Introduction to JBoss Seam 2.1

Priyatam Mudivarti
Sr Engineer, Cramer

www.reverttoconsole.com

Agenda

- Common Webapp Development Patterns – 5 mins
- Why Seam? – 5 mins
- Demos Showcase ~ 30 mins
  - Environment Setup, seam-gen
  - Hotel Booking without security
  - Hotel Booking with security under 5 minutes
  - Advanced Hotel Booking with Rules, Identity Management
  - Seambay (ebay spoof)
- Features ~ 20 mins
- Q&A – 5 mins
- Unlimited Q&A @Asguards after beer ☺

Today's Web Applications

- Too many layers
- Too many integration points
- Too much Xml configuration
- For new features, you have to learn a new framework
- "Stateless" Architecture
- JSF could be great, but falls short
- Not a platform, merely a set of libraries and wrappers
- Documentation is not at one place (too many dependent frameworks)

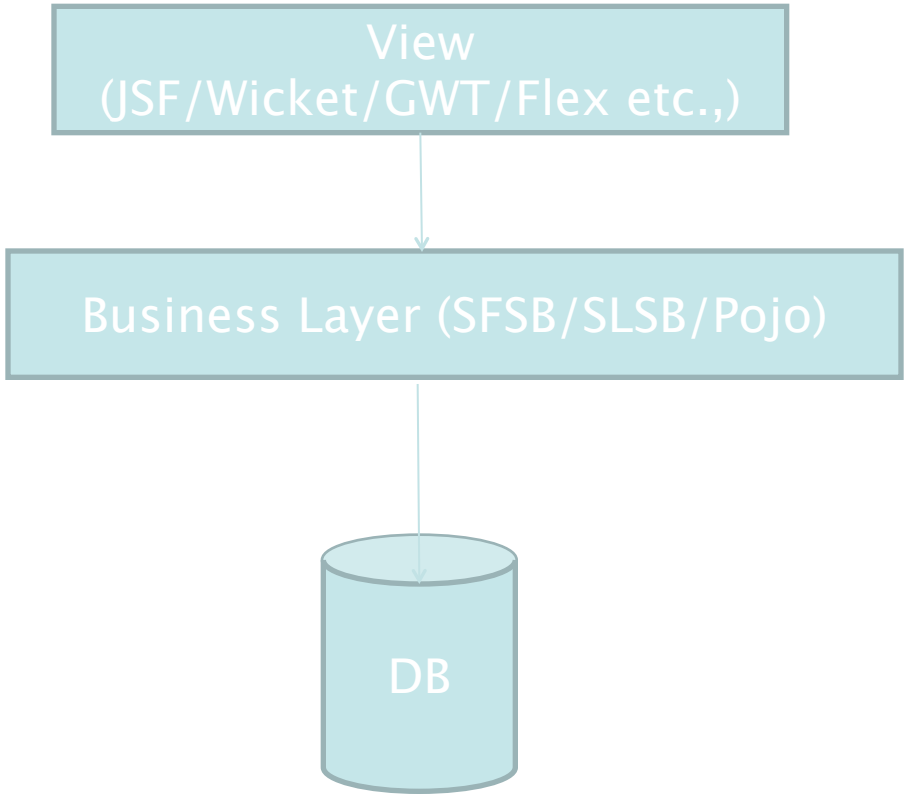
Why Seam

- Is it good? Yes.

Some good reasons

- Code is beauty. Seam makes it beautiful.
- Configuration by Exception
- JSF just got better. Other Views are welcome.
- Easiest way to get started with EJB 3.0
- CRUD is insanely simple.
- App Generator via seam-gen (aka scaffolding)
- It makes persistence a breeze
- Annotations over XML, end to end
- Automated integration testing using TestNg
- Central Component Registry and unified EL
- Event Model built in
- Security with Identity Management out of the box
- And many more … (Rules Engine, BPM, Ajax Support, Web Remoting, Pdf, Excel generators, RESTful …)
- Open source. Open standards. Future of Java EE.

Layers in Seam



Hey, No Integration Layer!

