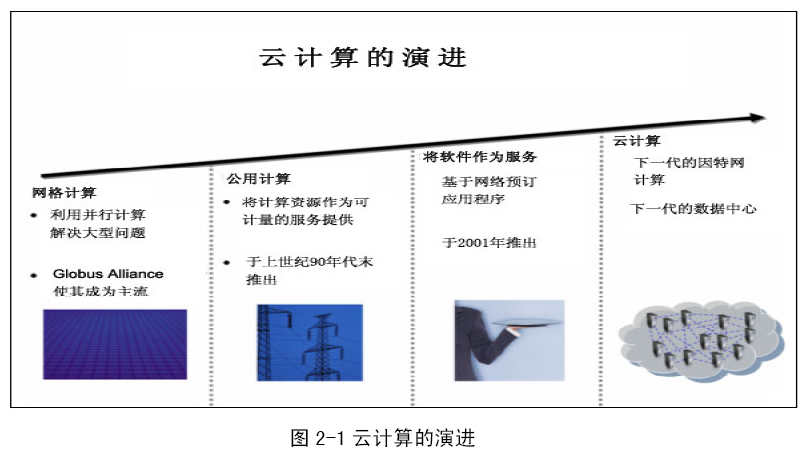
**第2章 系统框架设计与实现**

**2.1 系统框架设计**

2.1.1 云计算技术应用

2.1.1.1 云计算简介

云计算是在分布式计算（Distributed Computing）、网格计算（Grid Computing）和并行计算（Parallel Computing）基础上发展起来的，其前身是利用并行计算解决大型问题的网格计算和将计算资源作为可计量服务提供的公用计算，在互联网宽带技术和虚拟化技术高速发展后发展起来【】，云计算的演化过程如下图所示：



对于云计算的概念说法不一，但云计算所具备的特征如下：

1)能够整合大规模异构计算资源

传统的分布式计算一般仅能应用于一个小范围的计算网络(如局域网)，且对计算资源的同构性要求较高，难以处理在计算和存储能力、操作系统、开发平台等方面存在很大差异的计算资源。而通过云计算则可以整合分布在一个广阔地域内的、分属于若干个组织的计算资源，形成一个功能非常强大的计算和存储平台。

2)易于动态扩展

可扩展性是云计算与传统计算模式相比的最大优势之一。由于云计算能够集成硬件种类、网络类型、操作系统、软件平台等各不相同的各种计算设备，因此，在需要时云计算平台的计算和存储能力可以得到方便和快速的扩展。与传统计算平台需要几天甚至几个星期的系统升级时间相比，云计算平台的升级一般仅需要几分钟，且可以在不影响系统整体运行的情况下动态进行。此外，云计算平台可以建立在现有的硬件基础上，在升级时也只需按照需求增添相应的设备，而不需要像升级传统计算平台那样将设备完全更换，从而可节省大量硬件购置成本。

3)虚拟化与服务

虚拟化也是云计算的一个重要特征。无论一个云计算平台实际整合了多少计算设备，在用户看来其就是一个单一实体，也是获得计算服务的唯一接口。由于应用了虚拟化技术，云计算平台既可以将多个计算任务放在同一台功能强大的设备(如大型工作站)上运行，也可以将一个计算任务拆分成若干部分，分别在多台设备上运行。这样，就可以最大限度地利用系统内的闲置计算资源。此外，通过利用虚拟化技术，云计算平台可以根据客户的需求动态分配计算资源和构造系统平台。此外，若干设备的故障不影响云计算平台整体运行，也不会中断向用户提供服务。

2.1.1.2 云计算平台应用

通过分析云计算的特征，可以看出，对于面向社区的疾病诊断决策支持系统，通过应用云计算技术，首先可以解决过去由于分散式系统建设模式导致的系统建设周期长、多点维护成本高、故障处理缓慢的问题，全面提升了xxx；其次，云计算将海量存储计算的资源汇集成资源池，各应用系统可以根据需求获取动态地部署、配置及回收计算机资源；最后在云计算解决方案中，运算和存储远程集中托管，与具体应用环境隔离，在解决数据共享问题的基础上还有效保证了数据的安全性。因此，采用云计算技术建设面向社区的疾病诊断决策支持系统较之传统的构建模式有着明显的优势，即降低了社区医疗机构的IT负担，又xxx

从体系结构的角度来看，一个云计算系统是为配置各种计算机软件、硬件和网络资源、运行客户应用程序而搭建的平台。美国国家标准和技术研究院的云计算定义中明确了三种服务模式：

