A 

APACHE FORTRESS

September 30, 2020

ApacheCon @Home

Main Objective

Discuss how the Groovy programming language can improve existing (mature) Java projects.

Secondary Objective

Discuss security authorization using Apache Fortress.

Presentation and Sample Projects

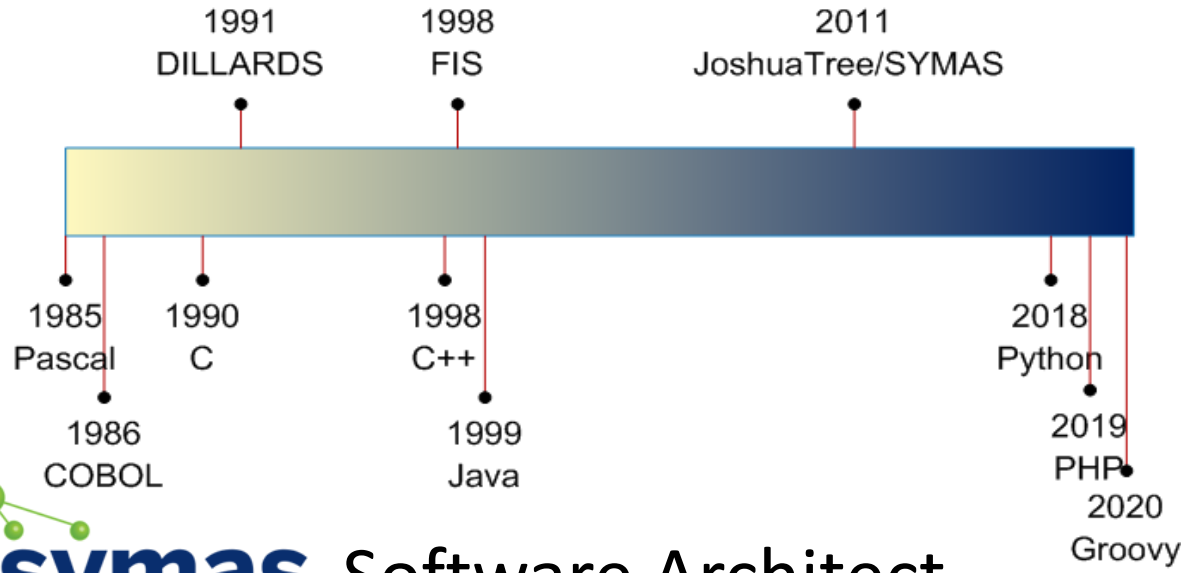
https://iamfortress.net/2020/09/30/a-groovy-apache-fortress/

Intro

Shawn McKinney

https://github.com/shawnmckinney

Code Monkey



- symas Software Architect
- Apache Directory PMC Chair
- OpenLDAP™ PROJECT Engineering Team

Session Agenda

- Project Goals
- Before Groovy
- For Example
- After Groovy
- Demo

symas

Project Goals

- Experiment with new API signatures
- Iterate quickly
- Improve user readability and comprehension
- Enhance programmer productivity
- Overcome biases (functional fixedness)

B.G. (Before Groovy)

Groovy 1.0 was released on January 2, 2007

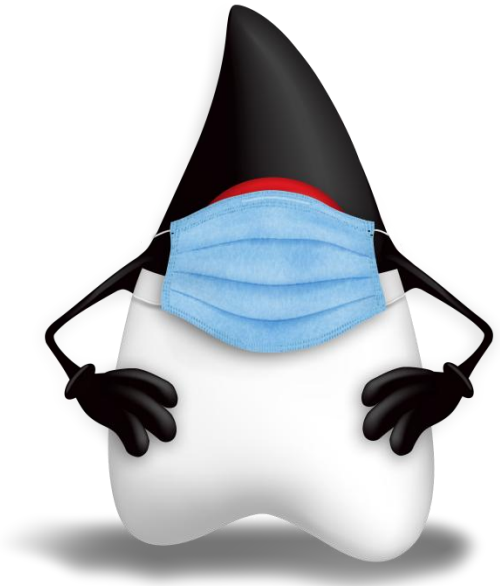
B.G.

https://wiki.openjdk.java.net/display/duke/Gallery

- Java had been in use for 12 years
- Service Oriented Architecture (SOA)
- J2EE was starting to show signs of age.
- Spring had emerged as dominant framework.

