloader

PART 2

TEEING OFF WITH SEAM

seam-gen



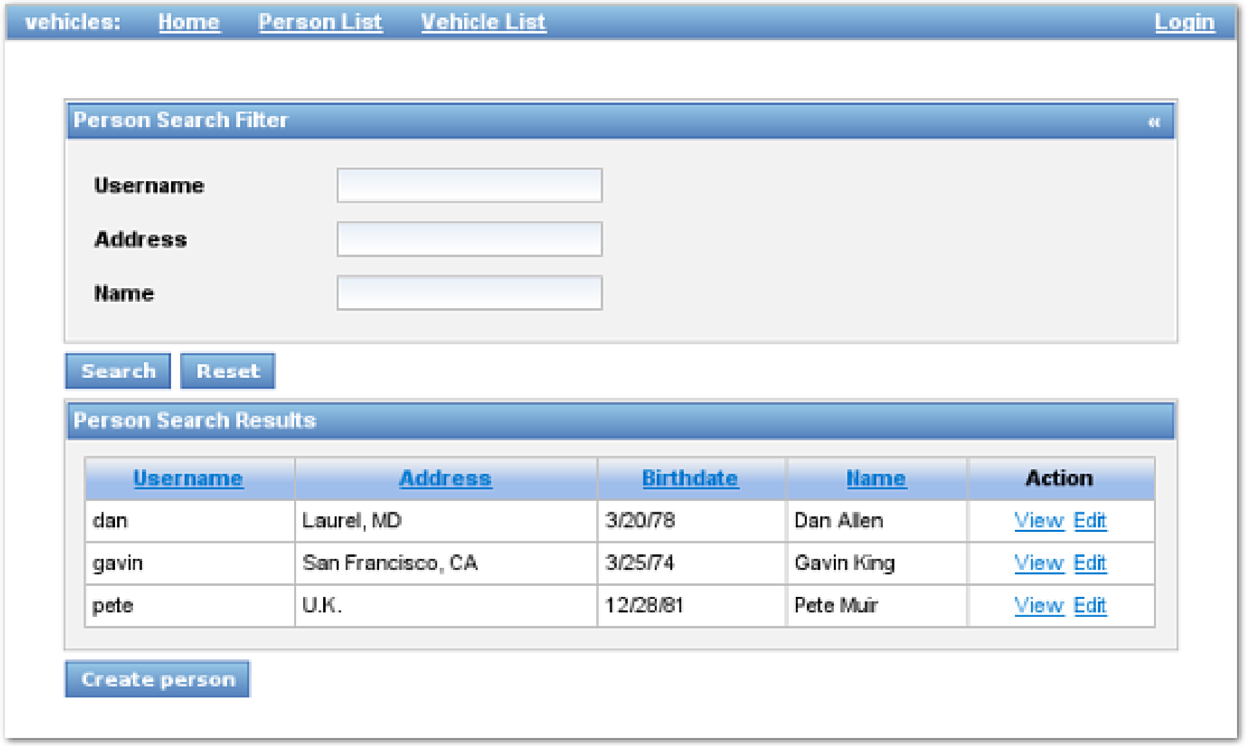


```
open18.war
  img
  layout
  stylesheet
  WEB-INF
    classes
    dev
    lib
    components.xml
    faces-config.xml
    jboss-web.xml
    pages.xml
    web.xml
  Course.page.xml
  Course.xhtml
```

- Creates IDE project files
  - Eclipse, NetBeans and IntelliJ IDEA
- Deploys to JBoss AS (default)
  - Incremental hot deployment for "instant change"
- Prepares three profiles: dev, prod, test
  - Test profile bootstraps Embedded JBoss

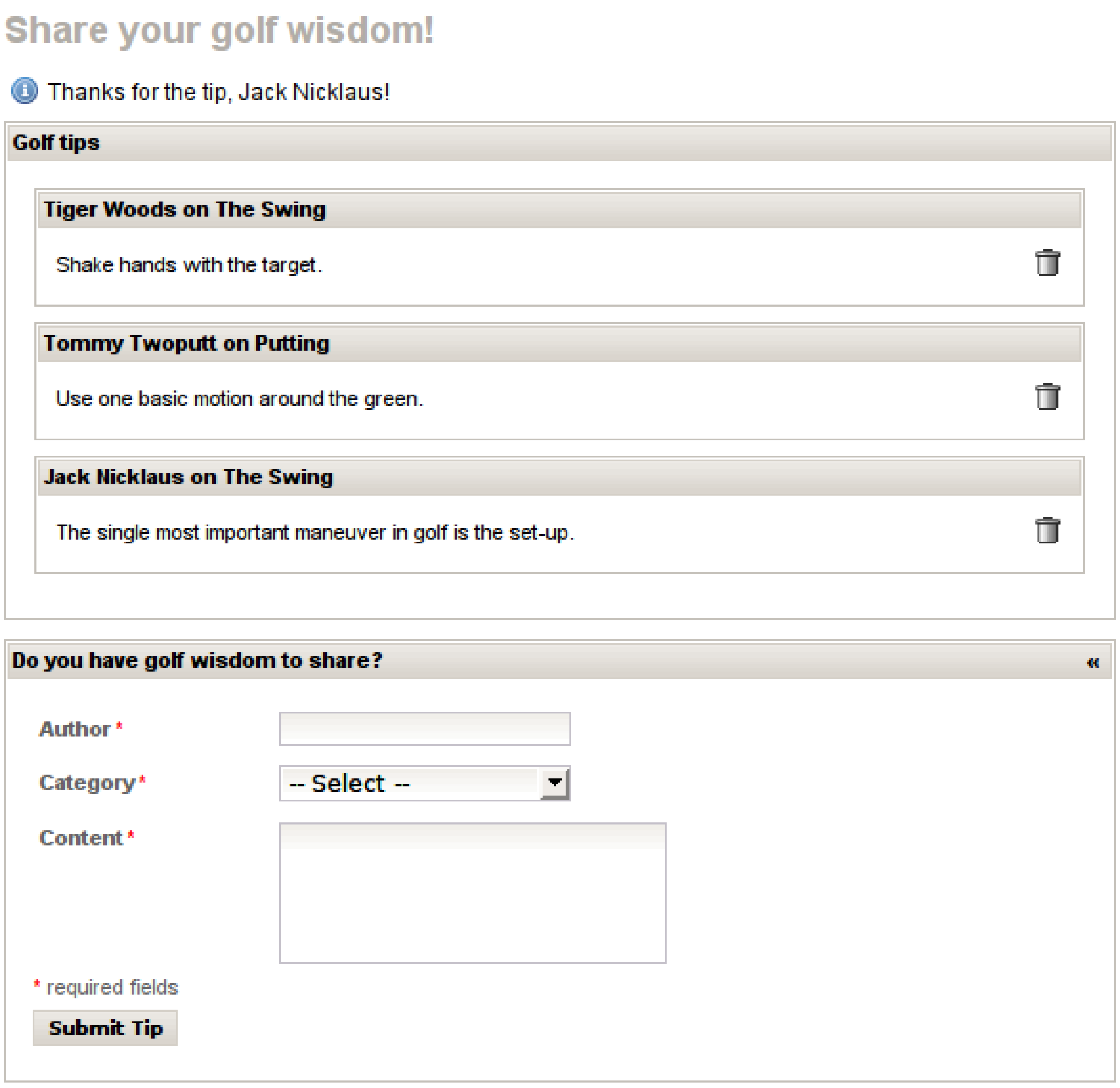
why seam-gen?