* 软件即服务 (SaaS)：消费者使用应用程序，但并不掌控操作系统、硬件或运作的网络基础架构。是一种服务观念的基础，软件服务供应商，以租赁的概念提供客户服务，而非购买，比较常见的模式是提供一组帐号密码。例如: Microsoft CRM与Salesforce.com
* 平台即服务 (PaaS)：消费者使用主机操作应用程序。消费者掌控运作应用程序的环境（也拥有主机部分掌控权），但并不掌控操作系统、硬件或运作的网络基础架构。平台通常是应用程序基础架构。例如: Google App Engine。
* 基础架构即服务 (IaaS)：消费者使用“基础计算资源”，如处理能力、存储空间、网络组件或中间件。消费者能掌控操作系统、存储空间、已部署的应用程序及网络组件（如防火墙、负载平衡器等），但并不掌控云基础架构。例如: Amazon AWS、Rackspace。

是基于PaaS构建面向社区的临床决策支持软件服务，主要精力在于构建应用程序的业务逻辑上，而不在于基础平台的构建、运维以及平台的扩展性和灾难防备上。另一方面由于医疗数据涉及隐私问题，放在公有云上存在安全性的问题。因此，本文旨在提供基于云计算的系统框架的技术方案。

2.1.2 架构模式选取

2.1.2.1 B/S架构简介

C/S结构，即Client/Server（客户机/服务器）结构，它通过将任务合理分配到Client端和Server端，利用两端的硬件环境，降低系统的通讯开销， 为早期软件系统首选的架构。

随着Internet技术的兴起，出现了C/S结构的一种变化或者改进的结构，是 B/S结构，即Browser/Server（浏览器/服务器）结构。在这种结构下，用户界面完全通过WWW浏览器实现，主要事务逻辑在服务器端实现，仅少部分事务逻辑在前端实现。它主要是利用了不断成熟的WWW浏览器技术，结合浏览器的多种Script语言（VBScript、javascript）和ActiveX用通用浏览器就实现了原来需要复杂专用软件才能实现的强大功能，并节约了开发成本，是一种全新的软件系统构造技术。

目前大部分的临床决策支持系统采用C/S结构开发实现，但对于面向社区的疾病诊断决策支持系统，应用的环境为广域网，用户数量庞大，分散各地。然而C/S结构采用的是Intranet技术，适用于局域网环境，在用户数量增多时，性能会明显下降，而B\S架构可满足广域网环境大量的用户，可动态配置ＷＥＢ、服务器满足访问量增多时的性能稳定。另一方面，B/S架构在系统的维护与升级的简易性方面也有不可比拟的优势，对于分布在各地的社区机构来说减少了IT相关的工作量。因此，系统选用B/S架构作为系统框架的开发架构。

2.1.2.2 MVC模式简介

MVC的英文是Model-View-Controller，概念上将视图从流程控制、业务逻辑独立开来，并定义了相互间作用的机制，使得各个模块的开发相对独立，这样一个应用被分成三个层——模型层、视图层、控制层。

1．Model

事务逻辑模块，通常是系统的核心部分，它包含数据操作和业务规则。它表示的是解决方案空间的真正的逻辑。模型的构成与具体的应用问题紧密相关。通常模型包括数据访问、业务逻辑和业务规则。

2．View

用户视图模块，这是用户界面部分。此模块承担连接用户和应用程序之间的接口角色，是一个系统中与用户关系最为密切的部分，也是需求变化最容易发生的部分。一方面，View模块将用户的请求传递给应用程序，触发应用程序对用户请求的处理逻辑；另一方面，View模块将请求的处理结果返回给用户。

3．Controller

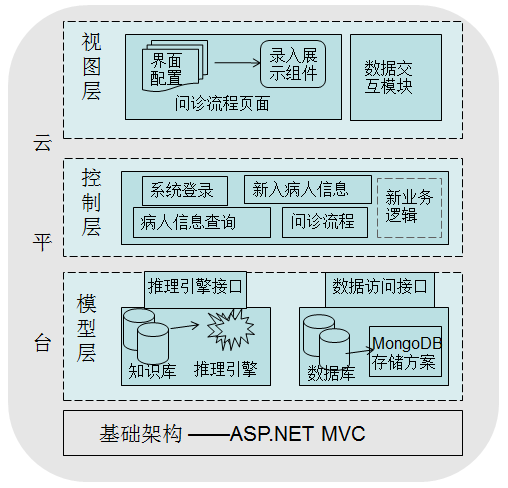
流程控制模块，这是前面两者的接口。控制器根据用户的操作判断所要执行的业务逻辑，关主要用于接收和转发用户请求。通常，从视图接收到用户请求后根据用户提供的业务信息传递给相应的模型处理，再将结果结合相应的视图返回展示给用户。



