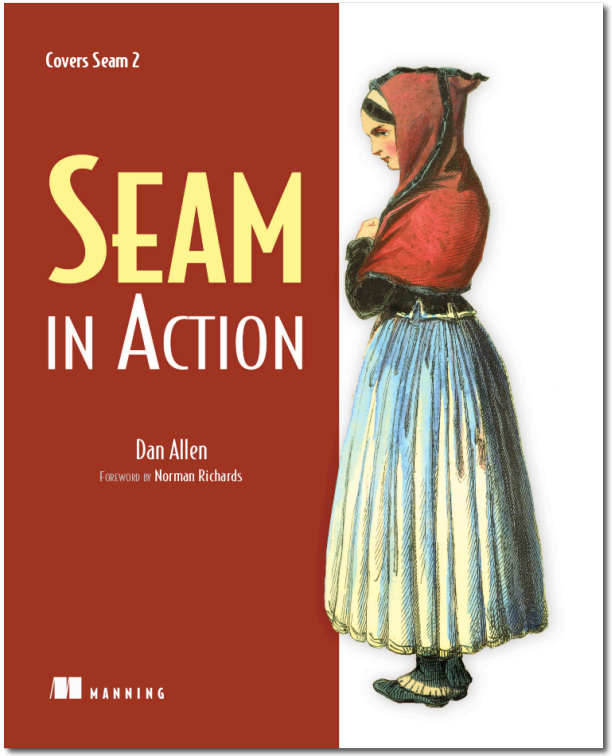
Maturing your application's security with Seam

Dan Allen
Senior Software Engineer
JBoss, a division of Red Hat

Who am I?

- **Author of Seam in Action**
- **Seam project member**
- **JSF user from the trenches**



Seam security assumptions

- **You are looking for a better security solution**
- **You are using Seam, JSF or both**
- **If not, you'll want to use Seam after this talk ;)**
- **Contrary to popular belief...**
  - **Seam is not invasive or heavyweight**
  - **Seam works on any major application server or servlet container**

Outline

- How JAAS left us hanging
- Security principles
- Authentication in 3 steps
- Identity management / declarative authentication
- Open ID: delegation of trust
- Four styles of authorization
- Permission management

JAAS, a surviving remnant of J2EE

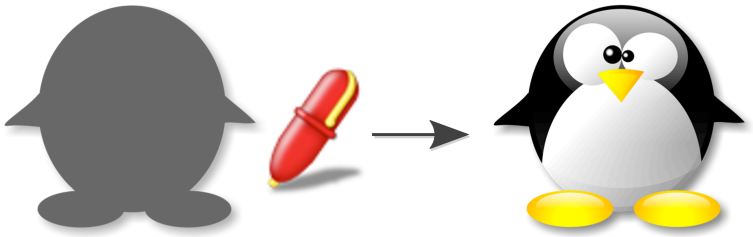
- **Entirely too complicated to setup**
- **Too container dependent**
- **Obscure configuration formats**
- **Poorly documented (in terms of examples)**
- **Pluggable? At what cost?**
- **Let's get back to basics**
  - **Borrow the APIs generic enough to reuse**

The security needs of a developer

- **It should be simple to setup**
- **It should be easy to manage**
- **The application should not outgrow it**

Security principles

- **Identity**
  - Who you are (security principal)
  - Isolates you from the guests
  - Granted roles and groups



- **Authentication**
  - Proving that you are you
  - Based on a secret you know
- **Authorization**
  - Resource control based on credentials

Authentication with Seam in 3 steps

1. **Designate an authentication method**
2. **Create a JSF login form**
3. **Write the authentication method**

Step 0: No prerequisites

- **Security is a core concern in Seam**
- **Eager or lazy authentication**
  - **Includes built-in support for routing user to login page**
  - **Use events to capture current view and redirect user back to it**
  - **Customized using navigation rules**
- **Authentication already setup in seam-gen projects**

Demo: Authentication in seam-gen

Step 1: Switching on authentication

- **Declare an authentication method in components.xml**

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

- **Authentication method requirements:**
  - **No arguments**
  - **Return boolean indicating if credentials are valid**
  - **Must be accessible via the EL**
- **Otherwise, the method can:**
  - **Have any name**
  - **Reside on any class (doesn't have to implement any special interfaces)**
- **Used behind the scenes by JAAS**

Step 2: Create a JSF login form

- **Native JSF support!**
  - **Bind credentials to properties on built-in identity component**
  - **Attach form action to login() method to kick off authentication**
- **Built-in remember me support**
  - **auto login or username only**