- Get to work quickly and be productive
- Use Seam in its *natural* environment
  - No bizarre exceptions to battle
- Generates the boring CRUD stuff



Username	
Address	
Name	

Username	Address	Birthdate	Name	Action
dan	Laurel, MD	3/20/78	Dan Allen	View Edit
gavin	San Francisco, CA	3/25/74	Gavin King	View Edit
pete	U.K.	12/28/81	Pete Muir	View Edit



Share your golf wisdom!

Thanks for the tip, Jack Nicklaus!

Golf tips

Tiger Woods on The Swing

Shake hands with the target.

Tommy Twoputt on Putting

Use one basic motion around the green.

Jack Nicklaus on The Swing

The single most important maneuver in golf is the set-up.

Do you have golf wisdom to share? «

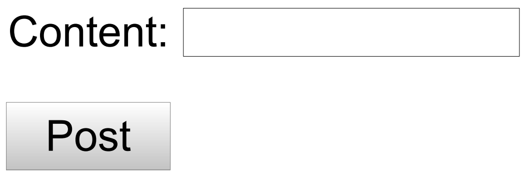
Author *	
Category *	-- Select --
Content *	

* required fields

Submit Tip

Form Binding

```
<h:form>
    <h:outputLabel for="content"
        value="Content:"/>
    <h:inputText id="content"
        value="#{tip.content}"/>
    <h:commandButton value="Post"
        action="#{tipBoard.post}"/>
</h:form>
```



Client

```
public class Tip
{
    private String content;
    // getters and setters
}


public class TipBoard
{
    public String post() {
        ...
        return "success";
    }
}
```

Server

```
<managed-bean>
    <managed-bean-name>tipBoard</managed-bean-name>
    <managed-bean-class>
        org.open18.golftips.action.TipBoard
    </managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

JSF managed bean

```java
@Name("tipBoard")
@Scope(EVENT)
public class TipBoard
{
    public String post()
    {
        ...
        return "/golftips.xhtml";
    }
}
```

Seam JavaBean component

```
@Name("tipBoard")
//@Scope(EVENT)
public class TipBoard
{
    public String post()
    {
        ...
        return "/golftips.xhtml";
    }
}
```

Automatic redirect after POST

Seam JavaBean component

```
@Local
public interface TipBoard
{
    String post();
}

@Stateless
@Name("tipBoard")
public class TipBoardBean
    implements TipBoard
{
    public String post() {
        ...
        return "/golftips.xhtml";
    }
}
```

Seam stateless session bean (SLSB) component

Invoking an EJB from JSF

- ...is not a crime!

```
<h:form>
    ...
    <h:commandButton value="Post"
        action="#{tipBoard.post}"/>
</h:form>
```

```
@Stateless
@Name("tipBoard")
public class TipBoardBean
    implements TipBoard
{
    public String post() { ... }
}
```

Client

Server

```
@Local
public interface TipBoard
{
    String post();
    void remove();
}

@Stateful
@Name("tipBoard")
@Scope(CONVERSATION)
public class TipBoardBean
    implements TipBoard
{
    public String post() {
        ...
        return "/golftips.xhtml";
    }

    @Remove public void remove() {}
}
```

Seam stateful session bean (SFSB) component

```
@Local
public interface TipBoard
{
    String post();
    void remove();
}

@Stateful
@Name("tipBoard")
//@Scope(CONVERSATION)
public class TipBoardBean
    implements TipBoard
{
    public String post() {
        ...
        return "/golftips.xhtml";
    }

    @Remove public void remove() {}
}
```

Seam stateful session bean (SFSB) component

SEEDING COMPONENTS

- **new-action** – *Generates a stateless component (SLSB) with one method and a JSF view that invokes it*
- **new-form** – *Generates a stateful component (SFSB) with one method and one property and an accompanying JSF view with form*
- **new-conversation** – *Generates a conversation-scoped component (SFSB) with a begin and end method and a counter (represents state)*
- **new-entity** – *Generates a basic JPA entity class with required annotations*

Are Annotions a Good Thing?

```java
public class Tip implements Serializable
{
    private Long id;
    private int version;
    private Date posted;
    private String author;
    private TipCategory category;
    private String content;

    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public int getVersion() { return version; }
    private void setVersion(version) { this.version = version; }

    public Date getPosted() { return posted; }
    public void setPosted(Date date) { posted = date; }

    public String getAuthor() { return author; }
    public void setAuthor(String name) { author = name; }

    public TipCategory getCategory() { return category; }
    public void setTipCategory(TipCategory cat) { category = cat; }

    public String getContent() { return content; }
    public void setContent(String content) { this.content = content; }
}
```

Model class

In the XML Age...

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

    <class name="org.open18.golftips.model.Tip" table="tip">
        <id name="id" type="long" unsaved-value="null">
            <generator class="identity"/>
        </id>
        <version name="version" type="integer"/>
        <property name="posted" type="timestamp" not-null="true"/>
        <property name="author" type="string"/>
        <many-to-one name="category" column="category_id"
            class="org.open18.golftips.model.TipCategory" not-null="true"/>
        <property name="content" type="string" not-null="true">
            <column sql-type="text"/>
        </property>
    </class>

</hibernate-mapping>
```

```
@Entity @Table(name = "tip")
public class Tip implements Serializable
{
    private Long id;
    private int version;
    private Date posted;
    private String author;
    private TipCategory category;
    private String content;

    @Id @GeneratedValue
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    @Version
    public int getVersion() { return version; }
    private void setVersion(int version) { this.version = version; }

    @Temporal(TIMESTAMP)
    public Date getPosted() { return posted; }
    public void setPosted(Date date) { posted = date; }

    public String getAuthor() { return author; }
    public void setAuthor(String name) { author = name; }

    @ManyToOne @JoinColumn(name = "category_id", nullable = false)
    public TipCategory getCategory() { return category; }
    public void setCategory(TipCategory cat) { category = cat; }

    @Lob
    public String getContent() { return content; }
    public void setContent(String content) { this.content = content; }
}
```

JPA entity class

```
@Entity @Table(name = "tip") @Name("tip")
public class Tip implements Serializable
{
    private Long id;
    private int version;
    private Date posted;
    private String author;
    private TipCategory category;
    private String content;

    @Id @GeneratedValue
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    @Version
    public int getVersion() { return version; }
    private void setVersion(int version) { this.version = version; }

    @Temporal(TIMESTAMP) @NotNull
    public Date getPosted() { return posted; }
    public void setPosted(Date date) { posted = date; }

    @NotNull @Length(min = 3, max = 50)
    public String getAuthor() { return author; }
    public void setAuthor(String name) { author = name; }

    @ManyToOne @JoinColumn(name = "category_id", nullable = false) @NotNull
    public TipCategory getCategory() { return category; }
    public void setCategory(TipCategory cat) { category = cat; }

    @Lob @NotNull @Length(min = 3, max = 10000)
    public String getContent() { return content; }
    public void setContent(String content) { this.content = content; }
}
```

Validations enforced in the UI using <s:validate> and by Hibernate before writes

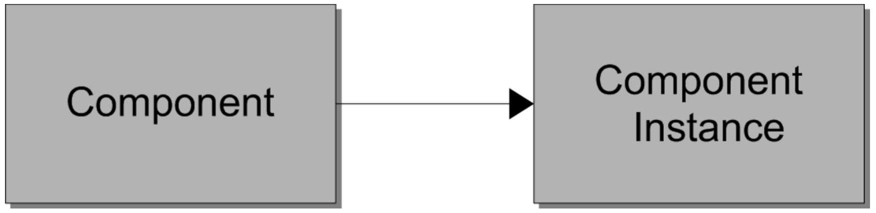
JPA entity class and Seam component with Hibernate Validator constraints

