

Service Component Architecture

Web services and SOA with the
Service Component Architecture

Graham Charters

gcc@php.net (aka charters@uk.ibm.com)

IBM

March 16, 2007



Agenda

- Background - the problems SCA addresses
- Writing and consuming simple Services
- Working with data structures
- Switching protocols
- Different styles of Web services
- Writing a custom protocols
- Wrap-up



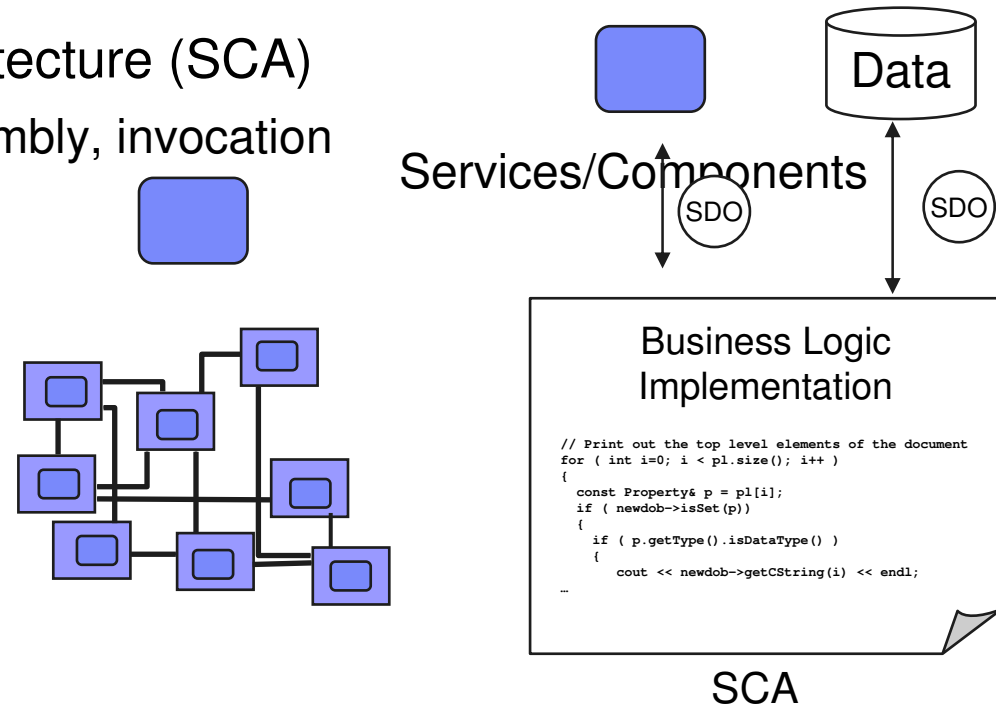
What's the problem?

- Provide simplicity and consistency in PHP applications when integrating Rich Internet Applications (Web2.0?) and other services (Enterprise SOA, Web ...)
 - Describe services so that others can use them
 - Use services that others have described
 - Do this flexibly and simply so that services can be reorganized and reused as requirements change
- We want to be able to do all this within PHP
 - write a script, drop it into Apache and away you go.

Service Component Architecture (SCA)

Given some business logic, how do you make that as useful and reusable as possible?

- Service Component Architecture (SCA)
 - Service definition, assembly, invocation and mediation

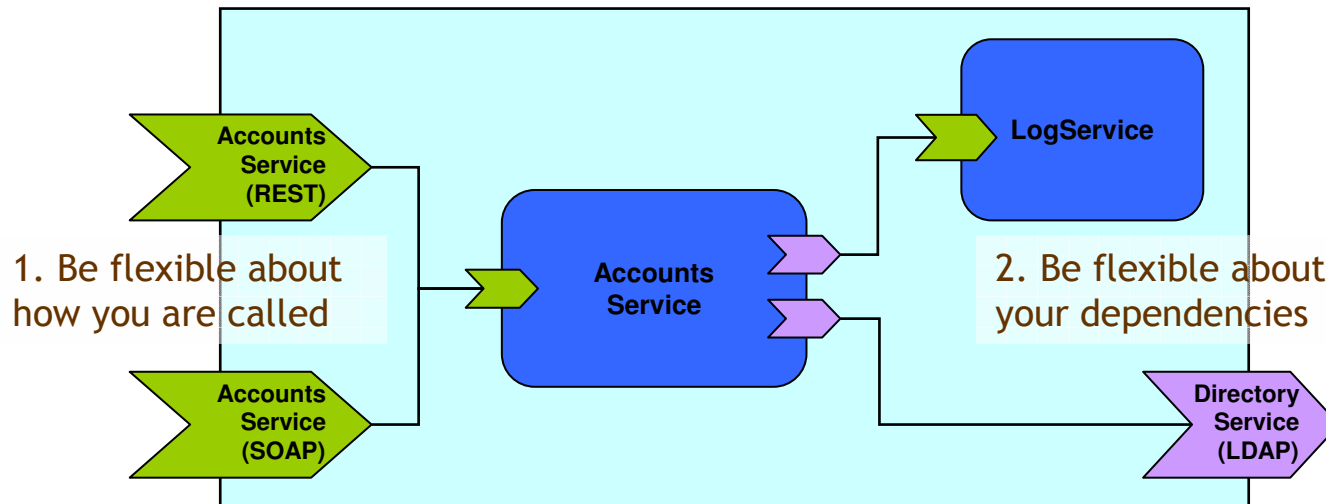


- Service Data Objects (SDO)
 - Data transfer between services and to/from a persistent store
 - SCA can work without SDO but SDO adds a common view of data

