spinspire

Healthcare Portal With Drupal

Sergey Cheban
Micah Forster

Sergey Cheban

Software Engineer/Consultant

Born and raised in Ukraine

Micah Forster

Software Engineer/Consultant

Loves implementing new technologies

Who we are...

- Web & Mobile development company
- Based out of Jacksonville, FL
- From corporate to medium sized clients
- Over a dozen large Drupal projects - built from scratch as well as migrated to Drupal

Agenda

Healthcare Insurance Portal in Drupal

- Site features
- Technologies used
- Challenges
- What we learned
- Future developments (AngularJS)

Let’s Begin

Site Features

- Consumer Information
- Plan Listings
- Enterprise Service Integration
  - Pricing engine, physician directory, Healthcare Exchange (ObamaCare)
- Dual DB Schema
- Shopping cart, plan selection wizard
- Site packaging and deployment

Technologies Used

- LAMP Stack, Drupal 7
- Contrib Modules
  - VBO, Views, Feeds, Features, GA
- DB2 PHP Driver
- Rsync for file system replication
- Drush automation, shell scripts
  - drush dl drupal-7.x
  - drush site-install standard --account-name=admin --account-pass=admin --db-url=mysql://YourMySQLUser:RandomPassword@localhost/YourMySQLDatabase
  - drush -y en module_name

Custom Modules

- Shopping Cart, Physician Directory
- Healthcare marketplace interface
- Views plugins to merge rates into views
- Campaign tracking, custom landing pages
- Custom touchpoint logging

Plan Listing Pages

- VBO & Contextual Filters for plan comparison
- Custom templates for pages
  - ex: custom/plan/list → page--custom--plan--list.tpl.php
- jQuery filtering by plan benefits
- Physician directory lookup
- Various shopping wizards
- Rates integration

Plan Listing Pages Contin.

- Integrating prices into Drupal plan listing view

```
function post_execute(&$result) {          //result is the array of objects in the view
  $demographics = $_SESSION[‘demographics’];
  $plans = array(‘2’,’3’,’14’);
  $rates = rate_request($demographics,$plans,'health');   //rates request to pricing engine
       foreach($result as $index => $values) {   //values points to the plan object in the view
         $values->{$alias} = $_SESSION[‘health’][‘2’][‘plan_price’];
       }
}
```

Enterprise Services

- Pricing Engine
  - Prices vary based on consumer info & plan type
  - XML request and response
- Physician directory lookup
  - Use Hessian web service to pull list of doctors
  - Use google geolocation API
- HealthCare.gov Interface

Pricing Engine

- Enterprise service that returns prices for plans based on the following conditions
  - User information (demographics)
  - Plan ids, plan type
  - Coverage date
- Drupal_http_request
  - Sends xml to enterprise pricing engine, and returns xml with prices for each plan

Pricing Engine cont…

- Sample pricing request code snippet:

```
$plans_arr = array();     //stores the plan ids with their respective prices
$prices_xml = drupal_http_request($url, array(
                        ‘method’ => ‘POST’,
                         ‘data’ => $request_xml,       //custom with user info, and plans id, etc.
                         ‘headers’ => array(
                                ‘application/x-www-form-urlencoded’,
                            ),
                         );
$prices_array = _parse_price_xml_to_array($prices_xml);
```

Pricing Engine cont...

Here is a little snippet of a sample price request call

```
$prices_array = _parse_price_xml_to_array($prices_xml);
store_prices($prices_array);

function store_prices($arr) {
    $plans_arr = array();
    foreach($arr['health'] as $plan) {  //iterate over each plan in the xml response
        $plan_id = $plan['plan_id'];
        $plan_premium = $plan['premium'];
        $plans_arr[$plan_id] = $plan_premium; //load $plans_arr with plan id and premium
    }
    $_SESSION['health'] = $plans_arr; //place rates array into session for plan list integration
}
```

Interfacing with HealthCare.gov

- Drupal menu path for response/request
- SOAP web service calls
- SAML generation/validation (DataPower)
- Subsidy eligibility request
- Integration of subsidy amount with Drupal view

HealthCare.gov contin.

- SAML Data Encryption before sending user to Exchange site
  1. Send user to HealthCare.gov through DataPower
  2. DP generates the SAML request
  3. DP encrypts the user data and signs with a certificate
  4. DP sends the encrypted SAML to HealthCare.gov API
  5. HealthCare.gov validates the SAML and user begins application for subsidy

HealthCare.gov contin.

