



SCA & SDO

(Service Component Architecture)

(Service Data Objects)

Uma Introdução

Silvano Girardi Jr.

silvano@phpsc.com.br

<http://www.silvano.pro.br>



Apresentação

- Natural de Rio Azul, Paraná
- Morando em Florianópolis há 8 anos
- Gerente de Desenvolvimento de Aplicações da InPhonex.com, empresa americana provedora de serviço VoIP
- Membro do PHPSC
- Contribuições para PEAR e SCA_SDO



Agenda

- Visão geral
 - História
 - SCA
 - SDO
 - Avaliação
 - Referências
 - Perguntas?
 - Agradecimentos



O que é isso?!

- SCA
 - Service Component Architecture
 - Arquitetura de Componentes de Serviço
 - “SCA possibilita ao desenvolvedor PHP criar componentes reutilizáveis que podem ser chamados de diversas maneiras, com mínimo esforço.”
- SDO
 - Service Data Object
 - Objeto de Dados de Serviço
 - “SDO possibilita que uma aplicação trabalhe com dados de diferentes fontes (como exemplos: um arquivo XML, um banco de dados, uma planilha) utilizando uma única interface.”



Agenda

- Visão geral
- História
 - SCA
 - SDO
 - Avaliação
 - Referências
 - Perguntas?
 - Agradecimentos



História – SCA & SDO

- SOA – Service Oriented Architecture



- Versão 0.9 – Novembro de 2005
- OSOA – Open Service Oriented Architecture
- Zend, IBM, Sybase, Oracle, BEA, entre outros.



- Versão 1.0 – Março de 2007
- OASIS - Organization for the Advancement of Structured Information Standards



História – SCA & SDO Para PHP

- Julho 2005 – Implementação de SDO em PHP enviada para o PHP PECL pela IBM
- Novembro 2005 – Um grupo de empresas, incluindo IBM, anunciam união para desenvolvimento das especificações para SCA e SDO
- Março 2006 – PHP SDO v1.0.0 (primeira versão estável)
- Agosto 2006 – PHP SDO v1.0.3 (agora depende da implementação SDO Tuscany C++ do Apache)
- Setembro 2006 – PHP SCA v0.1.0 (primeira versão lançada)
- Novembro 2006 – PHP SCA_SDO v1.1.0 (combinados SDO e SCA)
- Maio 2007 – PHP SCA_SDO v1.2.0 (com bindings plugáveis)
- Junho 2007 – PHP SCA_SDO v1.2.2 (binding para eBay SOAP)
- Março 2008 – PHP SCA_SDO v1.2.4 (versão atual)



Agenda

- Visão geral
- História
- SCA
- SDO
- Avaliação
- Referências
- Perguntas?
- Agradecimentos



SCA

- Existem várias maneiras de criar webservices: SOAP, REST, JSON-RPC, ATOM, XMLRPC, e muitas outras. (Cada uma com seus pontos fortes e fracos)
- Disponibiliza uma maneira consistente de trabalhar com diferentes “estilos” de webservices
- Uma linha de código pode significar a diferença entre disponibilizar um serviço via SOAP, JSON-RPC ou REST, por exemplo
- Uma linha de código também pode significar a diferença de como uma dependência (serviço) será chamada

