SCA Service Component Architecture

SCA服务构件架构

JAVA构件实现规范

满江红redsaga.com

SCA v1.00， 2007.2.15

SCA Service Component Architecture

Java Component Implementation Specification

SCA Version 1.00, 15 February 2007

Technical Contacts:

Ron Barack	SAP AG
Michael Beisiegel	IBM Corporation
Henning Blohm	SAP AG
Dave Booz	IBM Corporation
Jeremy Boynes	Independent
Ching-Yun Chao	IBM Corporation
Adrian Colyer	Interface21
Mike Edwards	IBM Corporation
Hal Hildebrand	Oracle
Sabin Ielceanu	TIBCO Software, Inc
Anish Karmarkar	Oracle
Daniel Kulp	IONA Technologies plc.
Ashok Malhotra	Oracle
Jim Marino	BEA Systems, Inc.
Michael Rowley	BEA Systems, Inc.
Ken Tam	BEA Systems, Inc
Scott Vorthmann	TIBCO Software, Inc
Lance Waterman	Sybase, Inc.

Copyright Notice

License

Status of this Document

This specification may change before final release and you are cautioned against relying on the content of this specification. The authors are currently soliciting your contributions and suggestions. Licenses are available for the purposes of feedback and (optionally) for implementation.

目录

1. Java实现类型

1.1 简介

此规范是《SCA 装配模型规范》[1]的扩展，定义了 Java 类如何提供 SCA 构件的实现，以及做为一种组件实现类型的该类在 SCA 中是如何被使用的。

此规范需要用到《SCA Java 通用注解注解和 API 规范》[2]中定义的所有注解和 API。在《SCA Java 通用注解和 API 规范》中，所有被引用的的注解和 API 都是定义良好的，除非另行说明。此外在这些 Java 通用注解和 API 规范中定义的语义都是标准化的。

1.2 Java 实现类型

此节详细说明 Java 类是如何提供 SCA 构件实现的，包括构件的各种特性，如服务、引用以及属性。另外，它还详细列出了在 Java 类被用作构件实现类型时，《SCA Java 通用注解注解和 API 规范》[2]中定义的元数据和 Java API 的使用。

1.2.1 服务

基于 Java 类的构件实现可以提供一个或多个服务。

基于 Java 的实现所提供的服务，可以有一个按下列方法之一定义的接口：

- Java 接口
- Java 类
- 产生自 Web Services Description Language [3] (WSDL) portType 的 Java 接口

Java 实现类必须实现服务接口定义的所有操作。如果服务接口是用一个 Java 接口定义的，那么基于 Java 的构件要么实现此 Java 接口，要么实现接口的所有操作。

由 Java 类定义接口的服务（与 Java 接口相对）不是远程的。产生自 WSDL portType 的 Java 接口是远程的，细节请查看《SCA Java 通用注解和 API 规范》的 WSDL2Java 和 Java2WSDL 节。

Java 实现类型可以通过使用@Service 显式地指定其提供的服务。在以下所定义的某种情形中，@Service 的使用并不是必须的，并且 Java 实现类型提供的服务可以从实现类自身推断而来（译者注：也许可以使用反射技术从实现类自身获得其提供的服务）。

1.2.1.1. @Service 的使用

服务接口可以被指定为Java接口。做为构件实现的Java类可以通过实现某个指定了服务合约的Java接口来提供服务。因为一个Java类可以实现多个接口，而这些接口中有些并没有定义SCA服务，所以@Service注解可以用于指示实现所提供的服务以及服务对应的Java接口定义。

如下是Java服务接口以及使用此接口提供服务的Java实现的一个例子：

接口:

```
public interface HelloService {
      String hello(String message);
}
```

实现类:

```
@Service(HelloService.class)
public class HelloServiceImpl implements HelloService {
public String hello(String message) {
      ...
      }
}
```

为了阐述的更加明了，下面给出了关于此实现的构件类型的XML描述。实际上，因为可以从Java类反射得到，所以没有必要书写构件类型。

```
<?xml version="1.0" encoding="ASCII"?>
<componentType xmlns="http://www.osoa.org/xmlns/sca/0.9">
    <service name="HelloService">
          <interface.java interface="services.hello.HelloService"/>
    </service>
</componentType>
```