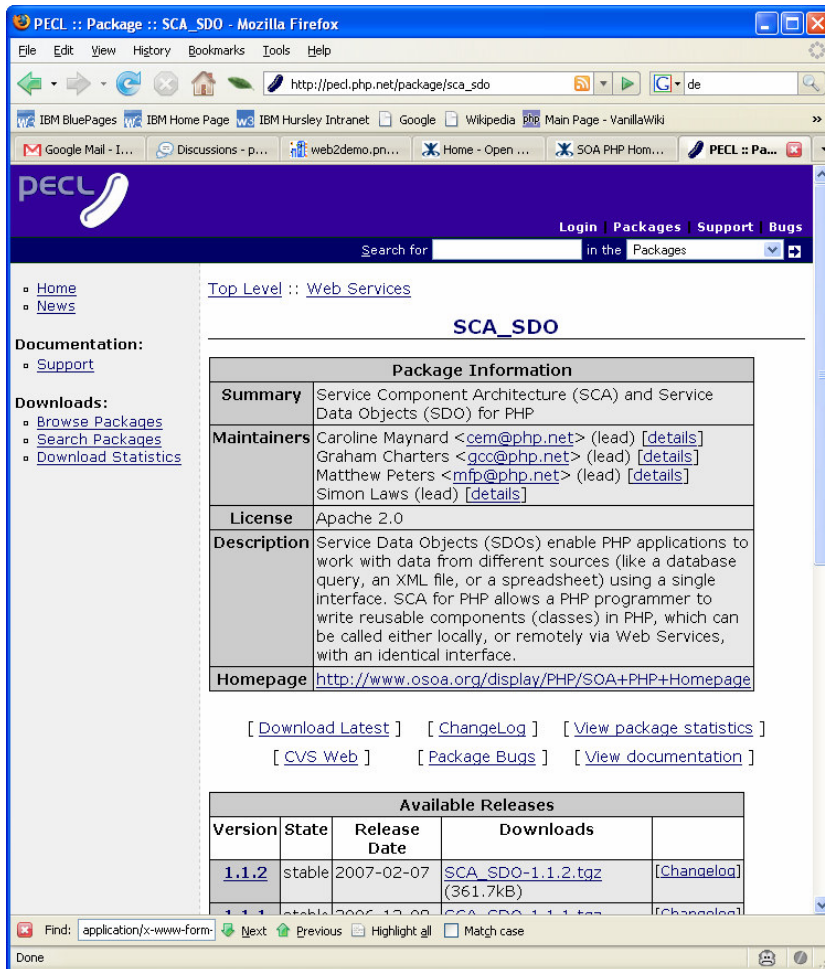
Wiring

How Does SCA Help Solve Our Problem?

- It's all about “separation of concerns”
- Be flexible about how you are called
 - Expose as many ‘bindings’ as needed – make sure your business logic does not need to know how it was called
- Be flexible about your dependencies
 - Define your service interface dependencies – make sure your business logic does not need to know how to resolve these
 - Ideally get something else to “wire up” the components (Inversion of Control; Dependency Injection patterns)



