外文翻译

建立可扩展的远程教学门户系统

摘要-当应用程序不断增长时，web应用就变成了一个巨大的问题。尤其是当新的算法被测试比较时，这个新算法所带来的新功能应该能够从核心系统中分离出来。仅仅用标准框架来处理这个问题是远远不够的，而是需要为应用系统计划一个具有扩展性的框架

本篇论文描述了一个基于django开发框架的可扩展系统是如何设计并实施的。在实施系统的同时我们还避免了过量的编码。本架构的优点和缺点是由一个样式评判插件进行评测。同时，本文还展望了在未来，扩展框架开发中使用本框架的可能性

关键词：插件式系统，远程教学，django，评测

Ⅰ.介绍

当设计一个真是世界中被一大批用户所使用的互联网应用时，有一个稳定的基础架构是十分重要的。它一定要保证应用的基础功能能够无时不刻的以良好的状态运行，同时其他的附加功能的运作绝对不能影响到应用的基础功能的运转。

一个实现远程教学节点的科研项目，一定要能够比较不同解决方案之间的差别以及具备测试新算法的能力。对于一个远程教学节点项目而言，有两个主要的研究领域：第一个是这类应用服务需要与各类用户交互。用户交互是未来网络应用程序中的一个方面。【1】因此，必须要为这个整合社群提供一个简单结局方案，例如评分【2】和标注【3】功能。进而，这个功能应该被整合进入核心功能，以制造出流畅的使用体验。

第二个方面是很多的研究都在注重如何提高寻找数据的效率。针对搜索录像讲座的搜索算法的补充，其需要很多的大量的数据。这些数据不仅来源于声频，视频数据【4】，同样也可能由用户来提供。即便是静态数据，也应适用于这种算法。目前已经有许多研究来整合这些不同的方法【5】.现如今，很多使用网络语义技术所实现的技术已经被提出。

因此，本文从这些应用所提供的数据以及一些小定义的插件说起。之后谈论两种实现的架构，包括使用和不使用插件来比较这些插件命名的优点和缺点。最后给这些工作做一个未来的展望，本节点可以适用于整个架构。

1. 关于“远程任务”节点简介

已经实施的远程教学节点“远程任务”提供很多线上讲座和视频，他们之中绝大多数是由HPI主办的。这些视频是由一个音频流和两个视频流同时捕捉的，这些流可以分离出来编程讲座和数字化的演讲。这套从2002年就开始使用的捕捉系统使得我们不用费很多力气就能创造出大量的讲座录音。

视频门户本身就提供了将视频内容传播到全世界的平台，这能让所有人都能观看到各个研究机构的大部分讲座。视频门户上有大约3000个讲座可以被搜索到。为了让用户能够更加便捷的搜索到这些视频，很多讲座被剪裁成具有很多额外大量数据的小片段。

这些所有的数据为研究真实数据，真实用户的构建了一个很好的基础。

B. 什么是插件？

很多网络框架都使用术语插件，有时候也被称作附加功能，即便他们有时候并不能提供任何被称为插件的功能。这也正是很多网络应用开发者对于插件这个名词有误解的原因。在本篇论文中，插件被定义为与在经典软件开发领域中的定义一样。

一个插件可以被看作是一个特殊的界面描述。这个界面定义了一些可以被开发者用来进行扩展基础的软件的函数和类。拥有很多用途的大型软件会使用这个概念来使得软件更具有可扩展性。举一个例子，软件开发者可以给火狐浏览器写一个插件，这样能够使得浏览器的功能更加多样。软件甚至能全部由插件组成，例如eclipse。

这些界面给予了开发者编写函数的功能，并且能够自动的加入系统内部而不对任何系统核心功能做出改变。但是在网络应用开发中，插件这个名词经常被用作为库的概念，库也可以被当作插件插入系统并且被开发人员所用，但这并不是插件原有的意思。

事实上插件架构就是一个控制权反转。这一点不仅是写插件的作者知道，使用这些插件创作交互界面的人也知道这些插件将需要什么数据以及什么样的函数。但是他们并不知道这些函数将以什么样的方式实施。举个例子，一个交互界面开发工程师为了数据增强开发了一个交互界面。他仅知道他会得到增强的数据，但他并不知道谁会这么做以及这个到底是怎么实现的。