和接口相反，Java实现类本身也能定义构件提供的服务。这种情况下，@Service用于显式声明此实现类定义了实现所提供的服务。同情形下，构件只能提供@Service声明的服务。如下所示：

```
@Service(HelloServiceImpl.class)
public class HelloServiceImpl implements AnotherInterface {
public String hello(String message) {
      ...
      }
…
}
```

在上述例子中，HelloWorldServiceImpl提供了由实现类中的public方法定义的服务。这里，AnotherInterface接口没有指定构件提供的服务。如下是由内省得到的构件类型的XML描述：

```
<?xml version="1.0" encoding="ASCII"?>
<componentType xmlns="http://www.osoa.org/xmlns/sca/0.9">
      <service name="HelloServiceImpl">
            <interface.java interface="services.hello.HelloServiceImpl"/>
      </service>
```

```
</componentType>
```

@Service可以用于指定一个实现提供的多个服务：

```
@Service(interfaces={HelloService.class, AnotherInterface.class})
public class HelloServiceImpl implements HelloService, AnotherInterface {

public String hello(String message) {
      ...
      }
…
}
```

如下片段演示了此实现由内省得到的构件类型：

```
<?xml version="1.0" encoding="ASCII"?>
<componentType xmlns="http://www.osoa.org/xmlns/sca/1.0">

      <service name="HelloService">
            <interface.java interface="services.hello.HelloService"/>
      </service>
      <service name="AnotherService">
            <interface.java interface="services.hello.AnotherService"/>
      </service>

</componentType>
```

1.2.1.2 本地服务和远程服务

接口或实现类定义的Java服务合约，可以使用@Remotable来声明服务遵循远程服务的语义，其中远程服务在《SCA装配规范》中做了定义。如下例子演示了@Remotable的使用：

```
package services.hello;

@Remotable
public interface HelloService {

      String hello(String message);
}
```

除非声明@Remotable，否则Java接口或实现类定义的服务被认为是《SCA装配模型规范》中定义的本地服务。

如果实现类实现了未用@Remotable注解标注的接口，那么认为此实现类实现了单个本地服务。而该本地服务的类型由此实现类定义。（注：本地服务可以用Java接口或Java类指定类型）

实现类可以对SCA运行时提供建议，关于是否实现类能通过使用@AllowsPassByReference注解来得到传值语义而不用拷贝方式。

（补充翻译：实现类可以对SCA运行时提供建议，关于是否实现类能通过使用@AllowsPassByReference注解来避免使用拷贝方式的传值语义。

译者注原文为：An implementation class may provide hints to the SCA runtime about whether it can achieve pass-by-value semantics without making a copy by using the @AllowsPassByReference.

)

1.2.1.3 内省Java实现提供的服务

在下述情况中，Java实现类提供的服务可以通过内省得到，可以省略@Service的使用。如下的算法用于决定服务是如何从实现类内省来的。

如果在实现类上，SCA服务的接口没有指定@Service注解，则假定所有注解了@Remotable的已实现接口都是构件提供的服务接口。如果被实现接口都不是远程的，则默认为实现提供了单个服务，此服务的类别（type）就是实现类。（译者注：服务的类别就是实现类的类别）

1.2.1.4 非阻塞服务操作

Java接口或实现类定义的服务操作可以使用@OneWay来声明客户调用服务操作时，SCA运行时必须遵循非堵塞性语义。此非堵塞性语义在《SCA装配规范》中定义。

1.2.1.5. 非会话型服务和会话型服务

Java实现类型支持所有的会话型服务注解，这些会话型注解在《SCA Java通用注解和API规范》中定义：@Conversational、@EndsConversational和@ConversationAttributes。

如下语义适用于由Java接口或实现类定义的服务合约。由Java接口或实现类定义的服务合约，除非使用了@Conversational注解进行标注，否则将会被认为是《SCA装配规范》中定义的非会话型服务合约。在使用@Conversational的情况下，@Conversational用于声明提供服务的构件实现将实现《SCA装配规范》中定义的会话型语义。

1.2.1.6. 回调服务

在一个Java类所实现的服务接口上使用@Callback注解来声明一个回调接口。

1.2.2 引用

引用可以通过注入或ComponentContext API（此API在《SCA Java通用注解与API规范》中定义）获得。推荐尽可能使用注入方式来访问引用。

1.2.2.1 引用注入

通过使用@Reference注解，Java实现类型可以显式地指定其使用的引用，如下所示：

```
public class ClientComponentImpl implements Client {
      private HelloService service;

      @Reference
      public void setHelloService(HelloService service) {
            this.service =      service;
      }
}
```