FILLING @IN THE DOTS

```java
@Logger private Log log;
@In EntityManager entityManager;
@In StatusMessages statusMessages;
@In @Out(required = false) Tip tip;
@Out(required = false) Tip savedTip;

public String post()
{
    log.debug("New tip posted by #{tip.author}");
    tip.setPosted(new Date());
    entityManager.persist(tip);
    savedTip = tip;
    tip = null;
    statusMessages.add("Thanks for the tip, #{savedTip.author}!");
    return "/golftips.xhtml";
}
```

CONTEXTUAL COMPONENT

- Component
  - Class
  - Name
  - Scope



- Component instance
  - Created by container when name is requested
  - Stored in scope (i.e., context), holds state
  - Life cycle managed by container
- Annotations
  - Define interactions and behavior

SEAM EJB COMPONENT

EJB Local Reference in web.xml (not required in JBoss AS)

```
<ejb-local-ref>
    <ejb-ref-name>golftipsEE/TipBoardBean/local</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <local>org.open18.golftips.action.TipBoardBean</local>
</ejb-local-ref>
```

EJB Local Interface and Seam SLSB component

```
@Local
public interface TipBoard
{
    String post();
}
```

```
@Stateless
@Name("tipBoard")
public class TipBoardBean
    implements TipBoard
{
    public String post() { ... }
}
```

How does Seam make the connection?

Resolving an EJB in Seam

- EJBs have dual citizenship
- First option: @JndiName
- Second option: Resolve JNDI from pattern
  - #{ejbName}
    - Value of name attribute on @Stateful/@Stateless
    - Unqualified class name of component
    - Value of <ejb-name> in ejb-jar.xml or web.xml
  - Plug into Seam's org.jboss.seam.core.init.jndiPattern
    - i.e., golfertipsEE/#{ejbName}/local
- SFSB references stored in Seam context
  - Default is conversation context

GETTING GROOVY

```
@Name("tipSearch")
@Scope(CONVERSATION)
class TipSearch
{
    @In protected def entityManager
    @DataModel def tips

    @Begin(join = true) void search()
    {
        tips = entityManager
            .createQuery("select t from Tip t").resultList
    }

    void deleteSelected() {
        tips.findAll { t -> t.selected }
            .each { t -> entityManager.remove t }
        search()
        "/golftips.xhtml"
    }
}
```

Types not required!

EJBs Can Be Groovy Too!

```
@Stateful
@Name("tipSearch")
class TipSearchBean
    implements TipSearch
{
    @In protected def entityManager
    @DataModel def tips

    @Begin(join = true) void search()
    {
        tips = entityManager
            .createQuery("select t from Tip t").resultList
    }

    void deleteSelected() {
        tips.findAll { t -> t.selected }
            .each { t -> entityManager.remove t }
        search()
        "/golftips.xhtml"
    }

    @Remove void remove();
}
```

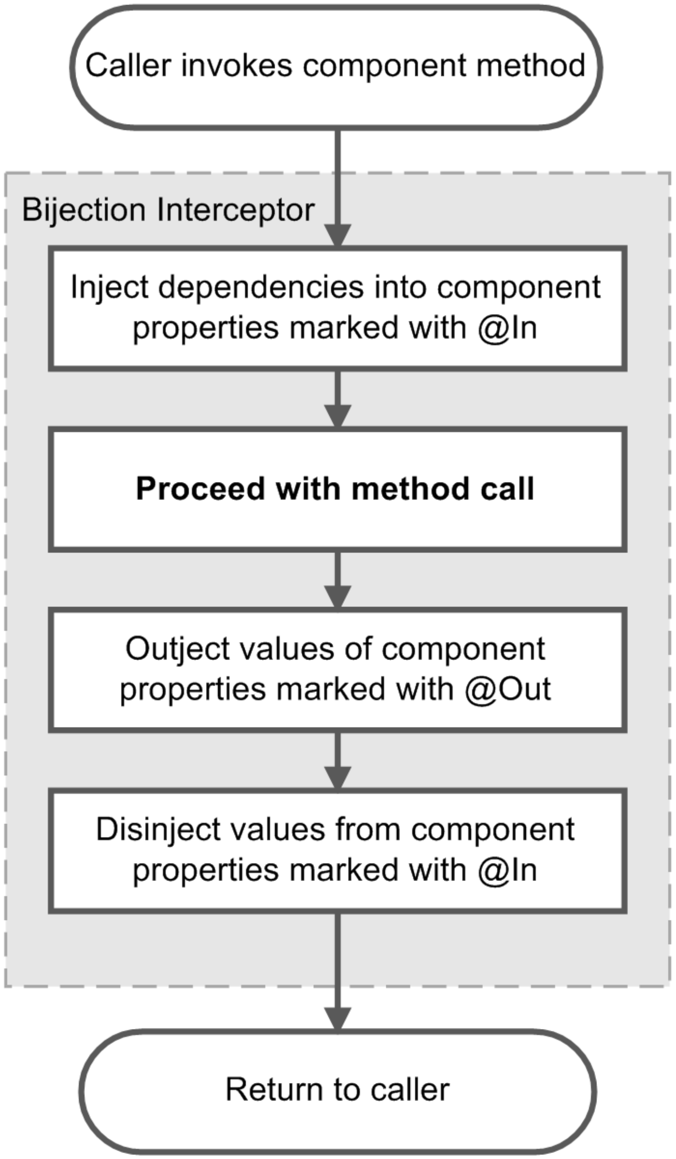
Bijection

- Similar in nature to dependency injection
  - Container satisfies the needs of components
- Occurs on *every* method invocation
- References to dependencies are transient
  - Adapts to change in state
  - Components in different scopes can safely interact

*Bi*jection = Injection + Outjection

Bijection, Your Caddy

THE 4 STEPS OF BIJECTION


Caller invokes component method
Bijection Interceptor
Inject dependencies into component properties marked with @In
Proceed with method call
Outject values of component properties marked with @Out
Disinject values from component properties marked with @In
Return to caller

PART 3

JSF: CLEANING HOUSE

JSF Triage

- Define components with annotations
- EJB functionality for JavaBeans
- Page actions, page-level security, GET
- Intelligent stateless & stateful navigation
- Context variable initializers
- Unified EL extensions
  - Parameterized expressions, pseudo-properties, etc.
- Transparent data model selections
  - Incorporated into bijection
- Global, managed transactions

Getting Data to the View

JSF doesn't provide either option!

