

Seam and Web Beans

Not just evolutionary, revolutionary

Jacob Orshalick

Focus IT Solutions, LLC

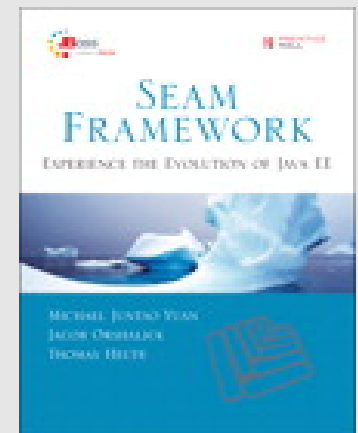
jacob@solutionsfit.com

<http://solutionsfit.com/blog/>

About Me

- Consultant for Focus IT Solutions, LLC
- Seam project committer
- Often blogging on Seam and related technologies
- Co-author of Seam Framework: Experience the Evolution of Java EE

Michael Yuan, Jacob Orshalick, Thomas Huete



What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, No LIEs
- RESTful URLs
- Rapid Application Development
- Web Beans (JSR-299)

What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, No LIEs
- RESTful URLs
- Rapid Application Development
- Web Beans (JSR-299)

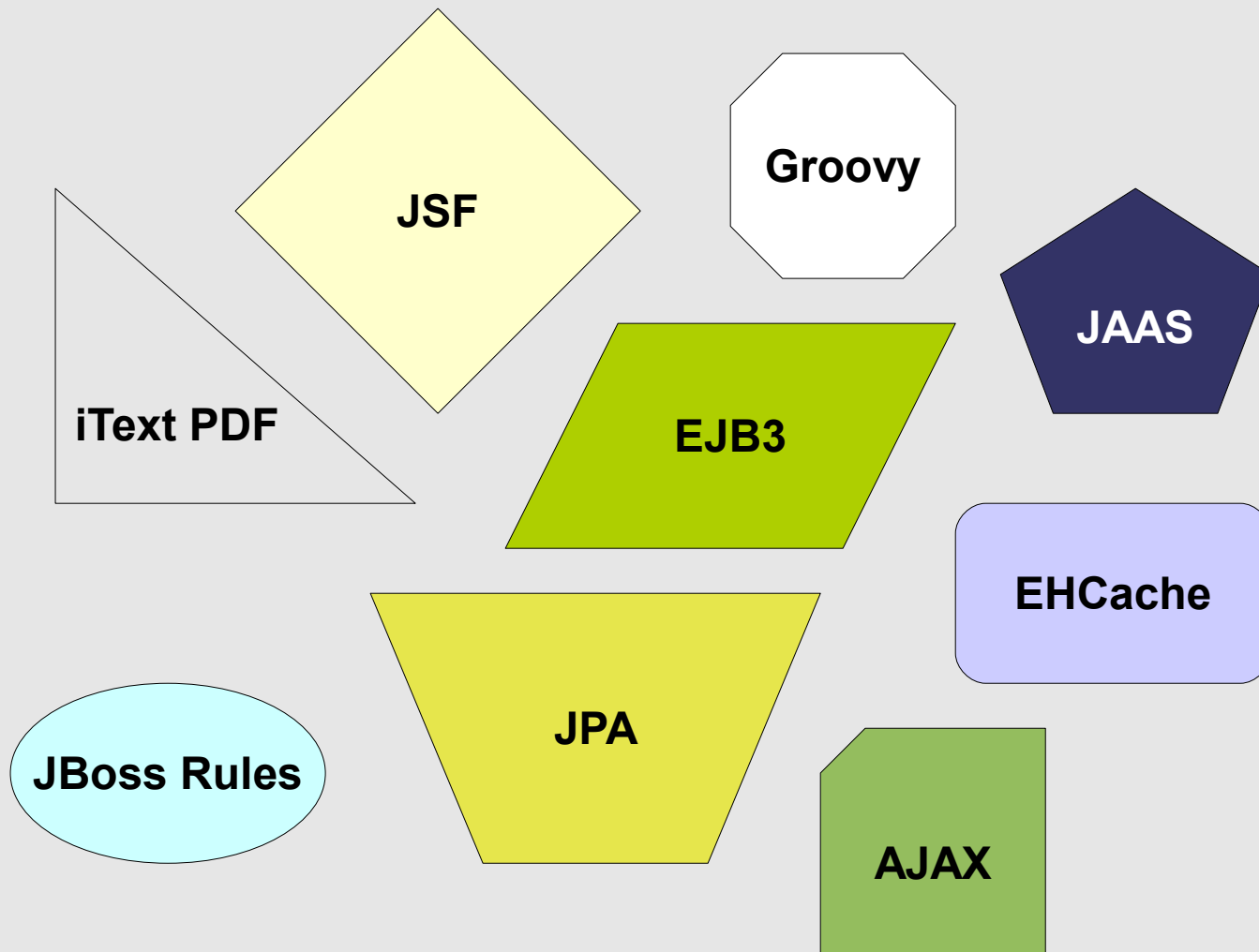
Integration Framework

Integration is one of the hardest problems in enterprise application development.