如果@Reference标注了一个public或protected的setter方法，则要求SCA运行时提供服务引用合约的相应实现，且此服务引用合约由setter方法的参数类型指定。以上必须通过调用实现实例的setter方法来实现。什么时候注入由实现的作用域（scope）决定。然而，注入总是发生在第一个服务方法被调用之前。

如果@Reference标注了一个public或protected的属性域，则要求SCA运行时提供服务引用合约的相应实现，且属性域类型指定此服务引用合约。以上必须通过设置实现实例的属性域来实现。什么时候注入由实现的作用域决定。

如果@Reference标注了构造函数的某个参数，则要求在实现实例化期间，SCA运行时提供服务引用合约的相应实现，构造函数参数指定服务引用合约。

引用也可以根据1.2.7.节定义的规则并通过内省实现类来指定。

引用可被声明为是可选的，《SCA Java通用注解和API规范》中对此进行了定义。

1.2.2.2. 动态引用访问

引用可以通过ComponentContext.getService()和ComponentContext.getServiceReference(..)方法来动态访问。在《Java通用注解与API规范》中对这些方法进行了描述。

1.2.3 属性

1.2.3.1. 属性注入

属性可以通过注入或ComponentContext API（ComponentContext API在《SCA Java通用注解与API规范》中定义）来获得。推荐尽可能使用注入方式来访问属性。

Java实现类型可以通过使用@Property注解显式地指定其属性，如下所示：

```
public class ClientComponentImpl implements Client {
      private int maxRetries;

      @Property
      public void setRetries(int maxRetries) {
            this. maxRetries = maxRetries;
      }
```

```
}
```

如果@Property标注了一个public或protected的setter方法，则要求SCA运行时提供相应的属性值。以上必须通过调用实现实例的setter方法实现。什么时候进行注入由实现的作用域（scope）决定。

如果@Property标注了一个public或protected的属性域，则要求SCA运行时提供相应的属性值。什么时候进行注入由实现的作用域（scope）决定。

如果@Property标注了构造函数的某个参数，则要求SCA运行时在实现实例进行实例化期间提供相应的属性值。什么时候进行注入由实现的作用域（scope）决定。

属性也可以根据1.2.7.节定义的规则并通过内省实现类来指定。

属性可被声明为是可选的，在《SCA Java通用注解和API规范》中对此进行了定义。

1.2.3.2. 动态属性访问

属性可以通过ComponentContext. getProperty ()方法（此方法在《SCA Java通用注解与API规范》中定义）来动态访问。

1.2.4 实现实例的实例化

Java实现类必须提供一个public或protected的构造函数，SCA运行时用此构造函数实例化实现的实例。构造函数可以有参数，当存在参数时，SCA容器会在调用构造函数时传递可用的属性或引用值。未通过构造函数设置的任何属性或引用值，都将会在任一服务方法被调用之前，被设置给属性域或被传递给与其关联的setter方法。

按照如下方式，由容器选择要用的构造函数：

1. 用@Constructor注解标注的显式构造函数

2. 无二义性地标识了所有属性和引用值的显式构造函数

3. 无参数的构造函数

@Constructor注解只能在一个构造函数上使用；如果多个构造函数标注了@Constructor，SCA容器将报错。

与构造函数的每个参数相关联的属性或引用以如下方式标识：

- 通过@Constructor注解中的name（如果存在）
- 通过在参数声明上使用的@Property或@Reference注解
- 通过将属性类型或引用的类型与参数类型唯一地进行匹配

构件间的循环引用发生的话，容器会用以下两种办法之一进行处理：

- 如果循环中的任意引用是可选的，那么容器就会在构造期间注入null值。保证调用任何服务之前都已经注入了目标的引用。
- 容器也可以注入一个目标服务的代理；调用代理上的方法会导致ServiceUnavailableException异常。

以下是合法的Java构件构造函数声明的例子：

```
/** Simple class taking a single property value */
public class Impl1 {
String someProperty;
public Impl1(String propval) {...}
}

/** Simple class taking a property and reference in the constructor;
* The values are not injected into the fields.
*//
public class Impl2 {
public String someProperty;
public SomeService someReference;
public Impl2(String a, SomeService b) {...}
}

/** Class declaring a named property and reference through the constructor
*/
public class Impl3 {
@Constructor({"someProperty", "someReference"})
public Impl3(String a, SomeService b) {...}
}

/** Class declaring a named property and reference through parameters */
public class Impl3b {
public Impl3b(
@Property("someProperty") String a,
@Reference("someReference) SomeService b
) {...}
```

```
}

/** Additional property set through a method */
public class Impl4 {
public String someProperty;
public SomeService someReference;
public Impl2(String a, SomeService b) {...}
@Property public void setAnotherProperty(int x) {...}
}
```