Java's Many Strengths

- Platform endurance
- Abundance of common libs
- Strong ecosystem
- Continues to evolve
- Adapts well to new tech
- Remains popular to this day



https://wiki.openjdk.java.net/display/duke/Gallery

But… there are drawbacks

- e.g. Boilerplate code

https://wiki.openjdk.java.net/display/duke/Gallery

Boilerplate

In computer programming, **boilerplate code** or just **boilerplate** are sections of code that have to be included in many places with little or no alteration. When using languages that are considered *verbose*, the programmer must write a lot of code to accomplish only minor functionality. Such code is called *boilerplate*.[1]

https://en.wikipedia.org/wiki/Boilerplate_code

Boilerplate Examples

- Exception blocks `try, catch`
- Getters, Setters `getX, setX`
- Package imports `import java.lang…`
- Constructors `Foo<Bar> x = new…`
- Punctuation `;, (, ), [, ], {,…`

Java Code Sample

```
List<RoleConstraint> constraints = new ArrayList();
RoleConstraint constraint = new RoleConstraint();
constraint.setKey( "locale" );
constraint.setValue( "East" );
constraints.add( constraint );

constraint = new RoleConstraint();
constraint.setKey( "strength" );
constraint.setValue( "2fa" );
constraints.add( constraint );

try
{
    AccessMgr accessMgr = AccessMgrFactory.createInstance( );
    Session session = accessMgr.createSession( new User( "Louie"), constraints, true );

    List<UserRole> userRoles = session.getRoles();
    assertFalse( userRoles.contains( new UserRole( user.getUserId(), TIds.WASHER ) ) );
    assertFalse( accessMgr.checkAccess( session, new Permission("MONEY", "dry" ) ) );
    assertFalse( accessMgr.checkAccess( session, new Permission("MONEY", "rinse" ) ) );
    assertFalse( accessMgr.checkAccess( session, new Permission("MONEY", "soak" ) ) );

    assertTrue( userRoles.contains( new UserRole( user.getUserId(), TIds.TELLER ) ) );
    assertTrue( accessMgr.checkAccess( session, new Permission("ACCT", "deposit" ) ) );
    assertTrue( accessMgr.checkAccess( session, new Permission("ACCT", "inquiry" ) ) );
    assertTrue( accessMgr.checkAccess( session, new Permission("ACCT", "withdrawal" ) ) );
}
catch ( SecurityException ex )
{
      ...
}
```

https://github.com/shawnmckinney/fortress-abac-demo/blob/master/src/main/java/com/mycompany/MyBasePage.java

