A smart conference for a smarter planet

Web 2.0 and RESTful Applications with WebSphere sMash and PHP

Roland Barcia - STSM, WebSphere Lab Services, Lead Web 2.0 Architect
Agenda

• WebSphere sMash
• WebSphere sMash Programming Model
• PHP in WebSphere sMash
• PHP Applications
• Demonstrations
Agenda

- **WebSphere sMash**
- WebSphere sMash Programming Model
- PHP in WebSphere sMash
- PHP Applications
- Demonstrations
What is WebSphere sMash?

• WebSphere sMash is a new Agile Web Application Platform
  – Leveraging Dynamic Scripting Languages
  – Optimized for Producing....
    • REST-based Services
    • Integration Applications
    • Mash-ups
    • Rich Web interfaces
sMash Core Values

**Speed**
- Dynamic Scripting Support
- Rich User Interfaces Support

**Agility**
- Integrated runtime
- Pre-built services

**Simplicity**
- End to end web-based tooling
- Visual & programmatic editors
Dynamic Scripting

- WebSphere sMash is a dynamic scripting platform
- Application logic is created in a scripting language
  - Groovy (for people that prefer Java)
  - PHP (for the millions of existing PHP programmers)
- Java is positioned as the “system” language
Application Centric Runtime

• WebSphere sMash is an application-centric runtime
  – You create an application and run it
  – Each application runs in its own process (JVM)
  – Runtime is designed to be short lived

• WebSphere sMash is a full runtime stack
  – Everything needed to run the application is provided
    • including the HTTP stack
  – No external proxy or web server is required
Simple Deployment

• The deployment is essentially ZIP and Copy
• No machine specific information bound into the application
• Default mode is shared dependencies
  – Application dependencies are stored locally and pulled from the network as needed
• Standalone mode is supported as well
  – All application dependencies are included in the ZIP except the JVM
Scalability

- Programming model is single threaded
- Application instance holds many independent threads
- Greater scalability achieved via multiple instances with a sprayer
- Tooling is provided for Apache mod_proxy
- WebSphere Virtual Enterprise can be used for larger deployments
Browser-based Tooling

Dynamic Scripting Editor

Visual UI Editor

Visual Assembly Editor

Agility in Dynamic Scripting – Groovy / PHP

Simplicity in constructing web pages (Javascript)

Speed in scripting / assemble activities into a flow
## Versions of WebSphere sMash

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WebSphere sMash DE</strong></td>
<td>Free download! WebSphere sMash + development tooling. Restricted production use.</td>
</tr>
<tr>
<td><strong>WebSphere sMash</strong></td>
<td>The runtime environment. Available under a standard IBM commercial license.</td>
</tr>
<tr>
<td><strong>WebSphere sMash RTE</strong></td>
<td>Allows sMash apps to communicate with one another using asynchronous messages.</td>
</tr>
<tr>
<td><strong>Project Zero</strong></td>
<td>Free download! Bleeding edge version.</td>
</tr>
</tbody>
</table>

[http://projectzero.org](http://projectzero.org)
Version History

2008
- 1.0
- 1.0.0.1
- 1.0.0.2
- 1.0.0.3
- August
- July
- June

2009
- 1.1
- 1.1.0.1
- December
- October

Timeline:
- April 2009
- June 2008
- July 2008
- August 2008
- October 2008
- December 2008
- April 2009

Agenda

• WebSphere sMash
• WebSphere sMash Programming Model
• PHP in WebSphere sMash
• PHP Applications
• Demonstrations
Events

- All behavior in the system is modeled as a set of events
  - Applications are built by handling these events and providing desired behavior
  - Similar to AJAX model or classic UI programming
Event Handlers

- All handlers are stateless
- Can be implemented in Groovy, PHP, and Java

**PHP**
```php
<?php
class Employees {
    function onGET() {
        echo "Response from a GET request";
    }
    function onPUT() {
        echo "Response from a PUT request";
    }
    function onPOST() {
        echo "Response from a POST request";
    }
    function onDELETE() {
        echo "Response from a DELETE request";
    }
}
?>
```

**Groovy**
```groovy
println "Response from a \$request.method[]"
def onGET() {
    println "Response from a GET request"
}
def onPUT() {
    println "Response from a PUT request"
}
def onPOST() {
    println "Response from a POST request"
}
def onDELETE() {
    println "Response from a DELETE request"
}
```
Global Context – State Management

• The Global Context (GC) provides access to and management of all application state
  – Conceptually a map of data

• Externalizes all state from the application logic
  – Enables the restartability of the JVM without data loss
  – Enables clustering and scaling to be added transparently

• Simplifies and unifies access to application state and data structures and simplifies state passing within the application

• Contains information provided by both the runtime (such as request parameters) and by the application
# Global Context Zones

<table>
<thead>
<tr>
<th>Zone</th>
<th>Scope</th>
<th>Automatic Recycle</th>
<th>User Initiated Restart</th>
<th>User modified data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Request / Thread</td>
<td>State discarded</td>
<td>State discarded</td>
<td>discarded</td>
</tr>
<tr>
<td>Event</td>
<td>Event / Thread</td>
<td>State discarded</td>
<td>State discarded</td>
<td>discarded</td>
</tr>
<tr>
<td>Tmp</td>
<td>Application</td>
<td>State discarded</td>
<td>State discarded</td>
<td>discarded</td>
</tr>
<tr>
<td>Config</td>
<td>Application</td>
<td>State reloaded from config files</td>
<td>State reloaded from config files</td>
<td>discarded</td>
</tr>
<tr>
<td>Connection</td>
<td>Event / Thread</td>
<td>State discarded</td>
<td>State discarded</td>
<td>discarded</td>
</tr>
<tr>
<td>User</td>
<td>Session denoted by zsessionid</td>
<td>State preserved</td>
<td>State discarded</td>
<td>preserved</td>
</tr>
<tr>
<td>App</td>
<td>Application</td>
<td>State preserved</td>
<td>State discarded</td>
<td>preserved</td>
</tr>
<tr>
<td>Storage</td>
<td>Application</td>
<td>State preserved</td>
<td>State preserved</td>
<td>preserved</td>
</tr>
</tbody>
</table>
Accessing the Global Context