综上，MVC的工作流程是：控制器接收用户请求并调用相应的模型进行处理；模型处理用户请求并返回数据；控制器利用返回的数据渲染相应的视图展示给用户。

采用MVC模式进行系统框架的开发，可以降低数据描述和应用操作的耦合度，使系统结构清晰，各模块内的组件可复用性易于实现。而且各个部分的分离使得系统的维护性和扩展性得到极大的提高。

综合以上分析，系统采用云计算平台作为基础 ，将MVC模式应用于B\S 架构的系统框架的设计，因此系统的整体框架图如下所示。



**2.2 核心模块设计**

2.2.1基于Web Service的推理引擎模块

推理引擎部分是知识规则应用于问题求解的载体，根据系统输入数据的当前内容，利用知识库的内容，按照一定的推理策略，去解决当前的问题。由于系统需要解决的不是使用单个推理引擎解决某种疾病的诊断问题，因此推理引擎模块需要提供多种推理方式的统一的推理引擎接口。然而由于推理方式种类繁多，已有的实现方式多样，基于各种编程语言和平台。而基于web service 的推理引擎模块将推理的内部实现封装起来，以服务的形式向上层提供推理功能。也就是说，在保存接口一致的前提下，推理方法的调换或者更新都可以轻松实现，为简化系统后续的维护和更新的打下了基础。此外，web service的使用和管理都独立于系统其它部分，而且与开发语言无关，可以满足开发人员的协同工作和多样性的需求。

2.2.1.1 Web Service 概述

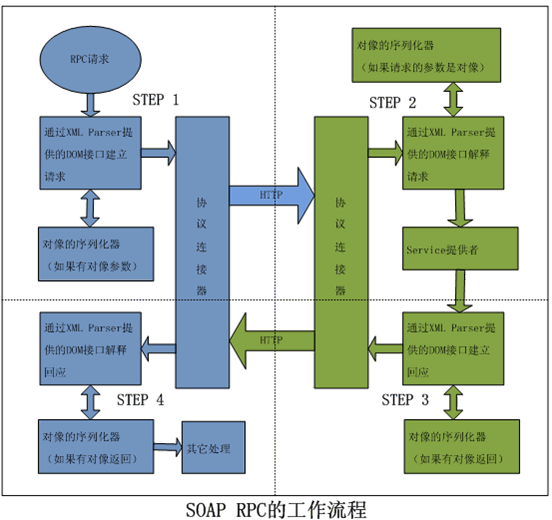
Web Service也叫XML Web Service，是一种可以接收从Internet或者Intranet上的其它系统中传递过来的请求，轻量级的独立的通讯技术。通过SOAP在Web上提供的软件服务，使用WSDL文件进行说明，并通过UDDI进行注册。它包含以下关键的技术：

（1）XML：(Extensible Markup Language)扩展型可标记语言。面向短期的临时数据处理、面向万维网络，是Soap的基础。

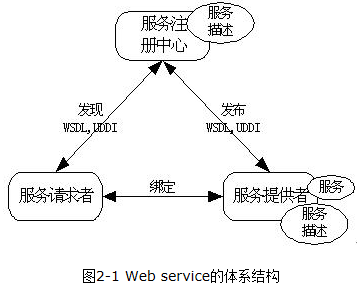
（2）Soap：(Simple Object Access Protocol)简单对象存取协议。是XML Web Service 的通信协议。当用户通过UDDI找到你的WSDL描述文档后，他通过可以SOAP调用你建立的Web服务中的一个或多个操作。SOAP是XML文档形式的调用方法的规范，它可以支持不同的底层接口，像HTTP(S)或者SMTP。

（3）WSDL：(Web Services Description Language) WSDL 文件是一个 XML 文档，用于说明一组 SOAP 消息以及如何交换这些消息。大多数情况下由软件自动生成和使用。

（4）UDDI (Universal Description, Discovery, and Integration) 是一个主要针对Web服务供应商和使用者的新项目。在用户能够调用Web服务之前，必须确定这个服务内包含哪些商务方法，找到被调用的接口定义，还要在服务端来编制软件，UDDI是一种根据描述文档来引导系统查找相应服务的机制。UDDI利用SOAP消息机制（标准的XML/HTTP）来发布，编辑，浏览以及查找注册信息。它采用XML格式来封装各种不同类型的数据，并且发送到注册中心或者由注册中心来返回需要的数据。