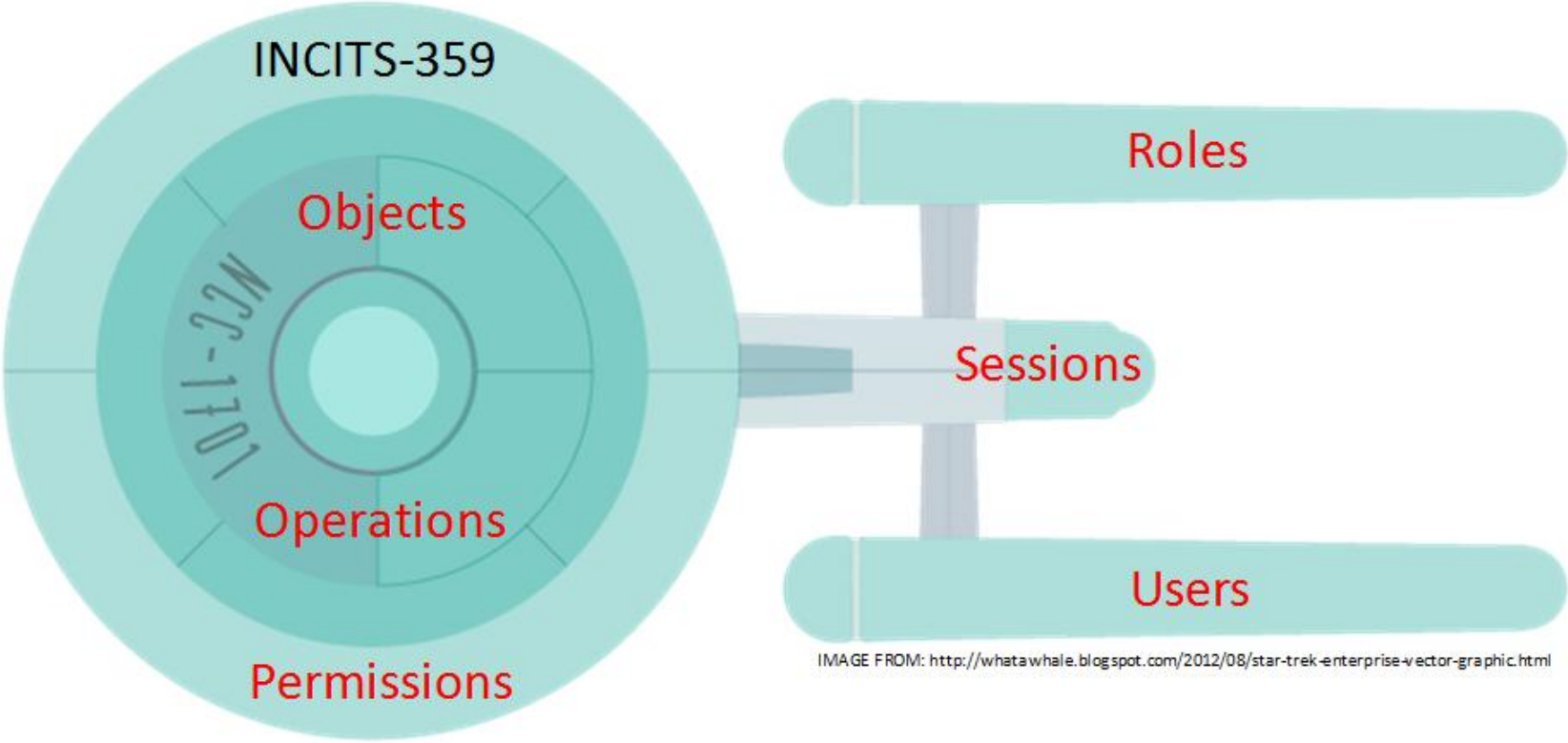
e.g.

Authorization

Authorization is the function of specifying access rights/privileges to resources, which is related to information security and computer security in general and to access control in particular. More formally, "to authorize" is to define an access policy.

https://en.wikipedia.org/wiki/Authorization

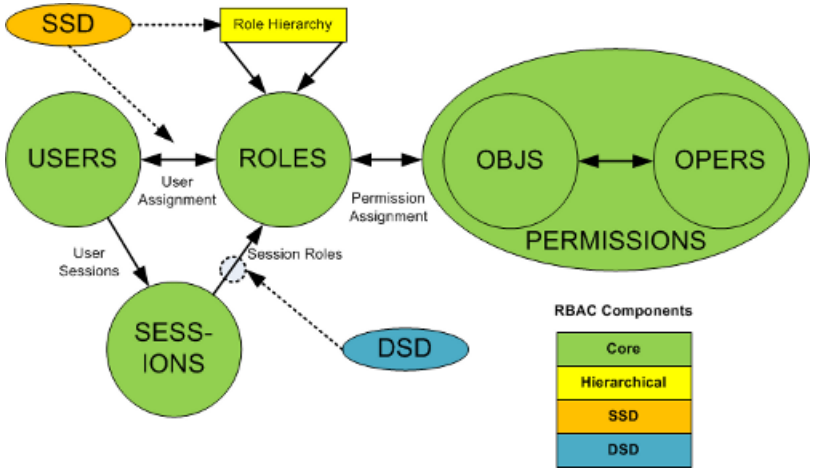
Role-Based Access Control (RBAC)



Role-Based Access Control (RBAC)

http://csrc.nist.gov/groups/SNS/rbac/

- RBAC0
  - Users, Roles, Perms, Sessions
- RBAC1
  - Hierarchical Roles
- RBAC2
  - Static Separation of Duties (SSD)
- RBAC3
  - Dynamic Separation of Duties (DSD)



ANSI INCITS 359

RBAC Functional Model

CreateSession(*user*, *session*)

This function creates a new session with a given user as owner and an active role set. The function is valid if and only if:

- the user is a member of the *USERS* data set, and
- the active role set is a subset of the roles assigned to that user. In a RBAC implementation, the session's active roles might actually be the groups that represent those roles.