Page Action

Before render

Factory

Any time

PAGE ACTIONS

- Associated with one or more JSF view IDs
- Executed prior to *Render Response* phase
- Can result in a navigation event
- Protect page from:
  - An invalid data request
  - An out of bounds request
  - A user with insufficient credentials
- Trigger invocation from a plain link
  - i.e., a registration link

INITIALIZING A VARIABLE LAZILY

- Notoriously difficult in JSF
  - Leads to bad practice of looking up data in getter
- Typical case in component-based model
- Factory
  - Called when no value bound to name
  - Component method or EL value or method expression
  - Only occurs once, until context ends
- Can wrap variable in manager component
  - Receive life-cycle callback methods
  - Observe events

A Context Variable @Factory

```
@Name("clubhouse")
public class Clubhouse
{
    @In protected EntityManager entityManager;

    @Factory("newGolfers")
    public List<Golfer> findNewGolfers()
    {
        return entityManager.createQuery(
            "select g from Golfer g order by g.dateJoined desc")
            .setMaxResults(5)
            .getResultList();
    }
}
```

Called when the context variable produced by this factory is referenced and is uninitialized or null.

```
<h:dataTable var="_golfer" value="#{newGolfers}">
    <h:column>#{_golfer.name}</h:column>
</h:dataTable>
```

A DATA MODEL @FACTORY

```
@Name("clubhouse")
public class Clubhouse
{
    @In protected EntityManager entityManager;

    @DataModel(scope = ScopeType.PAGE)
    private List<Golfer> newGolfers;

    @Factory("newGolfers")
    public void findNewGolfers()
    {
        newGolfers = entityManager.createQuery(
            "select g from Golfer g order by g.dateJoined desc")
            .setMaxResults(5)
            .getResultList();
    }
}
```

```
<h:dataTable var="_golfer" value="#{newGolfers}">
    <h:column>#{_golfer.name}</h:column>
</h:dataTable>
```

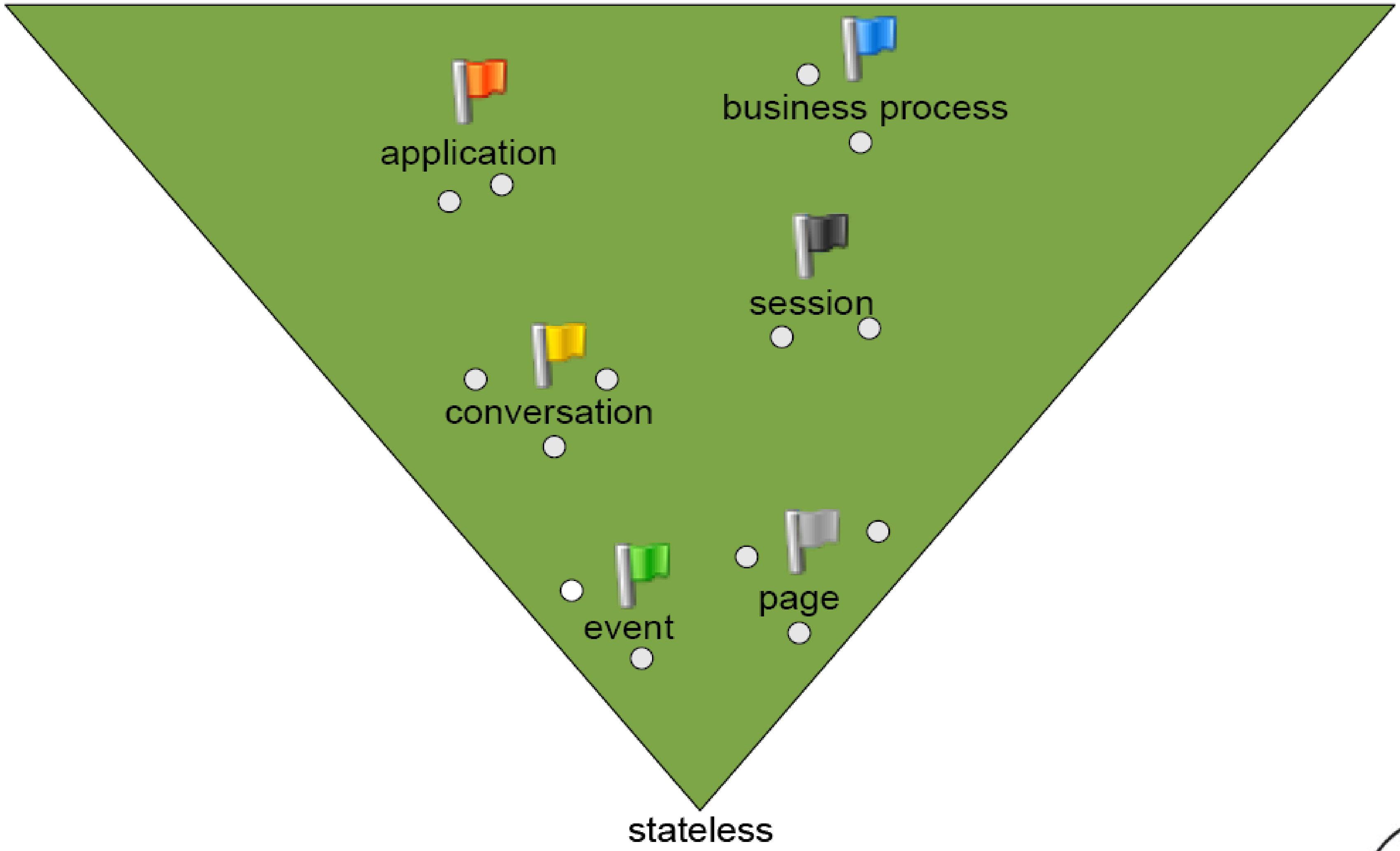
Java EE Integration Testing

- SeamTest
  - Based on TestNG
  - DBUnitSeamTest for advanced database setup
- Test application “as is”
  - Mimic JSF or non-JSF requests
  - Full JSF life cycle
  - Mock APIs where appropriate
  - JTA, EJB 3, EL
- Uses Embedded JBoss
  - Started once per suite
  - Has baggage, look forward to EJB 3.1

PART 4

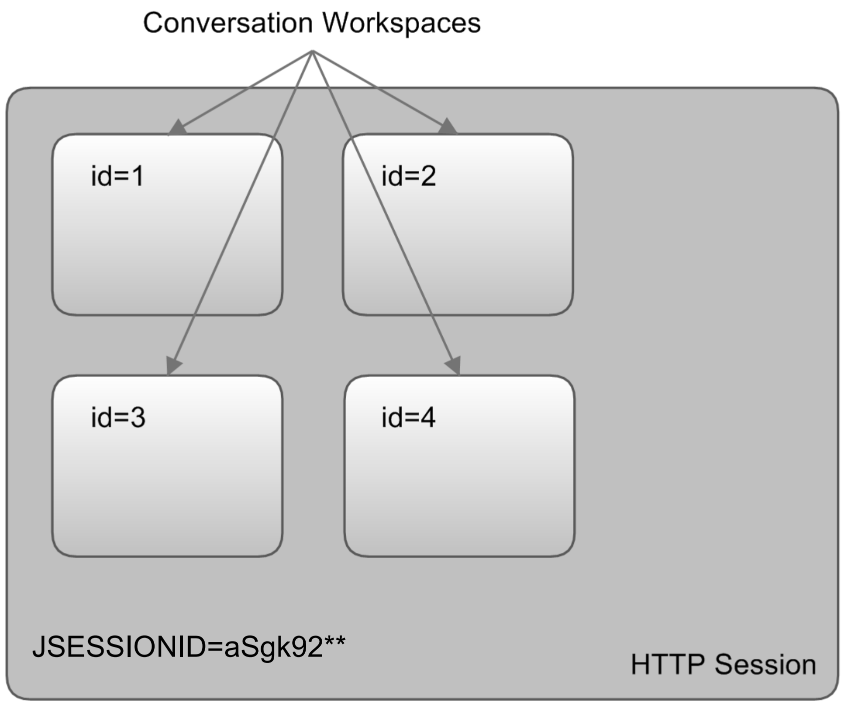
MANAGING STATE

“FORE!”


