Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

Pragmatic Identity 2.0: Simple, Open, Identity Services Using REST

Pat Patterson
Ron Ten-Hove
Sun Microsystems, Inc
Identity Services R&D

Sun microsystems

Good Programmers Need to be Lazy!

- Proactive Laziness
  - Automating repetitive work
  - Simplifying complex things
  - Avoiding unnecessary or difficult tasks
- Today we will show how to be very lazy when dealing with identity in your applications

About Pat Patterson

- Principal Engineer in Identity Management product group
- Has worked on security and IdM since 1997
  - Trustbase Transaction Manager, Access Manager
  - Federation standards technical architect
  - Community leader first for OpenSSO, now all of Sun's open-source IdM projects
- http://blogs.sun.com/superpat/

About Ron Ten-Hove

- Senior Staff Engineer in IdM product group
- Many years in SOA + integration product engineering
  - Many security / identity concerns from a SOA perspective, especially cross-domain
  - JSR 208 Spec Lead
- Murky past in industrial automation, oil & gas
- http://blogs.sun.com/rtenhove/
- Very lazy indeed :-)

Identity in Applications
Typical PHP Login Page

```
// If the user isn't logged in, try to log him in

if (!isset($_SESSION['user_id']) && isset($_POST['submit'])) {
  $dbc = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

  $username = mysqli_real_escape_string($dbc, trim($_POST['name']));
  $password = mysqli_real_escape_string($dbc, trim($_POST['pswd']));

  if (!empty($username) && !empty($password)) {
    $query = "SELECT user_id, username FROM users WHERE ".
        "username = '$username' AND password = SHA('$password')";
    $data = mysqli_query($dbc, $query);

    if (mysqli_num_rows($data) == 1) {
      // The login is OK so set the user ID session var and
      // redirect to the home page

      $row = mysqli_fetch_array($data);
      $_SESSION['user_id'] = $row['user_id'];
      // ...
```

Identity in Applications

Typical PHP Login Page

```
// If the user isn't logged in, try to log him in

if (!isset($_SESSION['user_id'])) {
  $dbc = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

  $username = mysqli_real_escape_string($dbc, trim($_POST['name']));
  $password = mysqli_real_escape_string($dbc, trim($_POST['pswd']));

  if (!empty($username) && !empty($password)) {
    $query = "SELECT user_id, username FROM users WHERE
      username = '$username' AND password = SHA('$password')";
    $data = mysqli_query($dbc, $query);

    if (mysqli_num_rows($data) == 1) {
      // The login is OK so set the user ID session var and
      // redirect to the home page

      $row = mysqli_fetch_array($data);
      $_SESSION['user_id'] = $row['user_id'];
      // ...
```

Not Lazy!

What Was Wrong With That?

- Affects every web app page
  - Lots of work; error prone
- Fine-grained access control hard coded
  - Based on user attributes
    - Roles, group membership, location, relationships...
  - Must determine what CRUD operations are allowed
  - Hard to change (one page at a time)
    - Mixes app & access control logic => more complex
    - Needs more testing => more work

What Was Wrong With That?
Continued

- Uses a hardcoded identity store (the db table)
  - Type, schema, access information baked in
  - Hard to change environments (dev, test, prod)
  - Changes to store or schema => lots of work
- Scalability risks
  - Changing ID store for scaling => painful / risky
- Uses a single identity store
  - Not realistic in most enterprises
  - Multiple ID store types used => complexity

Other Concerns

- Centralized policy administration
  - Consistency
  - Easy to change, reason about
- Conforming to legal mandates
  - Logging / auditing, changes, variability...
- Dealing with heterogenous environments
  - Acquisitions & mergers problem
- Software development life cycle
  - Changes to apps are expensive, slow

The Lazy Programmer's Dream

- I should *never* have to write my own identity management code again

The Lazy Programmer's Dream

- I should *never* have to write my own identity management code again
  - No matter what programming language I use

The Lazy Programmer's Dream

- I should *never* have to write my own identity management code again
  - No matter what programming language I use
  - No matter what back-ends I have existing identity data in

The Lazy Programmer's Dream

- I should *never* have to write my own identity management code again
  - No matter what programming language I use
  - No matter what back-ends I have existing identity data in
  - And I should not have to learn some ugly API instead

How To Be That Lazy?

Sounds like a lot of work...