- SAML Data validation after user returns from Exchange

1. DP validates the SAML response from HealthCare.gov
2. Returns the SAML assertion to our application
3. We parse the assertion and pull eligibility and user data
4. Validate the assertion for any exceptions
    a. User did not finish eligibility application
    b. Out of state zipcode and others..

4. Make eligibility request for data from HealthCare.gov to get subsidy information

HealthCare.gov contin.

Eligibility Request

```
//get response back from Healthcare.gov and send to DP to validate if the applicant is eligible
$eligibility_status = user_eligible($saml_response);

function user_eligible($saml_response) {
  $ch = curl_init();
  curl_setopt($ch, CURLOPT_POST, 1);
  curl_setopt($ch, CURLOPT_POSTFIELDS, $saml_response);
  $response = curl_exec($ch);
  curl_close();
  return $response;
}
```

HealthCare.gov contin.

Eligibility Request cont.

```
$obj_eligible_response = simplexml_load_string($eligibility_status); //convert xml to object
$arr_eligible_response = (array) $obj_eligible_response; //convert object to array
$applicant_eligibility = get_applicant_eligibility($arr_eligible_response); //get eligibility results

function get_applicant_eligibility($request) {
 //make another SOAP call to DP to request the eligibility results for applicant
 $response = call_dp_for_eligibility_results($request);
 return $response
}
```

HealthCare.gov contin.

Eligibility Response

- Store user demographics and subsidy in Drupal session

```
$applicant_eligibility = get_applicant_eligibility($arr_eligible_response); //get eligibility results

foreach($applicant_eligibility as $consumer) {     //iterate over applicants
  if($consumer['type'] == $primary) {   //check if applicant is primary
    $_SESSION['qualified_subsidy'] = $consumer['subsidy'];   //use primary subsidy for plan list
  }
  $_SESSION['user_demo'][] = $consumer;
}
```

Physician Directory Lookup

- Hessian uses binary protocol for web services
- Don’t have to worry about your SOAP templates and XML structure, hessian takes care of it

```
include_once( 'HessianClient.php' );
$testurl = 'http://localhost:8080/hessian/service/doctorService';
$proxy = &new HessianClient($testurl);
$search_criteria->name = ‘somename’;
$search_criteria->zipcode = ‘22333;
$doctors = $proxy->findDoctors($search_criteria, new stdClass()); //Remote method
$form_state[‘rebuild’] = TRUE; //rebuilds the search form for results
$form_state[‘storage’][‘doctors’] = $doctors;
```

For more info on Hessian: http://hessianphp.sourceforge.net/index.php

Dual DB Schema

- Allows us to separate consumer data (custom tables) from Drupal tables
  - Configure settings.php file
  - Copy & prefix the Drupal DB API functions in a custom module
  - db_set_active() to set active schema
  - set drupal variable to turn **on/off** the dual schema functionality

Dual DB Schema Contin.

- Update settings.php file

```
$databases['second_db'] = array('default' => array(
        'driver' => 'mysql',
        'database' => 'Name_of_your_Database',
        'username' => 'Database_username',
        'password' => 'some-password',
        'host' => 'mysql-server-host-name',
        'port' => 3360
));
```

Dual DB Schema Contin.

- Prefix db_select and add to your custom module

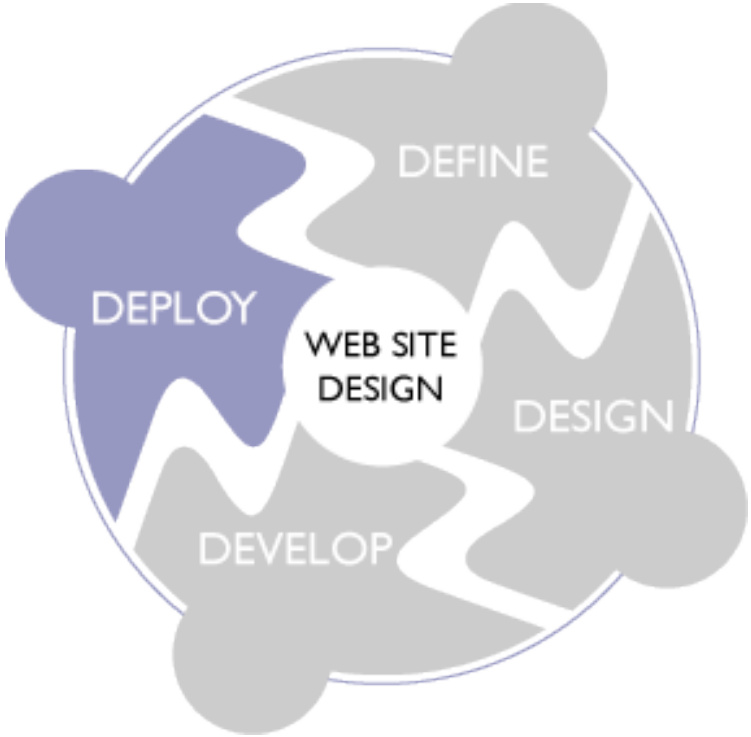
```
function prefix_db_select($table, $alias = NULL, array $options = array()) {
    db_set_active(variable_get('second_db', 'default'));
    if (empty($options['target'])) {
        $options['target'] = 'default';
    }
    $result = Database::getConnection($options['target'])->select($table, $alias,  $options);
        db_set_active('default');
        return $result;
}
```