application
business process
session
conversation
event
page
stateless

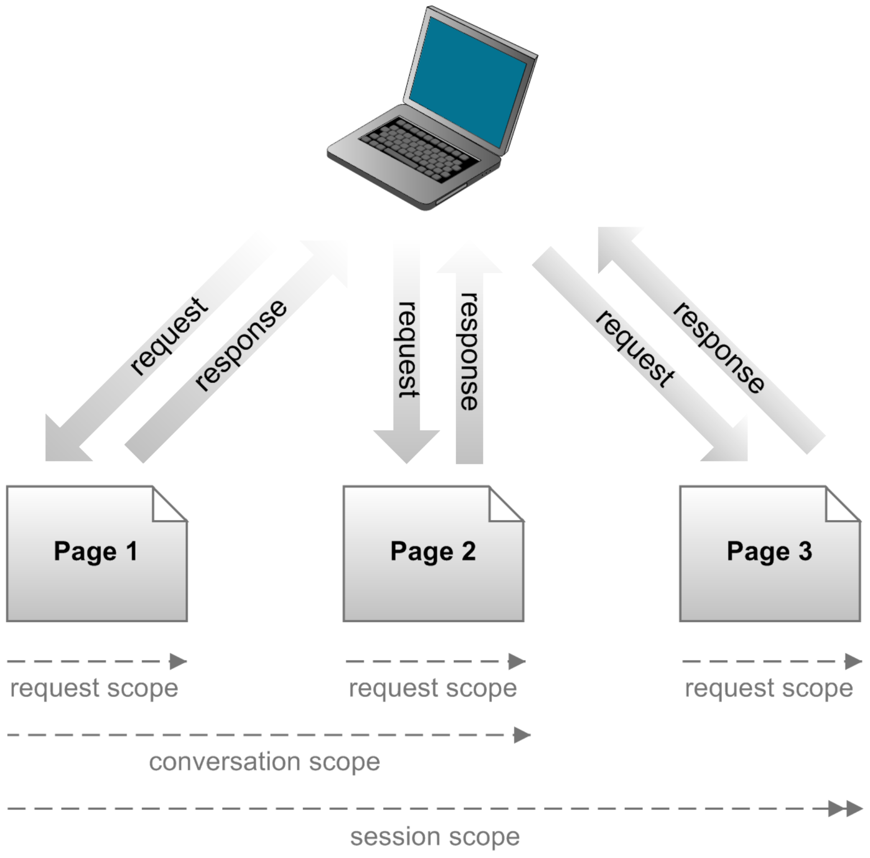
Carving out Conversations

- Isolated regions of HTTP session
  - Solves “session bleeding” problem
- Much shorter timeout than session
- Boundaries defined declaratively
- Not explicitly tied to navigation model



Conversation Propagation

- Conversation identifier sent with request
- Timeout period reset upon restore
- Optional directive signals boundaries
  - begin
  - join
  - end
  - nest
  - none



CONVERSATION USES

- Non-persistent information
  - Search criteria, selections, breadcrumb navigation
- Outstanding changes to persistent data
  - Managed entities that have changed
- Natural cache of query results
  - Aware of business context

TWO CONVERSATION STYLES

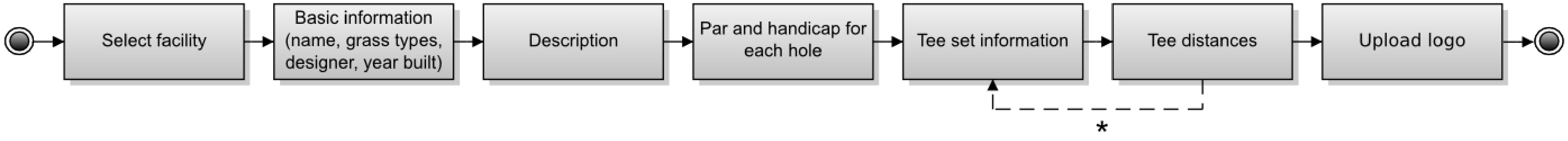
- Ad-hoc
  - "Open for business"
  - Can get tricky
  - Works well with named conversations
- Stateful pageflow
  - Based on jPDL
  - Every page request must be sanctioned
  - Can move action expressions to pageflow

AD-HOC CONVERSATION

```
@Name("courseComparison")
@Scope(CONVERSATION)
public class CourseComparison
{
    @In protected EntityManager entityManager;
    @RequestParameter protected Long courseId;
    @DataModel private Set<Course> markedCourses;
    @Out("readyToCompare") private boolean ready = false;

    @Begin(join = true)
    public void mark()
    {
        Course c = entityManger.find(Course.class, courseId);
        if (c == null) return;
        courses.add(c);
        ready = courses.size() > 1;
    }

    public void clear()
    {
        courses.clear(); ready = false;
    }

    @End(beforeRedirect = true)
    public void reset() {}
}
```

```
<h:commandLink value="Mark"
    action="#{courseComparison.mark}">
    <f:param name="courseId"
        value="#{_course.id}"/>
</h:commandLink>
```

CONVERSATIONAL PAGEFLOW (1)


Select facility
Basic information (name, grass types, designer, year built)
Description
Par and handicap for each hole
Tee set information
Tee distances
Upload logo
*

CONVERSATIONAL PAGEFLOW (2)

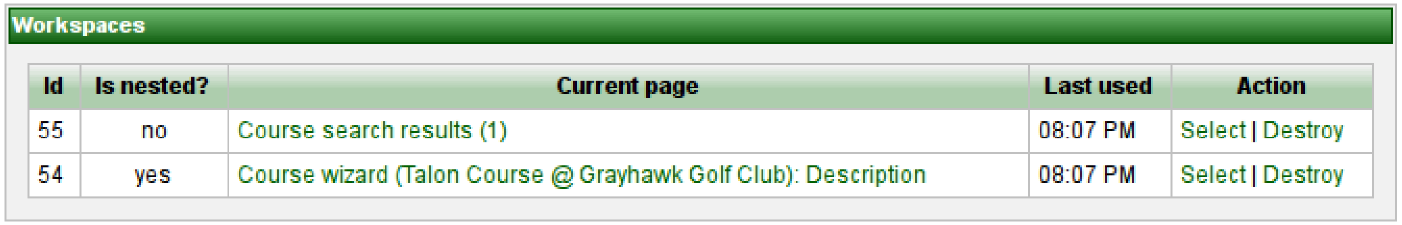
```
<?xml version="1.0" encoding="UTF-8"?>
<pageflow-definition>
    <start-state>
        <transition to="basicCourseInfo"/>
    </start-state>

    <page name="basicCourseInfo"
        view-id="/coursewizard/basicCourseInfo.xhtml" redirect="true">
       <transition name="cancel" to="cancel"/>
       <transition name="next" to="description"/>
    </page>
    ...
    <page name="review"
        view-id="/coursewizard/review.xhtml" redirect="true">
       <transition name="cancel" to="cancel"/>
       <transition name="success" to="end">
           <action expression="#{courseHome.setCourseId(course.id)}"/>
       </transition>
       <transition to="review"/>
    </page>
    <page name="end" view-id="/Course.xhtml" redirect="true">
        <end-conversation/>
    </page>
</pageflow-definition>
```

CONVERSATIONAL PAGEFLOW (3)

```
@Name("courseWizard")
@Scope(CONVERSATION)
public class CourseComparison
{
    @In protected EntityManager entityManager;
    @Out("newCourse") private Course course;

    @Begin(pageflow = "Course Wizard")
    public void enterNewCourse()
    {
        course = new Course();
    }

    @Conversational
    public void selectFacility(Facility facility)
    {
        course.setFacility(facility);
    }

    @End @Conversational
    public boolean save()
    {
        try { entityManager.persist(course); return true; }
        catch (Exception e) { return false; }
    }
}
```