- Data is organized by a URI structure
  - First part of URI is always the Zone name
    - /app, /user, /request, /config, /event, /tmp, etc...
- Access is modeled after REST
  - GET, PUT, POST, DELETE
  - zget(), zput(), zpost(), zdelete(), zlist(), zdump(), zcontains()
  - Groovy Short Cuts

Java

```java
String path = GlobalContext.zget("/request/path");
GlobalContext.zput("/tmp/cachedData", someVar);
ArrayList config = GlobalContext.zlist("/config");
```

Groovy Shortcuts

def path = request.path[];
def user.count[] = i
```

PHP

```php
<?php
// Get the URI from the request.
$uri = zget("/request/uri");

echo "<br/>$uri";

// Get the user agent
$header = zget("/request/headers/in/User-Agent");
echo "<br/>$header";

// List all the parameters.
$params = zlist("/request/params",false);

// Put the status.
zput("/request/status",200);

// Delete a key.
zdelete("/request/some_unwanted_key");

// Add a header value 'no-cache'
zpost("/request/headers/out/Cache-Control","no-cache");

// secured pages only.
$remoteU = zget("/request/subject#remoteUser");
$group = zget("/request/subject#groups");
$roles = zget("/request/subject#roles");
?>
```
Value Pathing

- The GC provides simplified access to certain data structures
  - Called **Value Pathing**
- Understands
  - Maps, List, First Element List, Objects
  - JSON (Implicitly through Maps, Lists, Objects)
- Allows read and write access to internals of the structure through the GC address

**Maps**
*(PHP Examples)*

```php
// Create or Replace existing Map
zput("/app/myMap", $arr)

// Add or replace an existing item in a Map
zput("/app/myMap#foo", "bar")

// Create or Merge into existing Map.
zpost("/app/myMap", $arr)

// Returns the map
$arr = zget("/app/myMap")

// Returns entry
$key = zget("/app/myMap#foo")

// Deletes Map
zdelete("/app/myMap")

// Removes entry
zdelete("/app/myMap#foo")
```

**Lists**
*(PHP Examples)*

```php
// Set or Replace list element
zput("/app/myList#0", "bar")
```

*(Groovy Example)*

```groovy
app.myMap['foo'] = 'bar'
```

*(Groovy Example)*

```
app.myList[0] = "bar"
```
# Application Directory Layout

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app</td>
<td>The scripts and templates for key components</td>
</tr>
<tr>
<td>classes</td>
<td>The java class files that are part of an application.</td>
</tr>
<tr>
<td>config</td>
<td>The configuration files of your application</td>
</tr>
<tr>
<td>java</td>
<td>The Java source files.</td>
</tr>
<tr>
<td>lib</td>
<td>Additional jar files that are required by your application.</td>
</tr>
<tr>
<td>logs</td>
<td>The log and trace files produced by your running application.</td>
</tr>
<tr>
<td>public</td>
<td>The Web accessible root folder of the application. Can contain html files, images, dynamic server scripts like .php and .groovy files, JavaScript, etc…</td>
</tr>
<tr>
<td>reports</td>
<td>The IVY dependency report shows details of the dependencies that you have on extension modules.</td>
</tr>
<tr>
<td>.zero</td>
<td>This directory is created by the WebSphere sMash runtime on behalf of the application to hold any generated files. Developers do not need to maintain or edit files in this directory.</td>
</tr>
</tbody>
</table>
### app and config

#### Directory Structure:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app/services</td>
<td>Custom error pages that handle specific errors</td>
</tr>
<tr>
<td>app/resources</td>
<td>Set of RESTful resources for an application.</td>
</tr>
<tr>
<td>app/scripts</td>
<td>Shared Scripts within an application, not directly accessible via URI.</td>
</tr>
<tr>
<td>app/views</td>
<td>Script implementation of views. Represents rendering logic.</td>
</tr>
<tr>
<td>app/models</td>
<td>JSON based Resources Models leveraging ZRM.</td>
</tr>
</tbody>
</table>

#### config Directory:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config/ivy.xml</td>
<td>Configuration for dependency management of your application.</td>
</tr>
<tr>
<td>config/zero.config</td>
<td>Runtime configuration file for your application. Populates the config zone of GC.</td>
</tr>
</tbody>
</table>
Dependencies and Ivy

- Apache Ivy is a tool for managing project dependencies.
- WebSphere sMash Leverage ivy technology for Dependencies.
- sMash applications have:
  - Local Repository
  - Remote Repository
    - Default (projectzero.org)
  - Dependency commands use to load modules into your app from local and remote repositories.
  - Strong Version Support

```xml
<ivy-module version="1.3">
  <info packagingType="unknown" module="ConsumerIncentiveApp" organisation="zero" revision="1.0.0">
    <license name="type of license" url="http://license.page"/>
    <ivyauthor name="author name" url="http://authors.home.page"/>
    <description homepage="http://module.description.page"/>
  </info>
  <publications>
    <artifact name="ConsumerIncentiveApp" type="zip" org="zero"/>
  </publications>
  <dependencies>
    <dependency org="zero" name="zero.core" rev="[1.0.0.0, 2.0.0.0["
    <dependency org="org.apache.derby" name="derby" rev="10+"/>
    <dependency org="zero" name="zero.data" rev="[1.0.0.0, 2.0.0.0["/>
  </dependencies>
</ivy-module>
```
Virtualized Directory

• WebSphere sMash provides seamless integration of directories across an application and its dependencies, while maintaining each as separate entities.

• All artifacts are searched within both the application and its declared dependencies
Configuration – zero.config

- **zero.config**
  - processed at the start of a Zero application
  - organized into "stanzas" of related key/value pairs.
  - Stanzas are associated with directives, such as
    - "store to the Global Context"
    - "include another configuration file."