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

Step 3: Write an authentication method

- **Adapts to any authentication backend**
  - **Cross reference credentials to identity store (e.g., database)**
- **Basic procedure**
  - credentials **component delivers username/password to be validated**
  - **If credentials look good, you grant roles using** identity **component**
  - identity **component tracks "logged in" state and username**

```
@Name("authenticator")
public class Authenticator {
    @In protected Identity identity;
    @In protected Credentials credentials;
    public boolean authenticate() {
        out.println("Login attempt by " + credentials.getUsername());
        identity.addRole("admin");
        return true;
    }
}
```

Data-driven authentication method

- **Query database using JPA EntityManager**

```
@Name("authenticator")
public class Authenticator {
    @In protected Identity identity;
    @In protected Credentials credentials;
    @In("entityManager") protected EntityManager em;
    public boolean authenticate() {
        try {
            UserAccount user = (UserAccount) em.createQuery(
                "select u from UserAccount u " +
                "where u.username = #{credentials.username}")
                .getSingleResult();
            if (user.getPassword().equals(credentials.getPassword())) {
                identity.addRole("member");
                return true;
            }
        } catch (NoResultException e) {}
        return false;
    }
}
```

Turning authentication over to Seam

- **Identity management framework**
  - **Annotation-based**
  - **Pluggable identity store (JPA and LDAP supported out of the box)**
  - **Built-in CRUD operations for users and roles/groups**
    - **Access controlled by permissions**
  - **JSF "controllers" for querying users and roles**
- **Eliminates authentication method**

The catch: some addition configuration is required

Demo: Identity management

Identity configuration

- **Select identity store implementation (JPA or LDAP)**
- **Identify User and Role classes**

```
<security:jpa-identity-store
    user-class="com.company.app.model.UserAccount"
    role-class="com.company.app.model.UserRole"/>
```

- **Annotate User and Role classes**

```
@Entity
public class UserAccount {
    @UserPrincipal public String getUsername() { ... }

    @UserPassword(hash = "MD5") public String getPasswordHash() { ... }

    @UserRoles @ManyToMany public Set<UserRole> getRoles() { ... }
}
```

```
@Entity
public class UserRole {
    @RoleName public String getName() { ... }
}
```

Actions for managing identities

- UserSearch
  - **Populates data model of users and handles user selection**
- UserAction
  - **Manages conversation for adding user or modifying selected user**
- RoleSearch
  - **Populates data model of roles and handles role selection**
- RoleAction
  - **Manages conversation for adding role or modifying selected role**

Delegating authentication to a third party

- **Open ID**
  - **Eliminates the need for multiple usernames across different websites**
  - **Users gets to choose who to trust with their credentials**
  - **You don't have the burden of maintaining authentication secrets**
- **Seam has a built-in openid component**
  - **Negotiates with third party to assign user an identity principal**
  - **Used in place of identity component on login page; no password!**
- **You may still want to create a local profile for the user**
  - **Can redirect new user to registration page after login**

Demo: Open ID

Open ID setup

- **Add phase listener for handling callback from provider**

```
<phase-listener>org.jboss.seam.security.openid.OpenIdPhaseListener</phase-listener>
```

- **Add Open ID libraries and dependencies to classpath**
  - **openid4java**
- **Create login page and configure navigation rules**