Web服务的体系结构是基于Web服务提供者、Web服务请求者、Web服务中介者三个角色和发布、发现、绑定三个动作构建的。如下图所示:



Web服务提供者是Web服务的拥有者，它耐心等待为其他服务和用户提供自己已有的功能；Web服务请求者是Web服务功能的使用者，它利用SOAP消息向Web服务提供者发送请求以获得服务；Web服务中介者的作用是把一个Web服务请求者与合适的Web服务提供者联系在一起，它充当管理者的角色，一般是UDDI。这三个角色是根据逻辑关系划分的，在实际应用中，角色之间很可能有交叉：一个Web服务既可以是Web服务提供者，也可以是Web服务请求者，或者二者兼而有之。显示了Web服务角色之间的关系：其中，“发布”是为了让用户或其他服务知道某个Web服务的存在和相关信息；“查找（发现）”是为了找到合适的Web服务；“绑定”则是在提供者与请求者之间建立某种联系

2.2.1.2 模块内部流程设计

为了满足系统的可扩展性需求，需要将推理实现的细节封装在推理模块内部，因此设计了如下的流程，Web Service 的函数接口为 Json Diagnosis(Json Data)。



1. SearchTemp: 接收到推理请求后，在注册表信息中查找对应疾病诊断的推理的以文件形式存储的知识和数据模型。如果存在该疾病的信息则将相关文件读入引擎，为推理作数据预处理准备；如果不存在则返回错误信息。
2. DataAdapter：将输入的数据根据推理引擎的需求处理为特定的数据格式传送给Inference进行推理，对于包含无效输入的数据则返回错误信息，中断流程。
3. Inference：根据输入的知识和数据根据一定的方法进行推理，推理结束后输出结论，通过DataAdapter返回给WebService的请求方

通过以上的方式，实现了对不同疾病的数据和知识库的开放性，知识库的更新和数据需求的变化都可以通过修改外部文件完成。

2.2.1.3 推理引擎服务周期概述

一个Web服务的周期主要包括服务的构建、构建完后服务的部署、服务的运行以及服务的维护四个阶段。从这四个阶段可以看出，推理引擎服务的创建过程其实与其他任何类型的应用程序的创建过程并没有什么区别，都是经历从应用程序的设计、代码的编写与部署到应用程序的最终使用的一个周期。

1)服务的构建。服务的构建包括服务的开发和服务的测试、服务接口描述的定义。可以通过根据新的推理需求创建新的推理引擎服务或者可以根据其他的推理引擎服务和应用程序组合成新的推理引擎服务。

2)服务的部署。服务的部署主要是在推理引擎服务实现完后，服务提供者向服务注册中心发布服务接口，并将服务的可执行文件部署在执行环境中。

3)服务的运行。在运行阶段，由于服务已经被部署，服务请求者就可以查找服务、调用服务。系统通过动态调用的方式，根据推理的方法的需求，获取相应的推理服务。

4)服务的维护。维护阶段包括管理和更新推理引擎服务应用程序，主要是对于推理方法或者知识库的改进和更新。

2.2.2 基于MongoDB的数据访问模块

医疗数据由于本身的复杂性，往往不是典型的关系型数据，数据类型繁杂且具有较多的层级结构。而关系型数据库的中的“行（row）”概念划定了严格的字段，MongoDB的“文档（Document）”模型灵活可变，通过内嵌文档或者数组的方式可支持复杂的层次关系。另一方面，就临床医疗数据的来说，需要对于医疗数据进行有效的检索及统计分析。MongoDB在文档的查询上，支持通用的辅助索引和复合索引，能够提供快速方便的查询；而统计分析方面，MongoDB支持MapReduce和其他聚合工具，对于大量数据的统计分析及数据挖掘提供良好的编程基础。

由于决策支持系统的实施和评估本身存在一个决策模型构建、系统构建应用、有效性评估、结果反馈、系统更新完善的循环过程。随着系统的不断更新，临床数据采集和存储的种类和数量随着知识的更新也会不断变更。这对于数据模型需要事先精确定义的RDBMS，哪怕是对数据模型的很小的变更，也需要数据库的重新设计实施。而数据模型的限制这一点上，NoSQL数据库要宽松的多，或者完全不存在，MongoDB允许应用在一个数据单元中存入它想要的任何结构，减少数据模型修改带来的麻烦，能够应对快速变化的数据需求。