- Identity Services
  - Delegate the work to someone else
- Architecture
  - Separation of concerns
  - Build identity and policy enforcement into the system, not apps
    - Solve once for all apps => very lazy!
- Combine the two above points
  - Completely hide the delegation from your apps

Three Flavors of IdM Services

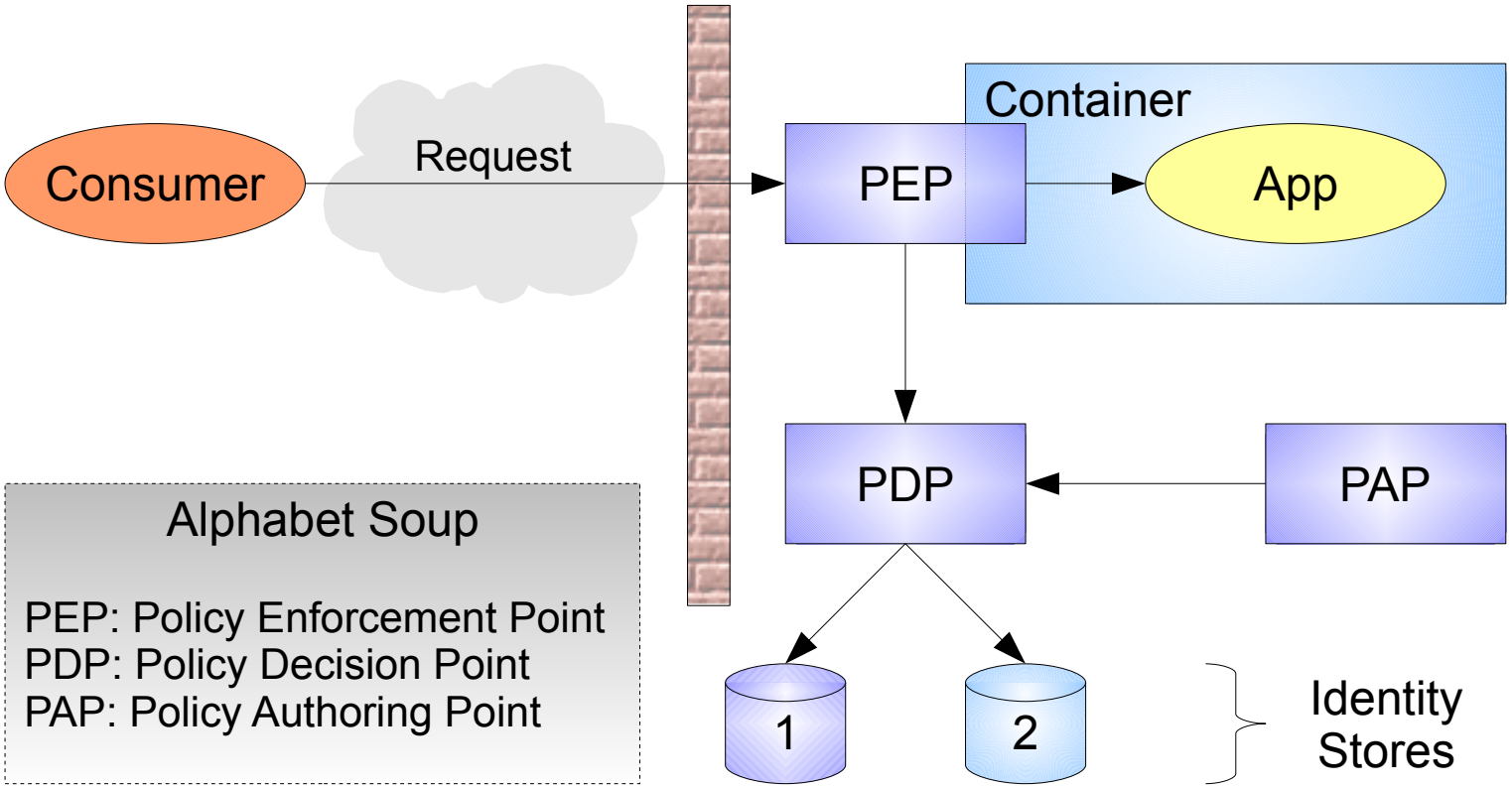
- Plug-ins and Connectors
  - For standard containers and ID stores
- Identity Standards
  - SAML, WS-*, DSML, SPML, ID-FF, …
  - Service-oriented architecture emphasized
- Resource-oriented architecture (REST)
  - Lightweight, simple to use
  - Programming language neutral
  - Single API for IdM

Plug-ins and Connectors

- Policy agents plug into containers
  - Intercept messages
  - Authenticate
  - Apply policies to authorize access
  - See project OpenSSO
- Connectors
  - Abstract different identity stores
  - Breaks dependencies on particular stores
  - See Identity Connectors framework