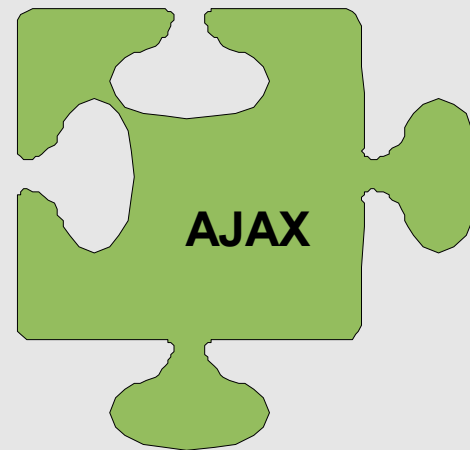
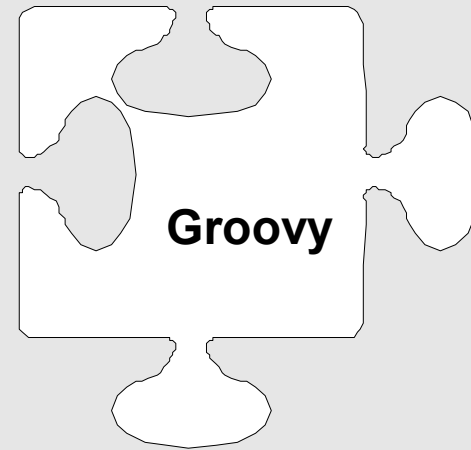
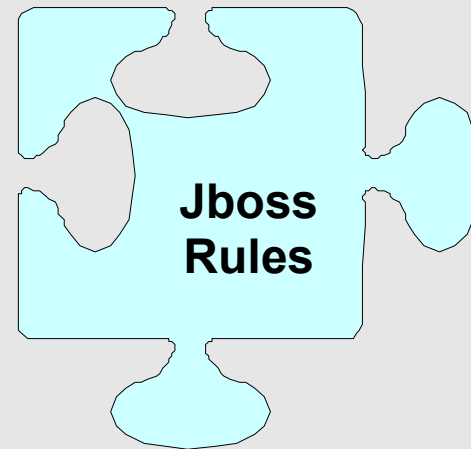
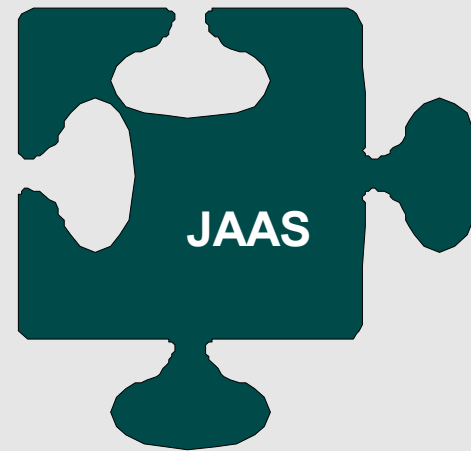
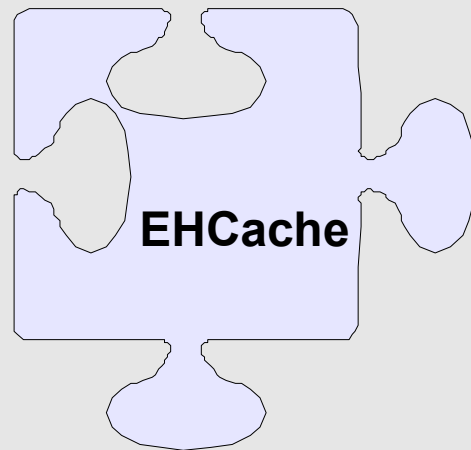
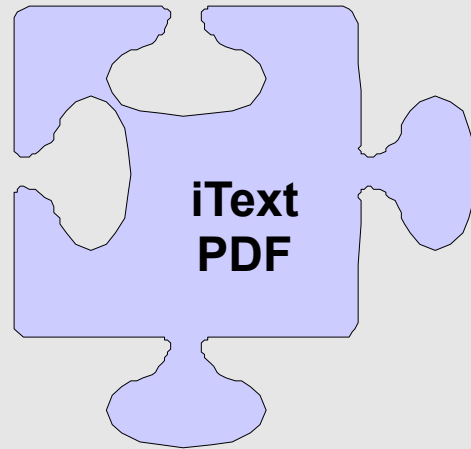
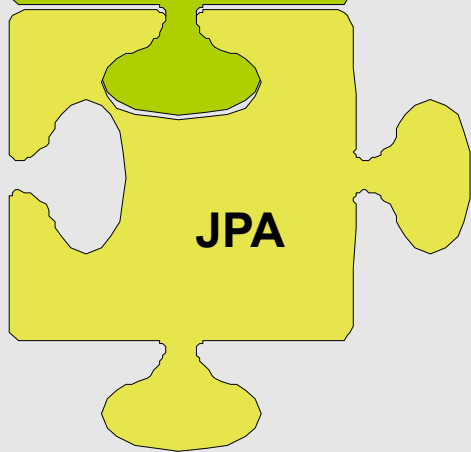
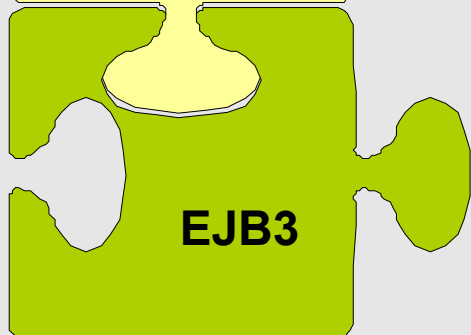
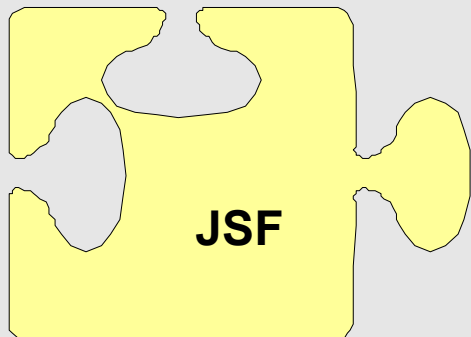
Integration Framework

- A multi-tier application is typically made up of many components:
 - transactional services, security, data persistence, asynchronous messaging, UI rendering, etc.
- Integrating these components can be a daunting task with all of the technologies in the Java EE stack!
- Seam makes integration simple with minimal configuration.

Integration Framework



Integration Framework



Integration Framework

- Finally a component model that crosses tiers:
 - Collapses artificial layer between EJB3 and JSF.
 - Provides consistent, annotation-based component model making components available across tiers.
 - Reduces or eliminates XML!

```
@Stateful
@Name("hotelBooking")
@Restrict("#{identity.loggedIn}")
public class HotelBookingAction implements HotelBooking
{
    @In private EntityManager em;
```

Component available through EL

```
<div class="input">
    <h:commandButton id="room_preference" value="Select Room"
        action="#{hotelBooking.setBookingDates}"/>&#160;
    <s:button id="cancel" value="Cancel" action="#{hotelBooking.cancel}"/>
</div>
```

Integration Framework

- EJB3 and JSF are not required!
 - This is a common misconception
 - Seam provides direct POJO support
 - Seam supports Spring beans
 - Use Flex, Wicket, Tapestry, or GWT for UI

Integration Framework

- Seam is Web 2.0 ready and provides direct AJAX support
- Challenges for Web 2.0 applications:
 - Increased database load with frequent server requests
 - Large object graphs in social networking applications
 - Concurrency as requests are generally asynchronous

Integration Framework

- How Seam handles it:
 - Stateful persistence context acts as an in-memory cache and provides correct support for lazy-loading
 - Direct support for multi-layered caching (web tier, business tier, and ORM)
 - Automated concurrency management

What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, No LIEs
- RESTful URLs
- Rapid Application Development
- Web Beans (JSR-299)

Simplifying JSF

Seam makes JSF worth using!

Simplifying JSF

- Direct Facelets support (<https://facelets.dev.java.net/>)
- EL Extension
 - Seam allows method expressions in EL to accept method parameters

```
<h:column>  
  <f:facet name="header">Action</f:facet>  
  <s:link id="viewHotel" value="View Hotel" action="#{hotelBooking.selectHotel(hot.id) }"/>  
</h:column>  
</h:dataTable>
```

- Seam also makes EL available beyond a web page

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

Simplifying JSF

- Additional Contexts
- Automating concurrency management
- Integrating ORM across application layers

Simplifying JSF

- Adhering to DRY with Validations
 - Direct integration of Hibernate bean validations

```
@NotNull(message="Credit card number is required")
@Length(min=16, max=16, message="Credit card number must 16 digits long")
@Pattern(regex="\\d*", message="Credit card number must be numeric")
private String creditCard;

public Booking() {}
```

- Integrated into the JSF validations phase

```
<div class="entry">
  <div class="label">
    Credit Card #:
  </div>
  <div class="input">
    <h:inputText id="creditCard" value="#{booking.creditCard}">
      <s:validate />
    </h:inputText>
    <s:message id="message" styleClass="error errors"/>
  </div>
</div>
```

Simplifying JSF

- Eliminating backing beans
 - Business layer can be directly invoked from page
 - Action components can be EJBs or POJOs
- Elimination of XML hell

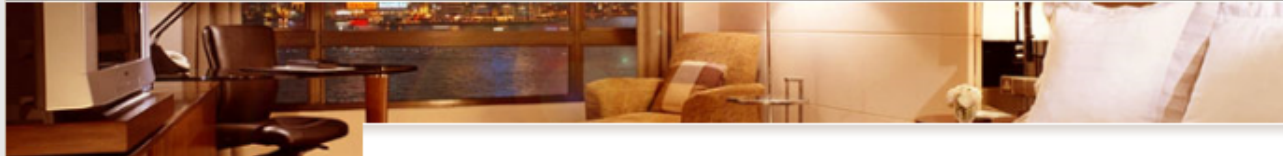
```
@Stateful
@Name("hotelBooking")
@Restrict("#{identity.loggedIn}")
public class HotelBookingAction implements HotelBooking
{
    @In private EntityManager em;
```

Simplifying JSF

jboss suites

seam framework demo

Welcome Demo User | Search | Settings | Logout



Nesting conversations

Nested conversations allow the application to capture a consistent continuable state at various points in a user interaction, thus insuring truly correct behavior in the face of backbuttoning and workspace management.

Continuing the conversation

Try going back and selecting a different room. You will notice a new nested conversation in the list of workspace entries. Each nested conversation is associated with the specific room selected ensuring consistent behavior.