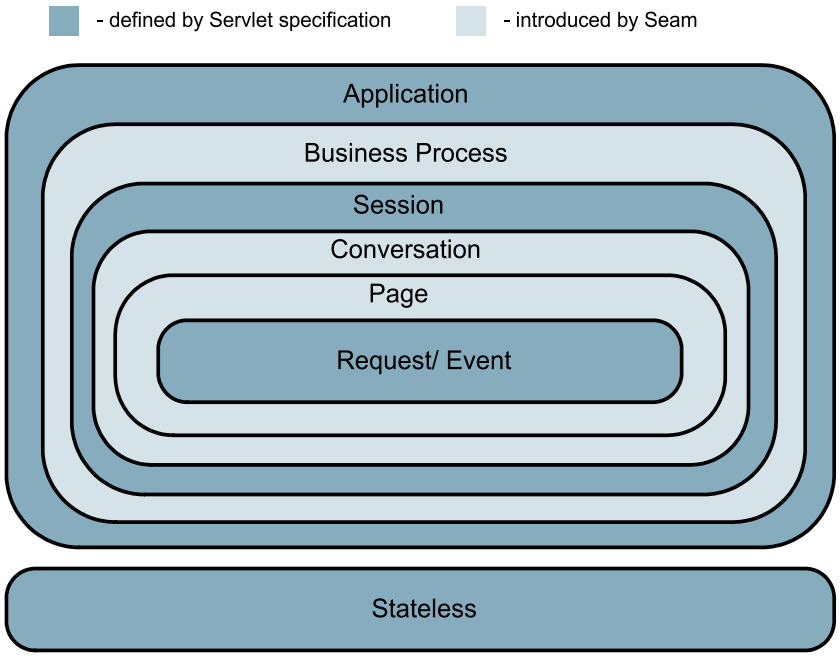
- Seam exposes Business Layer objects and their attributes on the front-end using EL expressions.
- No DAOs
- No DTOs

Contextual components

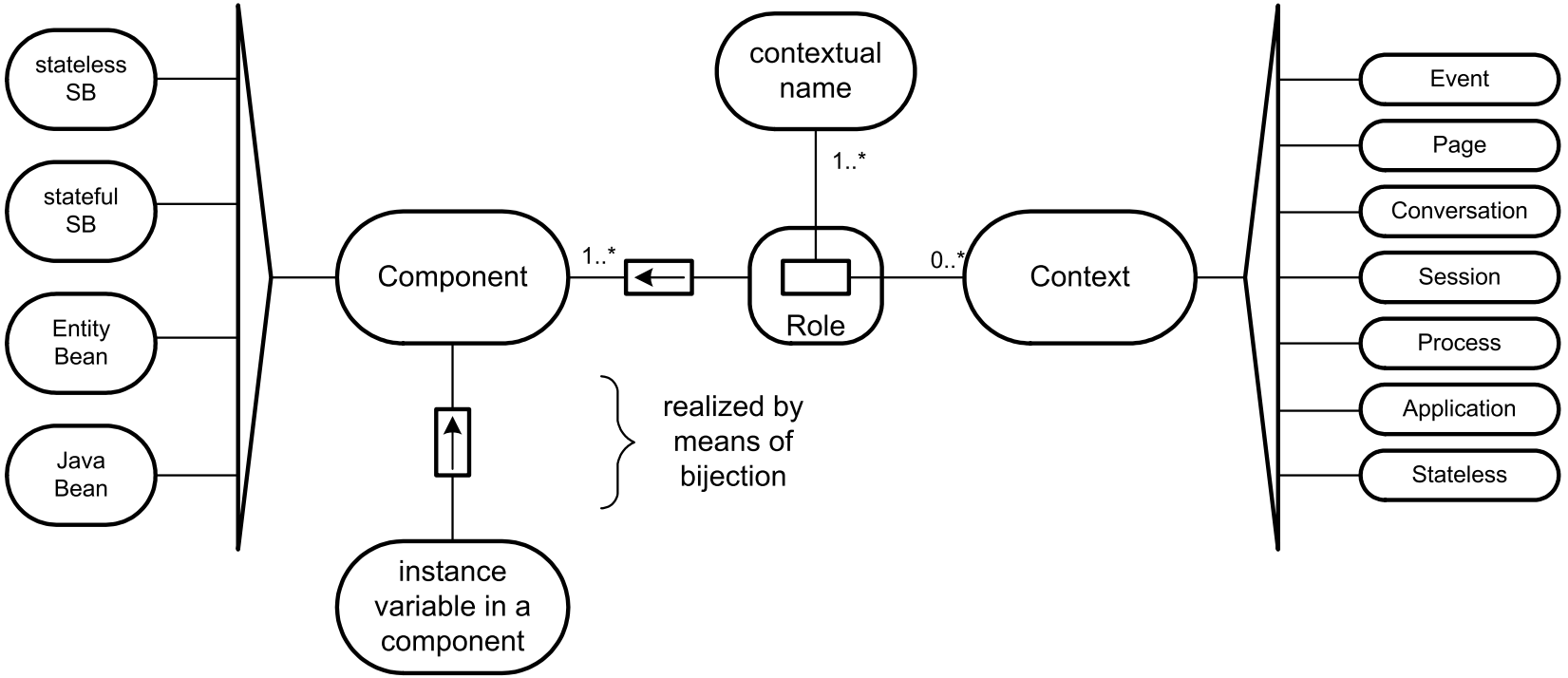
- Most of the problems relate to state management
  - Traditional contexts for Java Servlets focused on technology instead of on application
  - EJB itself has no strong model of state management
  - We need a richer context model that includes “logical” context
- Mismatch between the JSF and EJB 3.0 models
  - We should be able to use annotations everywhere
  - An EJB should be able to be a JSF managed bean (vice versa)
- Main idea: realization of desktop-like wizards and dialogs, possibly in parallel
- It makes sense to think of binding EJB components directly to the JSF view