这种插件的定义已知存在许多各式各样的问题。很多项目【8】【9】都描述了他们如何将一个已有的插件实施到他们的项目中，或者调用一个已经存在的交互界面。不仅如此，很多人不知道网络开发者和大多数大网站的架构师不会将插件架构的机制的包含到自己的项目中。因此这需要人们亲自动手去建造。

Ⅱ. 经典实现远程教学门户的方法

“远程任务”门户已经上线运营了有几年了，并且也经历了一些主要功能上改变。在2009年夏天，这个系统可以全部使用心得技术重新启动。这次重新启动允许重新设计数据库。在使用旧的门户的经验基础上，我们决定创建更多的模型和独立的函数。在本章开始，有一个关于django框架的简要介绍。之后是介绍本项目如何在没有插件架构的情况下构造。

A．Django的概念

Django是一个开源的互联网应用开发框架。这是一个基于Python语言和MVC模式的项目。准确的说，他应该被称作Django框架中的MTV模式，因为应用中的各个部分被分为模式、模板、视图三部分。

这三个术语描述了Django开发框架中最重要的三个部分。一下地三个简短的描述能够给与人们对Django框架中这三个部分的作用有大致的理解。

模式

模式是数据的代表，他能够在各种不同形式的数据库中进行存储。因此，这是一个从数据库表到对象表索引和类的一个映射关系。模式同样提供了增加协助性函数的可能性。举一个例子，可能有人会通过这种方式写一个函数将日期转化为其他形式。

他们同样为一些抽象数据库样式设计和隐藏例如连接两个数据表之间的开销数据表提供帮助。

视图

视图的任务是收集需要的数据并且提供出来。因此，视图可以创建一个字符串包含HTML代码，或者使用包含HTML文件的特殊模板语法的模板。其中后者为更加普遍的做法。视图从模式中取出数据然后做好将数据提供给模板的准备。

视图同样也是统一资源定位器的接入点。在一个特殊的配置文件中，包含每个统一资源定位器应该调用视图中的哪个特定的参数。配置文件中同样提供创建为特定统一资源定位器创建抽象命名的可能性，这样开发者就不需要知道哪个特定的统一资源定位器最终被使用。而开发者这需要知道统一资源定位器的名字即可。

模板

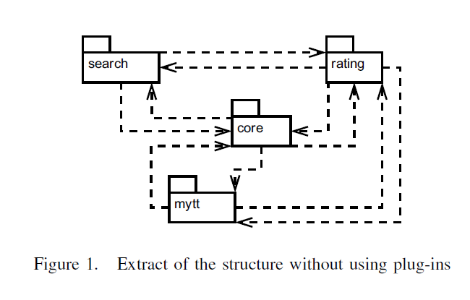
通常情况下，模板就是单纯的HTML语言文件，包含一些标签以便输出变量和便利列表。它同样也可以使用筛选器做一些附加样式，例如为时间戳变换格式。有了这些标签的帮助，程序员们就可以往模板里置入各式各样的函数了。

在django框架内部，模板可以被理解成像树形数据结构一样，拥有很多的节点。每个模板的标签就是这棵树的一个新的节点。这些模板的标签是可以互相嵌套的。因此这样一个复杂的标签书就这样被制造出来了。正因为这种结构，每个模板的标签都需要生成一个节点，来调用提交函数。更进一步，这个被创造出来的模板节点可以当作其他模板节点的孩子。

B．“远程任务”门户不使用插件架构的模式化架构

当设计一个应用时候，为不同的任务创建模式是一个很好的概念。这些模型的一个重要的功能就是能够设置访问限制并实现低耦合度。我们尝试按照任务划分模型时，结果分数的模块如下：登录模块，用户管理模块，神话模块，搜索模块和评分模块（见图1）。使用无插架解决方法，那么就会在几乎每个模块之间出现双向连接

这种现象的发生一方面是由于核心页面，例如应用的主入口界面需要调用增强模型页面，因此应用的用户可以调用所有的函数。一个这方面的例子就是当用户在主界面搜索对话时候，其就调用了搜索函数。而其他模型提供信息，称现在核心界面上也是完全有可能的。举个例子，一个讲座的评分应该在讲座界面上被显示出来，甚至可以被修改。