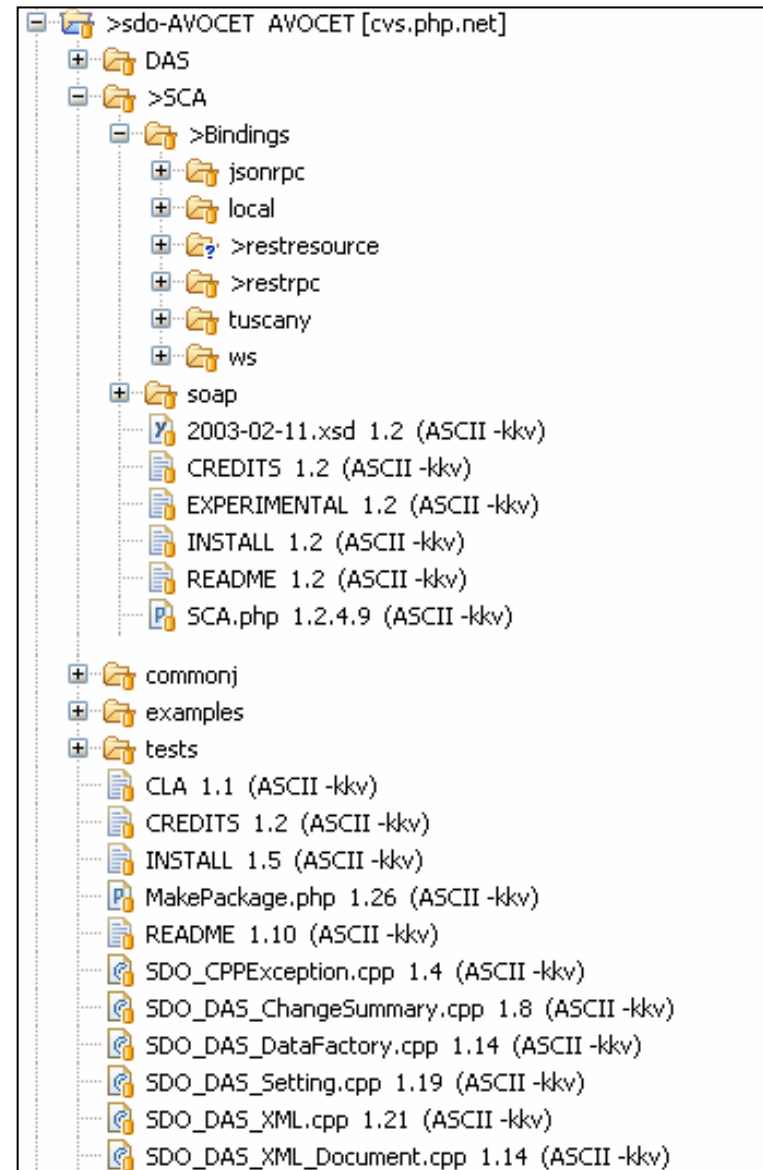
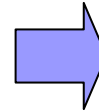
The SCA_SDO PECL Extension



The screenshot shows the PECL website page for the SCA_SDO package. The page includes a navigation menu, a search bar, and a main content area with the following sections:

- Documentation:** Support
- Downloads:** Browse Packages, Search Packages, Download Statistics
- Package Information:**
 - Summary:** Service Component Architecture (SCA) and Service Data Objects (SDO) for PHP
 - Maintainers:** Caroline Maynard <cem@php.net> (lead) [details], Graham Charters <gcc@php.net> (lead) [details], Matthew Peters <mfp@php.net> (lead) [details], Simon Laws (lead) [details]
 - License:** Apache 2.0
 - Description:** Service Data Objects (SDOs) enable PHP applications to work with data from different sources (like a database query, an XML file, or a spreadsheet) using a single interface. SCA for PHP allows a PHP programmer to write reusable components (classes) in PHP, which can be called either locally, or remotely via Web Services, with an identical interface.
 - Homepage:** <http://www.osoa.org/display/PHP/SOA+PHP+Homepage>
- Available Releases:**

Version	State	Release Date	Downloads	
1.1.2	stable	2007-02-07	SCA_SDO-1.1.2.tgz (361.7kB)	[ChangeLog]
1.1.1	stable	2006-12-08	SCA_SDO-1.1.1.tgz	[ChangeLog]



The screenshot shows a file tree view for the SCA_SDO package. The tree structure is as follows:

- >sdo-AVOCET AVOCET [cvs.php.net]
 - DAS
 - >SCA
 - >Bindings
 - jsonrpc
 - local
 - >restresource
 - >restrpc
 - tuscany
 - ws
 - soap
 - 2003-02-11.xsd 1.2 (ASCII -kqv)
 - CREDITS 1.2 (ASCII -kqv)
 - EXPERIMENTAL 1.2 (ASCII -kqv)
 - INSTALL 1.2 (ASCII -kqv)
 - README 1.2 (ASCII -kqv)
 - SCA.php 1.2.4.9 (ASCII -kqv)
 - commonj
 - examples
 - tests
 - CLA 1.1 (ASCII -kqv)
 - CREDITS 1.2 (ASCII -kqv)
 - INSTALL 1.5 (ASCII -kqv)
 - MakePackage.php 1.26 (ASCII -kqv)
 - README 1.10 (ASCII -kqv)
 - SDO_CPPEException.cpp 1.4 (ASCII -kqv)
 - SDO_DAS_ChangeSummary.cpp 1.8 (ASCII -kqv)
 - SDO_DAS_DataFactory.cpp 1.14 (ASCII -kqv)
 - SDO_DAS_Setting.cpp 1.19 (ASCII -kqv)
 - SDO_DAS_XML.cpp 1.21 (ASCII -kqv)
 - SDO_DAS_XML_Document.cpp 1.14 (ASCII -kqv)

PECL : http://pecl.php.net/sca_sdo

Mail : <http://groups.google.co.uk/group/phpsoa>

Web : <http://www.osoa.org/display/PHP>

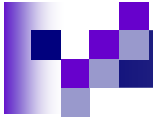
Specs: Google for "osoa"