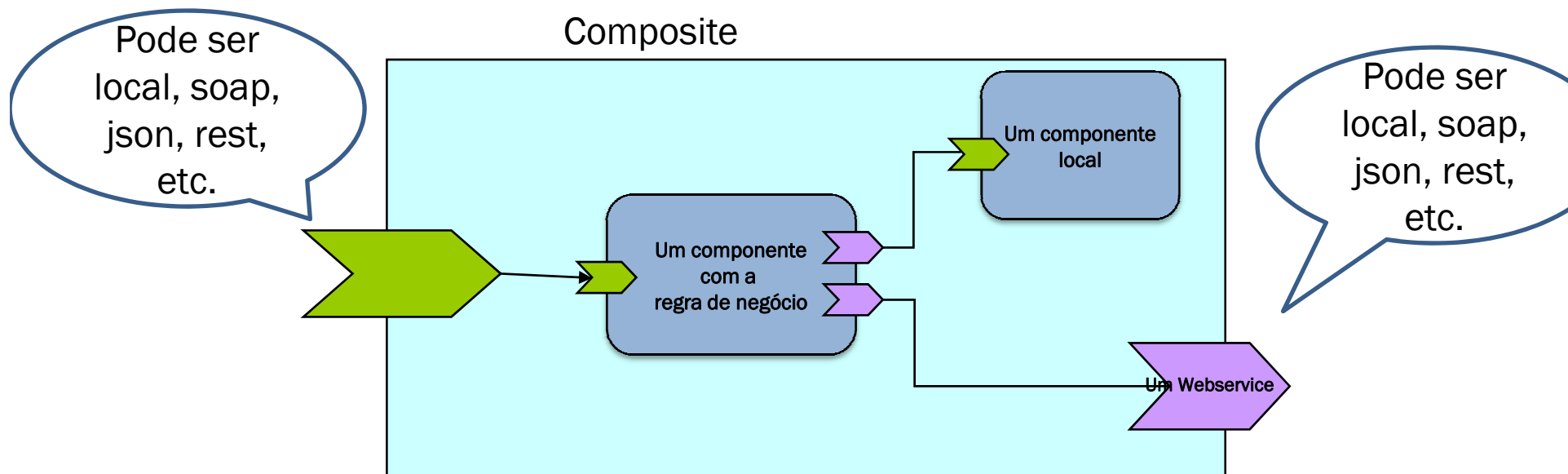
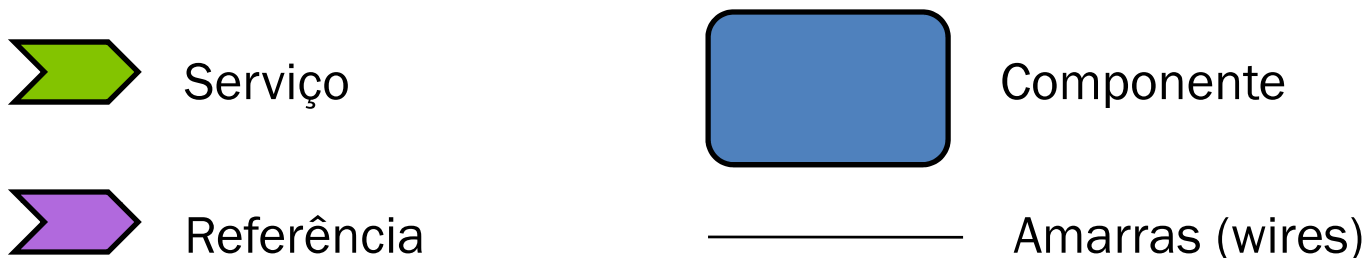


Separe as preocupações (Reusabilidade!)

- Mantenha a regra de negócio separada das “amarras”
- Mantenha a regra de negócio independente de como ela será encontrada e requisitada
 - Disponibilize quantos “bindings” forem necessários – certifique-se de que sua regra de negócio não precisa saber como ela foi chamada.
- Mantenha a regra de negócio independente de como chamar os serviços dos quais que ela depende
 - Declare as dependências – mas certifique-se de que sua regra de negócio não precisa saber como resolvê-las
 - De preferência, use alguma outra maneira para “amarrar” os componentes. (Padrões como Inversão de Controle, Injeção de Dependências)



Entendendo os diagramas





Um Componente PHP Simples

```
<?php

/**
 * Componente Saudação
 */
class Saudacao
{
    /**
     * @param string $nome
     * @return string
     */
    public function digaOla($nome) {
        return 'Ola ' . $nome;
    }
}

?>
```

```
<?php

require_once 'Saudacao.php';

$saudacao = new Saudacao();
echo $saudacao->digaOla('Santa Catarina');

?>
```



Um Componente PHP Simples com SCA

```
<?php  
→ include 'SCA/SCA.php';  
  
/**  
 * Componente Saudação  
 */  
→ * @service  
 */  
class Saudacao  
{  
    /**  
     * @param string $nome  
     * @return string  
     */  
    public function digaOla($nome) {  
        return 'Ola ' . $nome;  
    }  
}  
?>
```

```
<?php  
  
include 'SCA/SCA.php';  
  
$saudacao = SCA::getService('Saudacao.php');  
echo $saudacao->digaOla('Santa Catarina');  
  
?>
```



Um Serviço SOAP com SCA

```
<?php

include 'SCA/SCA.php';

/**
 * Componente Saudação
 *
 * @service
 * @binding.soap
 */
class Saudacao
{
    /**
     * @param string $nome
     * @return string
     */
    public function digaOla($nome) {
        return 'Ola ' . $nome;
    }
}

?>
```

```
<?php

// consumindo com SCA
include 'SCA/SCA.php';

$saudacao =
SCA::getService('http://www.exemplo.org/Saudacao.php?wsdl');
echo $saudacao->digaOla('Santa Catarina');

// consumindo com a extensão PHP SOAP
$saudacao = new
SoapClient('http://www.exemplo.org/Saudacao.php?wsdl');
echo $saudacao->digaOla(array('nome' => 'Santa Catarina'))-
>digaOlaReturn;

?>
```



Bindings – Os Diversos Estilos

Como provedor de serviços, clientes diferentes preferem ou precisam de “estilos” diferentes.