```plaintext
# Value set
/config/http/port = 8080

# List set
/config/resources/defaultExtensions = [".groovy"]

# List append
/config/bindings/.groovy += ["zero.core.groovysupport.bindings.DefaultBindings"]

# Map set
/config/test/map = { "a" : "b", "c" : "d" }

# Map append
/config/test/mapappend += { "a" : "b", "c" : "d" }
/config/test/mapappend += { "x" : "y", "w" : "z" }

# Event handler
/config/handlers += [{
    "events" : "GET",
    "handler" : "custom.Handler.class" }]

# Value reference (insert value read at config-load time)
/config/property/myPrefix = "/foo/bar"
/config/test/value = "${/config/property/myPrefix}/bat"

# Variable set/value reference
myPrefix = "/foo/bar"
/config/test/value = "${myPrefix}/bat"

# Include
@include "${/config/dependencies/zero.core}/config/security/form.config" {
    "formLoginPage" : "/login"
}
```
Web 2.0 Applications?

* A term coined by Tim O'Reilly

Simple to use

AJAX
• Highly Interactive
• Browser invoked services

JSON / XML / ATOM
• Information exchange
• JavaScript Friendly

Simple to access

REST
• Easily invoked
• HTTP-Centric Patterns
What is REST?

- Representational State Transfer
- Roy Fielding described this via his dissertation
- It’s the design rationale behind the web
- Architectural style for describing distributed hypermedia systems
  - Client/Server
  - Stateless
  - Cacheable
  - Uniform Interface
  - Layered Interface
  - Code-On-demand
- Architectural elements include
  - Data Elements (resources, identifiers, data representations, representational data, control data)
  - Connectors (client, server, cache, resolvers)
- Architecturally - it doesn’t really *have* to apply to HTTP
A RESTful Web service is formed like a sentence – it simplifies how developers access services

- **Verb** = HTTP Action (GET, POST, PUT, DELETE)
- **Noun** = the URI of the Service (the document)
- **Adjective** = MIME type of the resulting document
REST Misconceptions

• REST is just any XML over HTTP not using SOAP?
  – NO !!! REST is a Pattern of exchanging Resources.
  – RPC is Not REST

• REST is only useful for CRUD (Create, Read, Update, and Delete) semantics.
  – NO!!! Resources can be anything, from a Business Process to an Image.
  – No new Verbs, just new resources.

• REST replaces traditional Web Services
  – REST is about delivering resources through the HTTP Web channel.
  – Does not address Distributed transactions, other protocols like messaging.
REST Web Services

• Identifiable resources (URIs)

• Uniform Interface
  – GET, PUT, POST, DELETE

• Stateless Communication
  – Scalable, loose coupling

• Resource Representations
  – Multiple ways to represent (PDF, HTML, XML,) - via content types
  – HTTP has negotiation capabilities (e.g. ACCEPT)

• Hypermedia
  – Server provides links to resources
  – Allows for evolution
### Verbs (Actions)

- HTTP offers a uniform interface
  - i.e. constrains the user interface

<table>
<thead>
<tr>
<th>Operation</th>
<th>HTTP Method</th>
<th>API</th>
<th>Comments</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Resource</td>
<td>POST or PUT</td>
<td>HTTP PUT with data to the new resource URI</td>
<td>Creates the new resource, does not return the resource</td>
<td>200 (success), 201 (created), 400 (bad request), 401 (unauthorized), 403 (forbidden), 404 (not found), 410 (gone)</td>
</tr>
<tr>
<td>Retrieve Resource</td>
<td>GET</td>
<td>HTTP GET with the resource URI</td>
<td>Returns the resource if found</td>
<td>200 (success), 304 (unmodified), 403 (forbidden), 404 (not found), 410 (gone)</td>
</tr>
<tr>
<td>Update Resource</td>
<td>PUT</td>
<td>HTTP PUT with data to the existing resource URI</td>
<td>Replaces the current resource representation with the data</td>
<td>200 (success), 400 (bad request), 401 (unauthorized), 403 (forbidden), 404 (not found), 409 (conflict), 410 (gone)</td>
</tr>
<tr>
<td>Delete Resource</td>
<td>DELETE</td>
<td>HTTP DELETE with the resource URI</td>
<td>Deletes the resource, does not return the resource</td>
<td>200 (success), 204 (no content), 400 (bad request), 401 (unauthorized), 403 (forbidden), 404 (not found), 410 (gone)</td>
</tr>
<tr>
<td>Get information about a resource</td>
<td>OPTIONS</td>
<td>HTTP OPTIONS with the resource URI</td>
<td>Returns information about the options or requirements associated with the resource</td>
<td>200 (success), 404 (not found), 410 (gone)</td>
</tr>
<tr>
<td>Test a resource link</td>
<td>HEAD</td>
<td>HTTP HEAD with the resource URI</td>
<td>Returns same information as a GET without the body. Used for testing links</td>
<td>200 (success), 304 (unmodified), 403 (forbidden), 404 (not found), 410 (gone)</td>
</tr>
</tbody>
</table>
Algorithmic Resources

- Resources can be algorithms
  - Business Process, Façade, etc…
  - Should Follow HTTP Verb semantics like any other resource
  - Forces good auditing habits.
- Example: Consider Resource /Transfer
  - Transfers money from one account to another

<table>
<thead>
<tr>
<th>Verb</th>
<th>Collection</th>
<th>Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Returns a list of all previous transfers</td>
<td>/Transfer/344</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns record of specific Transfer</td>
</tr>
<tr>
<td>POST</td>
<td>Executes new Transfer.!!</td>
<td>Could retry failed transfer</td>
</tr>
<tr>
<td>PUT</td>
<td>Not Supported</td>
<td>Could Change parameters of transfer still in progress.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Not Supported</td>
<td>Cancel/Purge Transfer</td>
</tr>
</tbody>
</table>
Default Handlers

- Script as a Resources