Payment

Name: Marriott Courtyard

Address: Tower Place, Buckhead

City, State: Atlanta, GA

Zip: 30305

Country: USA

Room Preference: King Room

Total payment: \$120.00

Check In Date: Sat Oct 04 00:00:00 CDT 2008

Check Out Date: Sun Oct 05 00:00:00 CDT 2008

Credit Card #: *

Credit card number must 16 digits long

Credit Card Name: *

Credit Card Expiry:

[Proceed](#)

[Revise Room](#)

What will we cover?

- Integration Framework
- Simplifying JSF
- **Contextual Components**
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, No LIEs
- RESTful URLs
- Rapid Application Development
- Web Beans (JSR-299)

Contextual Components

- The component scopes:
 - Stateless: Components in this context are completely stateless and do not hold any state data of their own.
 - Event: This is the "narrowest" stateful context in Seam. Components in this context maintain their states throughout the processing of a single JSF request.
 - Page: Components in this context are tied to a specific page. You can have access to those components from all events emitted from this page.
 - Conversation: In Seam, a conversation is a series of web requests to accomplish a certain task (e.g., to checkout the shopping cart). Components tied to a conversation context maintain their state throughout the conversation.

Contextual Components

- The Component Scopes (continued)
 - Session: Components in the session context are managed in an HTTP session object. They maintain their states until the session expires. You can have multiple conversations in a session.
 - Business process: This context holds stateful components associated with a long running business process managed in the JBoss jBPM (Business Process Manager) engine. While all the previously discussed contexts manage stateful components for a single web user, the business process components can actually span across several users.
 - Application: This is a global context that holds static information. There is no concept of web users in this context.

What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, No LIEs
- RESTful URLs
- Rapid Application Development
- Web Beans (JSR-299)

Contextual Injection = Bijection

- Traditional Dependency Injection
 - Stateless by nature
 - Works great for stateless service applications
 - What about when state is required?

Contextual Injection = Bijection

- Bijection
 - Dependencies injected from context on invocation
 - Performs a scope search for dependencies
 - Dependencies disinjected once complete
 - Provides context to execute within

Contextual Injection = Bijection

```
@Stateful
@Name("hotelBooking")
@Restrict("#{identity.loggedIn}")
public class HotelBookingAction implements HotelBooking
{
    @In private Hotel hotel;
    @In private User user;
    @Out private Booking booking;

    public String bookHotel()
    {
        booking = new Booking(hotel, user);
        Calendar calendar = Calendar.getInstance();
        booking.setCheckinDate( calendar.getTime() );
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        booking.setCheckoutDate( calendar.getTime() );

        return "book";
    }
}
```

2) Inject dependencies

5) Disinject dependencies

4) Outject contributions

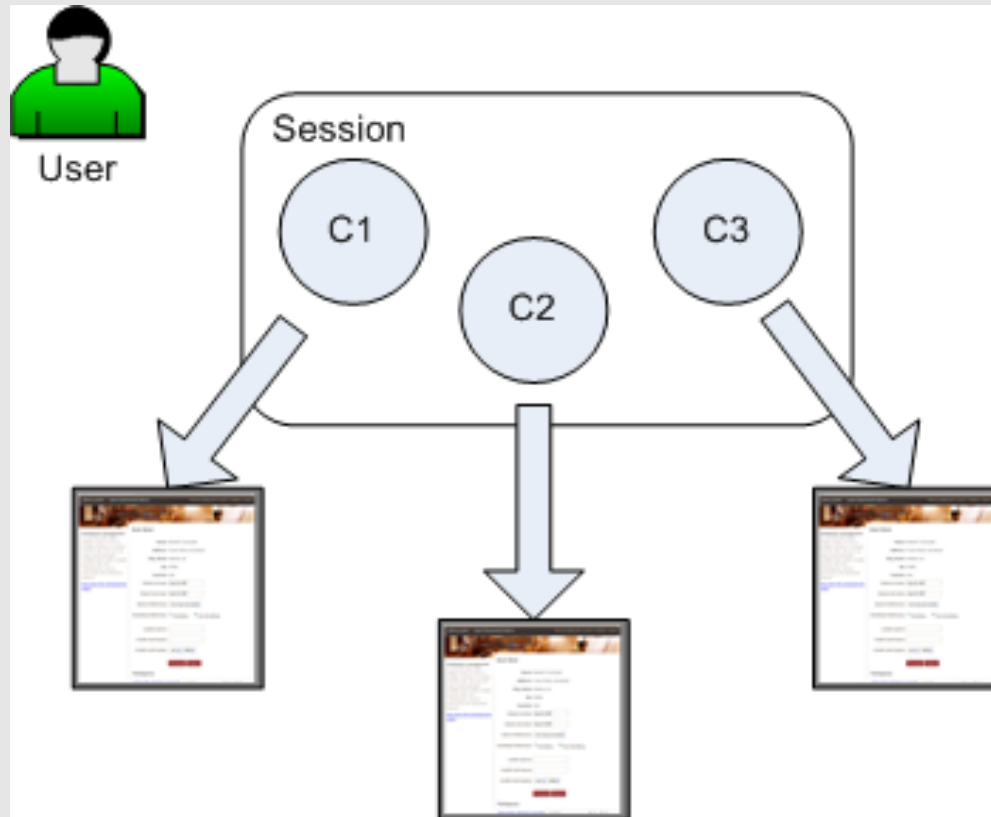
1) Invoke method

3) Execute method

What will we cover?

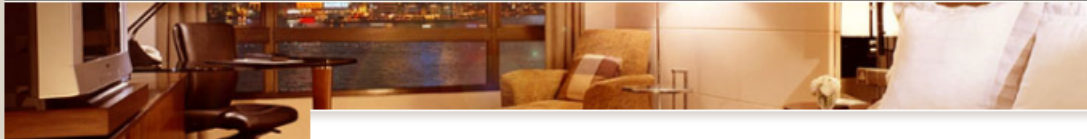
- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- **The Conversation Model**
- Web transactions, No LIEs
- RESTful URLs
- Rapid Application Development
- Web Beans (JSR-299)

The Conversation Model



The Conversation Model

jboss suites | seam framework demo Welcome Demo User | [Settings](#) | [Logout](#)



Stateful and contextual components
Now, you go through a series of pages to complete a booking transaction. Seam maintains the application state for you and only commits to the database when the transaction is successful.

[How Seam manages states?](#)

Furthermore, Seam also supports multiple workspaces. You can open multiple browser windows (or tabs) and have Seam maintaining the application state independently in each window. The "workspace" section in the main panel showcases this feature.

Workspaces

[New workspace](#) | [New window](#) | [What is it?](#)

Search Hotels

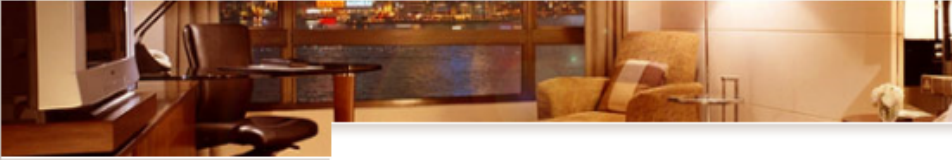
(e.g., enter word atlanta as the search string)

Maximum results to display: 10

Name	Address	City, State	Zip	Action
Marriott Courtyard	Tower Place, Buckhead	Atlanta, GA	30305	View Hotel
Doubletree	Tower Place, Buckhead	Atlanta, GA	30305	View Hotel
W Hotel	Union Square, Manhattan	NY, NY	10011	View Hotel
W Hotel	Lexington Ave, Manhattan	NY, NY	10011	View Hotel
Hotel Rouge	Dupont Circle	Washington, DC	20036	View Hotel
70 Park Avenue Hotel	70 Park Avenue	NY, NY	10011	View Hotel
Conrad Miami	1395 Brickell Ave	Miami, FL	33131	View Hotel
Sea Horse Inn	2106 N Clairemont Ave	Eau Claire, WI	54703	View Hotel
Super 8 Eau Claire Campus Area	1151 W Macarthur Ave	Eau Claire, WI	54701	View Hotel
Marriot Downtown	55 Fourth Street	San Francisco, CA	94103	View Hotel

The Conversation Model

jboss suites seam framework demo Welcome Demo



Stateful and contextual components
Now, you go through a series of pages to complete a booking transaction. Seam maintains the application state for you and only commits to the database when the transaction is successful.