1.2.5 实现作用域与生命周期回调

Java实现类型支持《SCA Java通用注解和API规范中》定义的所有作用域：STATELESS, REQUEST, CONVERSATION和COMPOSITE。实现通过使用@Scope注解指定它们的作用域：

```
@Scope(”COMPOSITE”)
public class ClientComponentImpl implements Client {
      // …
}
```

当实现类未指定@Scope注解，则其作用域默认为STATELESS。

Java构件实现通过使用@Init和@Destroy注解分别指定init和destroy回调。例如：

```
public class ClientComponentImpl implements Client {

      @Init
      public void init() {
            //…
      }

      @Destroy
      public void destroy() {
            //…
      }
}
```

1.2.5.1. 会话实现

作用域为CONVERSATION的Java实现类可以使用@ConversationID注解来持有当前的会话ID，此会话ID是通过public或protected属性域或setter方法注入的。可选地，Conversation API（在《SCA Java通用注解和API规范》中定义）能用来获取当前的会话ID。

作为会话型服务的提供者，需要在同一会话的连续方法调用之间维持状态数据。对于Java实现类型，有两种

可能的策略用于处理状态数据：

1．实现可以构建为无状态的代码段(本质上来说，代码认为每次方法调用都使用一个新的实例)。代码必须负责访问会话的conversationID，此conversationID由SCA运行时代码维护。在方法处理过程中任何必要的状态数据的持久化操作，以及对持久化状态数据的访问，都是由实现来负责的，而所有这些都是以conversationID作为key来使用。

2．实现可以构建为有状态的代码段，也就意味着实现把任意的状态数据存储在Java类实例的属性域中。实现必须用@Scope注解声明会话作用域。这指示SCA运行时环境，实现是有状态的，并且SCA运行时必须在客户函数调用与服务实现的某个特定实例之间扮演中介角色。如果运行时因为某种原因需要将实例清除出内存，运行时环境还要负责持久化及还原实现的实例。(注：潜在地，会话可以有非常长的生命期限，且SCA运行时可以使用集群系统。在集群系统中给定的实例对象可以在集群期间的各个节点中移动，以达到负载均衡的目的)

1.2.6 访问回调服务

需要回调服务的Java实现类可以用@Callback注解来获取与当前调用相关的回调服务引用。在一个public或protected属性域以及setter方法上注入此回调服务引用。

1.2.7 未注解的实现的语义

对于未显式地使用@Reference或@Property注解声明的Java构件实现，本节定义了其属性以及引用的判定规则。

在不存在@Property和@Reference注解时，一个类的属性和引用按以下规则定义：

1. 不包含在任何使用@Service注解的接口中的public setter方法
2. protected setter方法
3. 无对应public或protected setter方法的public或protected属性域

如下规则用于判断一个未注解的属性域或setter方法是否是一个属性或引用：

1) 如果Java类型是由@Remotable注解的接口，那么就是引用

2) 否则，如果Java类型是一个数组，且此数组元素的类型是由@Remotable注解的接口，那么就是引用

3) 否则，如果Java类型是java.util.Collection或其子接口或子类，并且collection的参数化类型是由@Remotable注解的接口，那么就是引用。

4) 否则就是属性。

1.2.8 在装配中指定 Java 实现类型

如下是为Java实现类型定义的实现元素的模式（schema）：

```
<implementation.java  class="NCName"  />
```

Implementation.java元素有如下属性：

- class(必须) –实现的Java类全限定名（即带包名的Java类名）。

1.2.9 指定构件类型（component type）

对于Java实现类，构件类型一般都是直接由Java类内省而来。

在配置文件中对构件类型的指定是可选的。component type配置文件由加载Java类的同一个类加载器发现。配置文件必须在与实现的命名空间一致的目录下，并与Java类名相同，且以.componentType扩展名替代.class扩展名。

构件类型配置文件如何与从构件实现反射得出的构件类型信息相结合，其规则在《SCA装配模型规范》中做了描述。如果构件类型信息与实现冲突，则产生错误。

如果构件类型配置文件是用WSDL接口指定的服务接口，那么Java类应该实现由JAX-WS的WSDL到Java接口映射产生的接口。请查看SCA Java通用注解和API规范[2]中的“WSDL2Java和Java2WSDL”一节。