The following schema formally describes the function. The *session* parameter, which represents the session identifier, is actually generated by the underlying system.

$$
\begin{aligned}
&CreateSession(user: NAME; ars: 2^{NAMES}; session: NAME) \triangleleft \\
&\quad user \in USERS; ars \subseteq \{r: ROLES \mid (user \mapsto r) \in UA\}; session \notin SESSIONS \\
&\quad SESSIONS' = SESSIONS \cup \{session\} \\
&\quad user\_sessions' = user\_sessions \setminus \{user \mapsto user\_sessions(user)\} \cup \\
&\qquad \{user \mapsto (user\_sessions(user) \cup \{session\})\} \\
&\quad session\_roles' = session\_roles \cup \{session \mapsto ars\} \triangleright
\end{aligned}
$$

Z-notation

ANSI RBAC Functional Model

Three standard interfaces:

1. Administrative – CRUD
2. Review – policy interrogation
3. System – policy enforcement

Functional Fixedness

Functional fixedness is a cognitive bias that limits a person to use an object only in the way it is traditionally used. The concept of functional fixedness originated in Gestalt psychology, a movement in psychology that emphasizes holistic processing. Karl Duncker defined functional fixedness as being a mental block against using an object in a new way that is required to solve a problem.[1] This "block" limits the ability of an individual to use components given to them to complete a task, as they cannot move past the original purpose of those components.

https://en.wikipedia.org/wiki/Functional_fixedness

Admin RBAC

[Link to AdminMgr javadoc]

Fortress Admin APIs map to the INCITS 359 specs

```
public interface AdminMgr {
  User addUser( User user );
  void deleteUser( User user );
  Role addRole( Role role );
  void deleteRole( Role role );
  void assignUser( UserRole uRole );
  void deassignUser( UserRole uRole );
  Permission addPermission( Permission perm );
  void deletePermission( Permission perm );
  void grantPermission( Permission perm, Role role );
  void addAscendant( Role childRole, Role parentRole);
  void addDescendant(Role parentRole, Role childRole);
  void addDsdRoleMember(SDSet dsdSet, Role role);
  void addInheritance(Role parentRole, Role childRole)
… http://git-wip-us.apache.org/repos/asf/directory-fortress-core.git
```

$$GrantPermission(object, operation, role{:}\ NAME) \triangleleft$$

$$(operation, object) \in PERMS; role \in ROLES$$

$$PA' = PA \cup \{(operation, object) \mapsto role\}$$

$$assigned\_permissions' = assigned\_permissions \setminus \{role \mapsto assigned\_permissions(roles)\} \cup \{role \mapsto (assigned\_permissions(role) \cup \{(operation, object)\})\} \triangleright$$

[Link to INCITS 359 spec]

Review RBAC

[Link to ReviewMgr javadoc]

Fortress Review APIs map to the INCITS 359 specs

```
public interface ReviewMgr {
    Permission readPermission( Permission permission );
    List<Permission> findPermissions( Permission permission );
    User readUser( User user );
    List<User> findUsers( OrgUnit ou );
    List<User> assignedUsers( Role role );
    Set<String> authorizedRoles( User user );
    List<Permission> rolePermissions( Role role );
    List<Permission> userPermissions( User user );
    Set<String> authorizedPermissionUsers(Permission perm);
    SDSet dsdRoleSet(SDSet set);
    Set<String> dsdRoleSetRoles(SDSet dsd);
    List<SDSet> dsdRoleSets(Role role);
    SDSet ssdRoleSet(SDSet set);
    Set<String> ssdRoleSetRoles(SDSet dsd);
    List<SDSet> ssdRoleSets(Role role);
    List<Role> findRoles(String searchVal);
…
```

http://git-wip-us.apache.org/repos/asf/directory-fortress-core.git

$$UserPermissions(user: NAME; result: 2^{PERMS}) \triangleleft$$

$$user \in USERS$$

$$result = \{r: ROLES; op: OPS; obj: OBJS \mid (user \mapsto r) \in UA \wedge ((op, obj) \mapsto r) \in PA \bullet (op, obj)\} \triangleright$$