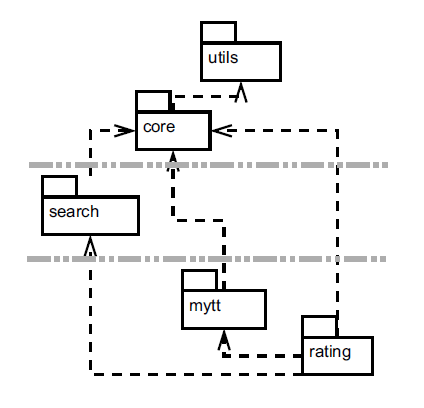
另一方面，附加的模型必须使用基础模型的数据。很多模型都是用自己函数总的视频和音频数据，这点非常显然。但他们也会使用核心模型中修改和展示数据的功能来实现一个一致的应用。

这些模型不仅仅与主模型之间进行交流，还会互相进行通信。举个例子，在普通搜索音频和视频的地方，会有一个搜索播放列表的功能。这个功能应该可以根据内容关联度的评分进行排序。这个神话模型，为所有已经登陆了的客户提供了特殊的函数，就需要与其他所有模型之间产生联系。

现存的问题就是，新的模型必须需要改变已有模型的数据。这很容易就造成巨大的应用复杂度的增长。而且也正因如此，在不影响其他组件的前提下，不可能去删除其他的模型。

III．插件式架构的概念

可扩展性应用的思路就是，最基本的模型不需要知道其他任何特定的模型。最终，它们会形成阶级性的模型，如图2（下图）所示。



正如你所见，只有特定的模型使用更为不同的模型的数据和交互界面。最基础的模型只提供能够被特定的模型所使用的插件点。

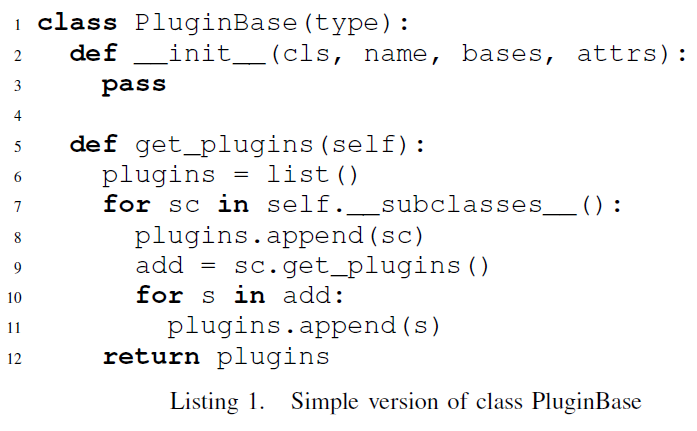
有了这样的结构，就能做到扩展现有的应用而不需要触碰任何核心的模型。同样，取出任何插件式模型也不需要动用任何其他的模型就能轻松完成。总的来说，插件式模型就是阶级树中的一个叶子。

为了完成这样一个架构，有两个关于概念的难题需要解决。首先是视图的延申：需要提供一个模板与插入级别机械之间的交互界面。其次是为多出来的数据设计方法。

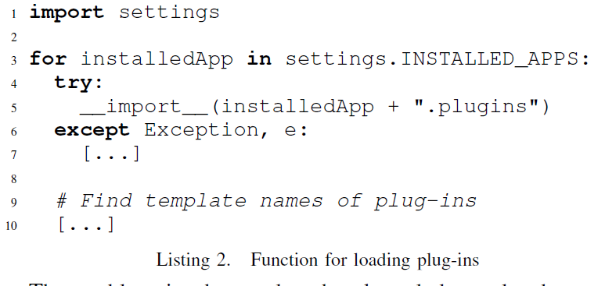
1. 扩展视图

Django框架没有针对设计插件扩展视图的概念。设计自己的插件系统对于开发者来说就是自己的任务了。

从Django的实现语言——Python开始入手解决这个问题是一个很好的方法。在Python中，有很多关于如何设计一个插入式系统的方法。一个简单的想法就是使用类中子类的方法。使用了这种方法，你只需要一个基本类，就可以借此询问它的所有子类。列表1是其一种实现方法