2. 附录

2.1 参考文献

[1] SCA Assembly Specification

http://www.osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf

[2] SCA Java Common Annotations and APIs

http://www.osoa.org/download/attachments/35/SCA_JavaCommonAnnotationsAndAPIs_V100.pdf

后记：SCA 中文规范项目

声明

SCA相关规范的中文文档得到 OSOA Chinese Community (OSOA中文社区) 的直接授权和大力的支持，其目的是在中文世界推广优秀的SOA相关技术标准。本次翻译活动由满江红开放技术研究组织(http://www.redsaga.com)和OSOA中文社区(http://www.osoa.org/pages/viewpage.action?pageId=416) 共同发起、组织，将翻译SCA的绝大多数规范，并得到goCom社区(http://www.gocom.cc/)的赞助。我们在此郑重宣布，本次翻译遵循原Service Component Architecture Specification的授权协议。在完整保留全部文本包括本版权页，并不违反Service Component Architecture Specification协议的前提下，允许和鼓励任何人进行全文转载及推广。我们在此宣布所有参与人员放弃除署名权外的一切权利。

致谢

SCA 规范的翻译对很多人来说都是一个挑战，再次感谢所有参与翻译、评审工作的同学们，没有你们的辛勤劳动，也就不会有中文规范的面试。

参与人员

规范分配情况，最后的校对和统稿由管理员统一完成。

Specification	翻译	一审	二审
SCA Assembly Model V1.00	ligang1111	wangfeng	wangfeng
SCA Policy Framework V1.00	liang_ma,max	pesome	needle
SCA Transaction Policy V1.00	max	jiangxd	guangh
SCA Java Common Annotations and APIs V1.00	ligang1111	HiugongGwok	dc
SCA Java Component Implementation V1.00	ligang1111	pinelygao	nekesai
SCA Spring Component Implementation V1.00	ligang1111	pinelygao	Caozw

SCA BPEL Client and Implementation V1.00	jiangxd	hongsoft	hongsoft
SCA C++ Client and Implementation V1.00	SpringWater	SpringWater s	SpringWater s
SCA Web Services Binding V1.00	xichengmylove	hongsoft	hongsoft
SCA JMS Binding V1.00	YuLimin/Echo	hongsoft	pegasus
SCA EJB Session Bean Binding V1.00	max	hongsoft	yanfei
SCA JCA Binding V1.00 （暂不发布）	bsspirit	jiangxd	benja

参与人员列表：

网名	姓名	网名	姓名
billytree	冯博	guangh	光华
ligang1111	李刚	Caozw	小曹
xichengmylove	严永华	yanfei	晏斐
pesome	张俊	benja	老贾
max	孙浩	HiugongGwok	郭晓刚
needle	needle	bsspirit	张丹
YuLimin	俞丽敏	Echo	杨春花
wangfeng	王锋	pinelygao	高松
hongsoft	洪波	liang_ma	马亮
jiangxd	姜晓东	Dc	王葱权
SpringWater	张世富	pegasus	马捷
chrischengzh	chris	nekesai	蔡永保
jiaoly	老焦	keqiang	克强

下列人员报名，但由于种种原因没有参与，在此感谢：

网名	姓名	网名	姓名
jetyang_1	杨文明	cenwenchu	岑文初
lafay	叶江	Eric	孟庆余
laodingshan	丁跃斌		

项目历程

SCA 规范一期 翻译项目进度汇总

2008 年 03 月 17 日

3 月 17 日，SCA 规范一期的翻译项目正式启动，Wiki 中增加了相应的 SCA 板块和对应的几个栏目。在 JavaEye 和 goCom 上发了招募贴，在满江红邮件列表中发了封邮件。

希望有更多的同学能够参加到这次的项目中，积极报名，争取能早日高质量地完成所有的工作，让更多人能享受到我们的劳动成果～

2008 年 03 月 22 日

相当郁闷，我在 JavaEye 上的帖子竟然被隐藏了，也忒欺负人了。经过沟通之后，终于又恢复了，又在 JavaEye 发了一遍新闻，呵呵。

2008 年 03 月 28 日

通过这段调查，发现 SCA 规范和 Spring 比较，相当于 阳春白雪 VS 下里巴人， 研究规范的都是比较专业的人士。 看来我们的翻译工作很有必要啊，在国内推广 SOA。

