AgileApps Development 101

A Whirlwind Tour of Platform Development Capabilities

Eric Armstrong
Developer, Doc Wrangler, Ux Advocate
eric.armstrong@softwareAG.com

What we're going to do…

- Overview
  - Where to find Information
  - Application Anatomy
- Hands On
  - A small app w/a JSP page and a controller
  - Additional information for Java developers
- JavaScript and REST
  - Add JavaScript to an Object Form
  - Using the REST APIs
- Summary of Tools

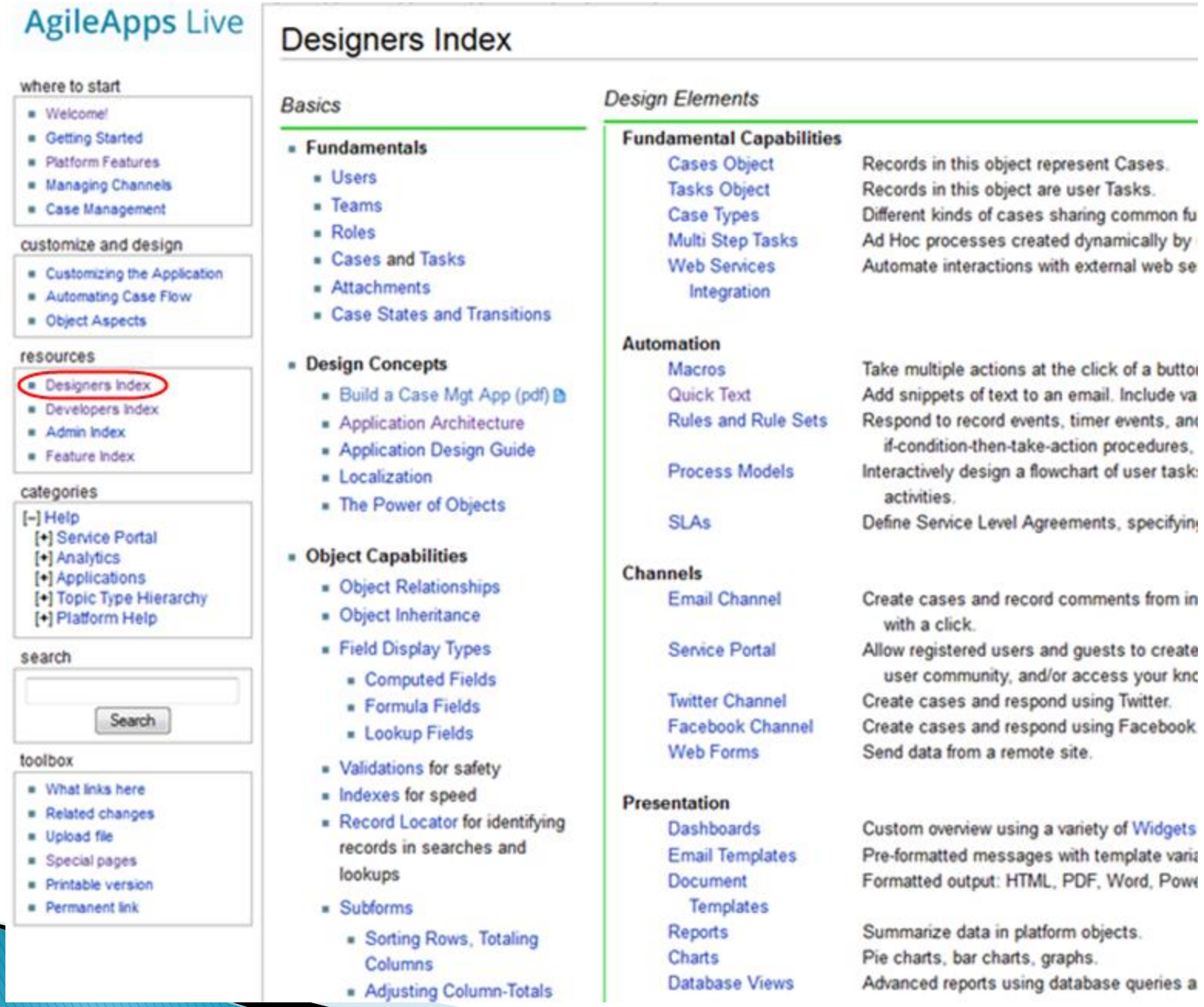
- Overview
  - Where to find Information ←
  - Application Overview
- Hands On
  - A small app w/a JSP page and a controller
  - Additional information for Java developers
- Demos
  - Add JavaScript to an Object Form
  - Using the REST APIs
- Summary of Tools