The Seam Context Model

- Seam defines a rich context model for stateful components, enabling container-management of application state



Seam's Context Model (contd.)



*source: Steffen Ryll

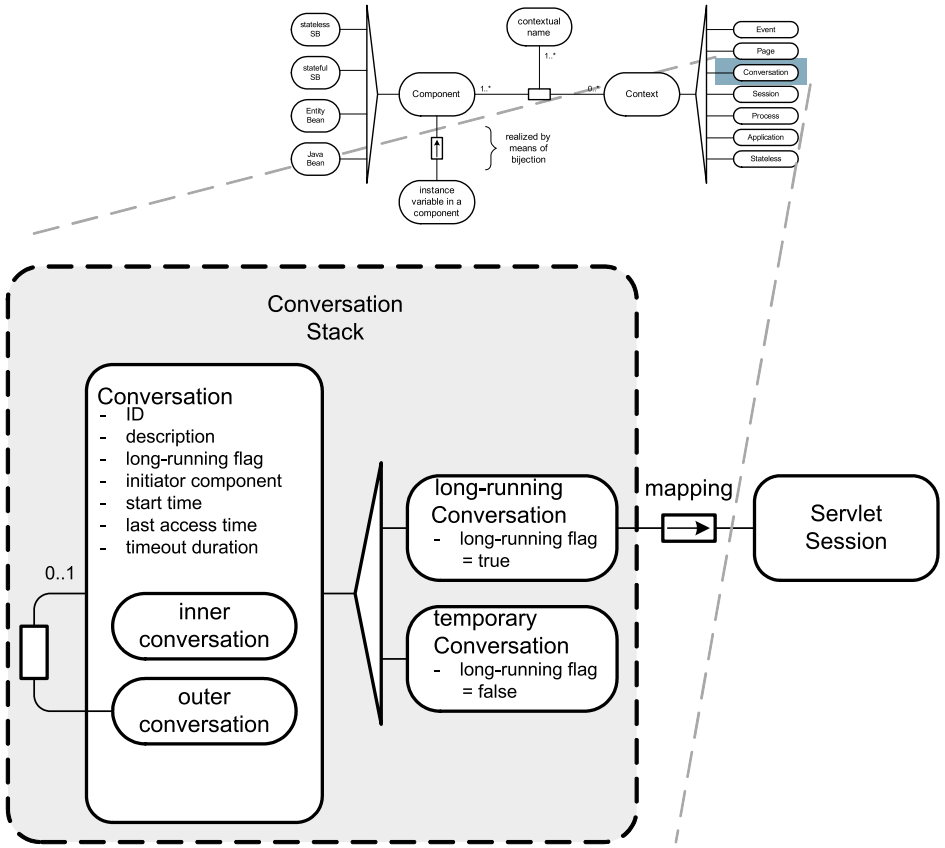
- Components may be attached to many Contexts
  - Role: pair of a contextual Name and a Context

Conversations

- Conversations are not that exciting until you really start thinking about them:
  - multi-window operation
  - “workspace management”
  - Back button operation
  - stack of continuable states (nested conversation)
- Two models for conversational pageflow
  - The stateless model: JSF navigation rules
    - ad hoc navigation (the app must handle backbutton)
    - actions tied to UI widgets
  - The stateful model: jBPM pageflow
    - no ad hoc navigation (back button bypassed)
    - actions tied to UI widgets or called directly from pageflow transitions