[How Seam manages states?](#)

Don't kill your database
Keeping conversational state in memory in the middle tier is a great way to improve your application's scalability. It saves hitting the database

Workspaces

[New workspace](#) | [New window](#) | [What is it?](#)

View hotel: Doubletree [current]	09
View hotel: Marriott Courtyard	09

View Hotel

Name: Doubletree

Address: Tower Place, Buckhead

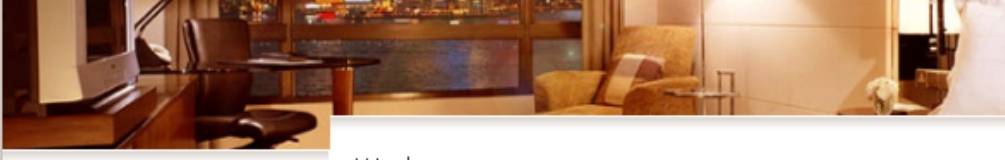
City: Atlanta

State: GA

Zip: 30305

[Book Hotel](#) [Back to Search](#)

jboss suites seam framework demo Welcome Demo User



Stateful and contextual components
Now, you go through a series of pages to complete a booking transaction. Seam maintains the application state for you and only commits to the database when the transaction is successful.

[How Seam manages states?](#)

Don't kill your database
Keeping conversational state in memory in the middle tier is a great way to improve your application's scalability. It saves hitting the database

Workspaces

[New workspace](#) | [New window](#) | [What is it?](#)

View hotel: Marriott Courtyard [current]	09:3
View hotel: Doubletree	09:3

View Hotel

Name: Marriott Courtyard

Address: Tower Place, Buckhead

City: Atlanta

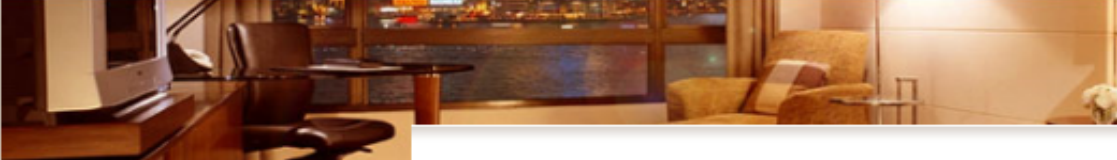
State: GA

Zip: 30305

[Book Hotel](#) [Back to Search](#)

The Conversation Model

jboss suites | seam framework demo Welcome



Stateful and contextual components
Now, you go through a series of pages to complete a booking transaction. Seam maintains the application state for you and only commits to the database when the transaction is successful.

[How Seam manages states?](#)

Furthermore, Seam also supports multiple workspaces. You can open multiple browser windows (or tabs) and have Seam maintaining the application state independently in each window. The "workspace" section in the main panel showcases this feature.

Workspaces

[New workspace](#) | [New window](#) | [What is it?](#)

[Confirm: Marriott Courtyard, Oct 8, 2008 to Oct 9, 2008](#)[current]

[Confirm: Doubletree, Oct 8, 2008 to Oct 9, 2008](#)

Confirm Hotel Booking

Name: Marriott Courtyard

Address: Tower Place, Buckhead

City: Atlanta

Zip: 30305

Check In Date: Oct 8, 2008

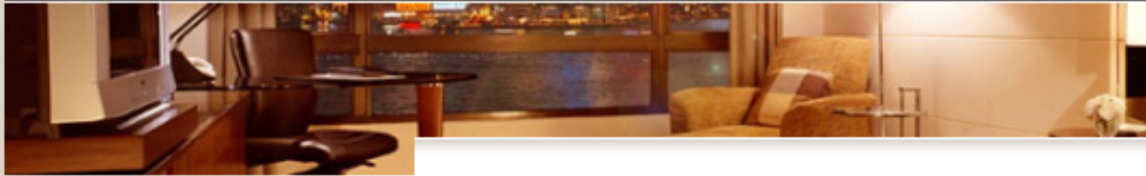
Check Out Date: Oct 9, 2008

Credit Card #: 1234123412341234

[Confirm](#) [Revise](#)

The Conversation Model

jboss suites | seam framework demo Welcome D



Stateful and contextual components
Now, you go through a series of pages to complete a booking transaction. Seam maintains the application state for you and only commits to the database when the transaction is successful.

[How Seam manages states?](#)

Furthermore, Seam also supports multiple workspaces. You can open multiple browser windows (or tabs) and have Seam maintaining the application state independently in each window. The "workspace" section in the

Workspaces

[New workspace](#) | [New window](#) | [What is it?](#)

[Confirm: Doubletree, Oct 8, 2008 to Oct 9, 2008](#)[cur

[View hotel: Marriott Courtyard](#)