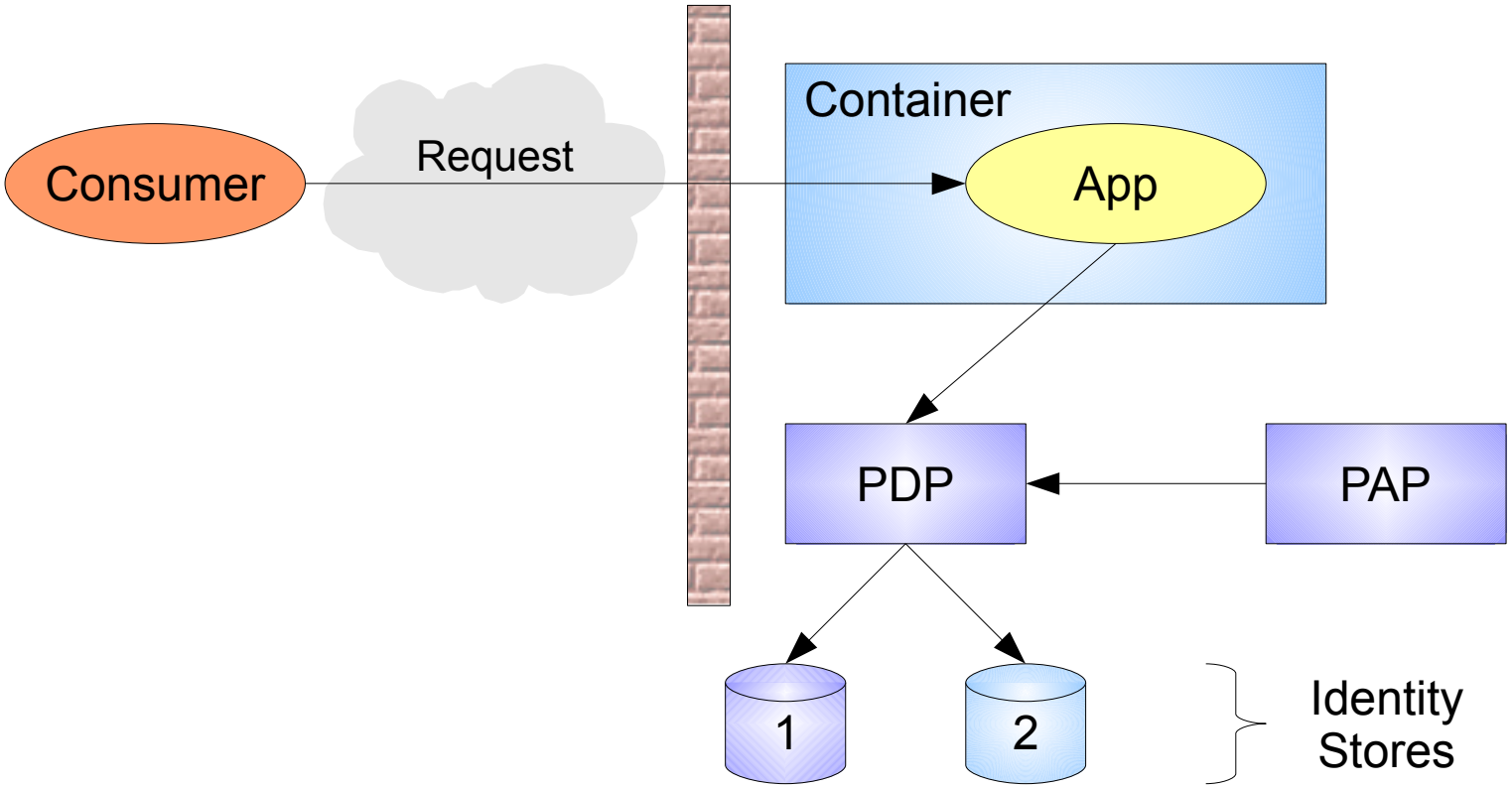
System With Plug-in Agent (PEP)

Zero app coding: very lazy



System Without Plug-in Agent

App acts as PEP: not quite as lazy



Accessing the PDP

- To be properly lazy
  - No API
  - No language-specific bindings
  - No complicated concepts to learn
- We need
  - REST
    - Interactions using standard HTTP interface only => easy to learn, easy to implement anywhere
  - Resource-oriented architecture
    - Intuitive resource hierarchy => easy to learn