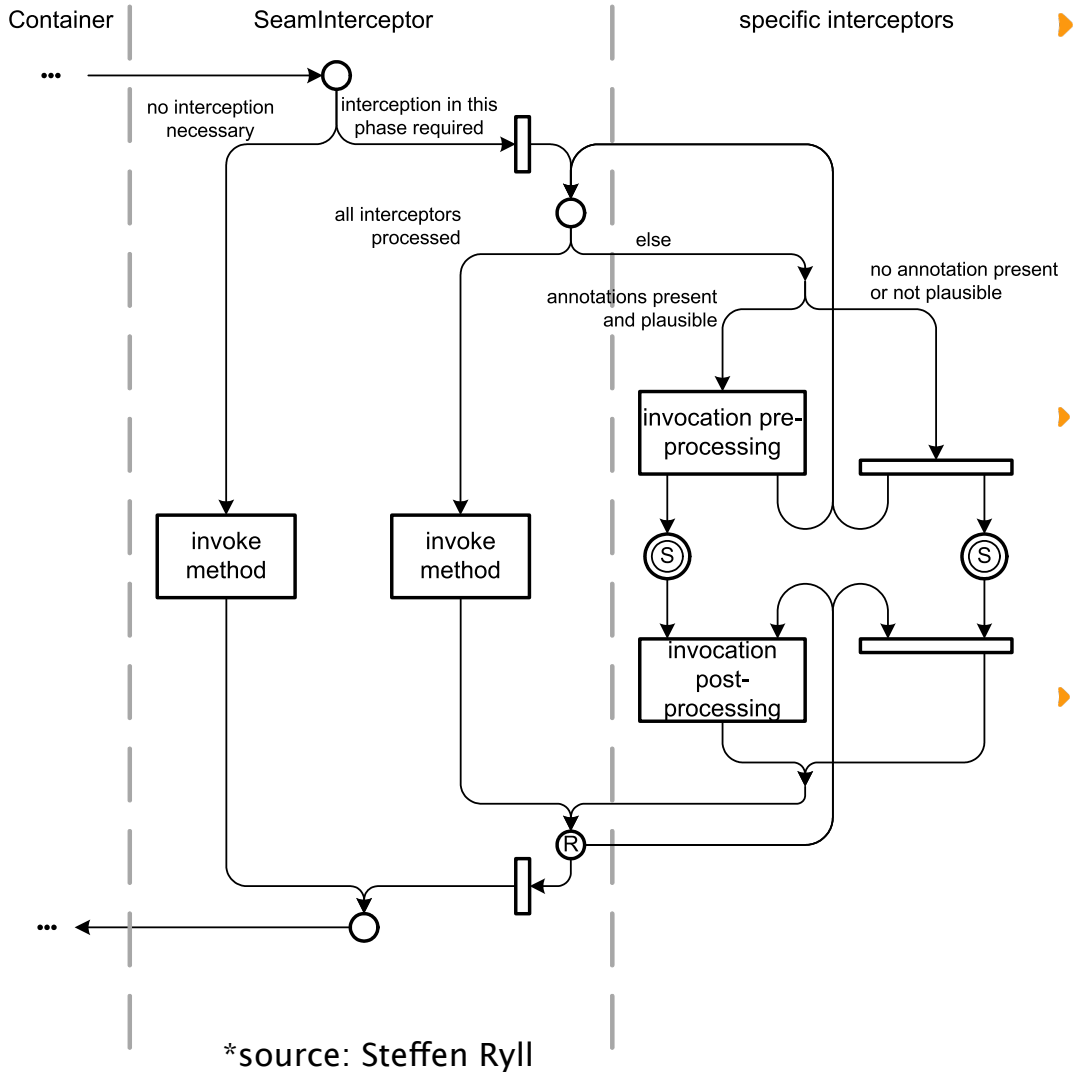
Conversations

- Conversation context usually held on server
  - serialization to client is supported as well
- Conversations can be nested
  - outer conversation continues when inner conversations are terminated



*source: Steffen Ryll

Interceptor-driven State Handling



*source: Steffen Ryll

- SeamInterceptor registered for all components
  - on method invocation, delegates to all other Seam interceptors
- annotations `@Around`, `@Within` allow to impose a partial order on interceptors
- Seam's interceptors
  - Conversation Interceptor
  - Bijection Interceptor
  - Business Process Interceptor
  - Validation Interceptor
  - and a few others

Page Flow / Navigation Rules

- Used to define process workflow / page flow
- Define transition between pages
- Define navigation rules for each page
- Flow is based on events and conditions

Business Process using JBpm

- last longer than login sessions
  - involve interaction with multiple users
  - potentially also several conversations with each user
- forms navigation graph with task nodes and transition edges
  - modeled with jPDL
  - graphical modeling tools available from Jboss
- process description interpreted by JBoss jBPM subsystem
  - takes care of making process state persistent
  - state handling is simply wrapped by Seam
  - process task nodes are mapped to JSF pages
  - Seam provides decision variables to jBPM subsystem