Como consumidor de serviços, não existe um único “estilo” suportado por todos os provedores de serviços.

- Service-oriented
 - soap
 - jsonrpc
 - xmlrpc
 - restrpc (GET+/methodname?param)
 - local
- Resource-oriented
 - restresource (GET/PUT/POST/DELETE)
 - atom
 - rss
- Other
 - message
 - ebaysoap
 - simpledb



Serviços SOAP + JSON-RPC com SCA

```
<?php
```

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

```
/**
```

```
 * Componente Saudação
```

```
 *
```

```
 * @service
```

```
 * @binding.soap
```

```
→ * @binding.jsonrpc
```

```
 */
```

```
class Saudacao
```

```
{
```

```
 /**
```

```
 * @param string $nome
```

```
 * @return string
```

```
 */
```

```
public function digaOla($nome) {
```

```
    return 'Ola ' . $nome;
```

```
}
```

```
}
```

```
?>
```

```
<?php
```

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

```
// consumir serviço SOAP
```

```
$saudacao =
```

```
SCA::getService('http://www.exemplo.com/Saudacao.php?wsdl');
```

```
echo $saudacao->digaOla('Santa Catarina');
```

```
// consumir serviço JSON-RPC
```

```
$saudacao =
```

```
SCA::getService('http://www.exemplo.com/Saudacao.php?smd');
```

```
echo $saudacao->digaOla('Santa Catarina');
```

```
?>
```




Referenciando outros Serviços

```
<?php
```

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

```
/**
```

```
 * Greeting Component
```

```
 *
```

```
 * @service
```

```
→ * @binding.xmlrpc
```

```
 */
```

```
class Greeting
```

```
{
```

```
 /**
```

```
  * @param string $name
```

```
  * @return string
```

```
  */
```

```
 public function sayHello($name) {
```

```
     return 'Hello' . $name;
```

```
 }
```

```
?>
```

```
<?php
```

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

```
/**
```

```
 * Saluto Componente
```

```
 *
```

```
 * @service
```

```
 * @binding.restrpc ←
```

```
 */
```

```
class Saluto
```

```
{
```

```
 /**
```

```
  * @param string $nome
```

```
  * @return string
```

```
  */
```

```
 public function direCiao($nome) {
```

```
     return 'Ciao' . $nome;
```