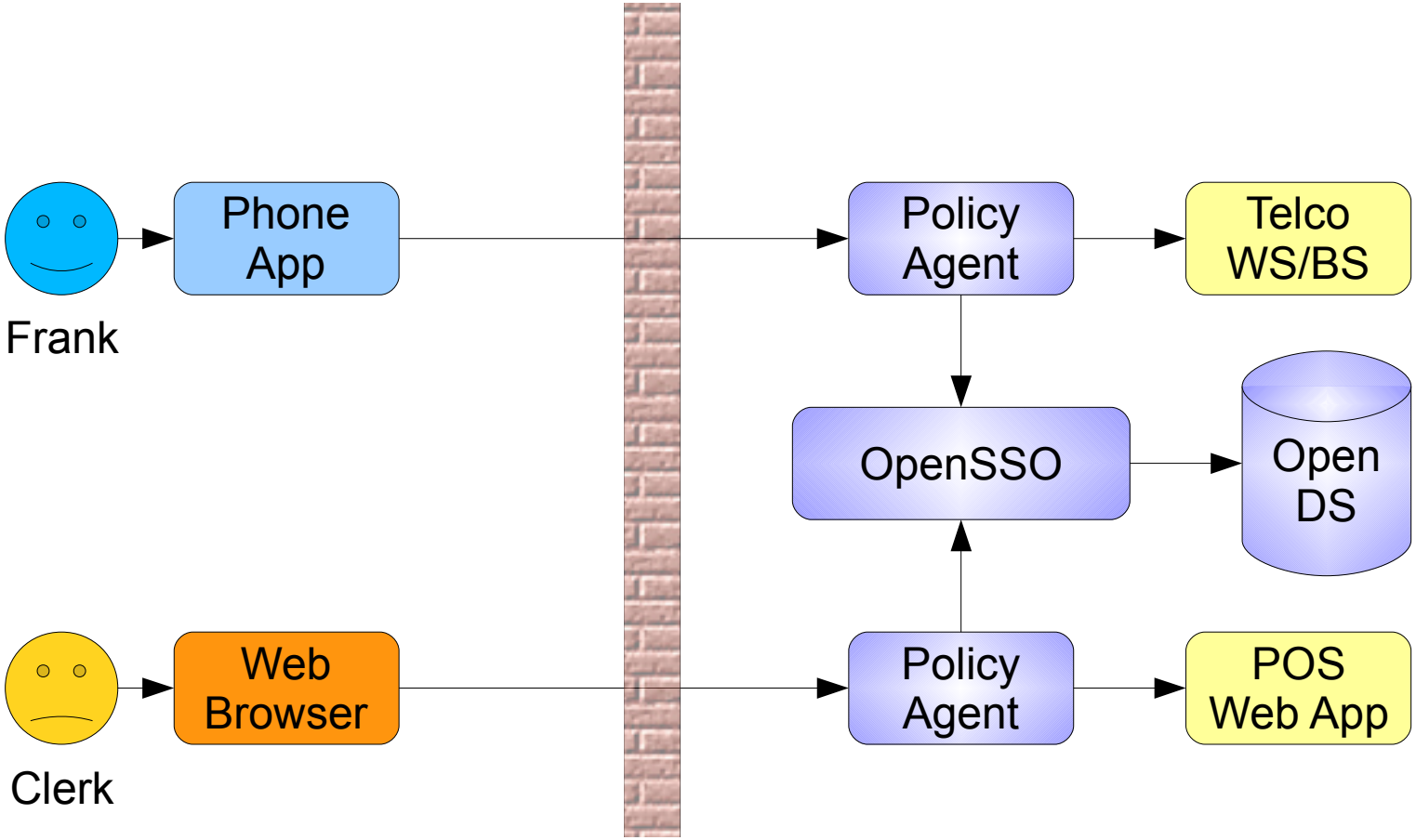
REST PDP Service Using HTTP

- GET /identity/authenticate?
  username=alice&password=foo
  - HTTP Status Code 200 OK + SSO token
  - 401 Unauthorized
- GET /identity/authorize?
  uri=http://example.com/foo/bar&action=GET&
  subjectid=<*SSOToken*>
  - 200 OK => user is authorized
  - 403 Forbidden

RESTful Account Provisioning

- Create: POST /targets/account
  201 Created +
  Location: /targets/account/alice
- Read: GET /targets/account/alice
  200 OK + account representation in body
- Update: PUT /targets/account/alice
  200 OK + account representation in body
- Delete: DELETE /targets/account/alice
  204 No Content
- Intuitive interface; resource based

Demo Overview



Demo

Summary

- Identity Services are
  - The lazy way to avoid writing code :-)
  - Separate policy decisions and enforcement from apps
  - Provide consistency across apps
  - Provide a simple interface
  - Provide those difficult extras:
    - Logging, auditing
    - Scalability
    - Isolation from identity store details & their changes

Q&A

References

- Project OpenSSO
  https://opensso.dev.java.net/
- Identity Connectors
  https://identityconnectors.dev.java.net/
- OpenDS
  https://opends.dev.java.net/

Thank You

Pat Patterson
superpat@sun.com

Ron Ten-Hove
rat@sun.com

https://opensso.dev.java.net