```php
<?php

class Employee {
    function onGET() {
        echo "Response from a GET request";
    }
    function onPUT() {
        echo "Response from a PUT request";
    }
    function onPOST() {
        echo "Response from a POST request";
    }
    function onDELETE() {
        echo "Response from a DELETE request";
    }
}
?>

<?php
switch (zget('/request/method')) {
    case 'GET':
        // GET handling
        break;
    case 'POST':
        // POST handling
        break;
    case 'DELETE':
        // DELETE handling
        break;
    case 'PUT':
        // PUT handling
        break;
}
?>
```
Custom Handlers

- Script as a Resources
  - `http://<host>:<port>/emp`
  - `http://<host>:<port>/emp/333`

**PHP Resource Handler**
```php
<?php
class Handler {
    function onGET() {
        echo "Response from a GET request";
    }
    function onPUT() {
        echo "Response from a PUT request";
    }
    function onPOST() {
        echo "Response from a POST request";
    }
    function onDelete() {
        echo "Response from a DELETE request";
    }
}
```
RESTful Resources

- RESTful Design
  - Collection Model
  - Action can be taken on the entire collection or a specified member of the collection
  - URI and HTTP method define the resource request

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/people</td>
<td>List members</td>
</tr>
<tr>
<td>POST</td>
<td>/people</td>
<td>Create member</td>
</tr>
<tr>
<td>GET</td>
<td>/people/1</td>
<td>Retrieve member</td>
</tr>
<tr>
<td>PUT</td>
<td>/people/1</td>
<td>Update member</td>
</tr>
<tr>
<td>DELETE</td>
<td>/people/1</td>
<td>Delete member</td>
</tr>
</tbody>
</table>

REST and WebSphere sMash

WebSphere sMash supports URI and HTTP method define the collection resource model
Each script in the `<apphome>/app/resources` directory represents a resource handler

URL convention for interacting with resources based on

```
/resources/<collectionName>[/<memberID>[/<pathInfo>]]
```

where the actions are defined as follows:

<table>
<thead>
<tr>
<th>Resource</th>
<th>GET</th>
<th>PUT</th>
<th>POST</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection</td>
<td>list</td>
<td>putCollection</td>
<td>create</td>
<td>deleteCollection</td>
</tr>
<tr>
<td>Member</td>
<td>retrieve</td>
<td>update</td>
<td>postMember</td>
<td>delete</td>
</tr>
</tbody>
</table>
Resources on the Web

- What are the URIs?
- Which methods are supported at each URI?
- What formats?

<table>
<thead>
<tr>
<th>Resource</th>
<th>URI</th>
<th>Method</th>
<th>Representation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebate list</td>
<td>/resources/rebate</td>
<td>GET</td>
<td>JSON (array)</td>
<td>List</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POST</td>
<td>JSON (object)</td>
<td>Create</td>
</tr>
<tr>
<td>Specific Rebate</td>
<td>/resources/rebate/{id}</td>
<td>GET</td>
<td>JSON (object)</td>
<td>Retrieve</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUT</td>
<td>JSON (object)</td>
<td>Update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE</td>
<td></td>
<td>Delete</td>
</tr>
</tbody>
</table>
Resource Handlers in Zero

- Basic event handlers for /resources/*

- Zero supports the collection model natively within the <apppheme>/app/resources virtualized directory. Each script within the resources directory represents a resource handler, which implements the collection and member operations. Resource handlers are accessed via a simple URL convention:

<table>
<thead>
<tr>
<th>URI pattern</th>
<th>Method</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/resources/collection</td>
<td>GET</td>
<td>list</td>
<td>List of all members</td>
</tr>
<tr>
<td>POST</td>
<td>create</td>
<td></td>
<td>Create member</td>
</tr>
<tr>
<td>/resources/collection/{id}</td>
<td>GET</td>
<td>retrieve</td>
<td>Retrieve one member</td>
</tr>
<tr>
<td>PUT</td>
<td>update</td>
<td></td>
<td>Replace member</td>
</tr>
<tr>
<td>DELETE</td>
<td>delete</td>
<td></td>
<td>Delete member</td>
</tr>
</tbody>
</table>
Resource Handlers  Example

http://<host>:<port>/resources/rebate
Creates list event
onList() handler in employee.groovy
handles event

```php
<?php

// Initialization common to all operations
$dataManager = data_manager('REBATE_DB');

class rebate {

    function onList() {
        global $dataManager;

        $renderType = $location = zget("/request/params/format");
        $rebateRecords = dataExec ($dataManager, "select * from rebate");
        zput('/request/headers/out/Content-Type', 'text/json');
        echo json_encode($rebateRecords);
    }

 ...
```
function onCreate() {
    global $dataManager;

    // Convert the raw JSON stream in to a PHP array
    $er = json_decode($HTTP_RAW_POST_DATA);

    $result = dataInsert($dataManager,
                          "INSERT INTO Rebate (name,description,rebatetype,validfrom,validto,website) ".
                          "VALUES (?, ?, ?, ?, ?, ?)",
                          array('rebateid'), array($er['name'], $er['description'],
                          $er['rebatetype'],
                          $er['validfrom'], $er['validto'], $er['website']));

    $locationUri = zget('/request/path') . "/" . @$result;
    zput('/request/headers/out/Location', $locationUri);
    zput('/request/status', 204);
}
Resource Handlers Example

app/resources/rebate.php (continued)
Similarly GET, PUT, and DELETE to /resources/rebate/333