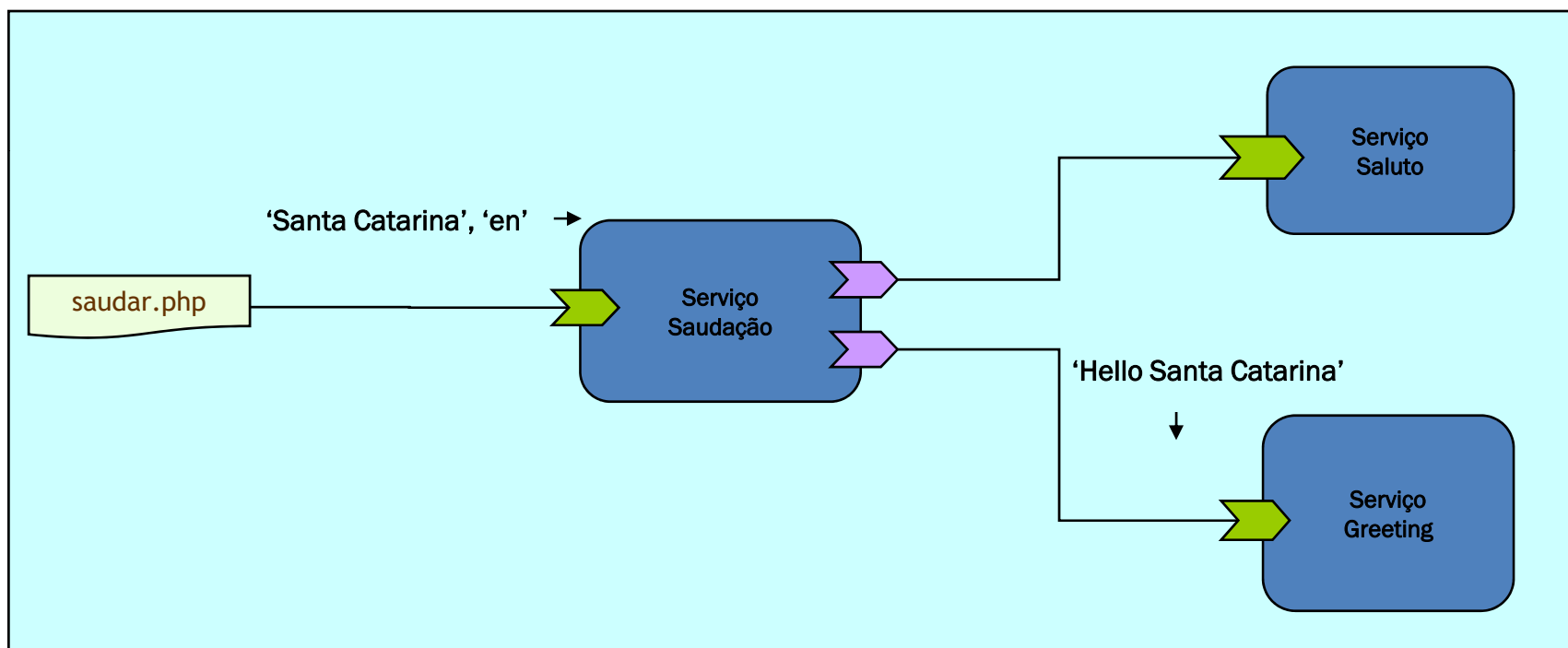
```
 }
```

```
}
```

```
?>
```



Referenciando outros Serviços





Referenciando outros Serviços

<?php

include 'SCA/SCA.php';

```
/**
 * Componente Saudação
 *
 * @service
 * @binding.soap
 * @binding.jsonrpc
 */
```

class Saudacao

```
{
    /**
     * @reference
     * @binding.xmlrpc http://www.example.org/Greeting.php
     */
    public $saudacao_en;
```

```
    /**
     * @reference
     * @binding.restrpc http://www.esempio.org/Saluto.php
     */
    public $saudacao_it;
```

```
    /**
     * @param string $nome
     * @param string $idioma
     * @return string
     */
```

public function digaOla(\$nome, \$idioma) {

switch (\$idioma) {

```
case 'en':
    return $this->saudacao_en->sayHello($nome);
    break;
```

```
case 'it':
    return $this->saudacao_it->direCiao($nome);
    break;
```

```
default:
    return 'Ola ' . $nome;
```