Dependency Injection (Bijection)

- Dependency injection was designed with J2EE-style stateless services in mind which is usually implemented in a static, unidirectional, and non-contextual way
- Dependency injection is broken for stateful components
  - A contextual variable can be written to, as well as read
  - A component in a wider scope must be able to have a reference to a component in a narrower scope
- For stateful components, we need bijection - dynamic, contextual, bidirectional
- Seam's Bijection:
  - Wiring of dependencies throughout the lifetime
  - "Outjecting" promotes the value of a component property to a context variable where it can be picked up by another component or referenced in a jsf-view,

Persistence Context

- The notion of persistence context is central to ORM
- A process-scoped persistence context is evil
  - requires in-memory locking and sophisticated deadlock detection
- A transaction-scoped persistence context has problems if you re-use objects across transactions
  - LazyInitializationException navigating lazy associations
  - NonUniqueObjectException reassociating detached instances
  - Less opportunity for caching (workaround: use a second-level cache, which is quite unscalable)
- EJB3 component-scoped persistence context is nice
  - not held open for entire request (while rendering view)

Security

- Authentication - an extensible, JAAS-based authentication layer that allows users to authenticate against any security provider.
- Identity Management - an API for managing a Seam application's users and roles at runtime.
- Authorization - an extremely comprehensive authorization framework, supporting user roles, persistent and rule-based permissions, and a pluggable permission resolver for customized security logic.
- Permission Management - a set of built-in Seam components to allow easy management of an application's security policy.
- CAPTCHA support
- Supports declarative security settings
  - Fine-grained security (including method & instance)

Ajax Support

- Seam's totally unique concurrency model and state-management model was conceived and designed with AJAX in mind.
- Page wide support (region, zone)
  - Add support to existing components
  - Sub view processing
  - Partial tree rendering, partial page refresh
  - Normal lifecycle
- Component Wide
  - Ajaxified components
  - Client validations
  - Client component interaction
  - Custom lifecycle
- Ajax Remoting
  - Similar to DWR
  - Access seam components from JS
  - JavaScript APIs
  - Expose server side components @WebRemote
  - Works with Ajax4Jsf, Dojo, GWT

Integration Testing

- Seam components can easily be tested in TestNG or Junit
- The JBoss Embeddable EJB3 container is a great platform for integration testing: perform an end to end testing in it's own embeddable container -- in a single unit test!
- Test he entire flow of a request or conversation
- Test all layers of Java code in the application, from presentation to persistence.

Seam-gen

- Generate a project structure with build
- Scaffolding
- Generate crud views
- Reverse engineering of pojos
- Lookup routine for establishing link to a related entity
- Entity model validations enforced with ajax feedback
- Incremental hot deploy of static resources
- Ready made project files for eclipse, netbeans, idea
- Basic page level authorization
- Seeding of database from import.sql on classpath
- Richfaces Ui components

Some Misconceptions

- JBoss Seam applications can run only on JBoss Application Server – false.
- JBoss Seam applications can use only RichFaces or ICEFaces JSF libraries as their front-end – false.
- Stateful session beans are unscalable!
  - Not true, at least, they are no more unscalable than **HttpSession**
  - JBoss EJB3 has very efficient stateful session bean replication built using JBoss Cache
- Needs EJB3: False
- Needs a container: False

Other Technologies?

- But what about Spring, Spring MVC, Grails, hibernate stack or JSF?
- Grails ... Probably.
- But Seam is:
  - An “application stack” not a “web framework”
  - A unified development platform of {programming model, frameworks, best practices and tooling}

Resources

- Reference docs, more than 30 examples, forums and best overall place – www.seamframework.org
- Seam Session handling by Steffen Ryll wendtstud1.hpi.uni-potsdam.de/sysmod-seminar/SS2006/presentations/17_ JBossSeam_Session_Handling.pdf
- Richfaces 3.3 – http://www.jboss.org/jbossrichfaces/docs/
- JSF 2 (includes "seam" like features) – http://nejug.org/events/show/91
- Webbeans JSR299 (inspired from Seam) –http://jcp.org/en/jsr/detail?id=299
- EJB 3.1 – http://jcp.org/en/jsr/detail?id=318
- Seam Books – **Seam in Action** (Dan Allen) & Seam Framework – Experience the evolution of Java EE (Jacob Orshalick)
- Tools: JBoss Tools for Eclipse. Intellij and Netbeans also have excellent support for Seam
- Refcards – http://refcardz.dzone.com/refcardz/core-seam
  – http://refcardz.dzone.com/refcardz/seam-ui
- My Tech Blog – www.reverttoconsole.com