[Link to INCITS 359 spec]

System RBAC

[Link to AccessMgr javadoc]

Fortress AccessMgr APIs map to the INCITS 359 specs

```
public interface AccessMgr {
  Session createSession( User user, boolean isTrusted );
  List<Permission> sessionPermissions( Session session );
  Set<String> authorizedRoles( Session session );
  void addActiveRole( Session session, UserRole role );
  void dropActiveRole( Session session, UserRole role );
  User getUser( Session session );
  boolean checkAccess( Session session, Permission perm);
}
```

http://git-wip-us.apache.org/repos/asf/directory-fortress-core.git

$$CheckAccess(session, operation, object: NAME; out\ result: BOOLEAN) \triangleleft$$

$$session \in SESSIONS; operation \in OPS; object \in OBJS$$

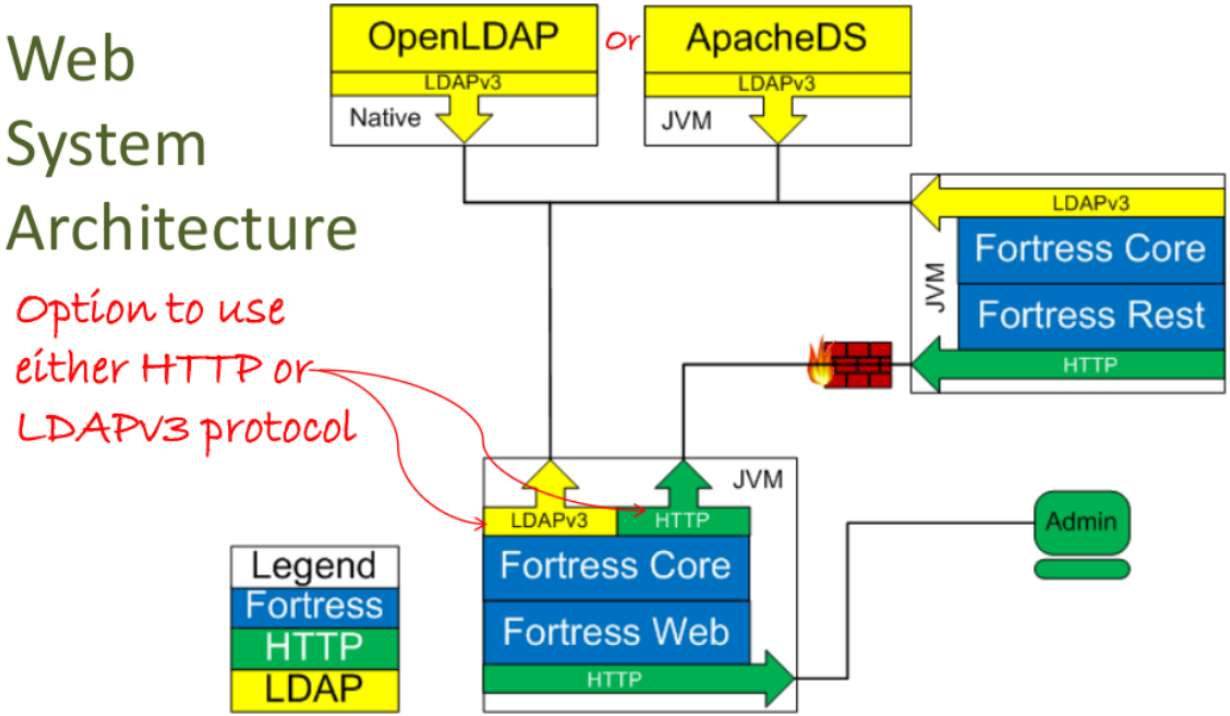
[Link to INCITS 359 spec]

$$result = (\exists r: ROLES \bullet r \in session\_roles(session) \wedge ((operation, object) \mapsto r) \in PA) \triangleright$$

Apache Fortress™

Access Management SDK and Web Components

A standards-based access management system, written in Java, supports ANSI INCITS 359 RBAC and more.