}

}

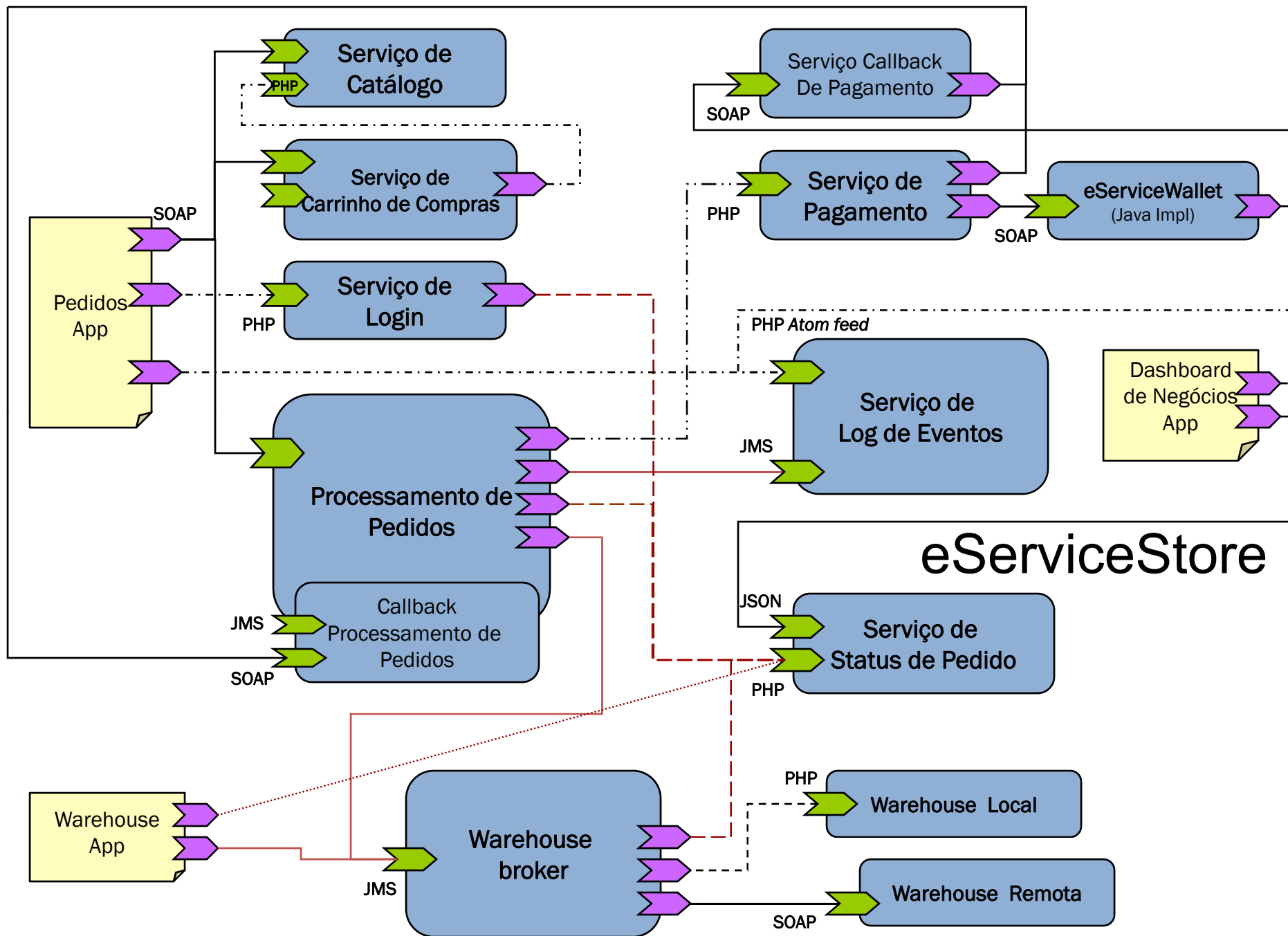
}

?>

```
/**
 * @reference
 * @binding.xmlrpc http://www.example.org/Greeting.php
 */
public $saudacao_en;
```

```
/**
 * @reference
 * @binding.restrpc http://www.esempio.org/Saluto.php
 */
public $saudacao_it;
```

```
case 'en':
    return $this->saudacao_en->sayHello($nome);
    break;
case 'it':
    return $this->saudacao_it->direCiao($nome);
    break;
```





Agenda

- Visão geral
- História
- SCA
- SDO