How To Write An SCA Service

- It's largely just a PHP class
- 'includes' the SCA runtime (must be last)
- Uses PHPDocumentor style annotations to *declare* capabilities and dependencies
- MUST assume pass-by-value
- But other than this, job done!
 - Need to make sure SCA_SDO PECL extension is loaded
 - Then drop this file into Apache
 - You have an EmailService exposed as a web service

```
<?php
include 'SCA/SCA.php';

/**
 * Service for sending emails
 *
 * @service
 * @binding.ws
 */
class EmailService {
    ...
    /**
     * Send a simple text email
     *
     * @param string $to The "to" email address
     * @param string $from The "from" email address
     * @param string $subject The subject of the email
     * @param string $message The email message
     * @return boolean
     */
    public function send($to, $from, $subject, $message) {
        ...
    }
}
?>
```



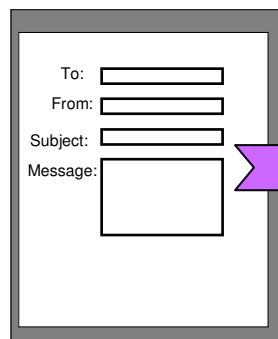
How to Consume a Service from a PHP Script

```
$to    = $_POST['to'];  
$from  = $_POST['from'];  
$subject = $_POST['subject'];  
$message = $_POST['message'];  
  
include 'SCA/SCA.php';  
  
$email_service = SCA::getService('EmailService.wsdl');  
$success = $email_service->send($to, $from, $subject, $message);
```

A Simple Email Form

- Write an SCA Component
- Expose it as a Web service
- Generate the WSDL
- Consume it in a client script

[EmailClient.php](#)



To:
From:
Subject:
Message:

[email_form.html](#)

Email
(SOAP)

Email
(SOAP)

Email

[EmailService.php](#)



Consuming Services From A Component

```
/**
 * Service for sending emails (supports shortnames)
 * @service
 */
class ContactEmailService {

    /**
     * @reference
     * @binding.ws ./EmailService.wsdl
     */
    public $email_service;

    /** ... */
    public function send($to, $from, $subject, $message) {

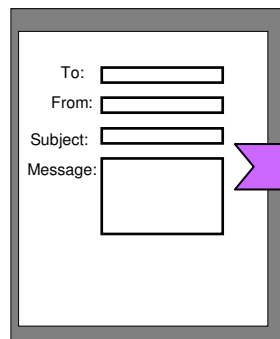
        /**
         * // a proxy to the service is 'injected' so we
         * // can just use it...
         * $this->email_service->
         *     send($to, $from, $subject, $message);
         */

        ...
    }
}
```

Add A Service With A Service Reference

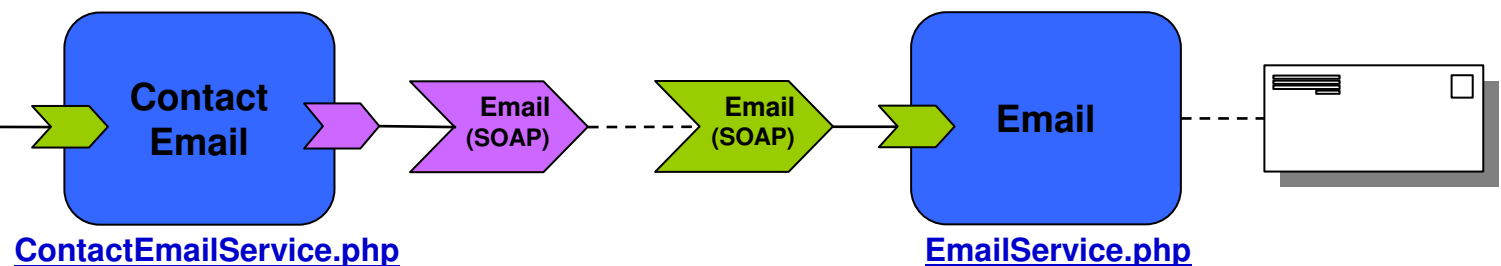
- Write a new SCA Component
- Expose it as a Local service
- Have it reference our Email service
- Consume it in our client script

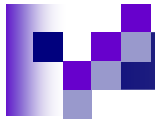
[EmailClient.php](#)



To:
From:
Subject:
Message:

[email_form.html](#)





Handling data structures

- Not all services exchange primitives!
- SCA uses Service Data Objects to handle data structures
- SDO requires a description of the data structures
 - Currently XML schema
 - Future: annotated PHP classes

Annotations for data structures

- Three steps to providing a service with complex types
 1. Create a schema for the data structure
 2. Annotate class to say which types are used
 3. Document the class methods to show where the types are used

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://example.org/contacts">

1 <element name="contact">
  <complexType>
    <sequence>
      <element name="shortname" type="string" />
      <element name="fullname" type="string" />
      <element name="email" type="string" />
    </sequence>
  </complexType>
</element>