SUPPORTING PARALLEL ACTIVITY

- Workspaces
  - Switch between parallel conversations
  - Workspace switcher is akin to browser tabs



Workspaces

Id	Is nested?	Current page	Last used	Action
55	no	Course search results (1)	08:07 PM	Select | Destroy
54	yes	Course wizard (Talon Course @ Grayhawk Golf Club): Description	08:07 PM	Select | Destroy

- Nested conversations
  - Isolate work within use case
  - Can easily navigate back to parent conversation
  - Terminated automatically if parent conversation ends

PART 5

RESPECT THE PERSISTENCE CONTEXT

MANAGING PERSISTENCE

- Persistence manager
  - JPA EntityManager or Hibernate Session
  - Manages object representations of database records
- Key services
  - First-level cache
  - Transparent database reads/writes
- Lifetime
  - Aligns well with use case (i.e., conversation)
  - ORM undervalued w/o proper lifetime

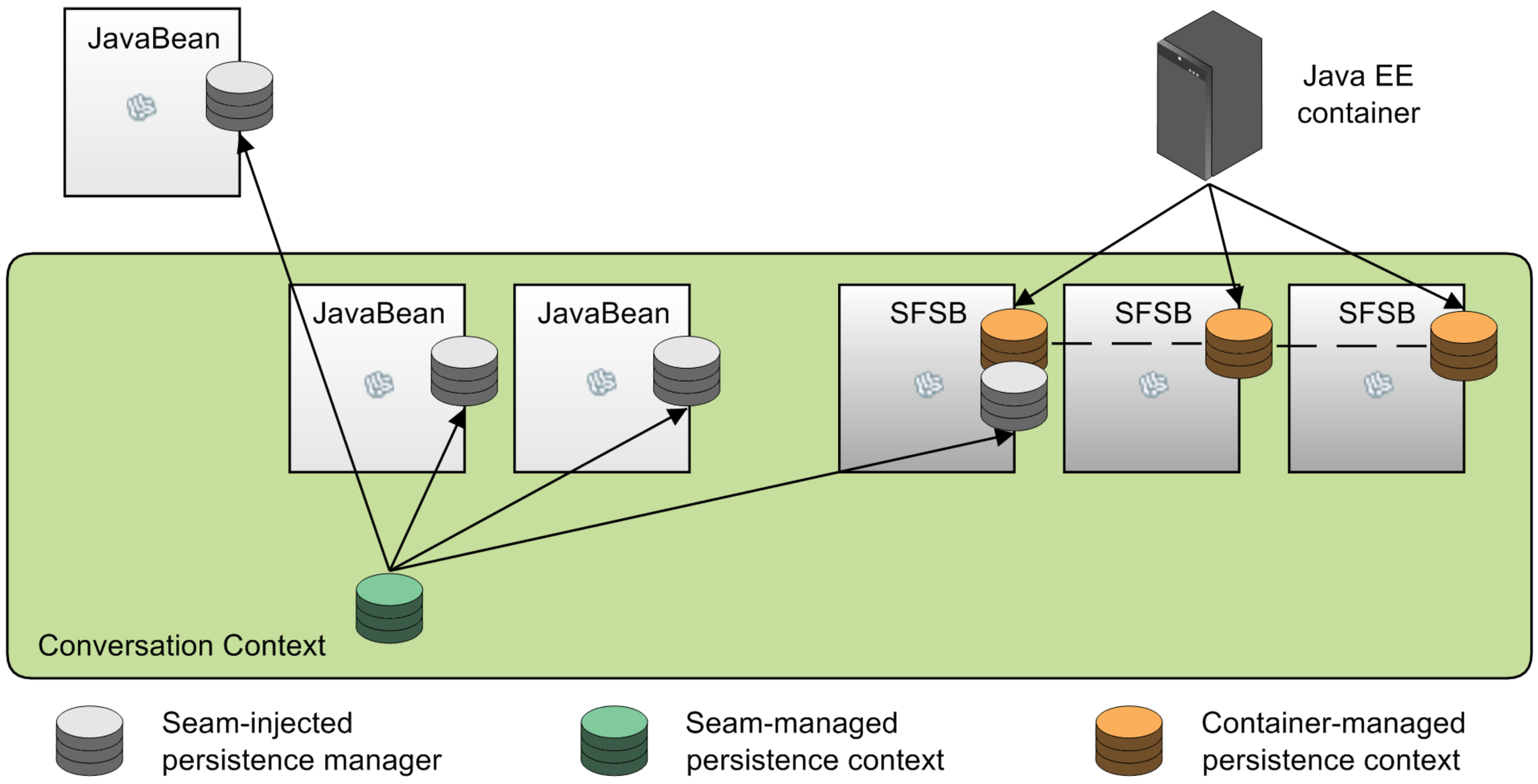
WHAT *IS* A PERSISTENCE CONTEXT?

- Map maintained by persistence manager
- Bucket of retrieved entity instances
  - In-memory cache
  - Instances are managed
  - One instance per database record
  - Performs automatic dirty checking, deferred DML
  - Persistence by reachability
- *All that stops working once it's closed!*
  - Entity instances become *detached*
  - The LazyInitializationException leaves its mark
  - Conversations solve this problem

EXTENDING THE PC

- Spans multiple requests
- Form values applied to *managed* entity
- Dirty checking ensures update
  - Update only happens if entity has changed
  - Flushing can be deferred to keep changes pending
- Leverage optimistic locking
- Multi-record editing for free

SEAM VS CONTAINER-MANAGED PC


JavaBean
Java EE container
JavaBean
JavaBean
SFSB
SFSB
SFSB
Conversation Context
Seam-injected persistence manager
Seam-managed persistence context
Container-managed persistence context

Seam (bi)jection

```
@In
EntityManager entityManager;
```

Java EE resource injection

```
@PersistenceContext
EntityManager entityManager;
```