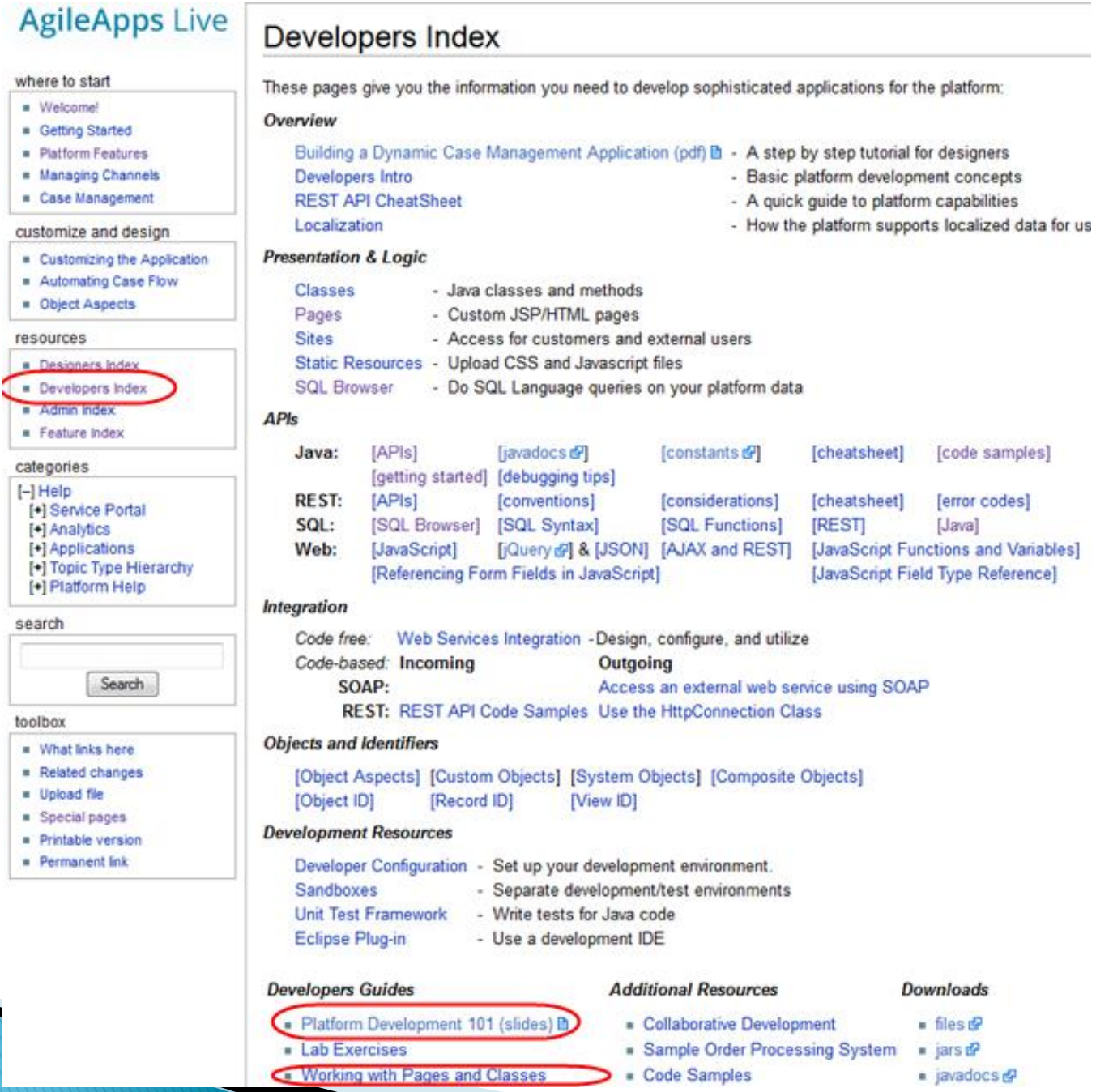
Sources of Information

- One Stop Shopping for Dev Info: http://agileappslive.info/wiki/Developers_Index

- Java APIs & Cheatsheet
- JavaDocs
- Rest APIs & Cheatsheet
- JavaScript APIs + jQuery
- SQL Browser, Syntax, & APIs
- Lots more…

- Search is your friend…


AgileApps Live
where to start
Welcome!
Getting Started
Platform Features
Managing Channels
Case Management
customize and design
Customizing the Application
Automating Case Flow
Object Aspects
resources
Designers Index
Developers Index
Admin Index
Feature Index
categories
[–] Help
[+] Service Portal
[+] Analytics
[+] Applications
[+] Topic Type Hierarchy
[+] Platform Help
search
Search
toolbox
What links here
Related changes
Upload file
Special pages
Printable version
Permanent link
Designers Index
Basics
Fundamentals
Users
Teams
Roles
Cases and Tasks
Attachments
Case States and Transitions
Design Concepts
Build a Case Mgt App (pdf)
Application Architecture
Application Design Guide
Localization
The Power of Objects
Object Capabilities
Object Relationships
Object Inheritance
Field Display Types
Computed Fields
Formula Fields
Lookup Fields
Validations for safety
Indexes for speed
Record Locator for identifying records in searches and lookups
Subforms
Sorting Rows, Totaling Columns
Adjusting Column-Totals
Design Elements
Fundamental Capabilities
Cases Object
Records in this object represent Cases.
Tasks Object
Records in this object are user Tasks.
Case Types
Different kinds of cases sharing common fu
Multi Step Tasks
Ad Hoc processes created dynamically by
Web Services Integration
Automate interactions with external web se
Automation
Macros
Take multiple actions at the click of a butto
Quick Text
Add snippets of text to an email. Include va
Rules and Rule Sets
Respond to record events, timer events, an
if-condition-then-take-action procedures,
Process Models
Interactively design a flowchart of user task
activities.
SLAs
Define Service Level Agreements, specifyin
Channels
Email Channel
Create cases and record comments from in
with a click.
Service Portal
Allow registered users and guests to create
user community, and/or access your kno
Twitter Channel
Create cases and respond using Twitter.
Facebook Channel
Create cases and respond using Facebook
Web Forms
Send data from a remote site.
Presentation
Dashboards
Custom overview using a variety of Widgets
Email Templates
Pre-formatted messages with template vari
Document Templates
Formatted output: HTML, PDF, Word, Powe
Reports
Summarize data in platform objects.
Charts
Pie charts, bar charts, graphs.
Database Views
Advanced reports using database queries a


AgileApps Live
where to start
Welcome!
Getting Started
Platform Features
Managing Channels
Case Management
customize and design
Customizing the Application
Automating Case Flow
Object Aspects
resources
Designers Index
Developers Index
Admin Index
Feature Index
categories
[–] Help
[+] Service Portal
[+] Analytics
[+] Applications
[+] Topic Type Hierarchy
[+] Platform Help
search
Search
toolbox
What links here
Related changes
Upload file
Special pages
Printable version
Permanent link
Developers Index
These pages give you the information you need to develop sophisticated applications for the platform:
Overview
Building a Dynamic Case Management Application (pdf) - A step by step tutorial for designers
Developers Intro - Basic platform development concepts
REST API CheatSheet - A quick guide to platform capabilities
Localization - How the platform supports localized data for us
Presentation & Logic
Classes - Java classes and methods
Pages - Custom JSP/HTML pages
Sites - Access for customers and external users
Static Resources - Upload CSS and Javascript files
SQL Browser - Do SQL Language queries on your platform data
APIs
Java: [APIs] [javadocs] [constants] [cheatsheet] [code samples]
[getting started] [debugging tips]
REST: [APIs] [conventions] [considerations] [cheatsheet] [error codes]
SQL: [SQL Browser] [SQL Syntax] [SQL Functions] [REST] [Java]
Web: [JavaScript] [jQuery] & [JSON] [AJAX and REST] [JavaScript Functions and Variables]
[Referencing Form Fields in JavaScript] [JavaScript Field Type Reference]
Integration
Code free: Web Services Integration - Design, configure, and utilize
Code-based: Incoming Outgoing
SOAP: Access an external web service using SOAP
REST: REST API Code Samples Use the HttpConnection Class
Objects and Identifiers
[Object Aspects] [Custom Objects] [System Objects] [Composite Objects]
[Object ID] [Record ID] [View ID]
Development Resources
Developer Configuration - Set up your development environment.
Sandboxes - Separate development/test environments
Unit Test Framework - Write tests for Java code
Eclipse Plug-in - Use a development IDE
Developers Guides
Platform Development 101 (slides)
Lab Exercises
Working with Pages and Classes
Additional Resources
Collaborative Development
Sample Order Processing System
Code Samples
Downloads
files
jars
javadocs

- Overview
  - Where to find Information
  - Application Overview ←
- Hands On
  - A small app w/a JSP page and a controller
  - Additional information for Java developers
- Demos
  - Add JavaScript to an Object Form
  - Using the REST APIs
- Summary of Tools

Automated Facets of Objects

- Fields:
  - Date/Time & Currency (localizable)
  - Formulas
  - …
- Forms (defaults, interactive builder)
- Validations (field/expression comparison)
- Rules, Processes & Web Services, SLAs
- Indexes
- Record Locators
- DB Configuration & access
- More …

The Power of Java Server Pages

- "JSP" Page
  - = HTML + Java Code
  - Interact with the Platform
  - Work with your data