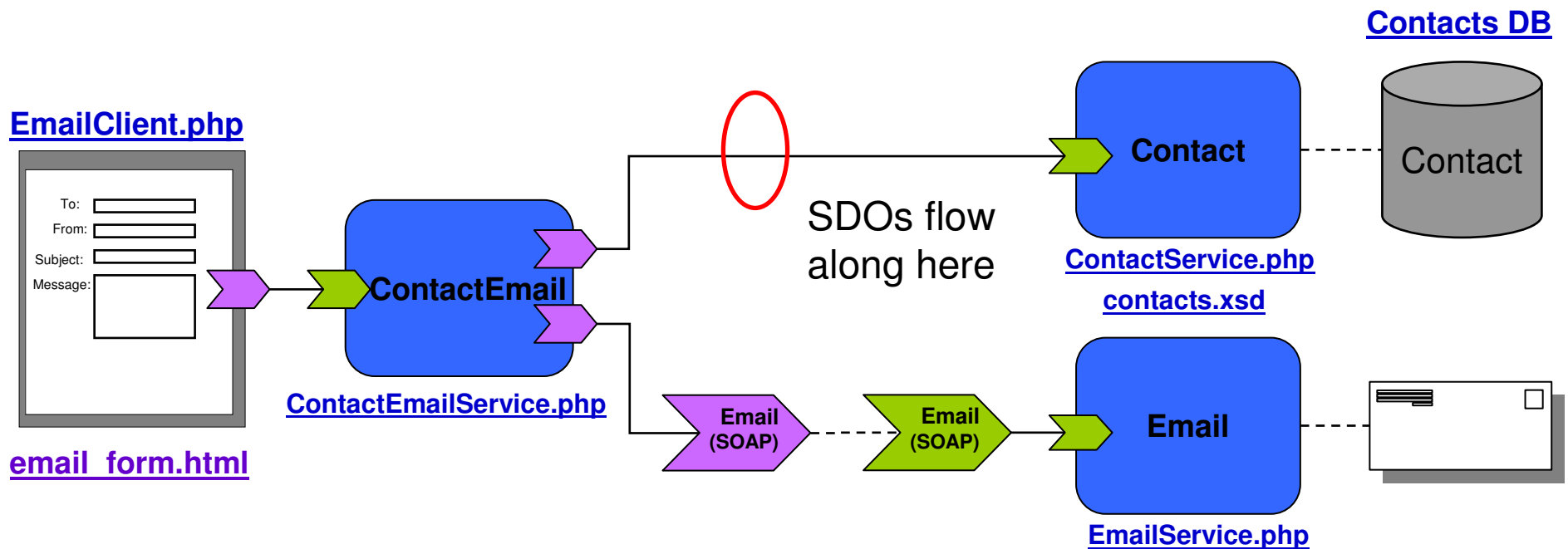
</schema>
```

```
/**
 * Service for managing email contacts
 * @service
2 * @types http://example.org/contacts contacts.xsd */
class ContactService {