Confirm Hotel Booking

Name: Doubletree

Address: Tower Place, Buckhead

City: Atlanta

Zip: 30305

Check In Date: Oct 8, 2008

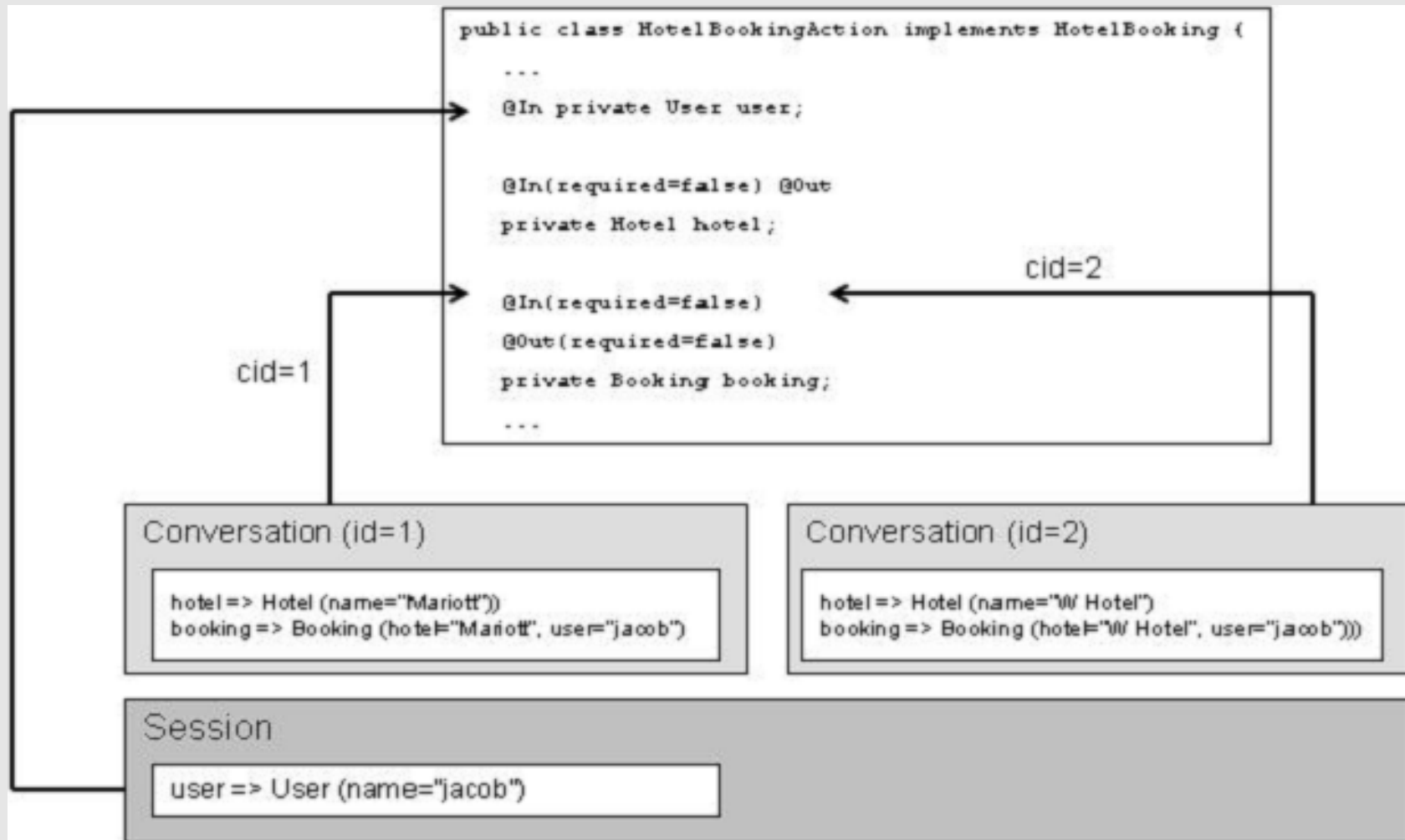
Check Out Date: Oct 9, 2008

Credit Card #: 1234123412341234

[Confirm](#) [Revise](#)

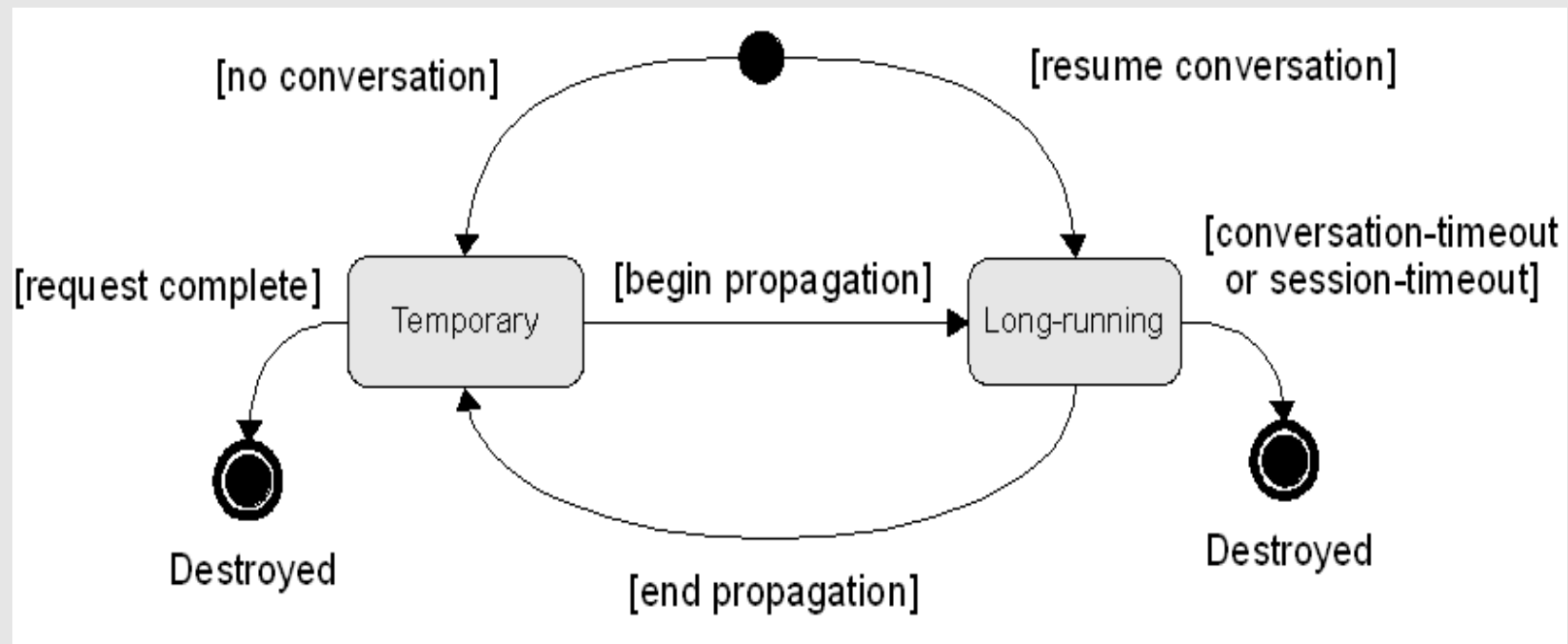
The Conversation Model

- Conversation vs. Session



The Conversation Model

- The Conversation Lifecycle



The Conversation Model

- Beginning a conversation

```
@Stateful
@Name("hotelBooking")
@Restrict("#{identity.loggedIn}")
public class HotelBookingAction implements HotelBooking
{
    @In private EntityManager em;

    @In(required=false) @Out
    private Hotel hotel;

    @Begin(flushMode=FlushModeType.MANUAL)
    public void selectHotel(Long id)
    {
        hotel = em.find(Hotel.class, id);
    }
}
```

The Conversation Model

- Ending the conversation

```
@End(root=true)
public void confirm()
{
    booking.setRoomPreference(roomSelection);
    processPayment(booking);
    em.persist(booking);
    statusMessages.add("Thank you, #{user.name}, your confirmation number " +
        "for #{hotel.name} is #{booking.id}");
    log.info("New booking: #{booking.id} for #{user.username}");
    events.raiseTransactionSuccessEvent(NEW_BOOKING_EVENT, booking);

    // when using manual flushmode we must use em.flush() to flush
    // the persistence context
    em.flush();
}

@End(beforeRedirect=true, root=true)
public void cancel() {}
```

What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- **Web transactions, No LIEs**
- **RESTful URLs**
- **Rapid Application Development**
- **Web Beans (JSR-299)**

Web Transactions, No LIEs

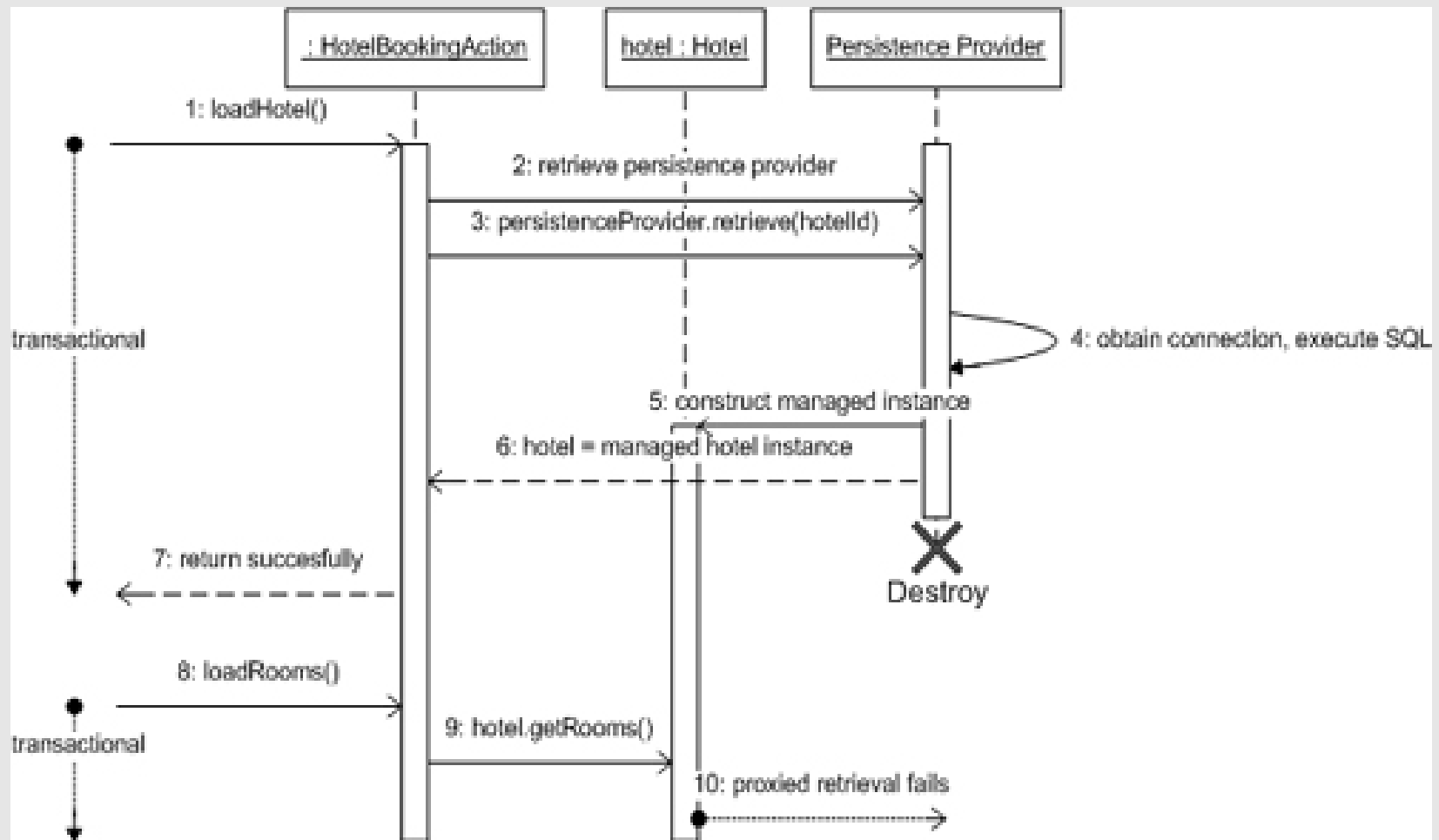
The big LIE we've been told is that the Persistence Context should be scoped to a transaction.