- JSP Basics
  - <% … code… %> -- do something
  - <%=variable %> -- insert data

- Conditionals and Loops
  - <% if (…) { %>
    … HTML …
    <% } %>

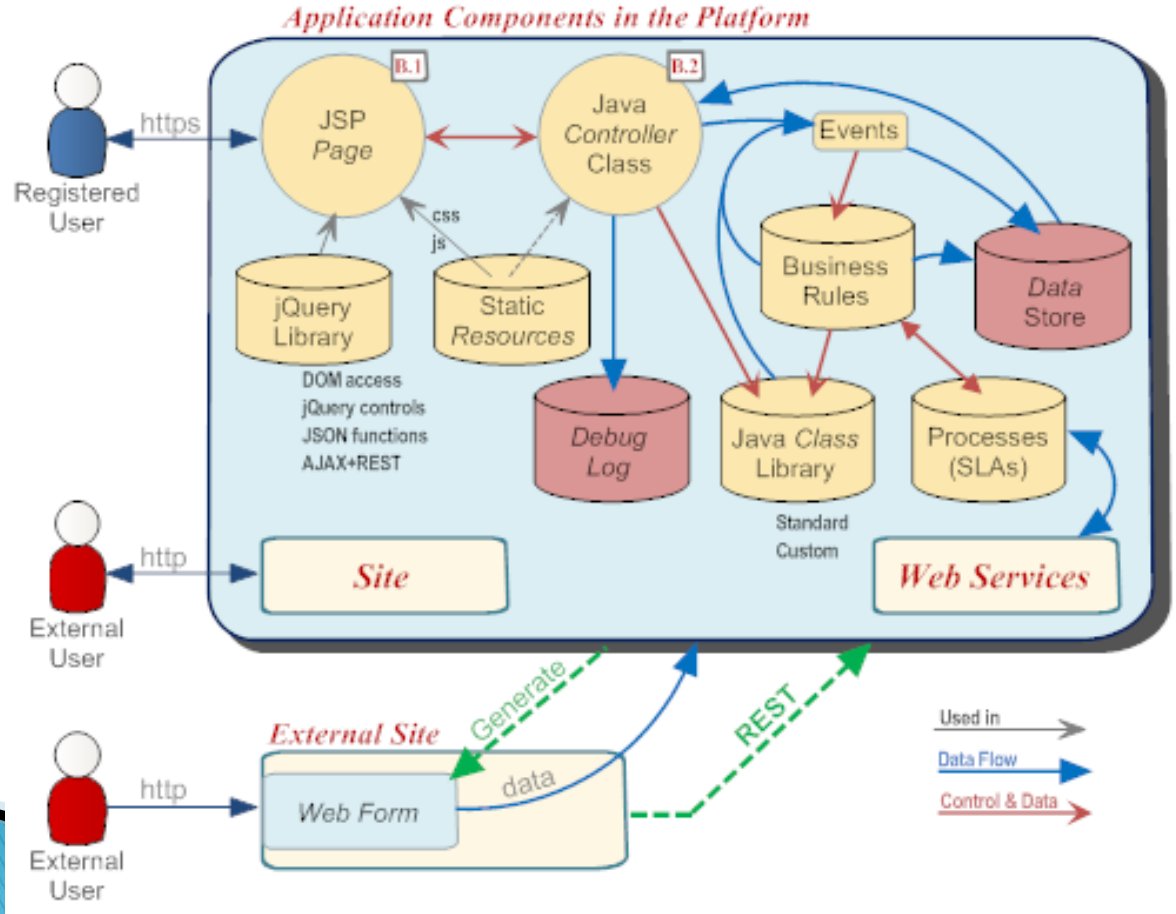
<% for (item:List) { %>
… HTML …
<% } %>

The Many Uses of JSP Pages

- Visit the URL (https://.../networking/pages/xyz.jsp)
  (no context—just the page)
- Web Tabs
  (appears like any other tab in the application)
- Dashboard Widgets (display data & graphics)
- Sites
- Custom Forms
- Custom Lookups

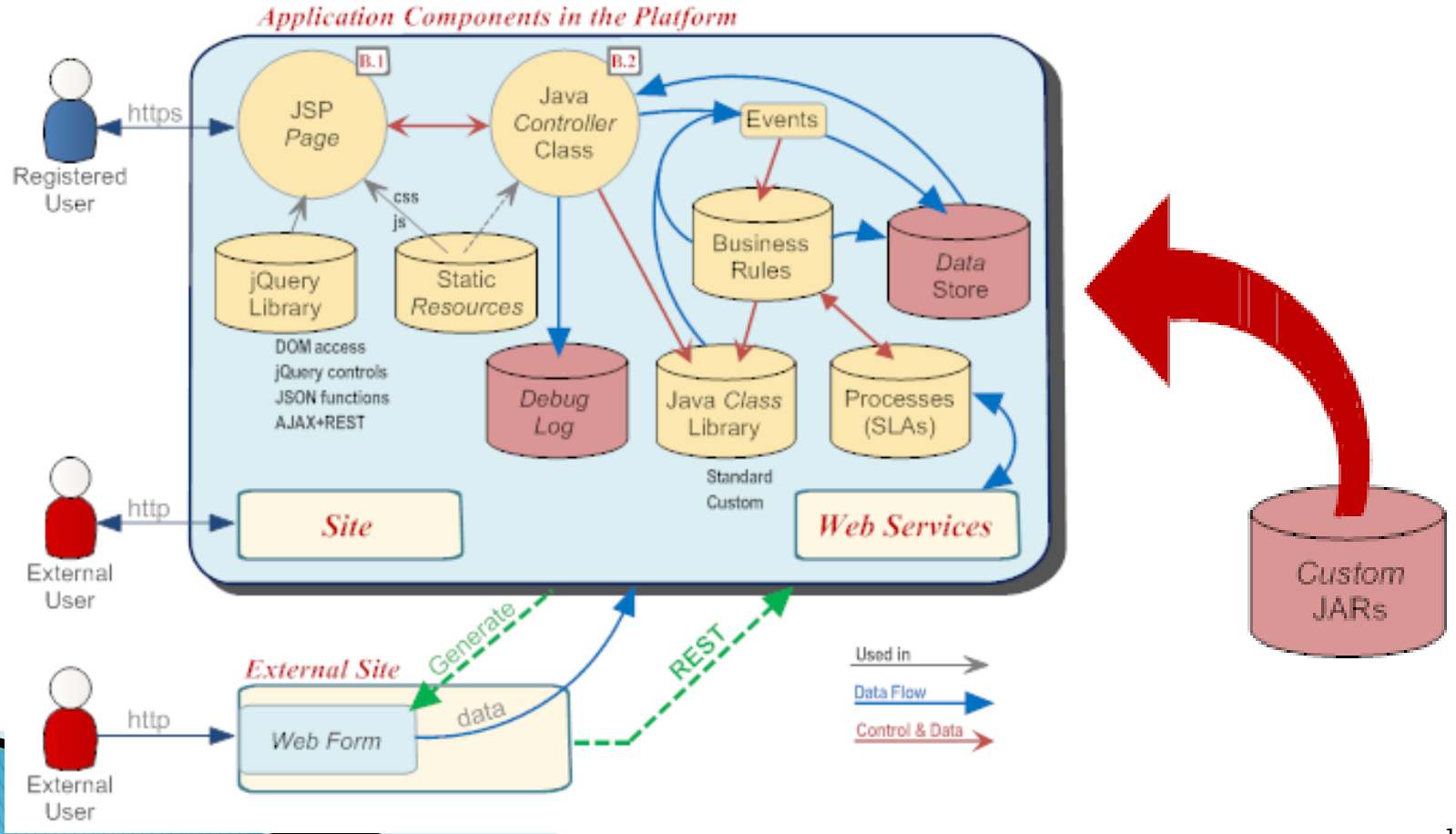
Anatomy of an Application

- Application Architecture
  http://agileappslive.info/wiki/Application_Architecture