这个列表只包含了一个很小的片段。当然，如果子类能够非常好的适应并且很完美的形成，做一些额外的检测会更好。

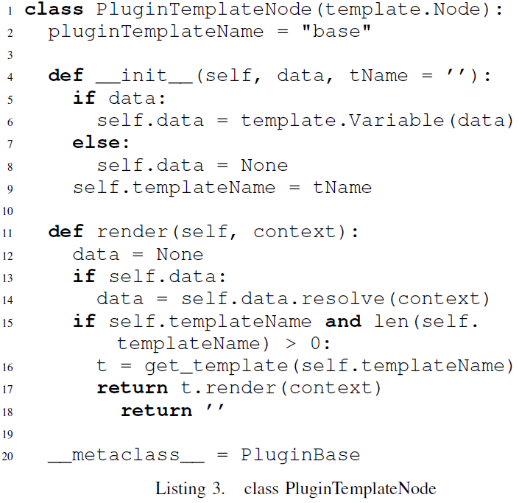


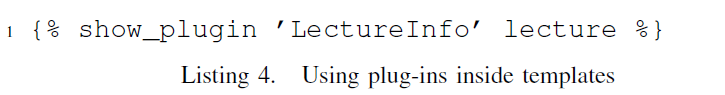
问题是，一个类只知道他们的子类什么时候被调用，在应用中的什么地方被调用了。因此，最方便的方法是保证这些子类在启动Django服务器的时候就被初始化完成。因此我们使用了“已安装的应用”的设置，如列表2所示。这其中包含了所有可用的模型以及用于其他模型初始化的模型

之后，所有已知的插件都可以被访问，并且在Python函数或类中被使用。解决模板类的插件问题，需要重新写一个特殊的模板标签。

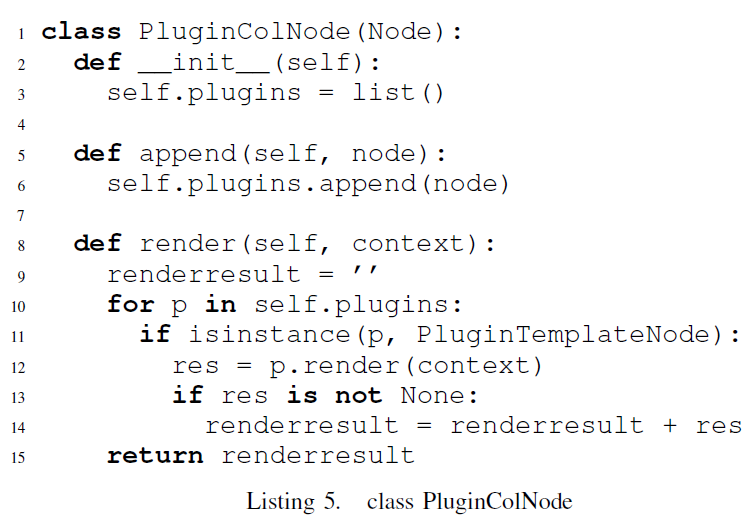
写模板标签的准备工作：模板标签是一个函数，返回一个模板节点对象。这些对象给予返回函数中必要的参数，他们在展示时，在网站浏览器内部为HTML创造数据。

因此，每个用于模板的插件类一定要继承模板节点类，这点非常重要。因此，写一个类似插件模板节点的基础类，包含一些将插件基础类当作大类的易用的函数，是十分方便的（如列表3所示）。

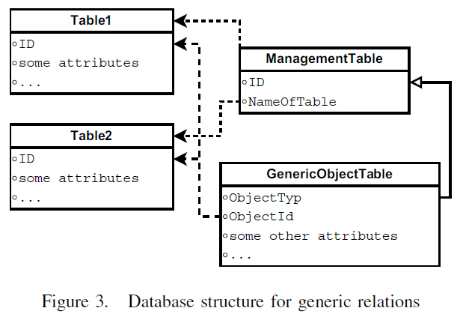


类变量“插件模板名称”用于获取模板标记唯一标识符。在列表4中，使用插件获取展示关于讲座的其他信息。对应的“插件模板节点”类会为“插件模板名称”设置“讲座信息”的数值。

因为插件可以创建多个模板节点，但只有一个模板节点可以被返回模板标记。解决这个问题最简单的解决方案就是：在调用自身函数时，收集所有的插件模板节点，并渲染他们。（如列表5所示）