Shopping Cart

- Save shopping cart for later
- Disclaimers using token replacement
  - check_markup(‘[content: /url/to/content: body]’, ‘full_html’);
- JSON posting to apply application

```
$json = ‘{“products”: [{“plnId”:”1234”,”effDate”:”04/01/2014”,”type”:”health”}]}’;
drupal_http_request($url, array(
    ‘method’=>’POST’, ‘data’ => $json,
    ‘headers’ => array(
            ‘application/x-www-form-url-encoded’)
    ));
```

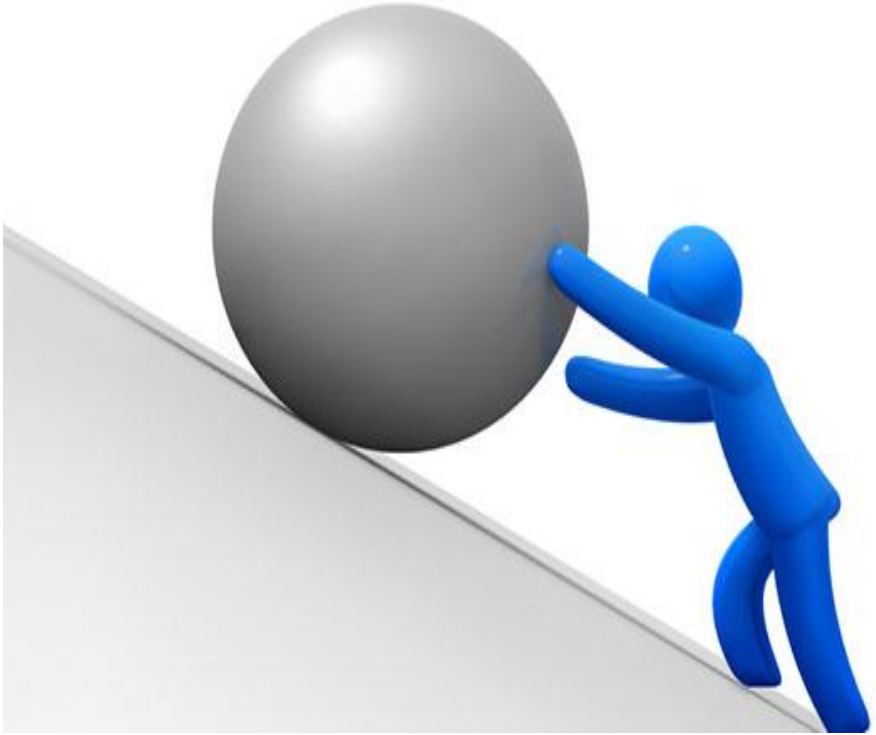
Site Packaging & Deployment



Site Packaging & Deployment

- Two production pipelines (Light & Dark)
- Use custom build tool for code deployment and content files
- Use features module to package your site and it’s configurations
- Custom tool to redirect traffic to other production environment with new changes.

Challenges



Challenges

- Service Integrations
- PI/PHI Regulations (HIPAA)
- Adhering to UX prototypes
  - Views with .tpl files and jQuery
- Development environment lockdown
- SVN instead of Git

What we learned

- Use Drupal as intended
  - Use a healthy combination of contrib and custom modules - 60% Contrib and 40% Custom
- Use drupal API functions if possible instead of pure PHP
  - Will save you headaches with security
- Drush saves time and effort!!!

Drupal with Angular?

Future Developments

AngularJS

- Client-side JavaScript MVC Framework
- Islands of application in the Ocean of content
- Site Optimization
  - No full page reloads
  - Cached template partials
  - Fewer server requests
- RESTful Drupal back-end

Questions?

Sergey Cheban - sergey@spinspire.com
Micah Forster - micah@spinspire.com

SpinSpire.com