```php
function onRetrieve() {
    global $dataManager;
    $rebateid = zget("/request/params/rebateId");
    $rebateRecords = dataQueryFirst($dataManager, "select * from rebate where rebateId = ?", array($rebateid));
    if(isset($rebateRecords)) {
        zput('/request/headers/out/Content-Type', 'text/json');
        echo json_encode($rebateRecords);
    } else {
        zput('/request/status', 404);
        $message = "incentiveid ". $incentiveid . " not found.";
        zput('/request/error/message', $message);
        zput('/request/view', 'error');
        render_view();
    }
}

function onUpdate() {
    global $dataManager;
    $rebateid = zget("/request/params/rebateId");
    $er = json_decode($HTTP_RAW_POST_DATA);
    $user = zget('/request/subject#remoteUser');
    $result = dataExec($dataManager, "UPDATE rebate SET name=?, description=?, ".
    "rebatetype=?, validfrom=?, validto=?, website=? WHERE rebateid=?", array($er['name'], $er['description'], $er['rebatetype'], $er['validfrom'],
    $er['validto'], $er['website'], $rebateid));
    if ($result != null) zput('/request/status', 204);
    else {zput('/request/status', 503); echo "Database query execution failure"; }
}

function onDelete() {
    global $dataManager;
    $rebateid = zget("/request/params/rebateId");
    $user = zget('/request/subject#remoteUser');
    $result = dataExec($dataManager, "DELETE FROM rebate WHERE rebateId=?", array($rebateid));
    if ($result != null) zput('/request/status', 204);
    else {zput('/request/status', 503); echo "Database query execution failure"; }
}
?>
```
Data formats – JavaScript Object Notation

• Encode

```php
<?php
$employee = array(
    'name' => "John Smith",
    'id' => '/resources/employee/JohnSmith",
    'mgr' => '/resources/employee/JaneDoe"
);

// Use json_encode()
zput(’/request.headers/out/Content-Type', 'application/json');
echo json_encode($employee);

// Alternatively use Zero JSON renderer
zput(’/request/view', 'JSON');
zput(’/request/json/output', $employee);
render_view();
?>
```

• Decode

```php
<?php
$employee = json_decode($HTTP_RAW_POST_DATA);

// Alternatively
$input = fopen("php://input", 'r');
$employee = json_decode(fread($input, 1024));
?>
```
Data Format - XML

```php
<?php
$address = array('line1' => 'This lane', 'line2' => 'Somewhere');
$employee = array('name' => 'Smith', 'address' => $address);

// Alternatively use the XML renderer
zput('/', 'request/view', 'XML');
zput('/', 'request/xml/output', $employee);
zput('/', 'request/xml/rootElement', 'employee');
zput('/', 'request/xml/idRefs', false);
render_view();
?>

<?xml version="1.0" encoding="UTF-8"?>
<employee>
  <name>Smith</name>
  <address>
    <line1>This lane</line1>
    <line2>Somewhere</line2>
  </address>
</employee>
```

Decode

```php
<?php
$xmlstring = <<<XML
<?xml version="1.0" encoding="UTF-8"?>
<employees>
  <employee id="345433">
      <name>John Smith</name>
  </employee>
  <employee id="343334">
      <name>Jane Doe</name>
  </employee>
</employees>
XML;

$employees = xml_decode($xmlstring);

$employee = $employees->employee[0];
$name = $employee->name;
$id = $employee->getAttribute('id');
?>
```
Data Format - ATOM

Atom document

```xml
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>http://localhost:8080/feed.php?id</id>
  <title type="text">http://localhost:8080/feed.php</title>
  <link href="http://localhost:8080/feed.php" rel="self"></link>
  <updated>1980-12-25T12:00:00.000Z</updated>
  <entry>
    <id>http://localhost:8080/feed.php/1</id>
    <title type="text">A Good Title is Important</title>
    <updated>1970-01-01T00:00:00.000Z</updated>
    <author>
      <name>John Doe</name>
    </author>
    <content type="text">Content is also important.</content>
    <link href="http://localhost:8080/feed.php/1" rel="edit"></link>
  </entry>
  <entry>
    <id>http://localhost:8080/feed.php/2</id>
    <title type="text">Bad Titles are Misleading</title>
    <updated>1980-12-25T12:00:00.000Z</updated>
    <author>
      <name>Jane Q. Sample</name>
    </author>
    <content type="text">Content is also important.</content>
    <link href="http://localhost:8080/feed.php/2" rel="edit"></link>
  </entry>
</feed>
```

Atom Feed

```php
<?php

// Rendering an Atom feed document.

$feed = array(
    array(
        "id" => 1,
        "title" => "A Good Title is Important",
        "authorname" => "John Doe",
        "updated" => "1970-01-01", // date format
        "contenttype" => "TEXT",
        "content" => "Content is also important."
    ),
    array(
        "id" => 2,
        "title" => "Bad Titles are Misleading",
        "authorname" => "Jane Q. Sample",
        "updated" => "1980-12-25 12:00:00", // date time format
        "contenttype" => "TEXT",
        "content" => "Content is also important."
    )
);

zput("/request/view","atom");
zput("/request/atom/output","$feed");
render_view();
?>
```
An alternative: Zero Resource Model (ZRM)

- Model application data
  - Constrained set of APIs encourages a RESTful application architecture
  - Data model that maps well into Atom feeds and JSON formats
  - Robust framework for persistence, validation, and serialization
  - Application Databases focus
ZRM Development life cycle

```json
app/models/employee.json
{
   "fields" : {
      "first_name": {"type":"string"},
      "last_name": {"type":"string"},
      "location": {"type":"string"}
   }
}
```

```php
<?php
zrm_delegate();
?>
```

```json
app/models/fixtures/initial_data.json
[
   {
      "type": "employee",
      "fields": {
         "first_name": "Alice",
         "last_name": "Rogers",
         "location": "Seattle"
      }
   },
   {
      "type": "employee",
      "fields": {
         "first_name": "Bill",
         "last_name": "Stevens",
         "location": "Seattle"
      }
   },
   {
      "type": "employee",
      "fields": {
         "first_name": "Cathy",
         "last_name": "Tomlin",
         "location": "Boston"
      }
   }
]
```