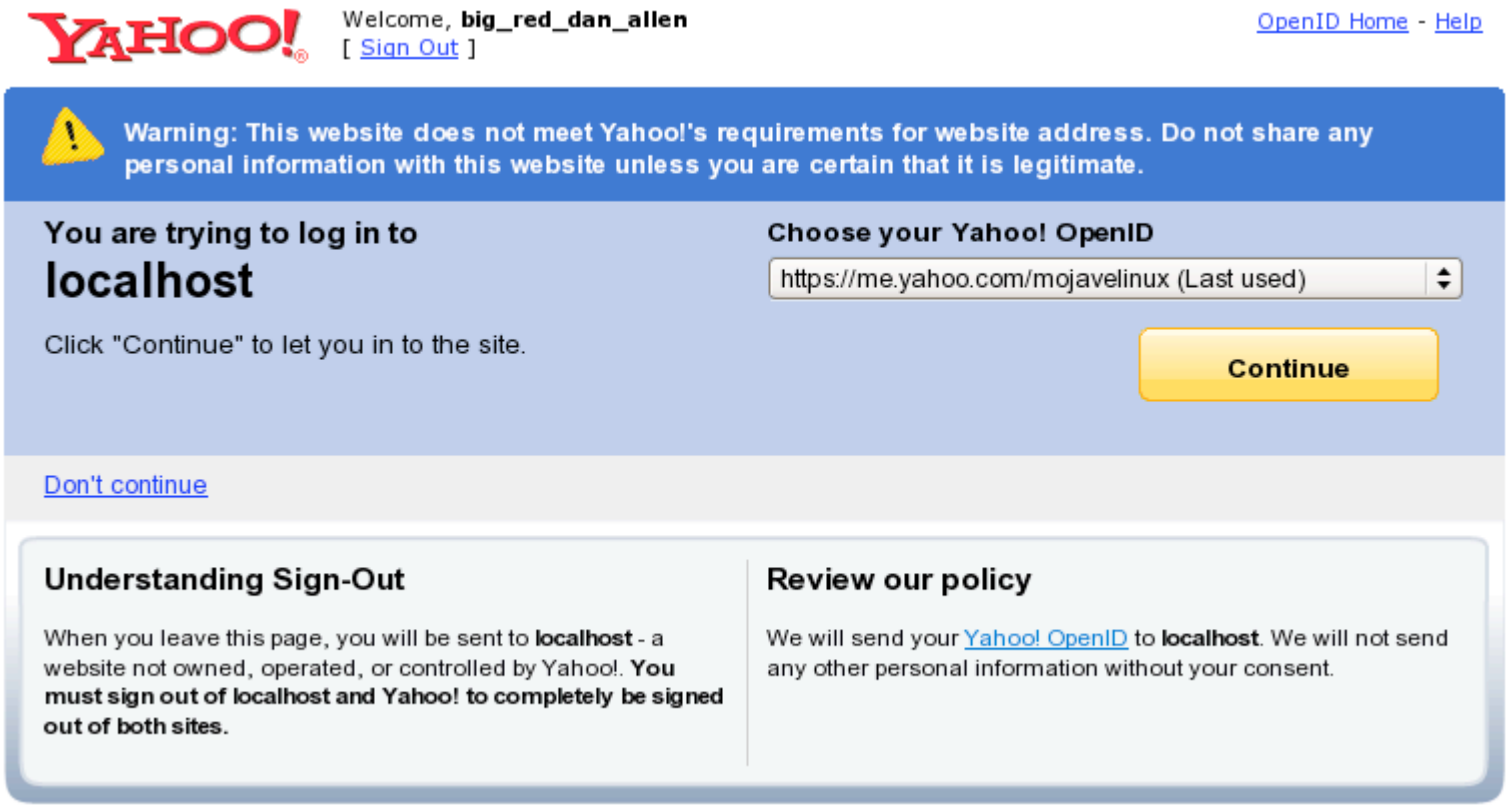
Open ID login page

- **User chooses provider (AOL, Blogger, Yahoo, etc)**
- **Seam negotiates hand-off (using openid4java)**

```
<h:form id="login">
    <h:outputLabel for="openid">Open ID</h:outputLabel>
    <h:inputText id="openid" value="#{openid.id}"/>
    <h:commandButton value="Login" action="#{openid.login}"/>
</h:form>
```

- **Returns to /openid.xhtml pseudo-view after login**
- **Using navigation rules, you can either...**
  - **Transfer Open ID account to user principal**
  - **Route user to registration page**
- **Open ID accessed using #{openid.validatedId}**

Open ID identity transfer



Authorization styles

- **Binary**
  - **Separates members from the guests**
- **Role-based**
  - **Stereotypes users**
- **Rule-based**
  - **Declarative and contextual rules**
- **Access Control Lists (ACLs)**
  - **Grants to specific object instances**
  - **Stored in database**

Binary authorization

- **Often first requirement**
- **Requires user to have an identity**
- **Identity component reports “logged in” state**

Java

```
if (identity.isLoggedIn()) {
    ...
}
```

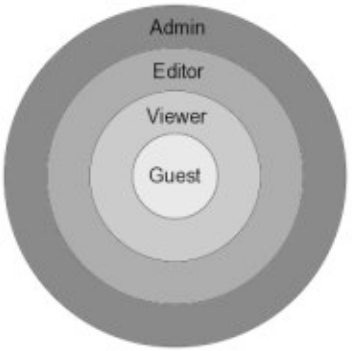
Seam page descriptor

```
<page view-id="/membersOnly.xhtml"
    login-required="true">
    ...
</page>
```