 - Avaliação
 - Referências
 - Perguntas?
 - Agradecimentos



Mas só posso usar escalares com SCA?

(escalar é uma variável que aceita um único valor por vez, exemplo: inteiro, string, float)

Respondendo à pergunta:

NÃO!

Solução: SDO



SDO

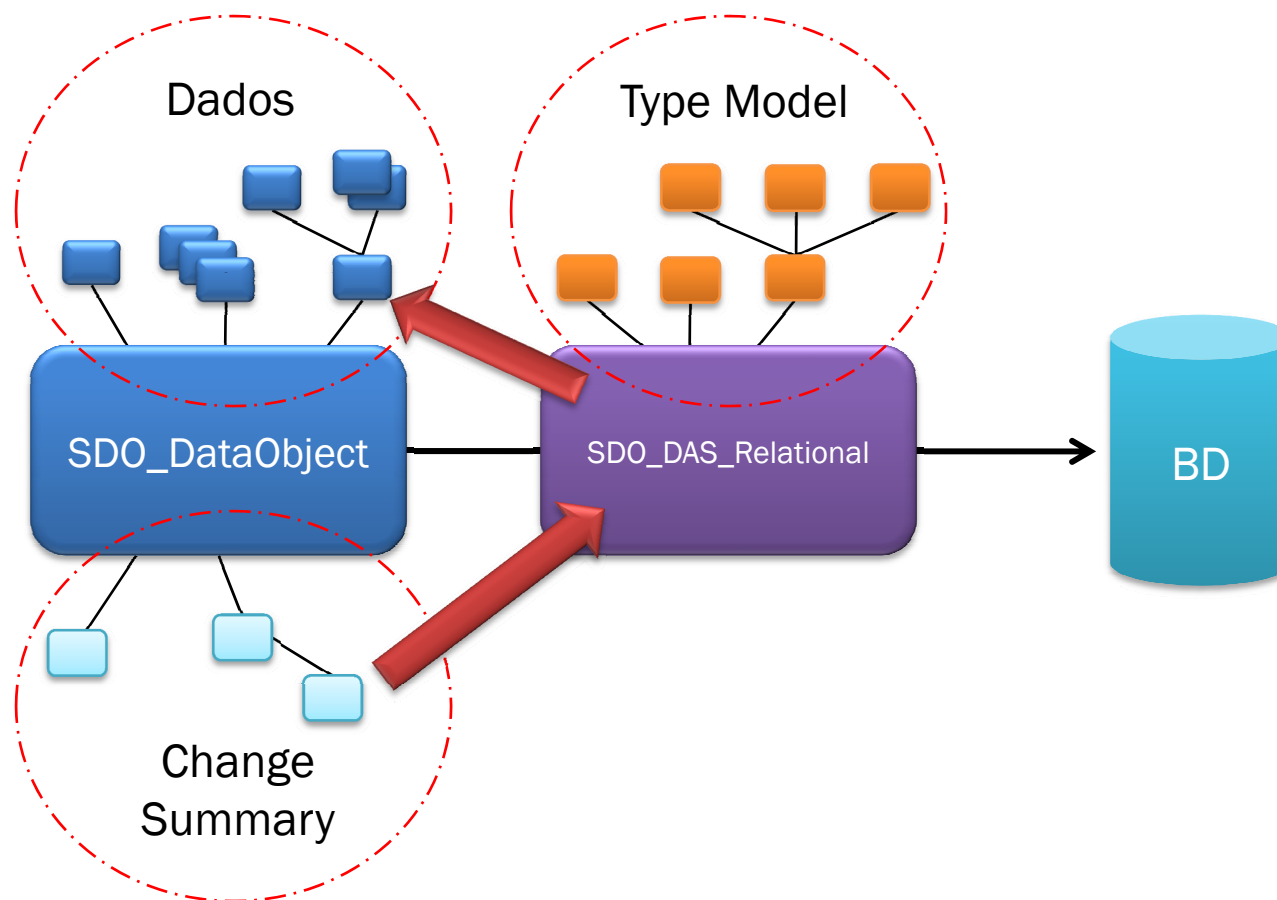
“SDO possibilita que uma aplicação trabalhe com dados de diferentes fontes (como exemplos: um arquivo XML, um banco de dados, uma planilha) utilizando uma única interface.”