```
roly-mac:zero bariaq$ zero model sync
```
Activity flows in a Nutshell

Rick builds things for the palette

Linda designs the Web UI

Frank strings together flow activities into solutions

George builds flows, Web UIs and ad-hoc scripts

Rick scripts ad-hoc activities
sMash Features and Services

- Dojo Toolkit
  - Drag and Drop Development in AppBuilder
  - sMash Dojo Dijits and sMash REST Store
- Data Access
  - Tools for generating and running DB Scripts
  - API based on pureQuery
- iWidget Creation
  - Integration with Lotus Mashup Center
- Security
  - LDAP Based Registries
  - Active Content Filtering
- Integration
  - Feed Support (RSS, ATOM)
  - Extended protocols (JMS, SFTP, Mail, REST to SOAP)
  - Timers
- Services
  - Excel Services
  - Open Services (Jazz Platform Integration)
Agenda

- WebSphere sMash
- WebSphere sMash Programming Model
- **PHP in WebSphere sMash**
- PHP Applications
- Demonstrations

let's build a smarter planet
Why PHP?

- 20M+ web domains use PHP
- 3M+ Programmers know PHP
- Huge repository of reusable modules, snippets, extensions.
- Easy language to learn -> Mashups
- Language has evolved to be easy to use

Gartner (Dec 2007)
- PHP Developers to grow from 3 to 5.5 million by 2013
- PHP Developers in Commercial or Corporate IT to grow from 13% to 60% by 2013
- “Pay special attention to opportunities to leverage PHP in combination with Java development efforts”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>=</td>
<td>Java</td>
<td>20.715%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>=</td>
<td>C</td>
<td>15.379%</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>↑↑</td>
<td>C++</td>
<td>10.715%</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>↓</td>
<td>(Visual) Basic</td>
<td>10.490%</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>↓</td>
<td>PHP</td>
<td>9.243%</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>↑↑</td>
<td>Python</td>
<td>5.012%</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>↓</td>
<td>Perl</td>
<td>4.841%</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>↓</td>
<td>C#</td>
<td>4.334%</td>
</tr>
</tbody>
</table>
WebSphere sMash PHP Support

• PHP runtime built in Java.
  – To PHP.net as Jruby is to Ruby and Jython is to Python
• Compile PHP into Java bytecodes and run on a Java Virtual Machine.
• Powerful blending of PHP and Java code.
  – Java and PHP code run in the same process on the same Thread
    • No need for Inter-process communication.
  – Efficient calls between PHP, Java, Groovy on the same stack.
  – Pass data between Java, Groovy and PHP without copying.
    • Avoids serialising and passing data between processes.
• Import Java classes as PHP Classes
  • Easy access to the many Java libraries from PHP code.
• http://www.projectzero.org/sMash/1.1.x/docs/zero.devguide.doc/zero.php/Core.html
PHP in WebSphere sMash

- Runs PHP 5 scripts
- Requires Java 5 SE or later.

- Extensibility via XAPI
  - XAPI-C for C extensions from php.net
  - XAPI-J for Java extensions, native libraries invoked over JNI and Project Zero interface
  - Extension language choice opaque to PHP script

- Java Bridge
- Debug using via xdebug protocol using Eclipse with PDT
Benefits of PHP in sMash.

• Develop quickly by using the best tools and materials for the job.
  – PHP code such as smarty, SimpleXML, drupal, phpBB,
  – Java code such as Apache Lucene, POI and Eclipse BIRT

• Start simple using sMash tooling such as ZRM, Flow
  – Customise and extend using PHP scripts and snippets

• Unleash agile teams using Java and PHP skills.
  – Allow teams to use their full range of skills.

• Build on a solid base.
  – PHP built on the Java VM at the heart of IBM’s enterprise software stack.
    • Familiar to many enterprises.
    • Vast investment in JIT, Garbage Collector, RAS and tools.
PHP – Java/Groovy Interaction

• PHP Java/Groovy Bridge allows PHP to:
  – Instantiate Java Classes
  – Call static and instance methods
  – Access static and instance fields
  – Extend Java Classes (not abstract)
  – Implement Java Interfaces.
  – Interact with Groovy Classes objects, Closures and Ranges

• Zero programming model allows PHP to:
  – Interact with other Modules built using Groovy, Java, PHP, Flow by:
    • Handle and fire Zero events
    • Fetch and store to the global context.
PHP – Java Bridge – Basic use

```php
<?php
$date = new java("java.util.Date", 70, 9, 4);
$map = new java("java.util.HashMap");
$map->put("title", "Java Bridge!");
$map->put("when", $date);
$array = array(1,2,3,4,5);
$map->put("stuff", $array);
$map->get("stuff");
?>
```

Basic access to Java methods and fields - types are automatically converted at boundary of PHP runtime

```php
<?php
$system = new javaClass("java.lang.System");
echo("Current time: " . $system->currentTimeMillis() . "\n");

$integerClass = new javaClass("java.lang.Integer");
$integerClass->parseInt("1234567890");
?>
```

Static methods and fields are accessible by using the built in JavaClass class

Java exceptions can be caught in PHP scripts

```php
<?php
try {
    $system = new javaClass("java.lang.System");
    $system->getProperty(FALSE);
} catch (javaException $exception) {
    echo "Cause: " . $exception->getCause() . "\n";
    var_dump($exception->getCause());
}
?>
```
PHP- Java Bridge – Iterators and overloads

```php
<?php
$list = new Java("java.util.ArrayList");
$list->add("Hello World!");
$list->add(FALSE);
$list->add(1234567890);
var_dump($list);
foreach ($list as $key => $value) {
    echo $key." ".$value."\n";
}
?>
```

Bridging between PHP and Java iterators

```php
<?php
$signature = new JavaSignature(JAVA_STRING);
$string = new Java("java.lang.String", $signature, "Hello World!");
$class = new JavaClass("java.lang.String");
$signature = new JavaSignature(JAVA_INT);
var_dump($class->valueOf($signature, 1234567890));
?>
```

Signatures provide explicit control for overloaded constructors and methods

```php
<?php
$string = new Java("java.lang.String", FALSE);
$string->lastIndexOf(FALSE);
?>
```

Notice: No signature on ambiguous call to method 'lastIndexOf' in ...
Java Bridge – Importing Java Classes.