  /**
   * Retrieve contact details
   *
   * @param string $shortname Short name of the contact
   * @return contact http://example.org/contacts The contact
3 */
  public function retrieve($shortname) {
    $contact = SCA::createDataObject(
      'http://example.org/contacts', 'contact');
    ...
    return $contact;
  }
}
```

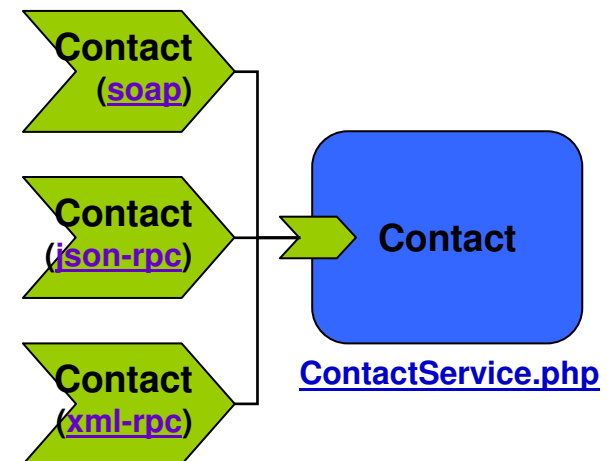
Separate Out The Contacts Functionality

- Create a new Contact service
- Design the data structure to represent a contact
- Adding data structures to the contact service
- Reference the Contact service from the ContactEmail service
- Use data structures in the ContactEmail service



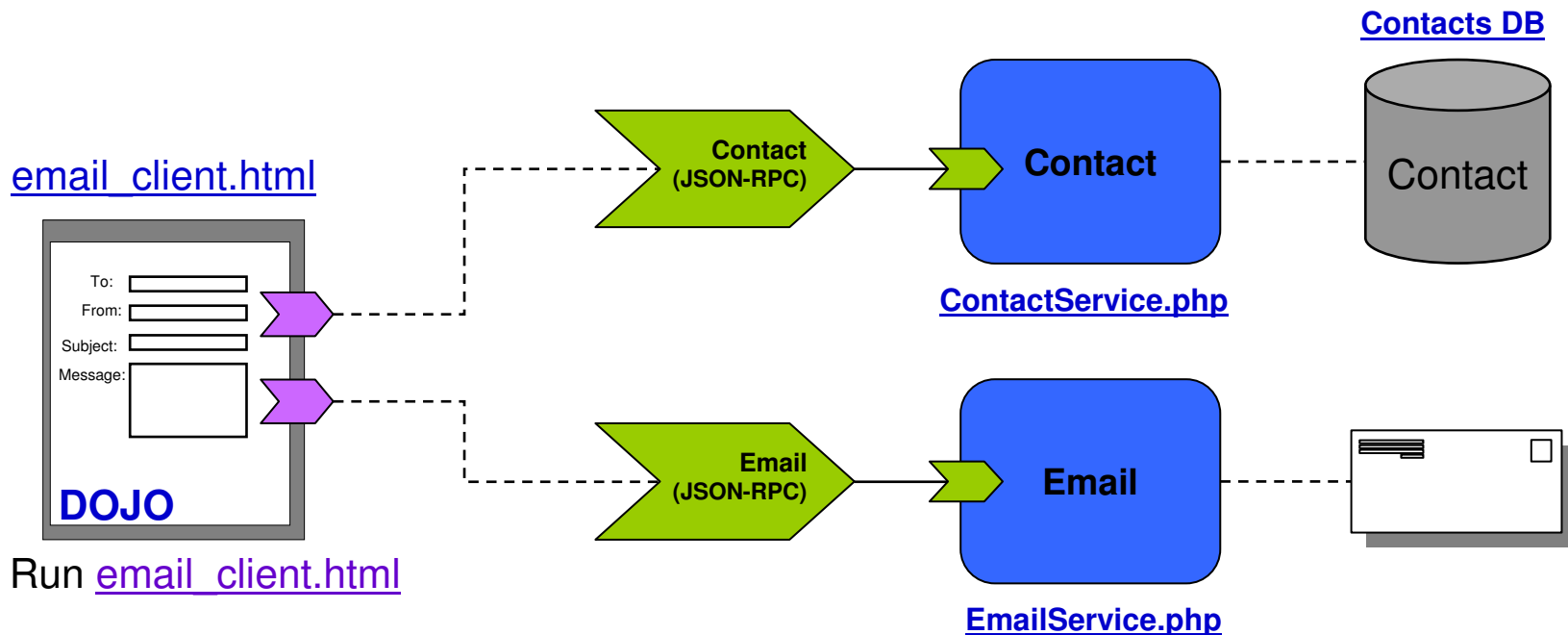
Changing Bindings

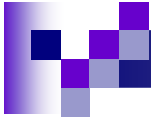
- Need to be able to choose protocols
 - As a provider: different clients (customers) prefer or require different protocols
 - Java client (soap/http), JavaScript client (json-rpc), ...
 - As a consumer: no one protocol is supported by all service providers
- Various bindings available
 - Local
 - WS (SOAP)
 - JSON-RPC
 - XML-RPC
 - REST-RPC
- Intend to provide others



AJAX Application Calling SCA

- Add a json-rpc binding to the Contact and Email services
- Call the services directly from a DOJO based AJAX application via json-rpc





Other styles of services

- What we've seen up to now is a number of rpc-style services
- Other styles exist that are equally valid
 - Resource-Oriented REST (**RE**presentational **S**tate **T**ransfer)
 - Plain Old XML (POX)
 - Syndication (Atompub, RSS)
 - ...and many more...
- No clean taxonomy/terminology exists
 - <http://www.intertwingly.net/blog/2006/11/03/REST-Web-Services>
 - <http://www.trachtenberg.com/blog/2006/11/06/rest-vs-httpox-vs-soap/>
 - <http://www-128.ibm.com/developerworks/xml/library/ws-restvsoap/>

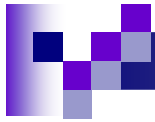


Resource-Oriented REST Background

- An architectural style for well-designed Web applications, not a standard
- Considers the Web to be a state machine
 - A network of Resources (e.g. Web pages) – a virtual state machine
 - Navigating resources via links results in **representations** of **states** being **transferred** to the user agent (e.g. browser)
- This concept is used to describe a class of Web services
 - URIs identify Resources on the Web
 - HTTP used to access and modify these Resources

HTTP Method	Operation
Post	Create
Get	Retrieve
Put	Update
Delete	Delete

- REST says nothing about the representations (formats) – might be HTML, XML, JSON, serialized PHP, ...

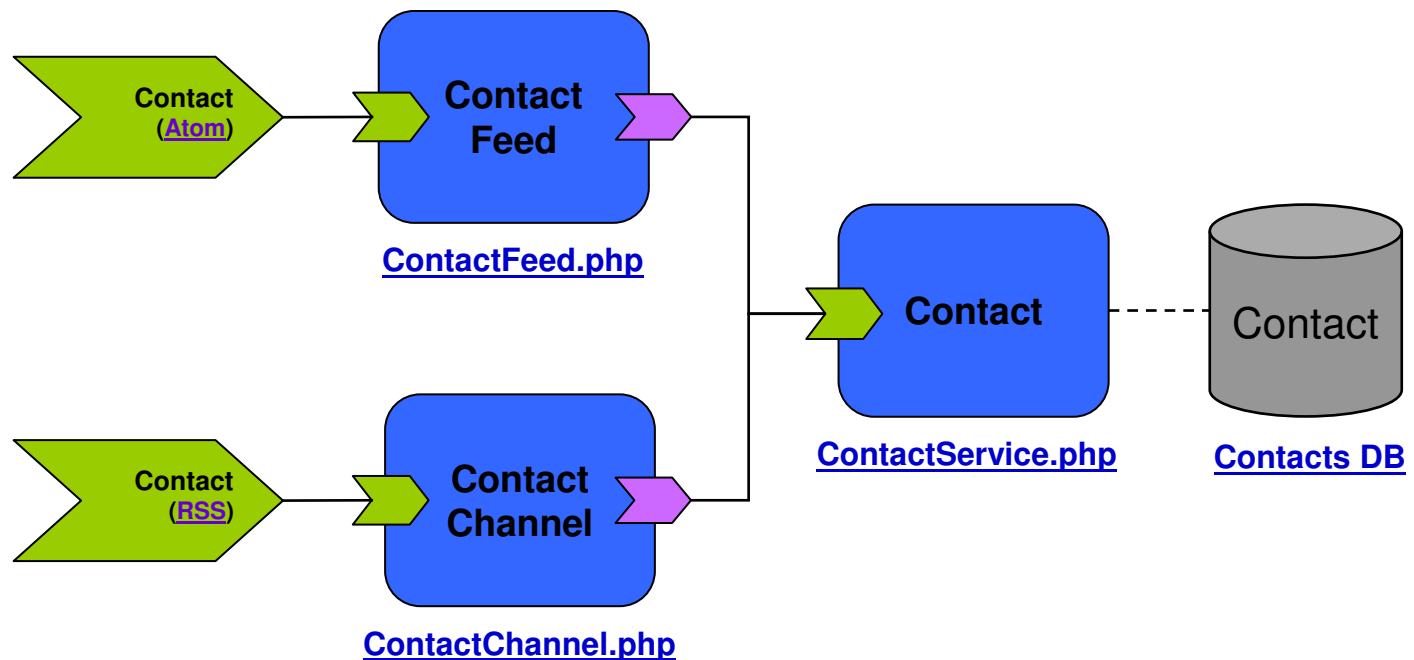


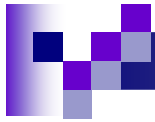
Syndication Services

- RSS and Atom are service types used to publish information
- Give the appearance of publish-subscribe but actually still request-response under the covers
- Not just about syndicating news feeds
- Can be thought of as standardized Resource-oriented REST services

Contact Syndication (VERY EXPERIMENTAL)

- What follows is some early thoughts on how to add syndication services





Custom Bindings

- Many real-world services are complex and difficult to call through a generic binding (e.g. eBay, Google GData, etc.)
- SCA allows people to write and contribute custom bindings



eBay SOAP binding example

- eBay Soap requires a client to provide:

- Soap Body (the main request)