Formado por 4 partes:

- **Service Data Objects (SDO)**
 - O objeto de dados manipulado dentro de um script PHP
- **Data Access Service (DAS)**
 - Disponibiliza meios de ler e escrever dados de/em uma fonte de dados
- **Type Model**
 - Descreve tipos de SDOs e tipos primitivos que podem aparecer (válidos) na estrutura do SDO. Pode ser criado através de uma API ou pela leitura de um XSD (no caso de um XML DAS)
- **Change Summary**
 - Registra as alterações de dados que ocorrem durante a execução do script. É utilizado pelo DAS para escrever dados de volta à fonte de dados.

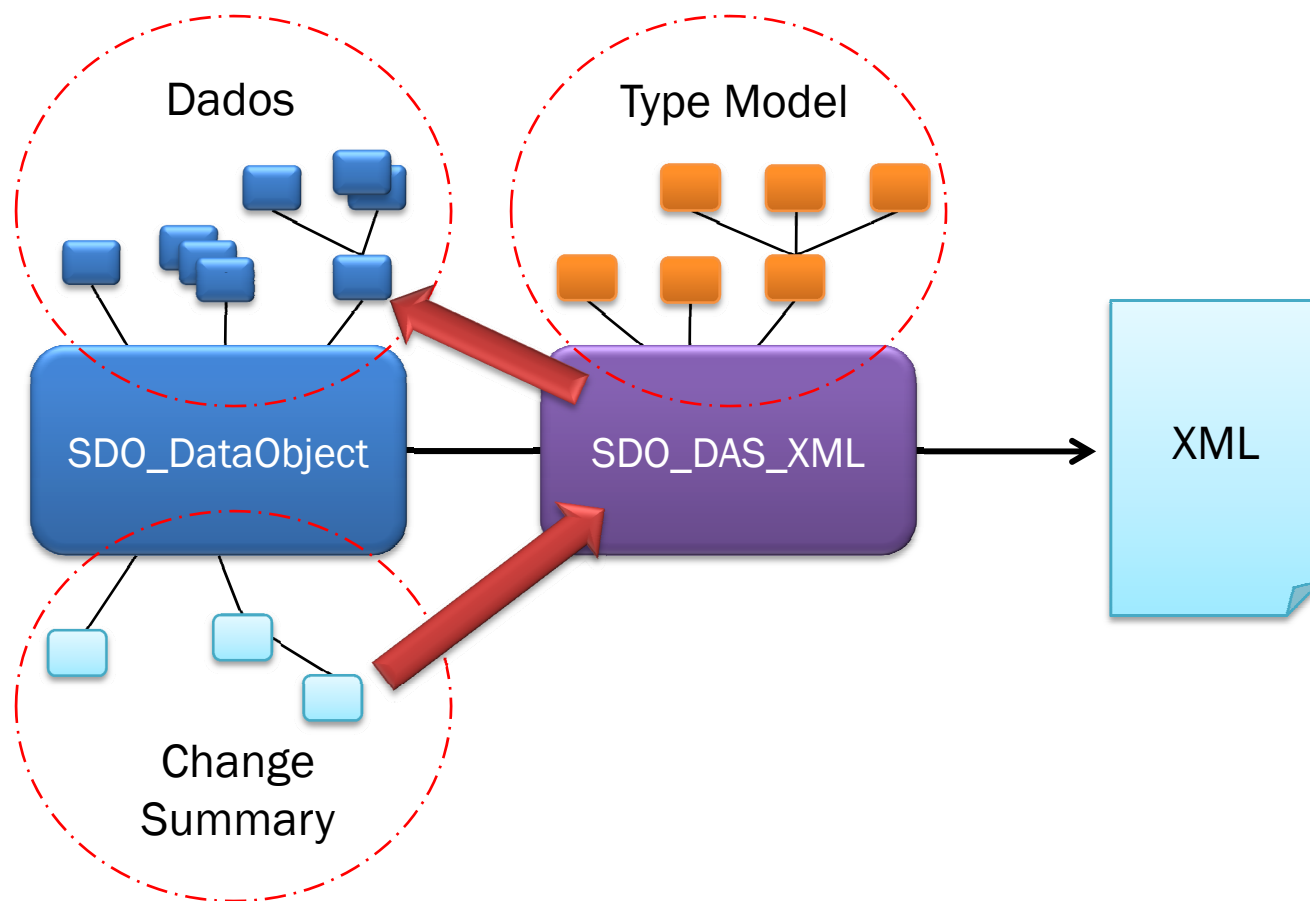


SDO





SDO





SDO no SCA

- Utilizado para trabalhar com tipos complexos
 - Ex: um pedido de compra, um endereço, etc.
- Suporte nativo
 - Criação de `SDO_DataObjects` utilizando o método `createDataObject()` diretamente em um serviço criado via `SCA::getService()`
- Não fica restrito apenas à utilização do SDO pelo método nativo do SCA. É possível utilizar o SDO de forma independente
- Utiliza `SDO_DAS_XML` como interface e XSD (XML Schema Definition) para descrever os tipos complexos



Vejam na prática

O serviço de Saudação recebeu uma atualização e agora não trabalha mais com um parâmetro string “nome”.

Agora é necessário enviar como parâmetro um tipo complexo “Pessoa”, cuja estrutura é a seguinte:

Tipo:

Pessoa

Propriedades:

nome

sobrenome

email

lista de telefones



O XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn://Pessoa" xmlns:tns="urn://Pessoa">

  <xs:element name="Pessoa">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nome" type="xs:string" />
        <xs:element name="sobrenome" type="xs:string" />
        <xs:element name="email" type="xs:string" />
        <xs:element name="telefone" type="tns:telefone" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="telefone">
    <xs:sequence>
      <xs:element name="ddd" type="xs:string" />
      <xs:element name="numero" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```



O Serviço Saudação Atualizado

```
<?php
```

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

```
/**
```

```
 * Componente Saudação
```

```
 *
```

```
 * @service
```

```
 * @binding.soap
```

```
→ * @types urn://Pessoa Pessoa.xsd
```

```
 */
```

```
class Saudacao
```

```
{
```

```
 /**
```

```
→ * @param Pessoa $Pessoa urn://Pessoa
```

```
 * @return string
```

```
 */
```

```
public function digaOla($Pessoa) {
```

```
    return 'Ola ' . $Pessoa->nome . ' ' . $Pessoa->sobrenome;
```

```
}
```

```
}
```

```
?>
```



Script saudar.php Atualizado

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

$saudacao = SCA::getService('http://www.exemplo.com/Saudacao.php?wsdl');

// criar SDO pessoa
$pessoa = $saudacao->createDataObject('urn://Pessoa', 'Pessoa');
$pessoa->nome = 'Silvano';
$pessoa->sobrenome = 'Girardi Jr.';
$pessoa->email = 'silvano@phpsc.com.br';

// criar SDO telefone, dentro do SDO pessoa
$telefone = $pessoa->createDataObject('telefone');
$telefone->ddd = '48';
$telefone->numero = '5555 5555';

// adicionando outro SDO telefone dentro da lista de telefones do SDO pessoa
$telefone = $pessoa->createDataObject('telefone');
$telefone->ddd = '48';
$telefone->numero = '9999 9999';

echo $saudacao->digaOla($pessoa);