有了这两个类，我们可以通过检索“插件式模板节点”所继承的所有差价类，并且将这些类附加到由模板标记的插件节点对象上，来轻松的编写通用模板标记。



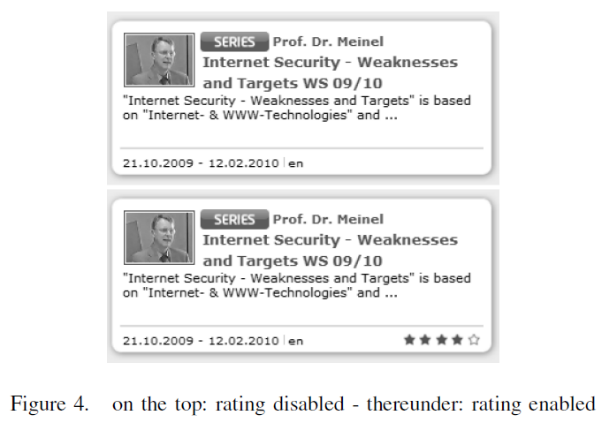
1. 模型的扩展

仅仅能够扩展应用程序的视图是不够的，因为通常情况下他们都需要使用额外的数据。而之前常用的数据库构造方法是构造一个新的表，并和原有的数据表建立外键以连接这些数据。只要你只想扩展一个表，那么这是一个很有效的构造方法。但是如果你想要扩展多个拥有相同功能的表，就必须使用泛型关系。（如图三所示）

这个概念是Django分布式的一部分，它允许多个数据表的外键保存表的类型和对象的id。因为Django需要处理所有的数据库请求，因此数据库结构和管理表对应用程序的开发人员都是不可见的。并且泛型的外键可以像普通的外键一样使用。

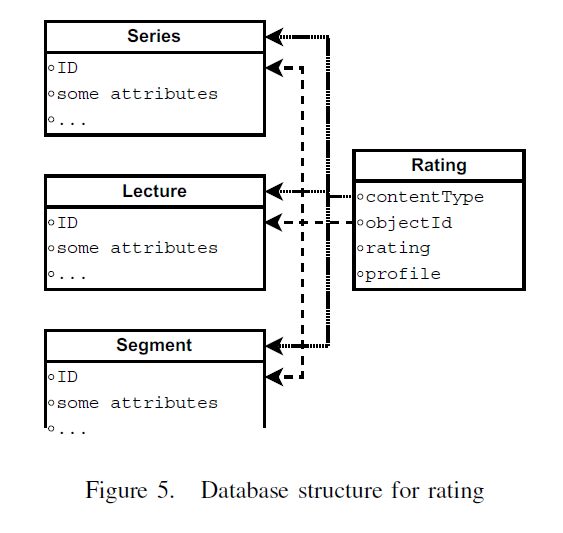
1. 例子：插件式架构的使用样例：评分函数

评分功能在远程教学门户中是使用插件式架构实现的，因为它并不是一个核心功能，因此只有部署在一个已经被实现的模型中才能与核心应用程序功能工作。（如图4所示）。在评价媒体的背景中，评分是对项目个人感知质量的量化。



在远程教学情景中，有几个层次可以被用来评分。通常远程教学门户网站由大部分嵌入子啊更大情景中的课堂录音而成，例如那些一整学期的课。更进一步，这些讲座通常被细分为更小的部分。这样做也是为了方便在需要下载内容的地方使用移动播放器，进行广播。并使用简单方法而且更加精确的收集和搜索元数据。由于三分层次都包含了远程教学的内同，因此它们之间可以单独进行评分。

评分会在新的从核心模型扩展出来的模型的帮助下存入数据库。这个模型评分结合了评级内容项的信息，发布福安与评级的用户信息，发布评级日期，和评级的值（如图5所示）。内容项通过内容类型和对象id的组合进行引用。这就产生了一个评分功能，它可以很容易的适应于系列，讲座，片段，还可以适用于心的内容类型，例如播放列表。

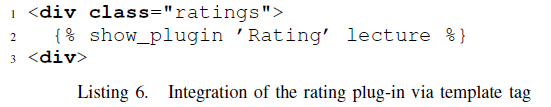


还有很多地方可以使用评级用来增强用户的交互界面。分级的结果应该显示在所有可以预览到内容的地方，例如视频存档及其类别中的搜索页，概述页。还应在与内同相关的所有可能评分的页面上给出评分，如课堂纤细信息页面和视频显示页面。这样才能确保用户能够轻松的访问并查看这新功能。评分页面应该检查登录状态，防止匿名用户的参与。