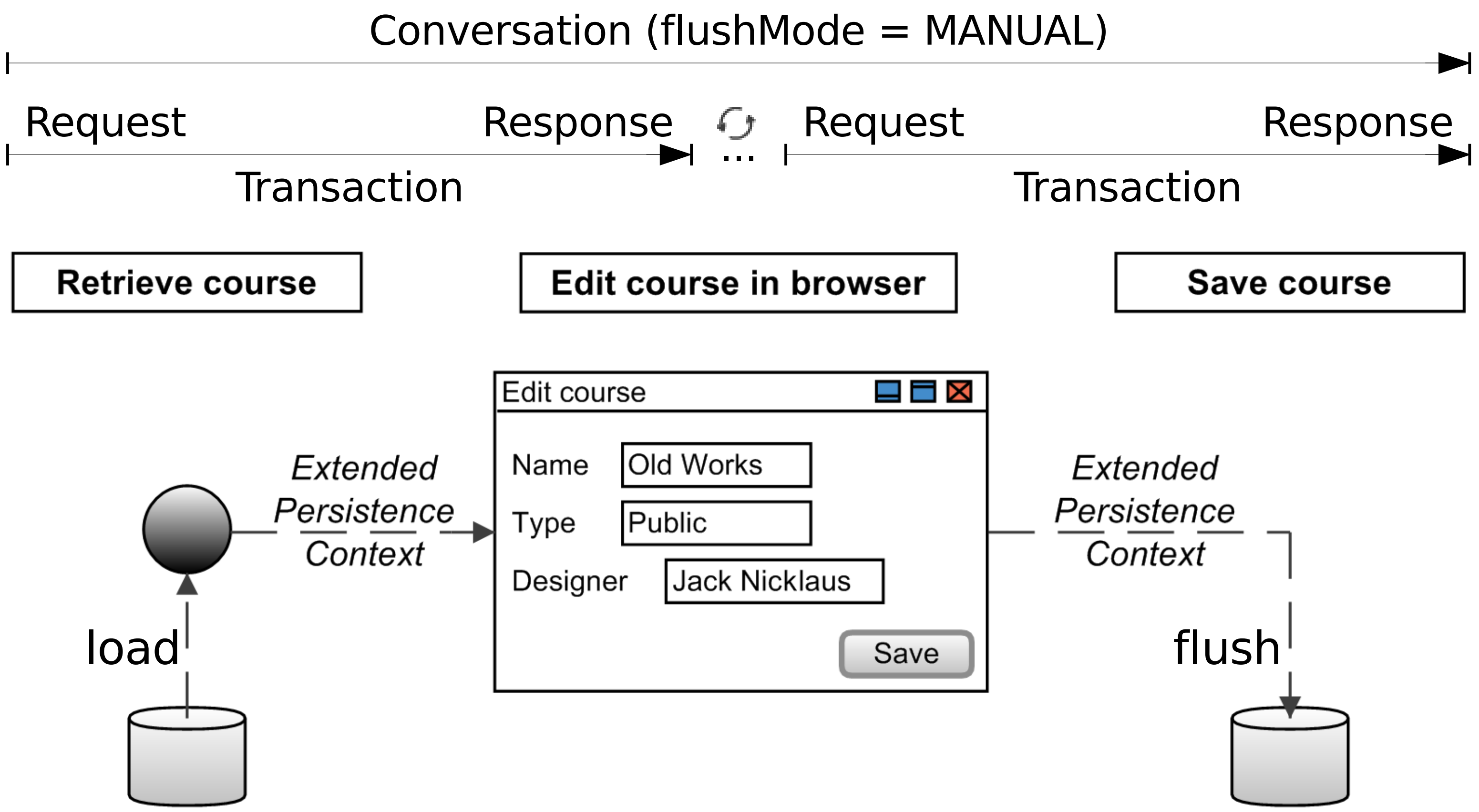
MANUAL FLUSHING

- Only available in Hibernate
- Defers updates until explicitly instructed
- Eliminates need for value objects
- Rollback need not involve database
- Can activate declaratively in Seam

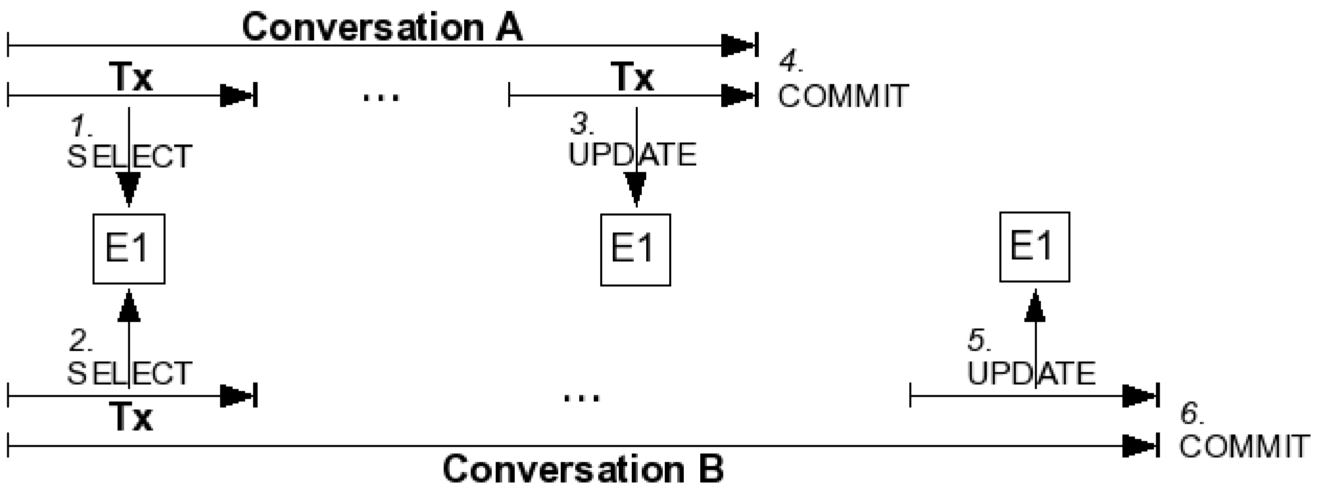
```
@Begin(flushMode = MANUAL)

<begin-conversation flush-mode="MANUAL"/>

<core:manager default-flush-mode="MANUAL"/>
```

- Conversations broken without it

Atomic Conversation



LAST COMMIT WINS?



```
private Integer version;

@Version
protected Integer getVersion() {
    return this.version;
}

protected void setVersion(Integer version) {
    this.version = version;
}
```

Optimistic locking

Verifies record in database hasn't changed. Aborts transaction if record is out of sync.

PART 6

LOCK DOWN

Ramping Up on Seam Security

- Single-method authentication
  - Configured as method expression in components.xml
  - Hides the complexities of JAAS
- Seam 2.1 offers built-in authentication
  - Database or LDAP
- 3 levels of authorization
  - Role-based
  - Rule-based (Drools)
  - ACLs (database)
- Identity and permissions management
- OpenID Single Sign-on (SSO)

Form-based Authentication:

The 3-Step Program

Step 1: Authentication Method

```
@Stateless
@Name("authenticator")
public class AuthenticatorBean implements Authenticator
{
    @Logger private Log log;
    @In Identity identity;
    @In Credentials credentials;

    public boolean authenticate()
    {
        log.info("authenticating {0}", credentials.getUsername());
        if ("admin".equals(credentials.getUsername()))
        {
            identity.addRole("admin");
            return true;
        }
        return false;
    }

}
```

STEP 2: SECURITY CONFIGURATION

```
<?xml version="1.0" encoding="UTF-8"?>
<components xmlns="http://jboss.com/products/seam/components"
    xmlns:security="http://jboss.com/products/seam/security"
    xsi:schemaLocation="
        http://jboss.com/products/seam/security
        http://jboss.com/products/seam/security-2.1.xsd
        http://jboss.com/products/seam/components
        http://jboss.com/products/seam/components-2.1.xsd">

    <security:identity authenticate-method="#{authenticator.authenticate}"/>

</components>
```

STEP 3: JSF LOGIN FORM

```
<h:form id="login">
    <h:panelGrid columns="2">
        <h:outputLabel for="username" value="Username"/>
        <h:inputText id="username" value="#{credentials.username}"/>
        <h:outputLabel for="password" value="Password"/>
        <h:inputSecret id="password" value="#{credentials.password}"/>
        <h:outputLabel for="remember" value="Remember me"/>
        <h:selectBooleanCheckbox id="remember" value="#{identity.rememberMe}"/>
    </h:panelGrid>
    <div>
        <h:commandButton id="login" action="#{identity.login}" value="Login"/>
    </div>
</h:form>
```

Declarative Authentication

User class

```
@UserPrincipal
public String getUsername() { return username; }

@UserPassword(hash = "MD5")
public String getPasswordHash() { return passwordHash; }

@UserRoles @ManyToMany
public Set<MemberRole> getRoles() { return roles; }
```

Role class

```
@RoleName
public String getName() { return name; }

@RoleGroups @ManyToMany
public Set<MemberRole> getGroups() { return groups; }
```