Overview

- Sub-project of Apache Directory
- Written in Java
- Four Components:
  - Core – Java APIs + utilities
  - Realm – Java EE policy enforcement
  - Web – Administrative UI
  - Rest – APIs over HTTP interface

Extensive API Set

- Role-Based Access Control
- Delegated Administration
- Password Policies
- Audit Log Interrogation
- Configuration and Properties
- Nearly 200 public APIs!

Larger API Sets -> Higher Complexity

- Extraordinarily difficult to use, even if experienced in the problem space.
- Add in a complex model and it gets even worse.
- Has to be a better way.

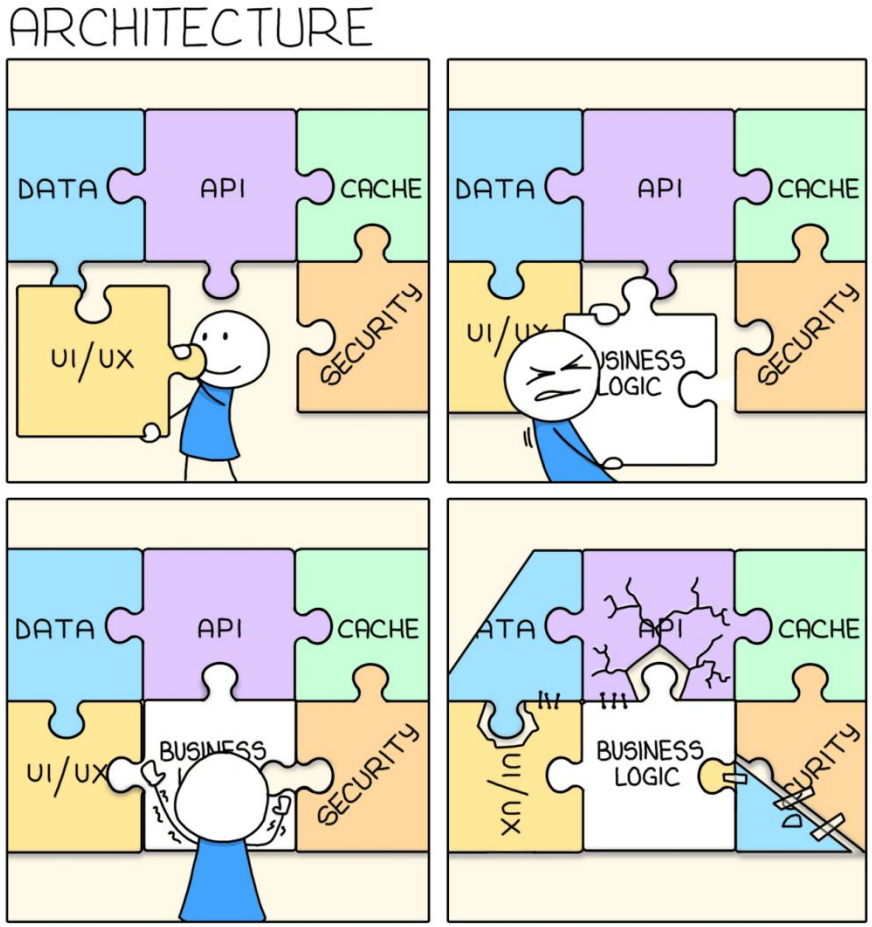
Can Groovy help…

- simplify things for the users?
- make it easier to maintain a project?

More Questions…

- Redesigned APIs more usable?
- Make it easier for users to integrate?
- Easier to write test cases?
- Easier to add new features?

What now?



https://www.monkeyuser.com/2018/architecture/

A.G. (After Groovy)

Groovy 1.0 was released on January 2, 2007

After Groovy

Not only is this code more concise, it is also much more readable. The signal-to-noise ratio is much better.