?>
```



Agenda

- Visão geral
- História
- SCA
- SDO
- Avaliação
 - Referências
 - Perguntas?
 - Agradecimentos



Avaliação

- Custo
 - Gratuitos e open source
- Instalação, licença e contribuições
 - Disponível no PECL
 - Licença Apache 2.0
 - É fácil contribuir
 - Através da lista de discussão
 - Através de produção de código
 - Para contribuir com código é preciso assinar um termo: CLA – Contributor's License Agreement
- Nem tudo são flores
 - Documentação escassa e desatualizada, além de estar parte no manual do PHP, parte no site OSOA
 - Algumas áreas não finalizadas
 - Desenvolvimento atual está lento
- Prós
 - O ganho em produtividade é indiscutível
 - Cumprem muito bem o papel ao qual se propõem
 - Possibilidade de criar outros bindings
- Futuro
 - SCA: <http://www.osoa.org/display/PHP/SCA+Wish+List>
 - SDO: <http://www.osoa.org/display/PHP/SDO+Wish+List>
- E vale a pena citar
 - O feedback dos desenvolvedores através do grupo phpsoa é excelente



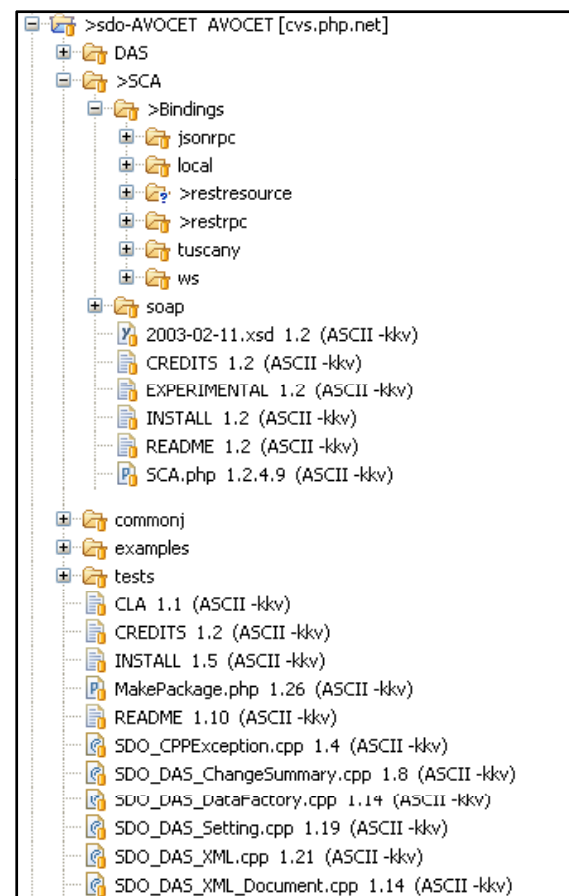
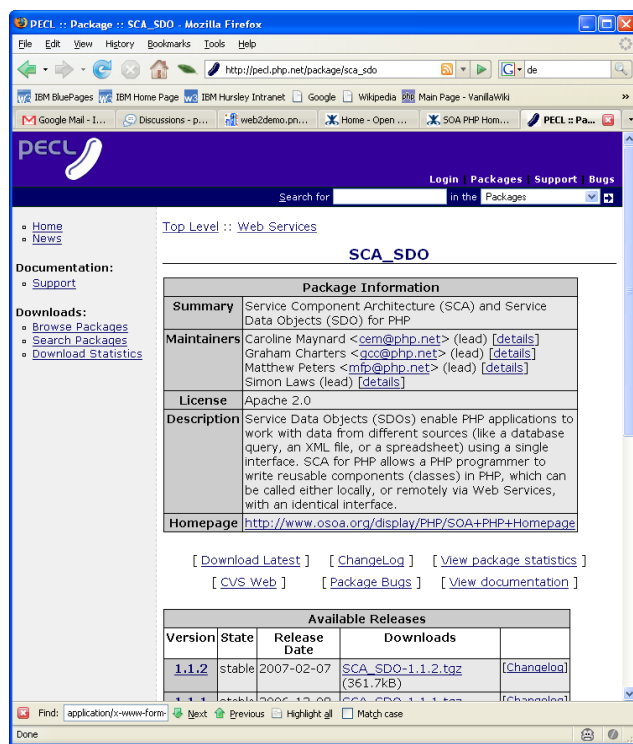
Agenda

- Visão geral
- História
- SCA
- SDO
- Avaliação
- Referências
- Perguntas?
- Agradecimentos



Quero instalar!

- Disponível no PECL.
 - <http://pecl.php.net>
 - `pecl install SCA_SDO`





Onde encontro mais informações?

- SCA e SDO para PHP
 - <http://.osoa.org/display/PHP/SOA+PHP+Homepage>
- Grupo de discussão
 - <http://groups.google.com/group/phpsoa>
- Manual do PHP para SCA
 - <http://www.php.net/SCA>
- Manual do PHP para SDO
 - <http://www.php.net/SDO>
- Meu Blog
 - <http://www.silvano.pro.br>
- Ferramentas de busca :)



Agenda

- Visão geral
- História
- SCA
- SDO
- Avaliação
- Referências
- Perguntas?
- Agradecimentos



Perguntas!?





Agenda

- Visão geral
- História
- SCA
- SDO
- Avaliação
- Referências
- Perguntas?
- Agradecimentos



Agradecimentos

- PHPSC (em especial Elton Minetto)
- UNOCHAPECÓ
- Desenvolvedores SCA_SDO
 - Matthew Peters
 - Caroline Maynard
 - Graham Charters
- Você!
- **MUITO OBRIGADO!**