2008 年 04 月 02 日

昨天是愚人节，结果自己被愚弄了一把，看到 Spring 被微软收购了，真是"很傻很天真"。

鉴于这次规范的翻译的特殊性，有相当多的规范已经有中文草稿，所以没有采取其他文档翻译按章节划分的做法。

截至今天为止参加人数已经有 10 几个人，一期计划翻译的 9 个规范都已经被领走了，ligang1111 同学一个人承包了 5 个规范，而且都是分量最重，而且内容最多的部分。最后一个 JMS binding 规范翻译是被 俞丽敏夫妇领走的，革命伴侣，令人羡慕！

希望有更多的人参与进来，现在正在招募一审，二审人员。

2008 年 04 月 06 日

CVS 服务器已经设置好，上传了规范英文文档，相关翻译人员分配了 CVS 帐号，大家可以开始干活了，呵呵。

2008 年 04 月 08 日

刚同学已经完成 SCA Spring Component Implementation Specification V100 翻译，在这里向他的辛勤劳动表示祝贺！

他为我们已经开了一个好头！

其他已经领取任务的同志们加油啊！

2008 年 04 月 10 日

昨天满江红网站出了故障访问不了，还好今天已经正常了。

xichengmylove 同学已经完成 SCA Web Services Binding V1.00 翻译，max 已经完成领取任务的 30%．在这里向他门的辛勤劳动表示祝贺！

其他已经领取任务的同志们加油啊！请及时上报翻译进度。

2008 年 04 月 11 日

我们又有新鲜血液注入拉： 让我们欢迎 jiangxd，SpringWater 。他们分别承担了 SCA BPEL client，SCA C++ Client 的翻译工作，鼓掌！

SCA BPEL Client and Implementation V1.00	jiangxd			已领取	15	2008.4.11
SCA C++ Client and Implementation V1.00	SpringWater			已领取	70	2008.4.11

2008 年 04 月 21 日

前面一周出差了，回来发现很多文档都翻译完了，但是有些翻译文档没有上传，请大家及时上传，有问题直接和我联系。

众人拾柴火焰高！原定二期翻译的 SCA C++ Client and Implementation V1.00 规范已经完成 70%的翻译，向 SpringWater 致敬！

目前比较头疼的是，一审二审工作，还望群策群力。

2008 年 04 月 24 日

昨天下午去听了 SOA 中国的关键任务 上海站 报告，会上有个仁兄替我们 SCA 翻译作了个小广告，弱弱的问下，是那位啊？

感谢李刚同学！ 翻译完成 **SCA Assembly Model V1.00** ，分量最重的这块。

让我们欢迎 teamlet！他将负责 **SCA Assembly Model V1.00** 的一审工作！

2008 年 04 月 28 日

SpingWater 今天将 SCA_ClientAndImplementationModel_Cpp-V100 翻译文档发给我了，我看了一下，其实我们计划的翻译阶段即将完成了！

再次感谢所有参与者的辛苦劳动！

现在面临的问题是 翻译文档的一审、二审工作，我将联系 OSOA 中文社区、Tuscany 中文社区、goCom 和 JavaEye 社区的 SOA 爱好者，也向所有的 SOA 爱好者发出邀请，希望大家热情参与！

2008 年 04 月 29 日

laodingshan 同学今天领取了 SCA ComponentImplementation_V100 和 SCA_SpringComponentImplementationSpecification-V100 的一审工作，虽然他是个老同志，但是参与 SOA 的热情很高！

对他的加入表示热烈欢迎！

2008 年 05 月 15 日

今天有了个想法，这次翻译工作我们也能拉到一部分赞助，因此我想征求大家的意见，设立一个伯乐奖，对于推荐熟悉 SOA 相关技术，参与标准翻译"千里马"的"伯乐"也给与这次翻译活动的纪念品，希望大家积极推荐！

2008 年 05 月 19 日

沉痛悼念汶川大地震死难同胞！

2008 年 05 年 20 日

liang_ma 同学主动承担了 **SCA Policy Framework V1.00** 规范的翻译工作，这个规范比较长，有 46 页，让我们为他加油打气！