Web Transactions, No LIEs

- A Seam application typically assembles and modifies database entity objects throughout a conversation
- At the end of the conversation, we commit all those entity objects into the database
- If anything goes wrong, the entire transaction fails and the database remains unchanged
- Seam allows us to achieve this transactional behavior even with user think-time

Web Transactions, No LIEs

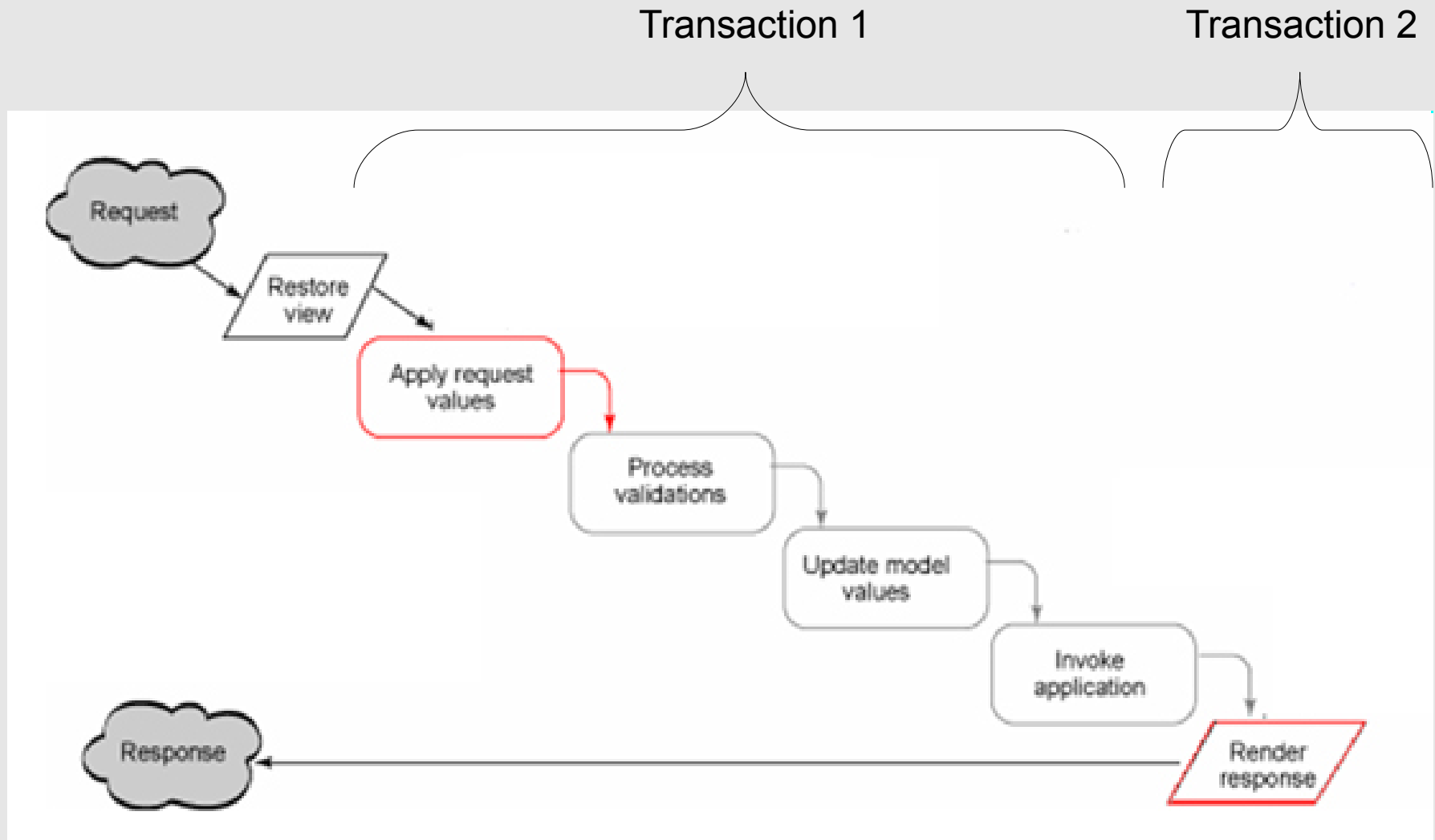
- LIEs (LazyInitializationExceptions) occur when an Entity becomes detached from the PersistenceContext



Web Transactions, No LIEs

- Seam avoids this by scoping the PersistenceContext to the conversation
- Entities remain attached throughout the conversation
- What about Open-Session-In-View?
 - Transactional boundaries are wrong
 - Seam uses two transactions per request; the first spans the beginning of the apply request values phase until the end of the invoke application phase; the second spans the render response phase

Web Transactions, No LIEs



Web Transactions, No LIEs

- SMPCs (Seam-managed Persistence Context)
 - Scoped to conversation and shared between components in conversation