让用户管理自己在个人领域中的操作的“神话”门户中，必须有自己偷拍的显示。同时更改选票，删除选票的权利应该被允许。

所有插件的功能都是通过新视图扩展核心视图集合来实现的。所有插件都可以继承的接口是在应用程序的核心部分实现的。有两个不同的插件提供评级结果的显示和评级接口，每个插件在核心应用程序中实现一个接口。一个插件旨在预览页面上显示较小的评级结果（如图4所示）。第二个插件实现了结果和评级界面的正常大小的显示。它用于所有内容项和视频的详细页面。

通过使用模板标记将功能集中到标准模板中。在评级输出应该出现的地方调用评级插件并传递到包含应该显示其分数的内容项（见列表6）。



“神话”为子页面的集成提供了自己的插件接口。一个接口函数在“神话”中实现菜单连接，另一个在接口中实现可视连接摁扭。两个链接都指向插入式子页。两个评级插件继承自这些函数，将“管理我的评分”接口集成到“神话”模型中。

IV．优点及缺点

因为每个技术插件都有其优缺点，如果使用它们是一个好的选择，则取决于用例。我们收集了团队中10位开发人员的经验，他们从不同的架构知识开始，进而获得了这些事实。

以下的缺点在本项目中尤为突出：

·学习任务：培养时间增长。一个新的开发者必须去学习插件的基本概念，并且必须去教他新的方法的规则。因此这需要开发者更高的开发能力。

·小任务的开销：一个小小的连接可能需要实现的整个函数或类，而不是只需更改HTML文件。

·小项目的高复杂性：如果一个项目只有一两个函数，其可以承担插件并且已知这个项目不会继续增长，那么插件式架构就不需要了。

当然，它也有很多优点：

·容易进行功能的开关：如果有插件产生错误，那么很容易就可以将其禁用。禁用有缺陷的插件也可以自动执行。它可以很好的实现替换函数的功能。

·运行系统和开发状态的不同：它可以使得开发的函数和项目内的代码在上线的应用中禁止运行。

·模型分离：其中模型是分离的，因此可以从交互界面中分辨出其很容易外购一些任务。

·只需要很少的系统知识：开发插件的人员不需要了解核心系统，他们是需要知道他们所使用的交互界面的特点。这使得学习成本大大降低。因此，对于大项目，理解插件式架构的时间能够得到很好的补偿。

·复杂性的降低：在大项目中设计的开销是相对来说较小的。更进一步，它减少了在模板和视图中的冗余测试，以及项目的复杂性。

·对于不同的开发任务造成更小的影响：由于不是所有的开发者都要使用到所有的插件，让开发者忽视早期的平台是更为轻松的方式。

V．总结和未来工作

实际的实现已经进入门户并运行了半年多。下一步是将这些插件封装到易于使用的项目中，这样它们就可以独立于项目使用。本项目将分享到Django社区。者可以让更多的人得到插件架构的经验。这些经验会被收集，并由此检查这些方法是否也能适用于其他项目。同样，我们研究团队中的一个项目团队也在等待自己测试架构的结果

除了插架式架构应该成为之后工作的一个工具外。还有一些其他的想法的实施也能贯穿整个架构。

实现基本搜索特征是一个很大的任务。有两点需要完成：首先需要有很多不同类型的对象以备查找。在这个核心系统中，包含系列，讲座，片段和人，但以后还必须搜索播放列表和其他数据。

第二点是如何定义搜索工作。核心搜索能够搜索到讲座或系列的元数据。但是用很多元数据是以不同方式生成的。例如音频或OCR分析，这对于增强搜索引擎搜索到的可能性、提高查询效率都是十分重要的。

在实现这个基本搜索功能时，应提供不同类型的元数据。我们计划给予不同讲座或系列之间更多的关联信息。

插件也会带来其他的可能性，例如嵌入到更多的社区功能之中，以提供新的研究方案。如果这些数据能够有助于丰富我们门户的用户体验，那么我们允许用户使用标签提供更多的元数据并进行评估。