2008 年 05 月 27 日

SCA Java Common Annotations and APIs V1.00 已经翻译完毕，祝贺 ligang1111；

HiugongGwok 同学将承担一审工作，欢迎 HiugongGwok ！

bsspirit 同学将承担 SCA_JCABindings v 1.0 的翻译，欢迎！

2008 年 06 月 05 日

这一周，大家很忙啊，进度没有及时汇报，兄弟们要加油啊，一鼓作气！

提前预祝端午节快乐！

2008 年 06 月 12 日

这是第一个法定的端午节假期，我们勤劳的 jiangxd 同志加班加点，完成了 **SCA Transaction Policy V1.00** 的一审工作。在此表示祝贺！

bsspirit 同学承担 SCA_JCABindings v 1.0 的翻译进度也很快，工作量已完成 70%了。

2008 年 06 月 17 日

bsspirit 同学完成 **SCA JCA Binding V1.00** 的翻译工作，在此表示祝贺！

由于 laodingshan 同志自身工作很忙，所以他负责的 SCA ComponentImplementation_V100 和 SCA_SpringComponentImplementationSpecification-V100 一审工作现在由 pinelygao 接替。

pinelygao 同志有着 8 年的 J2ee 开发、架构经历，将会给予我们很大的支援！

2008 年 06 月 20 日

在 **SCA Policy Framework V1.00** 规范翻译中，liang_ma 同学可能遇到了困难。进度比较缓慢。

我们不得不承认，这对他是一个挑战，而且他已经翻译了 40%了 ！！！

这个规范是最后一个正在翻译的规范了。我们惊喜的发现，现在翻译的规范数比我们原来计划的要多很多了。

在我们为他加油打气的同时，我们也希望，期待援助：圈内人士，如果有对此感兴趣，欢迎积极加入，SCA Policy Framework V1.00 是一个很重要的规范，也是下一步研究的热点。 规范总共 46 页。

希望大家踊跃认领，支持！

补充：伟大的 jiangxd 同志又承担了 **SCA JCA Binding V1.00** 一审工作，我已经不知道说什么感谢好了。。。。。

2008 年 06 月 23 日

max 同志将承担起 **SCA JCA Binding V1.00** 的剩余翻译工作。这样我们的工作将可以 7 月份完成。届时组织一次集体规范的二审工作。

接下来，SCA 相关规范的中文版将面世。

请 liang_ma 同学尽快将你已经翻译的部分上传或者发给我。

SCA Policy Framework V1.00 一审已经被 pesome 领取了，感谢 pesome ！！！

2008 年 06 月 25 日

今天把 liang_ma 负责的 SCA_Policy_Framework_V100 翻译上传了，他已经翻译到了 1.4 节。

max 同志终于可以开始工作了。

感谢天，感谢地，看来原定计划 不会难产了。

2008 年 06 月 26 日

今天是个好日子。

喜报！

hongsoft 同学负责的 SCA BPEL Client and Implementation V1.00 ， SCA EJB Session Bean Binding V1.00 一审完毕。

pinelygao 同学负责的 SCA Java Component Implementation V1.00 一审完毕

而且，hongsoft 同学还将接替 needle,完成 SCA Web Services Binding V1.00 和 SCA JMS Binding V1.00 的一审工作。

向勤劳的 hongsoft 大厨表示致敬！！！

2008 年 07 月 07 日

新的规范出来了，**SCA Java EE Integration Specification V1.00** ，看了下，ORACLE 和 BEA 的人加起来比 IBM 还要多。没有看到 Primeton 的人，看来我们中国厂商要从标准跟随者到领导者，还要多多努力。

SCA Policy Framework V1.00 的翻译工作非常迅速，感谢 max 的辛苦工作，今天他告诉我只剩下 1.4 节和 liang_ma 交界的地方了。

今天是鄙人的生日，假公济私一下，哈哈。。

2008 年 07 月 10 日

HiugongGwok 承担的 SCA Java Common Annotations and APIs V1.00 一审工作已经完成。

而且，max 同志承担的 **SCA JCA Binding V1.00** 翻译已经高质量的完成，细心的他将在今晚自己审查后，上传。对 max 同志的高度责任心，一丝不苟的认真态度，值得我们学习。

再次感谢 2 位同志，胜利的曙光已经越来越近了。

还有一个好消息，我已经联系了相关赞助，将为我们这次的 SOA 规范翻译活动出专门的纪念 T 恤，以为各位同志的辛勤工作给少许心理上的安慰。

2008 年 07 月 11 日

max 同志承担的 **SCA Policy Framework** 已经上传 CVS，pesome 同志要接着进行"火炬"传递，完成一审工作。

SCA JCA Binding V1.00 一审工作在伟大的 jiangxd 同志的辛苦工作下已经完成，他自己本身工作也很忙，在经常加班的情况下，还认真审阅了规范的翻译，进行了大量认真、细致的修改、校正工作。