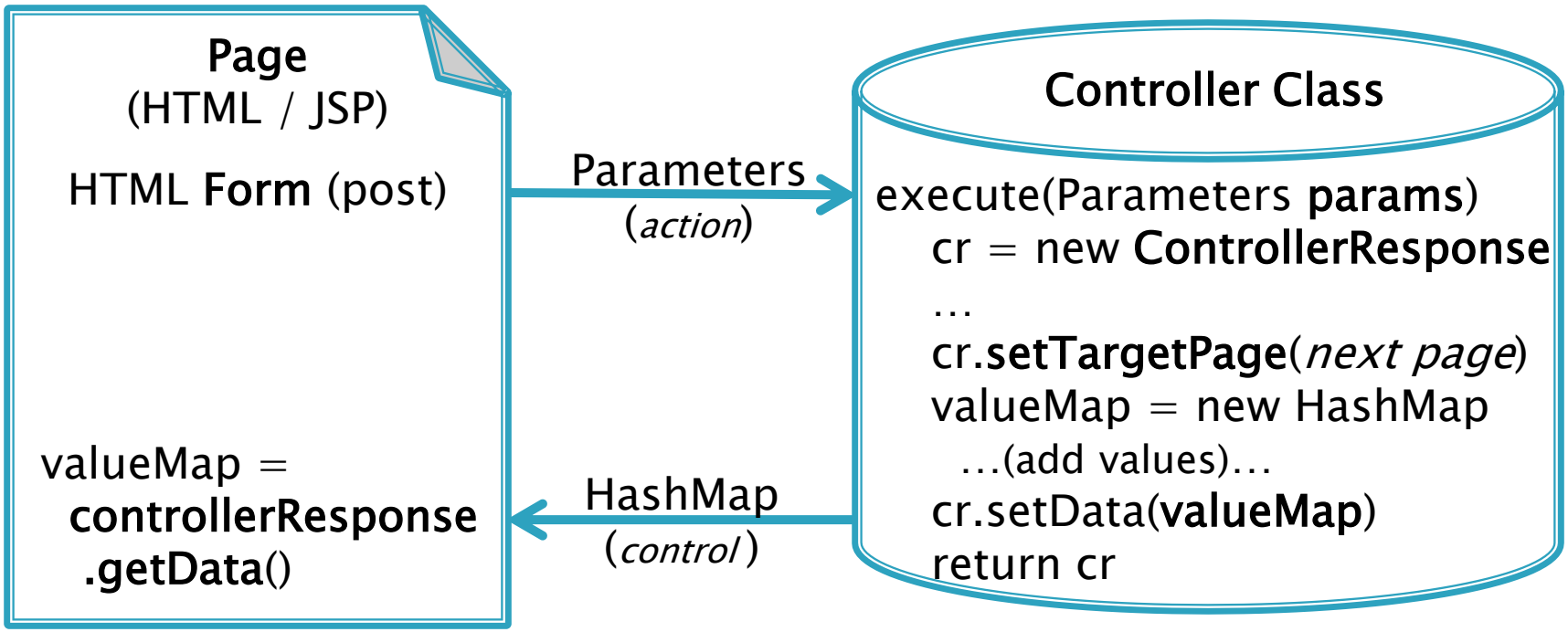


For an On-Premise Installation

- Add libraries (e.g. A 3rd party rules engine, or existing libraries that integrate w/external systems or with custom web services)



Pages and Controller Classes



- java.util.HashMap (*valueMap*)
- com.platform.api.Parameters (*params*)

  (used by all platform APIs)

Calling Platform APIs

- 99% in the Functions.class
  http://agileappslive.info/wiki/download/javadocs/
- *Going In:* com.platform.api.Parameters
  - Parameters params = Functions.getParametersInstance();
- *Coming Back Out:* com.platform.api.Result
  - Failure: getCode() < 0 -- getMessage() for info
    Search Success: getCode() >= 0 -- #of items returned
    Other Success: getCode() == 0
  - Exactly one value (getRecord) ➔ a Parameters object:
    Parameters result_params = result.getParameters();
  - One or more (search) ➔ A *list* of Parameters objects:
    ParametersIterator it = result.getIterator();
    Parameters result_params = it.next();
  - *Controller:* Put Result values into a HashMap for the JSP page

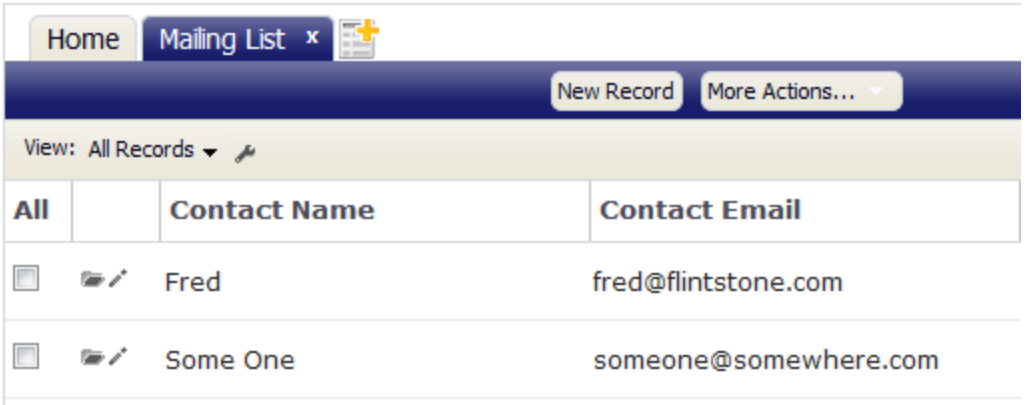
My Naming Conventions

- In an attempt to improve clarity…
  - ***valueMap*** = HashMap instance passed to a Page
  - ***page_control*** = passed *to* a Page in a valueMap
  - ***params*** =Parameters instance passed to an API
  - ***page_action*** = passed *from* a Page in the params
  - ***result*** = Result values passed back from an API
  - ***result_params*** = Parameters object from the result

- Overview
  - Where to find Information
  - Application Overview
- Hands On
  - A small app w/a JSP page and a controller ←
  - Additional information for Java developers
- Demos
  - Add JavaScript to an Object Form
  - Using the REST APIs
- Summary of Tools