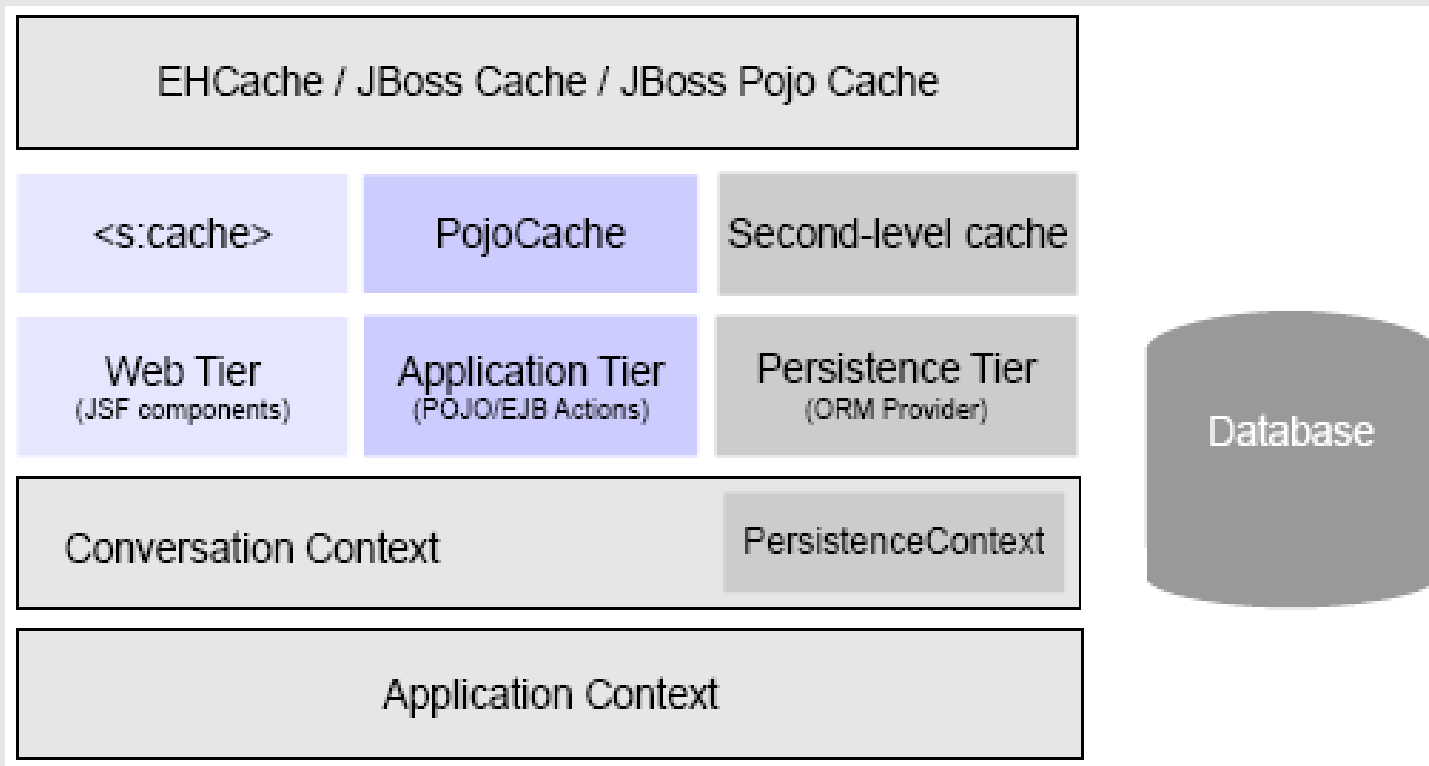
```
@Stateful
@Name("hotelBooking")
@Restrict("#{identity.loggedIn}")
public class HotelBookingAction implements HotelBooking
{
    @In private EntityManager em;
    @In private User user;
```

```
@Name("rewardsManager")
@Scope(ScopeType.CONVERSATION)
public class RewardsManager {
    @Logger private Log log;
    @In User user;
    @In EntityManager em;
```

Same conversation same EntityManager

Web Transactions, No LIEs

- Multi-layered caching in a Seam application



What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, No LIEs
- **RESTful URLs**
- **Rapid Application Development**
- **Web Beans (JSR-299)**

RESTful URLs

- Seam provides built-in support for REST
 - Useful for content-serving applications
 - Allows bookmarkable URLs
 - Meaningful URLs that describe content
- When searching for a hotel it would be useful to allow bookmarking.

RESTful URLs

- Built-in REST support

- The link can be written to the view as follows:

```
<h:outputLink value="/booking/hotel.seam?hotelId=#{hot.hotelId}">
    View Hotel
</h:outputLink>
```

- The resulting URL is a get request that loads the hotel through a factory method

```
@Factory(value="hotel")
public void loadHotel()
{
    // loads hotel into the conversation based on the RESTful id
    hotel = (Hotel) em.createQuery("select h from " +
        "Hotel h where hotelId = :identifier")
        .setParameter("identifier", hotelId)
        .getSingleResult();
}
```

RESTful URLs

- Mapping query parameters to beans is easy

```
<page view-id="/hotel.xhtml" login-required="true">
  <description>View hotel: #{hotel.name}</description>

  <param name="hotelId" value="#{hotelBooking.hotelId}" />
```

Page parameter set through EL expression

```
@Stateful
@Name("hotelBooking")
@Restrict("#{identity.loggedIn}")
public class HotelBookingAction implements HotelBooking
{
    private long hotelId;

    public long getHotelId() {
        return hotelId;
    }

    public void setHotelId(long hotelId) {
        this.hotelId = hotelId;
    }
}
```

RESTful URLs

- Meaningful URLs

- Direct integration of URLRewrite

```
<rule>
  <from>^/book/ ([A-Za-z0-9]*) $</from>
  <to last="true">/book.seam?hotelId=$1</to>
</rule>

<outbound-rule>
  <from>/book.seam\?hotelId= ([A-Za-z0-9]*) </from>
  <to>/book/$1</to>
</outbound-rule>
```

- URL rewritten:

- /book.seam?hotelId=MarriottCourtyardBuckhead
 - /book/MarriottCourtyardBuckhead
- Can be tied to natural conversations

What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, No LIEs
- RESTful URLs
- **Rapid Application Development**
- **Web Beans (JSR-299)**

Rapid Application Development

- Seam-gen is a RAD tool, that provides application scaffolding
- It's easy to create your first Seam project:
 - seam setup
 - seam create-project
- Or generate a project from a schema:
 - seam generate-entities
- Other commands:
 - seam new-form, seam new-action, seam deploy

What will we cover?

- Integration Framework
- Simplifying JSF
- Contextual Components
- Contextual Injection = Bijection
- The Conversation Model
- Web transactions, Combating LIEs
- RESTful URLs
- Rapid Application Development
- **Web Beans (JSR-299)**

Web Beans (JSR-299)

If Seam is evolutionary, Web Beans is revolutionary!

Web Beans (JSR-299)

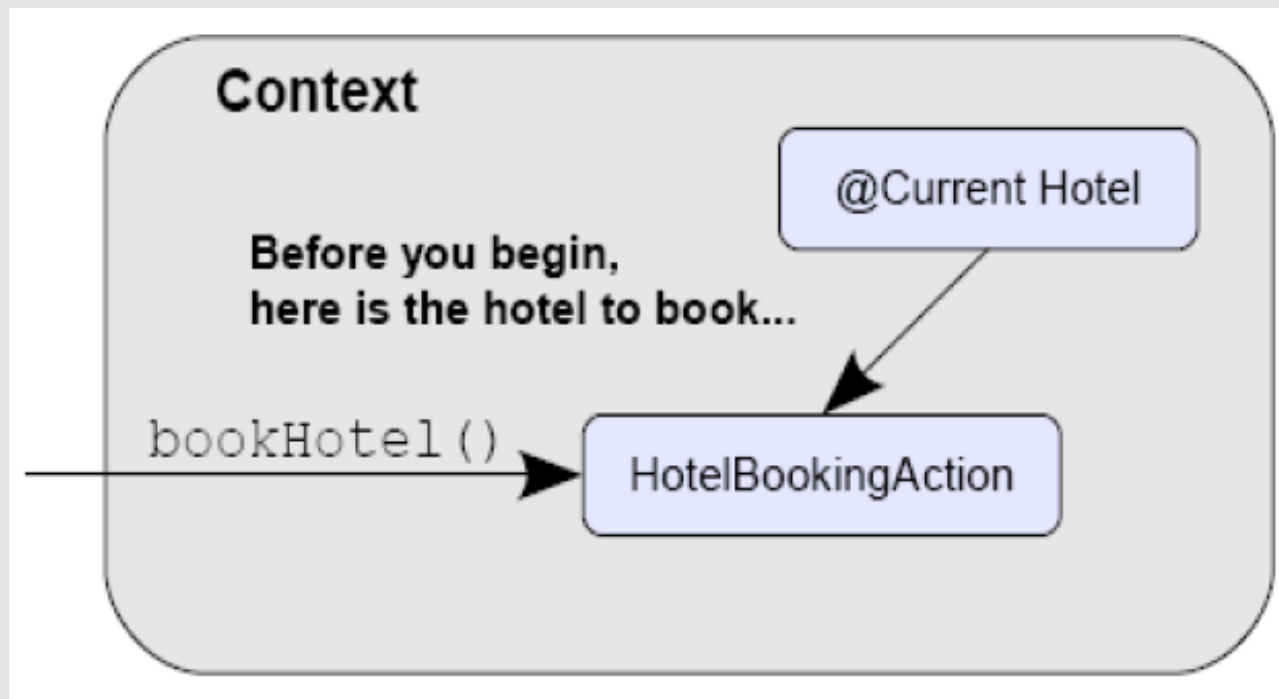
- The Web Beans specification (JSR-299) is collaborative community effort influenced by Seam and Google Guice.
- Web Beans will standardize a type-safe component model that applies across application tiers.
- This finally unifies the web tier and the EJB tier which greatly simplifies Java web development.