EL

```
<h:panelGroup rendered="#{identity.loggedIn}">
    Rate this post...
</h:panelGroup>
```

Role-based authorization



- **Coarse-grained security**
  - **Good for sectioning off areas of application**
- **Roles are assigned during authentication**
  - **identity.addRole("ROLE_NAME") for custom authentication**
  - **@Roles mapping when using identity store**
- **Seam doesn't dictate a naming convention for roles**

Java

```
if (identity.hasRole("admin")) {
    ...
}
```

JBoss EL

```
<s:link view="/admin/home.xhtml"
    rendered="#{identity.hasRole("admin")}"
    value="Admin Area"/>
```

Declarative restrictions

- **JSF views**

```
<page login-required="true"/>
```

```
<restrict>#{identity.hasRole("admin")}</restrict>
```

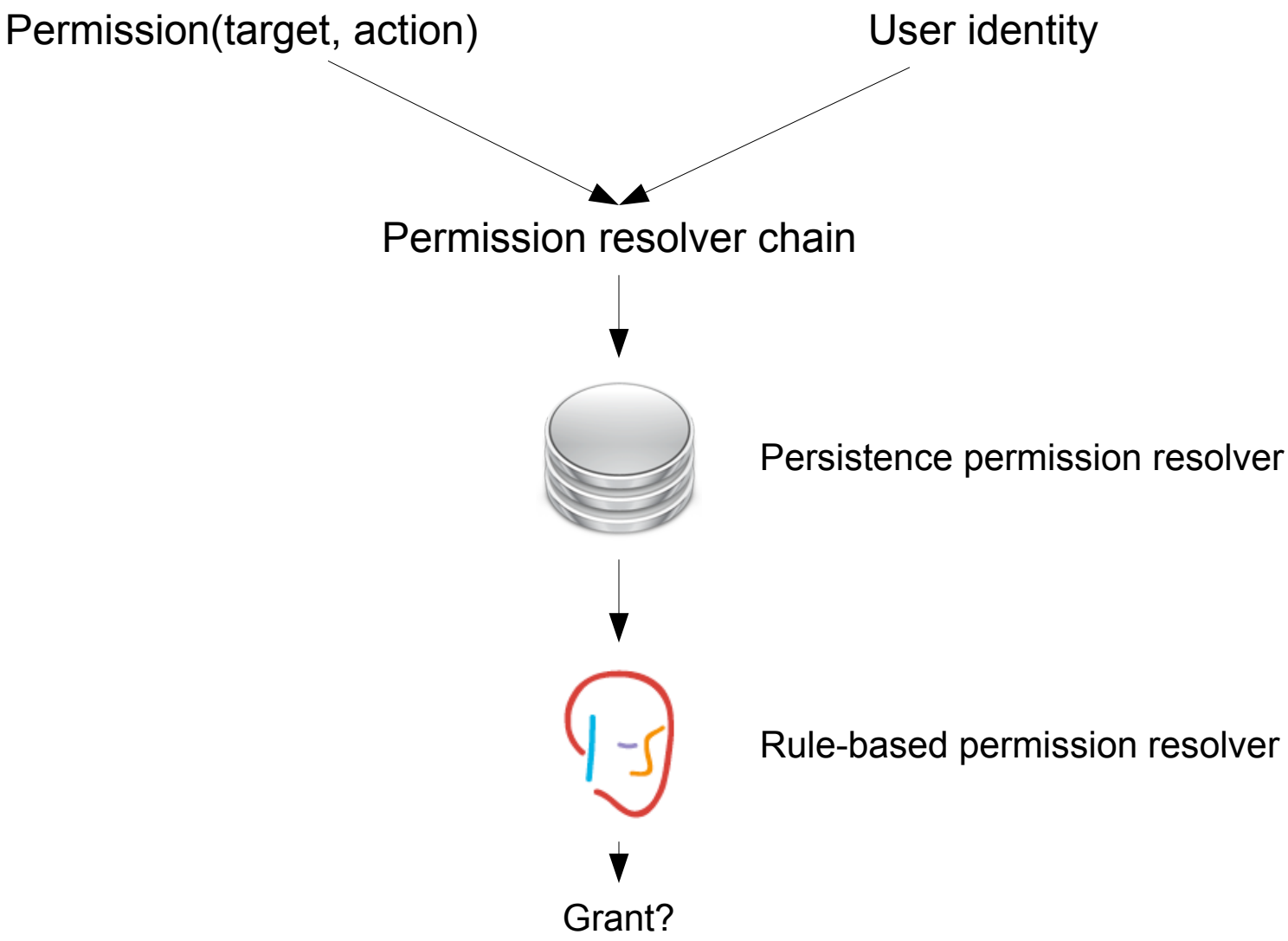
- **Classes and methods**

```
@Restrict("#{identity.loggedIn}")
```

```
@Restrict("#{identity.hasRole("admin")}")
```