Hands On: JSP Page & Controller

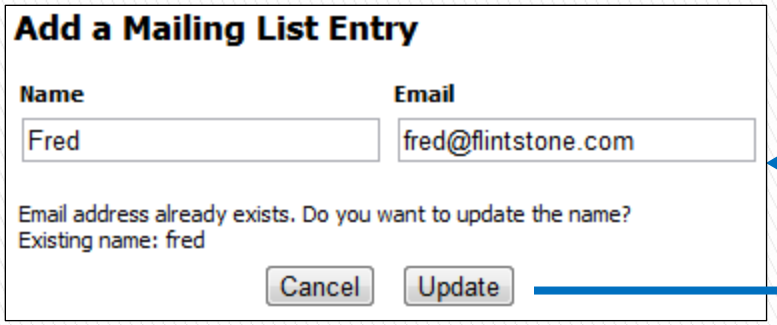
- Simple “Mailing List” object

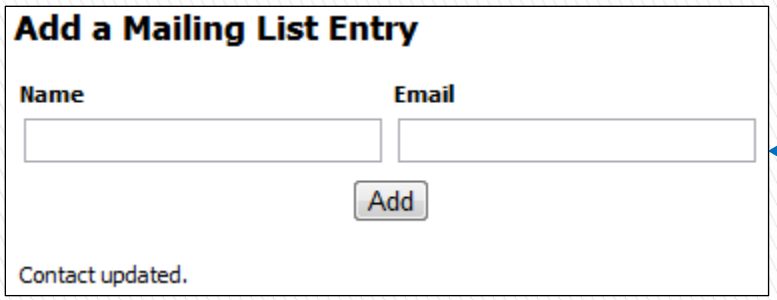


- Fields: **contact_name, contact_email**
- Functional Spec
  - Add new name/address pairs to the list
  - Don’t add if the email address already exists
  - Give user a chance to update the name

Usage Scenario

JSP Page





Java Controller

Email already exists!

update_query

Update

Update the record

updated

Getting started

- Login:
  - https://{yourDomain}.agileappslive.com
  - Supply Username & Password

- Setup:
  - **{gear} >Developer Resources**
  - **Developer Configuration > [Edit]**
    - *Namespace* – the *company* package name ("myCo")
      (You define subspaces when you add classes)
    - *Debug Log Level* – how much info you see
  - **Debug Log** (end of sidebar)

- Full Instructions
  - http://agileappslive.info/wiki/HowTo:Create_a_JSP_Page_and_Java_Controller

Create a JSP Page (Html + Java)

- **{gear} > Developer Resources > Pages > New Page**
- Name: AddUpdateYOURNAME.jsp (*extension, no _'s*)
- Header Files: Yes (so the page can appear as a tab)
- Copy content from http://agileappslive.info/wiki/download/training/AddUpdate.jsp
- Personalize debug msgs, namespace, & target Class

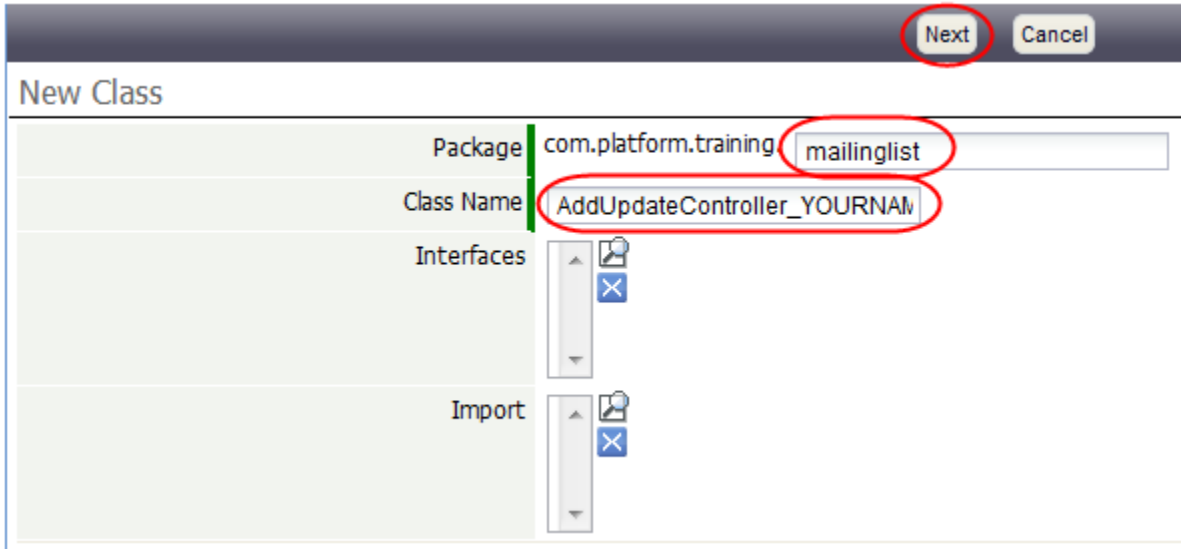
```
2  <%
3      java.util.HashMap valueMap = null;
4      if (controllerResponse == null) {
5          // Prevent null pointer errors the first time the page is called.
6          Logger.info("___New Run Started___","YOURNAME JSP");
7          valueMap = new java.util.HashMap();
8       } else {
9          // Get the parameters passed by the controller
10         valueMap = (java.util.HashMap)controllerResponse.getData();
11     }
12     String page_control =
13         valueMap.get("control") == null ? "init" : (String)valueMap.get("control");
14     Logger.info("Page running: " + page_control, "YOURNAME JSP");
15
32
33 <body><div style="margin-left:1em">
34 <h2>Add a Mailing List Entry</h2>
35 <p>
36     <!-- ADD YOUR NAME OR INITIALS TO ACCESS YOUR CLASS -->
37     <form name="mainForm"
38       action="/networking/controller/com/platform/acme/mailinglist/AddUpdateController/YOURNAME"
39       method="POST">
40         <table>
```

JSP Page Highlights

- Obtaining values passed by controller
  - java.util.HashMap valueMap =
    (java.util.HashMap)controllerResponse.getData();
- Form display controlled by passed values
  - <% if (page_control == "record_added") { %>
    . . .
- Form contents passed as name/value pairs
  - <input type="text“ name="contact_name"/>
  - <input type=“submit“ name=“action“ value=“Update"/>
  - <input type=“hidden“ name=“record_id"/>

Create the Controller Class (1)

- **{gear} > Developer Resources > Classes > [New Class]**



  Package: **mailinglist**
- Name: **AddUpdateController_YOURNAME** (*no extension*)
- **[Next]**

Create the Controller Class (2)