```
<SOAP-ENV:Body>
  <GetSearchResultsRequest...>
    <Version>495</Version>
    <Query>ipod</Query>
    <Pagination>
      <EntriesPerPage>10</EntriesPerPage>
    </Pagination>
  </GetSearchResultsRequest>
</SOAP-ENV:Body>
```

- Soap Header (the security information)

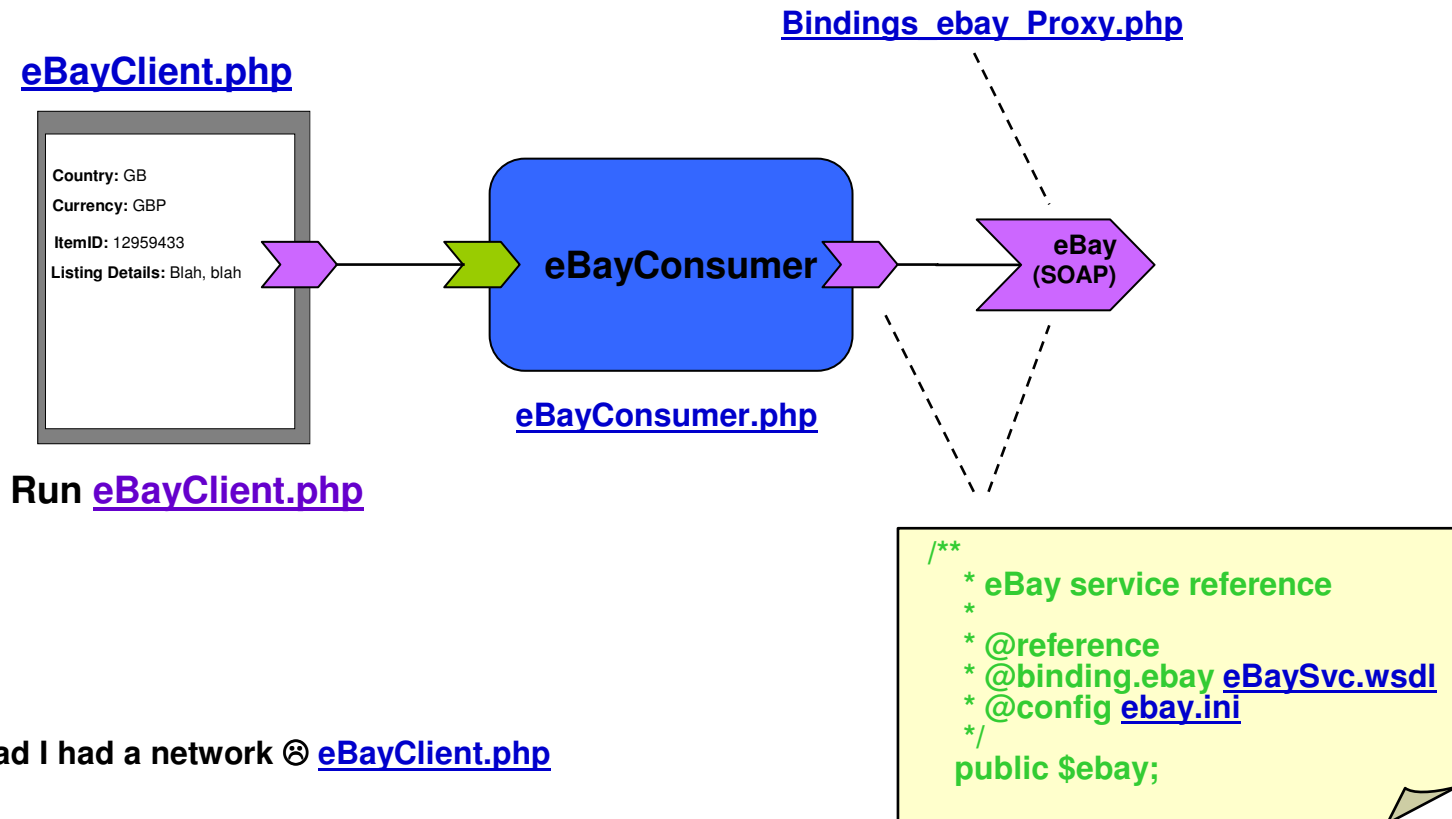
```
<SOAP-ENV:Header>
  <RequesterCredentials ...>
    <eBayAuthToken>AgAAAA**AQAAA...ST+aWf1</eBayAuthToken>
    <Credentials>
      <AppId>IBMUN...</AppId>
      <DevId>...</DevId>
      <AuthCert>...</AuthCert>
    </Credentials>
  </RequesterCredentials>
</SOAP-ENV:Header>
```

- Url Query String Parameters (for eBay to route requests)