胜利的曙光。。。。。。

2008 年 07 月 14 日

wangfeng 同学已经完成了 **SCA Assembly Model V1.00** 这个最长规范的一审工作，鉴于该同志一如既往地兢兢业业，经研究决定，对该同志进行通报表扬，呵呵！

2008 年 07 月 21 日

pesome 同学负责的 **SCA Policy Framework V1.00** 一审工作已经完成 50%，预计将于本周末全部完成。

这样我们将在 8 月 2 号组织一次集中的二审工作，这样整个一期规范将于 8 月中旬正式发布。

2008 年 07 月 23 日

昨天当了一次黄世仁，呵呵。

感谢 pinelygao 同学提交了 **SCA Java Component Implementation V1.00** ，**SCA Spring Component Implementation V1.00 一审文档。**

2008 年 07 月 28 日

pesome 同学负责的 **SCA Policy Framework V1.00** 一审工作已经完成,祝贺阿

一个不好的消息是 SCA JCA 规范需要因为翻译质量不过关，需要重新翻译

好消息是二审工作已经全面铺开了

2008 年 08 月 18 日

刘翔退赛了。

我们的 SCA 规范翻译二审工作还在继续。。。

SCA 规范目前正在进行紧张的二审工作，在欣赏到众多志愿者翻译成果的同时，也发现了不少的问题。

在此，我们诚征志愿者，对有问题的中文规范进行修改，我们提供中文版规范和翻译中问题的批注供您参考。并且你将会署名在发布的规范中。
另外，对于参与翻译工作的人员，我们提供了一些纪念品。目前的想法包括：

1. **按翻译页数奖励 goCom 币，每页奖励 goCom 币 10 分。**
（用此 G 币可以在 goShop 商城购买商品，凡卓越网上能看到的商品都可以在 goCom 上通过获得的 G 币进行购买）

2. **打造成为 goCom 专家组成员。**
（goCom 近期将推出专家组概念，专家组成员将在 goCom 网站专家页面享有独立页面用以展示其个人内容，goShop 商城中享受更高折扣，各级评比中享受 10%的加分，北京用户免费参与 goCom 户外活动)

3. **邀请参加 goCom 在线技术日活动，每次活动可奖励 1500goCom 币。**
4. **赠送最新银弹杂志与 goCom 纪念 T 恤。**

2008 年 08 月 29 日

日志好久没有更新了。二审工作确实比较辛苦，跟每个规范的翻译质量有很大关系，而且每个人对规范的理解也不一样。

郁闷。。。。

我们需要专业的 SCA 人员，作校对和最后的把关！

- 但是不管怎样，我们还要前行，接下来，将陆续发布规范，首先是装配规范和 java 注解和实现规范。*

希望大家多提宝贵意见，在拍砖的同时能够体会我们翻译者的苦与痛。

讨论、建议和勘误

进行规范的翻译确实是一个挑战，错误和翻译不当之处在所难免，请在尊重作者辛勤劳动的基础上，提出你们的宝贵建议。

我们在goCom OSOA中文专区（http://www.gocom.cc/modules/osoa/）上专门开辟了一个板块，欢迎指出错误和讨论。

技术圈子

欢迎您加入 **SOA技术圈子** http://groups.google.com/group/SOAer

SOAer 是一个 GoogleGroup，聚集了国内 SOA 方面的架构师、咨询师、技术专家和技术爱好者。主要活动包括：

1 SOA 相关开源的运作；

2 SOA 技术讨论；

3 SOA 相关热点问题的讨论；

4 SOA 规范：SCA/SDO 的翻译；

5 SOA 相关技术的高级培训；

6 特邀讲座及报道；

7 SOA 相关规范及书籍翻译；

我们本着“talk u like and do u like！”、“一起分享，一起成长！”的宗旨，一起探讨 SOA 的方方面面。

满江红

满江红开放技术研究组织(www.redsaga.com)长期以来致力于推动开放源代码事业在中国的发展。我们欢迎有能力、有热情的你加入我们,一起达到从未想过的高度！

招募

欢迎大家参与SCA、SDO后续规范的翻译。敬请关注：http://groups.google.com/group/SOAer