- Copy content from
  http://agileappslive.info/wiki/download/training
  --AddUpdateController.java
- Personalize the namespace, class, messages, & target page:

```
1  package com.platform.acme.mailinglist;
2
3  import com.platform.api.*;
4  import java.util.*;
5
6  public class AddUpdateController_YOURNAME implements com.platform.api.Controller
7  {
8      /**
9       * Substitute project name or your name to personalize the error message.
10      */
11     public void debug(String text) {
12         Logger.info(text, "YOURNAME Class");
13     }
14     public void errMsg(String text, String exceptionMsg) {
15         debug(text + ":\n  " + exceptionMsg);
16     }
17
18    /**
19     * Personalize the target page below, as well.
20     */
21     public ControllerResponse execute(HashMap valueMap) throws Exception
22     {
23         String page_action = (String)valueMap.get("action");
24         debug("Code is running: " + page_action);
25
26         ControllerResponse cr = new ControllerResponse();
27         cr.setTargetPage("AddUpdate_YOURNAME.jsp");
```

Controller Class Highlights (1)

- Getting Parameters passed by a Form

```
19      public ControllerResponse execute(HashMap valueMap) throws Exception
20      {
21          String page_action = (String)valueMap.get("action");
```

- Specifying target page

```
24          ControllerResponse cr = new ControllerResponse();
25          cr.setTargetPage("AddUpdate.jsp");
```

- Passing Values

```
44                  valueMap.put("control", "update_query");
45              }

55          }
56          cr.setData(valueMap);
57          return cr;
58      }
```

Controller Class Highlights (2)

- SQL Search in the `find()` function

```
71          String query = "SELECT id, contact_name FROM Mailing_List "
72                       + "WHERE contact_email = '" + email + "'";
73          Result result = Functions.execSQL(query);
```

- Other APIs in `add()` and `update()` functions

```
91      public Result add(HashMap valueMap) throws Exception
92      {
93          Result result = null;
94          try {
95              Parameters params = Functions.getParametersInstance();
96              params.add(valueMap);
97              result = Functions.addRecord("Mailing_List", params);
```

```
108     public Result update(HashMap valueMap) throws Exception
109     {
110         Result result = null;
111         try {
112            Parameters params = Functions.getParametersInstance();
113            params.add(valueMap);
114            String record_id = (String) valueMap.get("record_id");
115            result = Functions.updateRecord("Mailing_List", record_id, params);
```

Controller Class Highlights (3)

- execute() method = “Dispatch Table”

```
31          if ( page_action.equals("Add") ) {
32              String email = (String)valueMap.get("contact_email");
33              HashMap values = find(email);
34              if (values == null) {
35                  // No existing record. Add the new one
36                  add(valueMap);
37                  valueMap.put("control", "record_added");
38              } else {
39                  // Email address already exists.

44                  valueMap.put("control", "update_query");
45              }
46          } else if ( page_action.equals("Update") ) {
47              update(valueMap);

49              valueMap.put("control", "record_updated");
50          } else {
51              // User clicked "Cancel" after an update query.

54              valueMap.put("control", "init");
55          }
56          cr.setData(valueMap);
57          return cr;
58      }
```

Try it Out

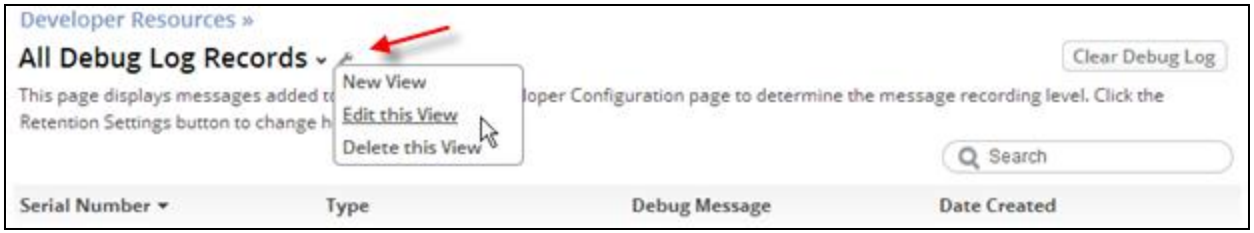
- Visit the page:
  https://{domain}/networking/pages/AddUpdate_*YourName*.jsp

- Experiment
  - Add a record, and view the data in the GUI *(refresh)*
  - Update a record *(refresh)*
  - Cancel an update
  - Add the page as a Web Tab
    http://agileappslive.info/wiki/Pages#Display_a_Page_as_a_Web_Tab

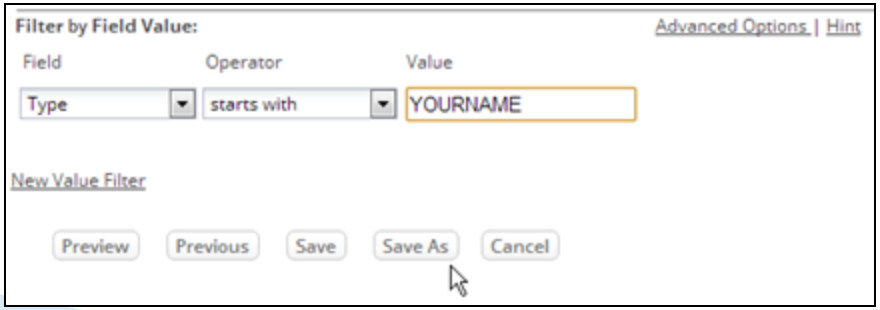
- Learn More
  - http://agileappslive.info/wiki/
    - Lab_Exercises#Basic_Programming_Exercises (from ground up)
    - HowTo:Create_a_JSP_Page_and_Java_Controller (this application)

View the Debug Log

- Go to the Log
  - {gear} > Developer Resources > Debug Log *(refresh)*
- Edit the view (hover over the wrench icon)



  - *Filter tab:* **Field** **Operator** **Value**
    Type starts with YourName
  - **[Save As]**



Extended Debugging

- Uncomment to add HashMap to Debug Log:

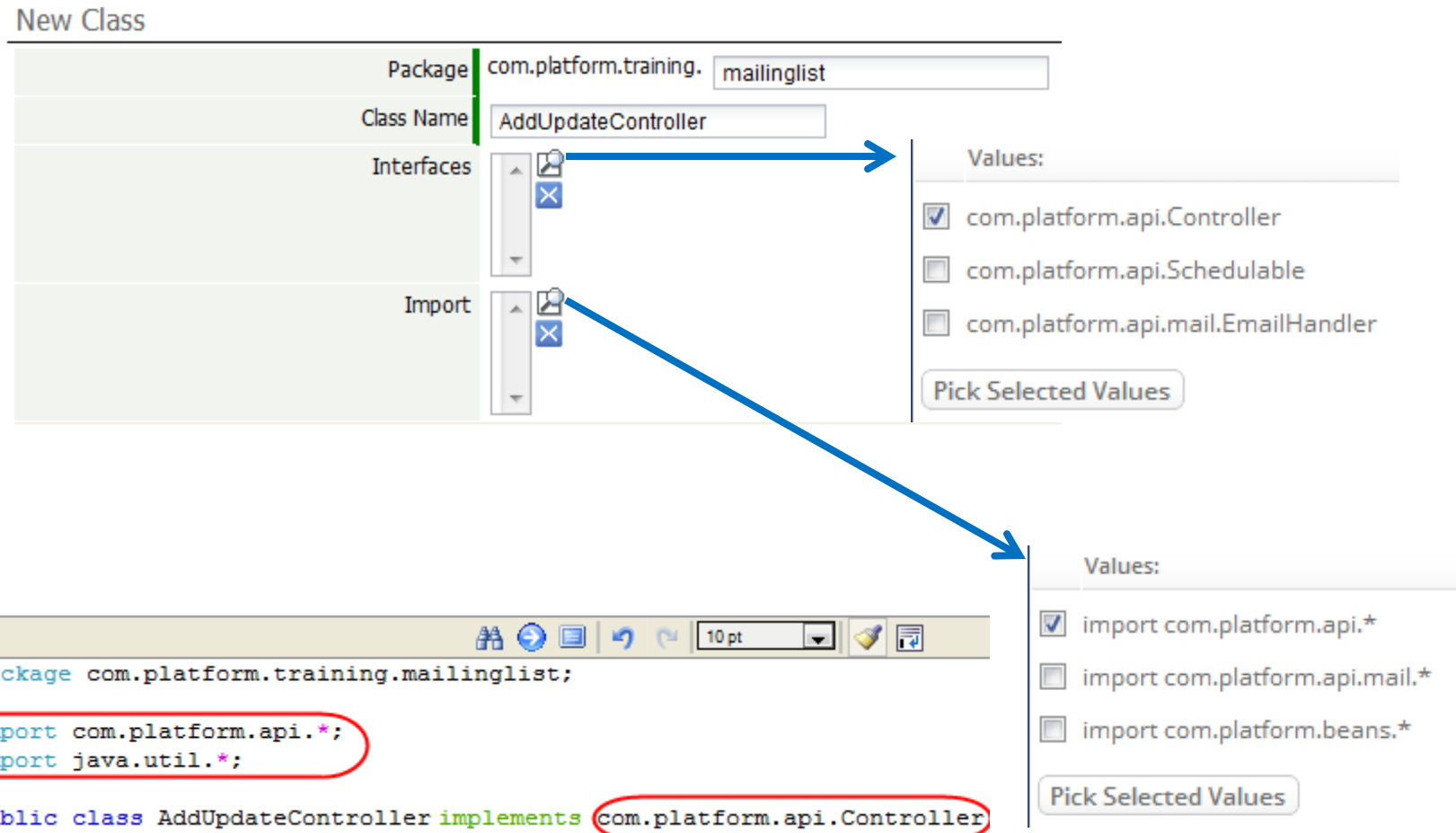
```
21    public ControllerResponse execute(HashMap valueMap) throws Exception
22    {
23        String page_action = (String)valueMap.get("action");
24        debug("Code is running: " + page_action);
25
26        ControllerResponse cr = new ControllerResponse();
27        cr.setTargetPage("AddUpdateYOURNAME.jsp");
28
29        // Uncomment this to see all incoming parameters
30        //debug("valueMap= " + valueMap);
```

- See the entry in the Debug Log (*refresh*)

- Overview
  - Where to find Information
  - Application Overview
- Hands On
  - A small app w/a JSP page and a controller
  - Additional information for Java developers ←
- Demos
  - Add JavaScript to an Object Form
  - Using the REST APIs
- Summary of Tools

Finding Interfaces & Imports

- When creating a new class:



Logging and Debugging

- Logger utility class

  /javadocs/com/platform/api/Logger.html

- Log messages at specified levels
  - Debug, Error, Fatal, Info, Trace, Warn

- More:

  http://agileappslive.info/wiki/Java_Debugging_Tips

  http://agileappslive.info/wiki/Debug_Log#Working_with_the_Debug_Log

Understanding Governors

- Limits on Resource Usage
  - Memory / String Heap / Statements executed
- Defaults built into the installation
- Can be set for each tenant (on-premise)
  - (helpful if you're billing by usage)
- Can be disabled (on-premise)
  - not recommended, except for development

- Overview
  - Where to find Information
  - Application Overview
- Hands On
  - A small app w/a JSP page and a controller
  - Additional information for Java developers
- Demos
  - Add JavaScript to an Object Form ←
  - Using the REST APIs
- Summary of Tools