-- Andrew Montalenti

https://amontalenti.com/2011/04/09/groovy-the-python-of-java

Fewer APIs → Easier to Learn

- Admin

```
boolean edit (Map<String,?> options, String op, String
    entity )
```

- Review

```
Map<String, ?> search (Map<String,?> options, String op,
    String entity )
```

- System

```
boolean start( Map<String,?> options=[:] )
boolean canDo ( String obj, String op, String objId =
    null )
```

Typical Object Oriented Programming

```
RoleConstraint strength =
   new RoleConstraint();
strength.setKey( "strength" );
strength.setValue( "2FA" );
strength.setId( “Teller” );
adminMgr.addRoleConstraint(
   new UserRole(“dewey”, “Teller”),
   strength );
```

Java

Dictionary Usage Simplifies

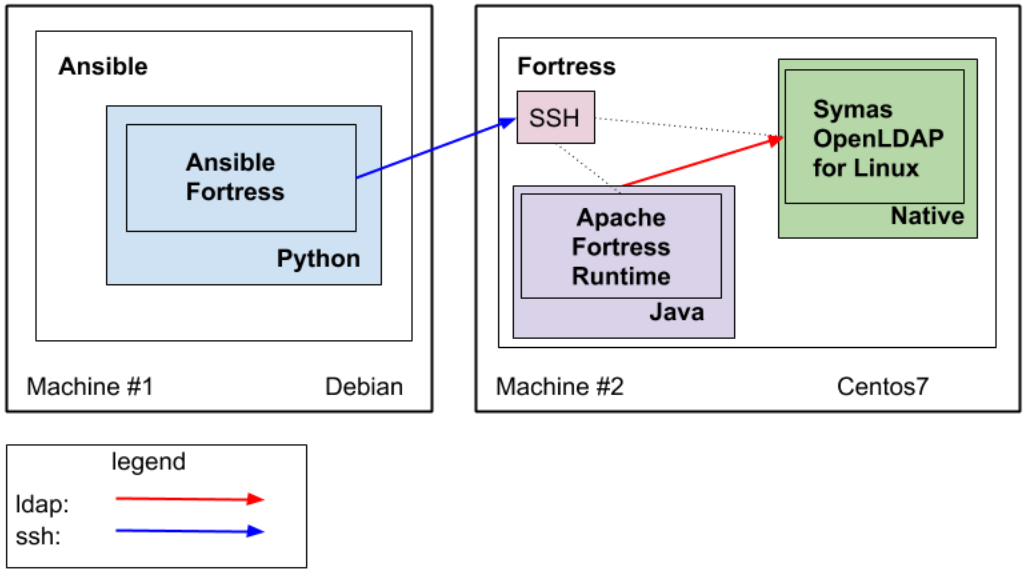
```
edit( 'add', 'roleconstraint',
        userId: 'dewey',
        id: TIds.TELLER,
        key: 'locale',
        value: 'North'
    )
```

Groovy

Demo


his is the first time you've seen this Stop error screen,
art your computer. If this screen appears again, follow
e steps:
to make sure any new hardware or software is properly installed.
his is a new installation, ask your hardware or software manufacti
any Windows updates you might need.
roblems continue, disable or remove any newly
lled hardware
oftware. Disable BIOS memory options such as
g or shadowing.
ou need to use Safe Mode to remov
disable
nents, restart
computer, press F8 to select Ad
Startup
ons, and then
ct Safe Mode.
nical information:
STOP: 0x00000050 (0xFD3094C2,0x00000
000)
SPCMDCON.SYS - Address FBFE7617 base at F
eStamp 3d6d

Demo System Architecture



Apache Fortress Samples

- https://gitlab.symas.net/symas-public/ansible-apache-fortress
- https://github.com/shawnmckinney/groovy-apache-fortress

https://github.com/shawnmckinney/groovy-apache-fortress

Java	Groovy
• AdminMgrTest: 182 lines • AccessMgrTest: 165 lines	• AdminMgrTest: 95 lines • AccessMgrTest: 92 lines

Less code to write -> higher productivity

Closing Thoughts

Groovy was written "to target" Java programmers — it is the dynamic language Sun should have built for the JVM as dynamic languages gained popularity and the Java language stagnated.

-- Andrew Montalenti

https://amontalenti.com/2011/04/09/groovy-the-python-of-java

https://directory.apache.org/fortress

APACHE FORTRESS

Contact Info