Configuration

```
<security:jpa-identity-store
    user-class="org.open18.model.Member"
    role-class="org.open18.model.MemberRole"/>
```

Anatomy of a Permission

- Two parts
  - Target
  - Action
- Resolved by permission chain
  - First affirmative vote grants access
  - Resolvers are pluggable

Authorization Points

- Page

```
<page view-id="/editCourse.xhtml" login-required="true"/>
<page view-id="/member/*" login-required="true"/>

<page view-id="/editCourse.xhtml">
    <restrict/>
</page>
```

- Method

```
@Restrict("#{identity.loggedIn}") public void findMembers() { ... }
@Restrict("#{course, 'modify')") public void updateCourse() { ... }
```

- Entity

```
@PrePersist @Restrict public void prePersist();
```

- Inline code

```
<h:panelGroup rendered="#{identity.loggedIn}"> ... </h:panelGroup>
<h:commandButton action="#{courseManager.edit}" value="Edit"
    rendered="#{s:hasPermission(course, 'modify')}"/>
```

PART 7

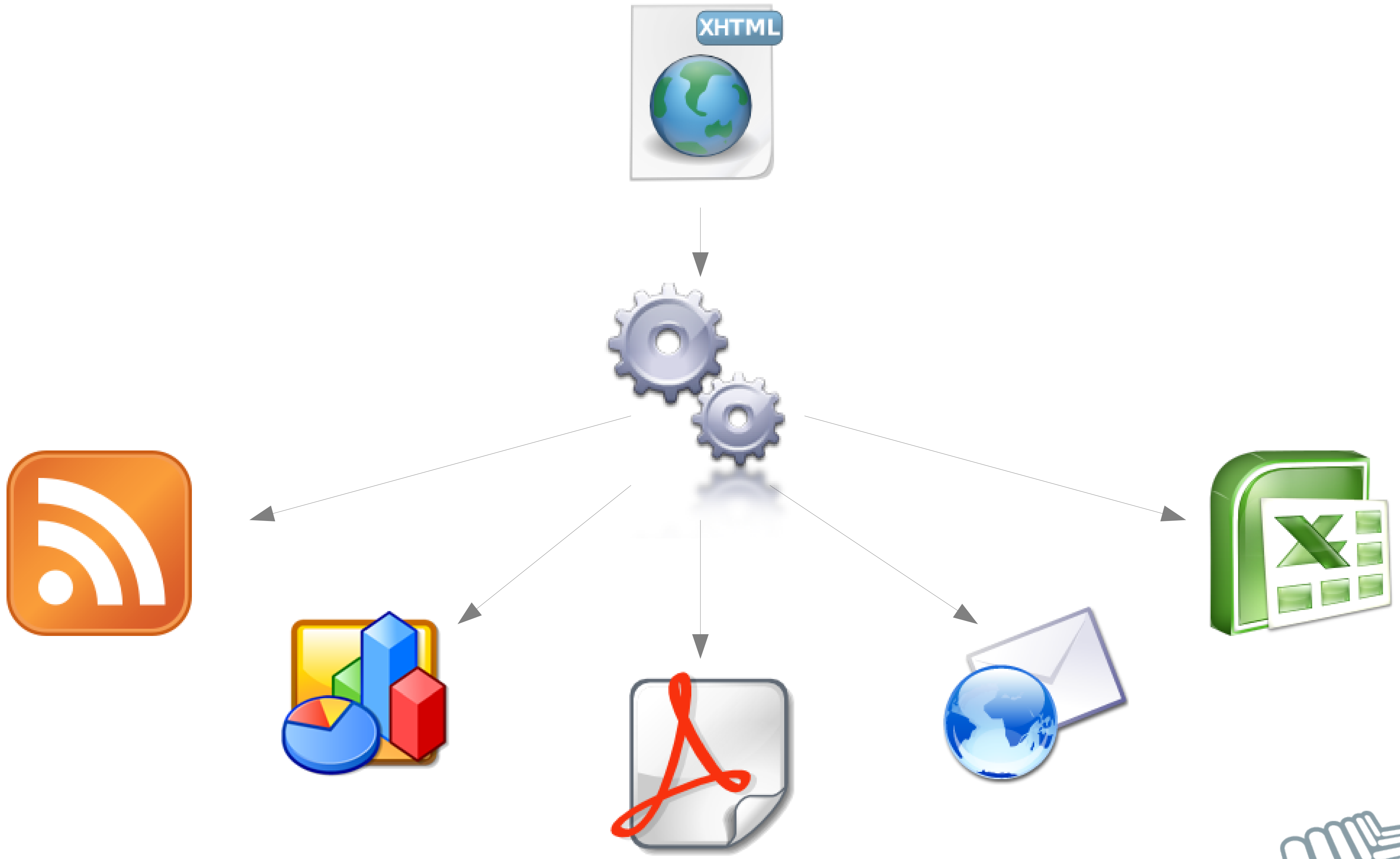
GET RICH

File Handling is a Breeze

- Upload component
  - Binds file input to byte[] property
  - Captures content type
- Graphics component
  - Generate image from byte[] property
  - Declarative transformations
- Document servlet
  - Serves binary content to browser
  - Supports file extensions in download URL
  - Can use it to serve custom content

Image courtesy of Aurigma, Inc.

Flexible Facelets



EVERY BUSINESS HAS A PROCESS

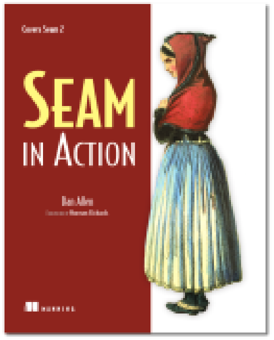
- Seam gives it a context
  - Integrates with jBPM
  - Process can “see” Seam components and context
- Process is a multi-user conversation
  - Each task is a single-user conversation
- Declarative boundaries
  - Annotations
    - @CreateProcess, @ResumeProcess
    - @StartTask, @BeginTask, @EndTask, @Transition
  - Page descriptor elements
  - JSF components

Web Beans: Seam Evolved

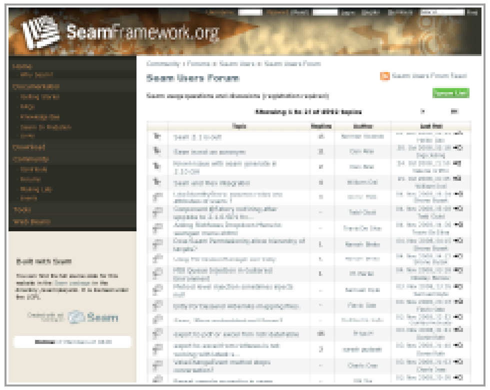
- Continued commitment to Java EE
- Standardizes Seam's core container
  - Annotations extend Java type system (“API type”)
  - Extensible context model
  - Conversations
- Type-safe resolution
  - Seam relies on string names
- Proxies instead of bijection
- Integration expected in Seam 3

RESOURCES

- Seam in Action (Manning, Sep 2008)
  - http://manning.com/dallen (4 free chapters)
  - http://code.google.com/p/seaminaction



- Seam, Web Beans and Hibernate blog
  - http://in.relation.to
- Seam community site & forums
  - http://seamframework.org



- Seam Issue Tracker
  - http://jira.jboss.org/jira/browse/JBSEAM
- Seam links (and lots of them)
  - http://delicious.com/seaminaction

SEAM IN ACTION

DAN ALLEN

JBoss, A DIVISION OF **RED HAT**

Thanks for attending!