Demo: Add JavaScript to a Form

- {gear} > Objects > Mailing List
- Forms > Default Layout
  - Field scripts (Hover -> Scroll icon: on change, on focus)
- > [Form Scripts] > [Edit]
  - On Load, On Save, Functions
- On Save Script:
  - ```
    if (getTextFieldValue(_sdForm, "contact_name") == "") {
       alert("Name is required");
    }
    ```
- Test it
    ```

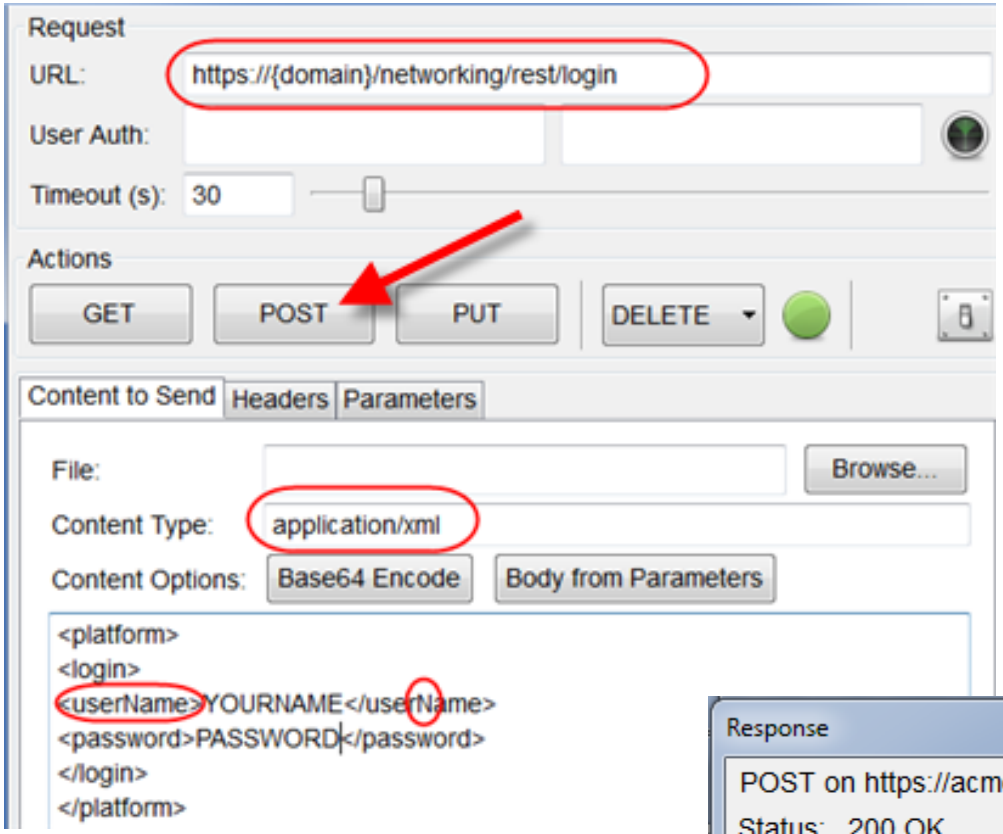
Built-in JavaScript Libraries

- jQuery library
  - Built-in (best way access & modify the DOM)
  - can also use jQuery components
  - and JSON functions
- JavaScript APIs
  http://agileappslive.info/wiki/JavaScript
- AJAX
  - Invoke REST APIs from JavaScript
  - Get back XML or JSON, as desired
  - …/wiki/HowTo_Guides#Build_Custom_Web_Pages

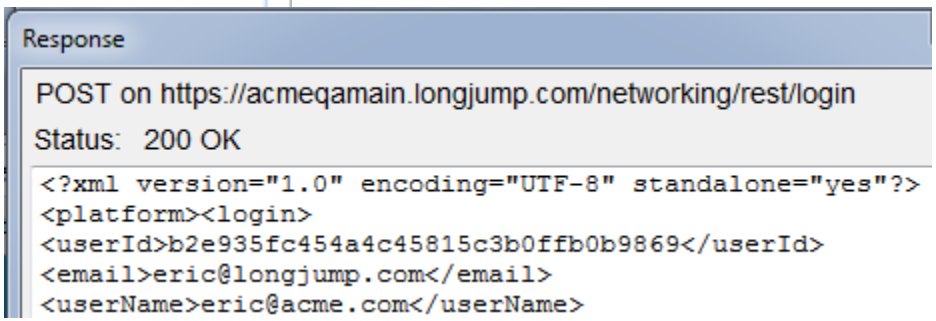
- Overview
  - Where to find Information
  - Application Overview
- Hands On
  - A small app w/a JSP page and a controller
  - Additional information for Java developers
- Demos
  - Add JavaScript to an Object Form
  - Using the REST APIs ←
- Summary of Tools

Login with REST

- Firefox > Tools > Poster (plugin)



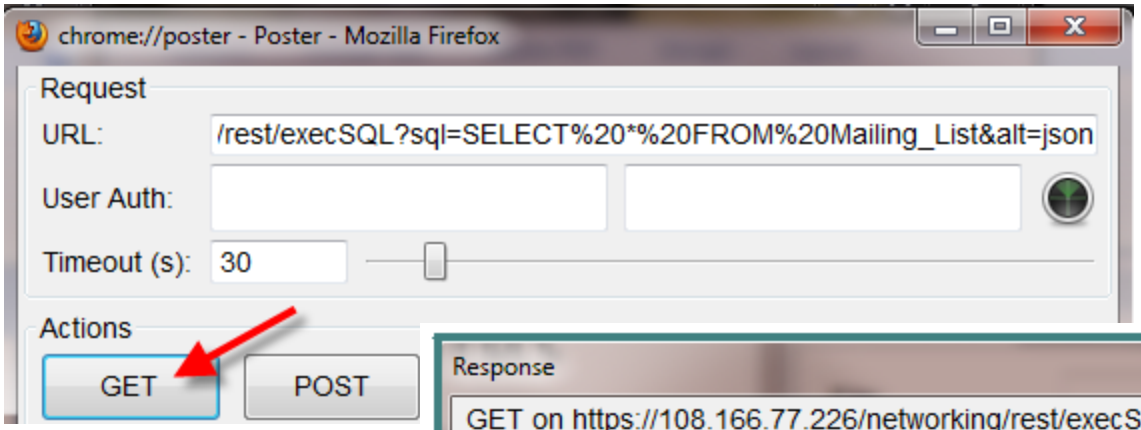
- XML Response (default)



Retrieve Data in JSON Format

/rest/execSQL?sql=**SELECT%20*%20FROM%20Mailing_List&alt=json**

Spaces encoded



Response

GET on https://108.166.77.226/networking/rest/execSQL?sql=SELECT%20*%20FROM%20Mailing_List

Status: 200 OK


```
{"platform": {
  "message": {
    "code": "0",
    "description": "Success"
  },
  "record": [
    {
      "contact_email": "fred@flintstone.com",
      "contact_name": "",
      "created_id": "78d817a52a804e349d40e9049f3ed114",
      "date_created": "2012-07-30T22:04:21.000Z",
      "date_modified": "2012-07-30T22:05:00.000Z",
      "id": "206587936",
      "modified_id": "78d817a52a804e349d40e9049f3ed114",
      "object_id": "93e571329ad84ffc81962b03f43c39d7",
      "owner_id": "78d817a52a804e349d40e9049f3ed114",
```

Encoding a Query String in Java

```
import com.platform.api.utility.Base64Codec;

String sql = "SELECT * FROM Mailing_List"

Base64Codec codec = new Base64Codec();
String encodedSQL = codec.encode(sql);

String base_url = ".../rest/execSQL?sql="
String url= base_url + encodedSQL;
```

- Overview
  - Where to find Information
  - Application Overview
- Hands On
  - A small app w/a JSP page and a controller
  - Additional information for Java developers
- Demos
  - Add JavaScript to an Object Form
  - Using the REST APIs
- Summary of Tools ←

Eclipse Plugin (Pages & Classes)

- Code completion, error reporting, highlighting
  - Download & installation instructions:
    http://agileappslive.info/wiki/Eclipse_Plug-In
  - Create a Project (give name, URL, username, password)
    File > New > LongJump Project
  - View
    Window > Open Perspective > Other > LongJump
  - Modify Login Settings
    Rt click (project) > **Properties > LongJump Authentication**
  - Synchronize
    Rt click (project, folder, or file) > **LongJump Server >**
    Save on Server - Refresh from Server - Delete from Server
    Refresh Libraries (for new APIs after a platform upgrade)

JSP Pages and SQL

- NotePad++ for JSP pages (great highlighting)
  --Use in conjunction with Eclipse Plugin

```
1   <%-- Mailing List Demo --%>
2   <%
3       java.util.HashMap valueMap = null;
4       if (controllerResponse == null) {
5           // Prevent null pointer errors the first time the page is called.
6           Functions.debug("YOURNAME: ___New Run Started___");
7           valueMap = new java.util.HashMap();
```

```
23      }
24      //Controller response can also pass a message:
25      // String message = controllerResponse == null? "" : controllerResponse.getMessage();
26  %>
27
28  <head>
29    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
30    <title>Add/Update Email Address</title>
31  </head>
```

- Built-in SQL Browser to test SQL statements
  http://agileappslive.info/wiki/SQL_Browser

Browser Tools

- Poster (Firefox plugin) for REST exploration
- FireBug (Firefox) or Dev'r Tools (Chrome)
  - GUI-select to view page components
  - View page interactions in the console

Backup Strategies

- Version-controlled files for pages & classes
  - Eclipse Plugin: platform ←→ files
  - Tortoise SVN: files ←→ shared repository
- Use a Package to Create a Snapshot
  - Publish, download, & add date
  - Remove date to re-install
- Sandboxes (production, Q/A, dev)

The End of the Beginning…

Resources

- http://agileappslive.info/wiki/download/training/PlatformDevelopment101.pdf (this talk, code files)
- http://agileappslive.info/wiki/HowTo:Create_a_JSP_Page_and_Java_Controller (writeup)