```
POST /?callname=GetSearchResults&siteid=1&version=495&appid=...&Routing=default HTTP/1.1
Host: api.sandbox.ebay.com
```

eBay SOAP binding example

- Solution: create “ebay” binding extending the “ws” binding to take eBay specific configuration



The output had I had a network ☹ [eBayClient.php](#)



Where might things go in the future?

- PHP classes for data structures
 - Simpler but less capable than xsd
- Simple database services
 - A CRUD service for a table
- Other bindings
 - Improve: Atom, RSS
 - New: Resource-oriented REST, Google GData, Yahoo!
- Annotation overriding
 - Externally changing service targets, bindings, properties



Summary

- SCA for PHP enables a PHP programmer to write components in PHP which are unaware of local/remote and protocol differences and can focus on providing reusable business logic.
- Components use PHP annotations both to declare their dependencies on other components, and to define the interface which they expose as a service. The SCA for PHP runtime resolves all of these.
- Deploying a PHP component as a 'Web service' can be as simple as copying it into a web server's document root. The SCA for PHP runtime automatically generates service descriptions (WSDL, SMD) for these when requested.



How To Find Out More...

- The PECL Extension
 - Go to PECL and search for SCA, SDO or SCA_SDO
 - http://pecl.php.net/package/sca_sdo
- Web Site
 - As well as the information in the PHP Manual there is a web site.
 - <http://www.osoa.org/display/PHP/SOA+PHP+Homepage>
- Mail List
 - For rants, questions, feedback etc. there is a Google Groups mail list called PHPSOA
 - <http://groups.google.com/group/phpsoa>
- Documents Describing SCA and SDO in more detail
 - Google for OSOA
 - <http://www.osoa.org/display/Main/Home>