- **If no criteria specified, a permission is implied**
  - **Target – object instance or view ID**
  - **Action – method name or JSF life cycle phase (restore or render)**

Resolving a permission



Rule-based security

- **Based on Drools**
- **Unique aspect of Seam security**
- **Rules are the raison d'être of security**
  - **You cannot enter the room with key**
  - **You cannot buy alcohol unless you are 21**
  - **You cannot fly if you have illegal weapons or 4 oz of shampoo**
  - **You cannot cash check unless it's endorsed**
- **Eliminates a lot of spaghetti business logic**
  - **Declarative and expressive**
  - **Hot swappable**

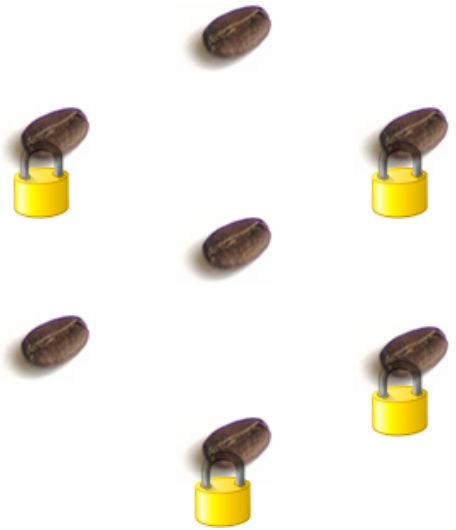
Checking your facts

- **The working memory contains facts**
  - **Facts are objects stored in a rules session**
- **You use facts to:**
  - **Make assertions**
  - **Perform operations when the rule is true**
- **Seam seeds working memory for security rules**
  - **PermissionCheck – the permission being requested**
  - **Principal – the security principal holding the username**
  - **Role – one or more roles assigned to the user**
  - **authenticated user account (when using identity management)**
  - **optional set of user-defined objects**

Demo: Rule-based security

Access control lists (ACLs)

- **Permission with a specific target**
- **Granted to a user or a role/group**
- **Can be managed by the application**
  - **Typically stored in a database**
- **Can be combined with rules**
  - **Conditional roles**



SELECT * FROM MEMBER_PERMISSION;

PERMISSIONID	ACTION	DISCRIMINATOR	RECIPIENT	TARGET
2	view	role	admin	Design:2
4	view	role	client	Design:1
5	view	role	admin	Design:1

Managing permissions in Seam

- **A persistent Permission object represents**
  - **A target**
  - **An action**
  - **A recipient (java.security.Principal)**
- **Choose provider and identify Permission class**

```
<security:jpa-permission-store
    user-class="com.company.app.model.UserPermission"/>
```

- **Seam provides built-in permissionManager component**
  - **List**
  - **Grant**
  - **Revoke**

Demo: Permission management

Run as: King for a thread

- **Elevated privileges for distinct operation**
  - **Self-registration on a web site**

```
new RunAsOperation() {
   public void execute() {
       identityManager.createUser(username, password);
   }
}.addRole("admin").run();
```

- **Could alternatively designate rule for this operation**

Summary

- **Seam security is easy to adopt**
- **Configuration is kept to a minimum**
- **Built-in security components**
  - **Configuration over custom code**
  - **Makes management of users, roles and permissions easy**
- **Many authorization options**
  - **Combination of rules and ACLs is powerful**
- **The security model matures with your application**

Questions?

Resources

- **Seam in Action, Manning 2008**
  - **http://manning.com/dallen**
  - **Chapter 11: Securing Seam Applications**
- **In Relation To... Blog**
  - **http://in.relation.to**
  - **Seam, Hibernate, Web Beans, JBoss Tools, RichFaces**
- **Seam community forums & wiki**
  - **http://seamframework.org**
- **Demo code**
  - **http://seaminaction.googlecode.com/svn/demos/presentations/tssjs09**