```php
<?php
java_import("java.lang.Integer", NULL, FALSE);
$value = new Integer(new JavaSignature(JAVA_STRING), "1234567890");
echo "Integer [".$value."]\\n";

java_import("java.util.Date");
$date = new Date(70, 9, 4);
$date->toLocaleString();

java_import("java.lang.Comparable");
ReflectionClass::export("Comparable");
?>
```

Importing a Java class creates a PHP class that has the same shape.

```php
<?php
java_import("java.lang.Integer", NULL, FALSE, "TestInteger");
$value = new TestInteger(new JavaSignature(JAVA_STRING), "1234567890");
echo "Integer [".$value."]\\n";
?>
```

```php
<?php
java_import("java.util.ArrayList", array("Traversable"), FALSE);
$list = new ArrayList();
$list->add("Hello World!");
$list->add(FALSE);
$list->add(1234567890);
foreach ($list as $key => $value) {
    echo $key." ".$value."\\n";
}
?>
```
Java Bridge – Extending Java in PHP

Extending a Java class inside PHP is possible but has some limitations.

```php
<?php
java_import("java.io.File");

class SuperFile extends File {
    function SuperFile($signature, $path) {
        parent::__construct($signature, $path);
    }
    function isThisCool() {
        return TRUE; // Way cool
    }
}

$file = new SuperFile(new JavaSignature(JAVA_STRING), "/");
$file->isDirectory();
$file->isThisCool();
?>
```

Java bean access maps field access onto get/set method calls.

```php
<?php
java_import("java.io.File");
$file = new File("/");
echo $file->Parent."\n";
echo $file->Name."\n";

java_import("java.lang.StringBuffer");
$buffer = new StringBuffer("Hello World!");
$buffer->Length = 5;
echo $buffer->toString()."\n";
?>
```
Groovy Bridge – Importing Scripts

```php
<?php
groovy_import("Import.groovy");
$import = new Import();
$result = $import->getDate();
?>
```

class Import {
    def getDate() {
        return new Date();
    }
}

Class [ <internal> class Import ] {
    - Constants [0] {
    }
    - Static properties [2] {
        Property [ public static $__timeStamp ]
        Property [ public static $__timeStamp__239_neverHappen1224852539937 ]
    }
    - Static methods [0] {
    }
    - Properties [0] {
    }
    - Methods [8] {
        Method [ <internal, ctor> public method Import ] {
        }
        Method [ <internal> public method getDate ] {
        }
        Method [ <internal> public method getMetaClass ] {
        }
    }
...
Other Groovy Bridge Features

• Method and Field Access
• Closures and Curry
  – PHP Functions can be passed to Groovy as a closure.
Invoking PHP - Events

- PHP Event Handler examples:
  - Timer
  - Custom Event
  - Flow, Security or Connection event.
  - Standard Request Event

```php
$confighandlers += ['
  "events" : "myevent",
  "handler" : "myeventhandler.php"
']
```

```php
// Event Handler for "myevent" stored in app/scripts/myeventhandler.php
$arr = array('foo' => 'bar');
put('/request/somekey', $arr);
?>
```
Invoking PHP - Script Activity in Flow

```php
$title = $get("/event/title");
$content = $get("/event/contentText");
```
Agenda

- WebSphere sMash
- PHP in WebSphere sMash
- WebSphere sMash Programming Model
- PHP Applications
- Demonstrations

let's build a smarter planet
PHP Applications that run on sMash

- Forums: phpBB
- CRM: SugarCRM
- wiki: MediaWiki
- Ajax Debugging: FirePHP
- Content Management: Drupal
- Desktop Virtualisation: eyeOS
- Blogging: WordPress
PHP Applications

phpBB
SugarCRM
WordPress
MediaWiki
FirePHP
Drupal
EyeOS
ZSL develops Web 2.0 Assets 3x faster with WebSphere sMash

- Downloaded sMash DE from projectzero.org
- Assets built in 3½ weeks
- Junior web development team
  - Dynamic Scripting Skills
  - Web Development Focus
  - Know very little about .Net and JEE
  - Understand concepts and functioning of Web Services, but may not have built or deployed them.

- 67% reduction in time-to-market for developing Web 2.0 assets
- 90% less time to implement best-of-breed programs
- Ability to reuse 25% of code
- Out-of-the-box functionality vs. 2½ days to install comparable software
Energy Commons Overview

www.energycommons.com (Situational Application)

• Concept
  – Standardized, private labeled social networking application offering targeted at businesses, consumers, and inter company collaboration on the energy topic

• Value Proposition
  – Interconnection of parties for knowledge share and exchange on demand in a silo’d industry through shared cost model for development and operations

• Innovative Aspects
  – Interconnected portals through “hub and spoke” utilizing global reach, large ecosystem, and utility channel to create scale
  – Viral expansion into energy value chain
  – Incentive Finder for Data Center incentives
  – 24x7 advertisement for IBM as global innovator for energy and climate
Energy Commons Architecture Overview
SugarCRM Integration scenario
Demonstrations
Agenda

- WebSphere sMash
- PHP in WebSphere sMash
- PHP Applications
- Demonstrations
Questions
References

• WebSphere sMash site  http://www.projectzero.org/

• WebSphere sMash forums http://www.projectzero.org/forum/


• PHP Applications that run on sMash:
We love your Feedback!

• Don’t forget to submit your Impact session and speaker feedback! Your feedback is very important to us, we use it to improve our conference for you next year.

• Go to www.impact09guide.com on a smartphone device or a loaner device

• From the Impact 2009 Online Conference Guide;
  – Select Agenda
  – Navigate to the session you want to give feedback on
  – Select the session or speaker feedback links
  – Submit your feedback
The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries:
ibm.com/legal/copytrade.shtml
AIX, CICS, CICSPlex, DataPower, DB2, DB2 Universal Database, i5/OS, IBM, the IBM logo, IMS/ESA, Power Systems, Lotus, OMEGAMON, OS/390, Parallel Sysplex, pureXML, Rational, Redbooks, Sametime, SMART SOA, System z, Tivoli, WebSphere, and z/OS.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Backup
Nested URIs

For example: `/resources/employees/Roland/accounts/FooTech`

- Development option 1:
  `/resources/employees/{employeesId}/{pathInfo}`

- Development option 2:
  `/resources/employees/{employeesId}/accounts/{acctcountId}`

and accounts.bnd:
`employees/accounts`