Web Beans (JSR-299)

- In the Web Beans Manifesto, the specification lead, Gavin King, described the theme of Web Beans as,
 - **"Loose coupling with strong typing"**
- Loose coupling provides the dynamic behavior that makes a system flexible to change.
- Unfortunately, loose coupling is often achieved by sacrificing type safety.

Web Beans (JSR-299)

- Contextual Injection



Web Beans (JSR-299)

- Defining a Web Beans component

Name a component for access through EL

Scoping a component to a context

```
@Named("hotelBookingAction")
@ConversationScoped
@Production
@Stateful
public class HotelBookingAction implements HotelBooking {
```

Deployment type

- Identify classes as Web Beans components.
- Deployment types specify install precedence and environment availability
- Specify your own deployment types with the `@DeploymentType` meta-annotation

Web Beans (JSR-299)

- Binding types

@Current is default injection

```
@Named("hotelBookingAction")
@ConversationScoped
@Production
@Stateful
public class HotelBookingAction implements HotelBooking {
    @Current User user;
    @CreditCard PaymentService creditCardPaymentService;
}
```

@CreditCard qualifies a type of PaymentService in a type-safe way.

```
@BindingType
@Target({METHOD, FIELD, PARAMETER, TYPE})
@Retention(RUNTIME)
public @interface CreditCard {}
```

```
@Production
@Named
@Stateless
@CreditCard
public class CreditCardPaymentService implements PaymentService {
```

Web Beans (JSR-299)

- Producer methods

```
@Produces @ConversationScoped @Named("hotels")
public List<Hotel> getAllHotels()
{
    return em.createQuery("select h from Hotel h").getResultList();
}

@Produces @ConversationScoped @Named("popularHotels")
public List<Hotel> getPopularHotels()
{
    return em.createQuery("select h from Hotel h where h.numBookings > 500")
        .getResultList();
}
```

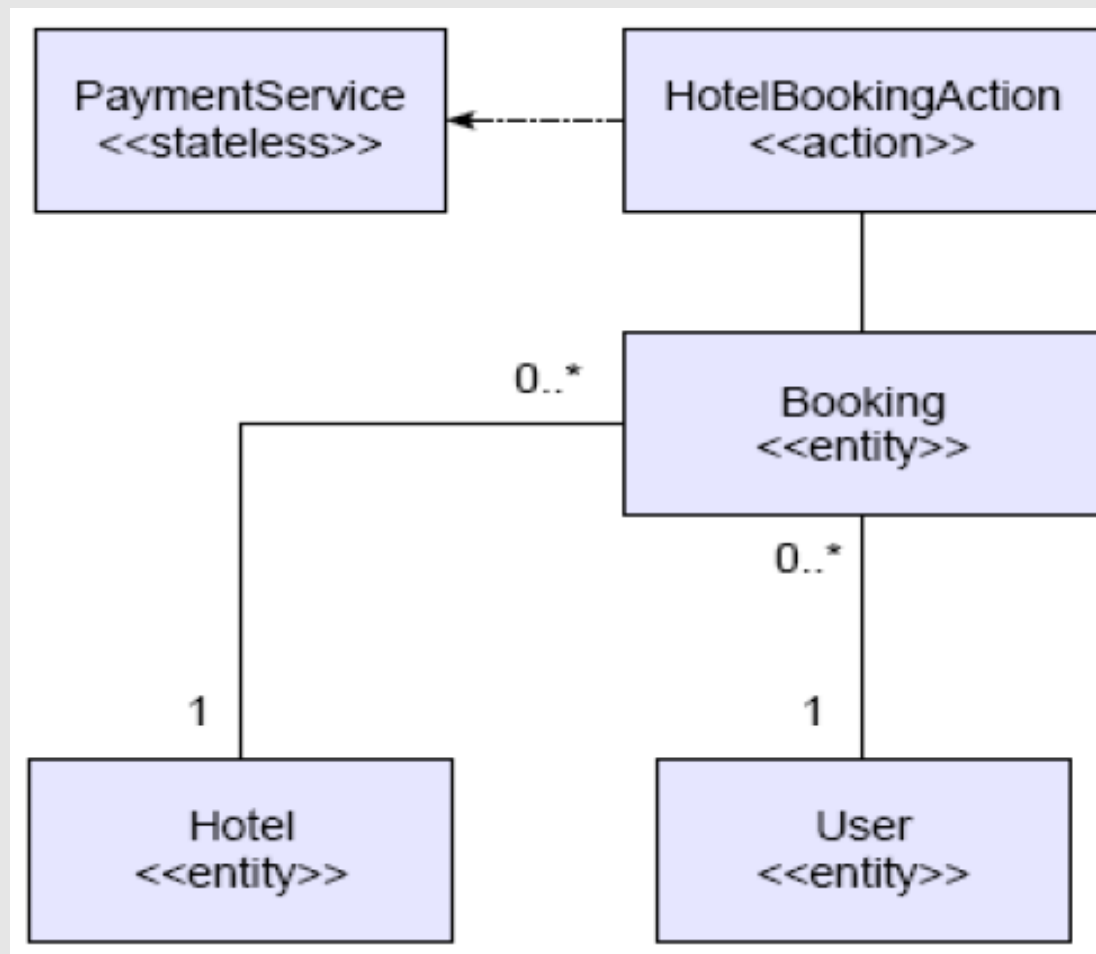
```
@Produces
public PaymentService getPaymentService(@Current Booking booking) {
    PaymentService paymentService;
    PaymentType paymentType = booking.getPayment().getPaymentType();

    switch(paymentType) {
        case CREDIT_CARD : paymentService = creditCardPaymentService;
            break;
        case REWARDS : paymentService = rewardsPaymentService;
            break;
        default: throw new IllegalStateException("A PaymentType has been defined without " +
            "an associated PaymentService");
    }

    return paymentService;
}
```

Web Beans (JSR-299)

- Stereotypes



Web Beans (JSR-299)

- Stereotypes

Common annotations

```
@Named("hotelSearchingAction")
@ConversationScoped
@Production
@Stateful
public class HotelSearchingAction implements HotelSearching {
```

```
@Named("hotelBookingAction")
@ConversationScoped
@Production
@Stateful
public class HotelBookingAction implements HotelBooking {
```

```
@Named
@ConversationScoped
@Production
@Stereotype(supportedScopes={RequestScoped.class, ConversationScoped.class})
@Target(TYPE)
@Retention(RUNTIME)
public @interface Action {}
```

So give it a go!

- There is so much more to cover...
- Just browse the Seam Reference Document, it's a great resource
- It's easy to get started, so give Seam a try!

Resources

- *Seam Framework: Experience the Evolution of Java EE*, Yuan, Orshalick, Huete
 - <http://solutionsfit.com/blog/2008/06/06/seam-framework-experience-the-evolution-of-java-ee/>
- Seam Home: <http://www.seamframework.org>
- Web Beans: <http://www.seamframework.org/WebBeans>
- My Blog: <http://solutionsfit.com/blog>
- Core Dev Blog: <http://in.relation.to>

Questions

- Anyone?