```groovy
def onList()
{
    def empld = request.employeeld[];
    def info = request.pathInfo[];
}
```
New Features
In-House Repository

- In-house repository
  - Publish your own modules
  - Hosted by a sMash Application or Web Server
Data support

• Externalized SQL statements

/config/db/reviewDB/statements = {
    "SELECT_ALL" : "SELECT * FROM table",
    "SELECT_WHERE_IN" : "SELECT * FROM table WHERE id IN (?,?,?,?)",
    "INSERT" : "INSERT INTO table (col1, col2) VALUES (?,?)"
}

def allrows = data.queryList('SELECT_ALL')

• Zero Resource Model for PHP
• SQL Server 2005
Excel Service

```
Uploading Excel File:
/Users/fraenkel/eclipse/11/ (Browse)

Resource Name:
division

Advanced

Overwrite existing resource

Worksheet Name:
End at row:
Start at row:

Create Resource

### division

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Date of employment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adam</td>
<td>Barr</td>
<td>East</td>
<td>M</td>
<td>1998-10-09</td>
<td>100</td>
<td>2009-04-20 12:53:18</td>
</tr>
</tbody>
</table>
```

Messages from service:

CWFZG70101: The resource named division was created.
Excel Service

ZRM

XLS

Data

{ "fields": {
    "Firstname": { "type": "string" },
    "Lastname": { "type": "string" },
    "Location": { "type": "string" },
    "Gender": { "type": "string" },
    "DateofEmployment": { "type": "date" }
}}

[ { "type": "employees", "fields": { }} ]
Unit Testing

• zero test
  – JUnit based
  – Tests can run
    • inside the application via a request
    • outside the application
  – Specialized task testing
Parallel Recycle

Requests

Without parallel recycle

With parallel recycle

sMash Server

sMash Server
Admin Console
sMash in the Cloud

• Available on Amazon EC2
  – sMash DE 1.1.0.1
    • AppBuilder enabled with security
• See http://tinyurl.com/sMashEC2
Samples

• Broken down into levels

• Cookbook
  - Recipes written by anyone
Tighter integration with IBM Mashup Center

- Generate iWidget templates
- Simple web page to test iWidget interactions
- Easy deployment into MashupHub:

```
Available commands: zero, svn, clear, help
* Commands are run from the current application root directory.
command> zero iwidget publish -title="Hello World Widget" -defURL=http://mysite
```
iWidget Editor

- Define events and payload
- Logical html pages
PHP Applications

phpBB
SugarCRM
WordPress
MediaWiki
FirePHP
Drupal
EyeOS
PHP to Java/Groovy Bridge

Access to Java types and methods

Automatic conversions

```
<?php
$date = new Java("java.util.Date", 70, 9, 4);
$map = new Java("java.util.HashMap");
$map->put("title", "Java Bridge");
$map->put("when", $date);
$array = array(1,2,3,4,5);
$map->put("stuff", $array);
$map->get("stuff");

try {
    $system = new JavaClass("java.lang.System");
    $system->getProperty(FALSE);
} catch (JavaException $exception) {
    echo "Cause: ", $exception->getCause()." \n";
    var_dump($exception->getCause());
}
```
Assemble Flow

- New activities
  - Web UI for simple collaboration
  - Invoke script - Groovy or PHP
- Flow persistence - app zone or DB
- User defined activities
AppBuilder
Source Editor

```python
1. import zero.resource.*
2. def onList() {
3.   def remoteUser = request.subject.remoteUser
4.   if (remoteUser == null) {
5.     request.status = 405
6.     request.view = 'error'
7.   } else {
8.     def collection = TypeCollection.retrieve('todo')
9.     def myTodos = collection.list(assignee: remoteUser)
10.    request.view = 'JSON'
11.    request.json.output = myTodos
12.  }
13.  request.json.output = myTodos
14.  render()
```
Page Editor
Search Dialog
Assemble Flow Editor
Zero Resource Model Editor
Zero Form Editor

Approve Travel Request
Select "Mark as approved" checkbox to approve this request.

- Employee's Email (e.g., hint)
  - This value is required
- Reason (e.g., hint)
- Destination (e.g., hint)
- Start Date (e.g., hint)
- Cost ($) (e.g., hint)
  - This value is required
- Mark as approved (e.g., hint)
Available commands: zero, svn, clear, help
* Commands are run from the current application root directory.
  command> help

The system commands will be run from the application root directory.
The PATH environment variable before running these commands in App.

Usage of shell constructs like piping or redirection is not supported.

Commands allowed by zero.config:
  zero
  svn

Built-in commands:
  clear: clear the console window (also deletes the log history)
  help: display this help message

command>
Debugger

```
println 'Welcome to the Debug demo'
def i = 0
i = i + 1
println 'The value of i is' + i
```
Import ZIP file