2.2.2.1 MongoDB概述

随着信息技术的发展，网络数据越来越偏向非结构化，数据并发读写要求高，且要求数据有一定的扩展性。针对这类数据的特点，NoSQL应运而生。NoSQL是Not Only SQL的缩写，是对传统关系型数据库以外的数据库统称，他们不遵循经典的RMDBS原理，且常与Web规模的大型数据库相关。NoSQL数据库不需要特定的表结构，通常不支持表的连接操作，不支持完整的ACID属性，而且一般拥有强大的可扩展性。下面是计算机学家Sourav Mazumder对NoSQL数据库的一个比较严谨的描述[19]：

（1）不像关系数据库会建立一个固定的关系模式，NoSQL会建立一个松散的、易扩展的数据模型来进行数据建模，比如文档、列形式。

（2）系统设计会按照CAP[20]原则，在跨多个节点之间建立数据分布式模型，使系统具有较强的水平扩展特性。

（3）有在磁盘以及内存中持久化数据的能力。

（4）进行数据访问的时候支持多种NoSQL接口。

根据特定场景和应用需求，NoSQL主要分为三类：一是面向高性能并发读写的Key-Value数据库，Key-Value数据库的主要特点是具有极高的并发读写性能，Redis、Tokyo Cabinet就是这类的代表；二是面向海量数据访问的面向文档数据库（Document Store），这类数据库的特点是可以在海量数据中快速查询数据，典型代表有MongoDB及CouthDB；三是面向可扩展的分布式数据库（Object Store），这类数据库解决了传统数据库的缺陷，可以适应数据量的增加及数据结构的变化。

MongoDB就是这样一种典型的NoSQL数据库，它由C++语言编写，是开源的一个基于分布式文件存储的数据库，旨在为WEB应用提供可扩展的高性能数据存储解决方案。MongoDB具有以下主要特性:

（1）**文档数据类型** 文档存储为BSON格式。BSON是JSON的二进制编码形式，结构类似嵌套键/值对。BSON是JSON的超集，额外支出一些类型，比如正则表达式、二进制数据和日期。每个文档都有唯一的标示符，如果数据在插入文档之前没有添加唯一标示符，则由MongoDB自动生成一个标示符ID。类似的存储格式如下：

{\_id:ObjectID:(“0d09a7c7d7a812a54c74d984a”), Title: ‟A”, Url:‟http\\www” }

表示的是一个文档中的存储数据。

（2）**即时查询** 说一个系统支持即时查询即是无需预先定义系统接收的查询类型。关系数据库用这个能力，他们会严格遵照指示执行任何完备的sql查询，无论有多少条件。虽然这个在关系数据库里面是理所当然的功能，但是在NoSQL中并非所有都支持动态查询。比如键值存储数据库只能按照一个维度-键来进行查询。

（3）**二级索引** MongoDB中的二级索引是用Bshuttle（B-Tree）实现的，B树索引也是绝大数关系数据库支持的索引，针对多种查询进行优化。

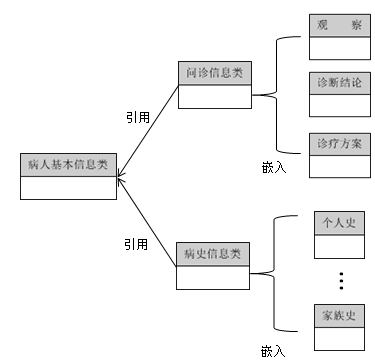
（4）**复制** MongoDB通过称为副本集（replica set）的拓扑结构提供复制功能。副本集将数据分布在多台机器上以实现冗余，在服务器和网络故障的时候，能提供自动故障转移。除此之外，复制功能还能用于扩展数据库的读能力。如果有一个读密集型的应用程序，可以把数据库的读操作分散到副本集群中的各台机器上。

（5）**速度和持久性** 在MongoDB中，用户可以选择写入语义，用户决定是否开启Journaling日志记录，通过这种方式来控制速度和持久性间的平衡。该日志记录将所有的写操作都提交到一个日志中，即使是服务器非正常关闭（比如断电故障），该日志也能保证重启服务后，恢复之前的写操作。

（6）**数据库扩展** MongoDB的水平扩展非常易于管理，它通过基于范围的分区机制，即自动分片来实现这一设计目标，自动分片机制会自动管理各个节点之间的数据分布。分片系统会自动处理节点的增加，帮助进行自动故障转移。

2.2.2.2数据库设计

为了充分发挥MongoDB的模式自由和扩展性，论文对于疾病诊断决策支持系统的医疗数据需求进行抽象后定义了基础信息类，各疾病可以进一步围绕这些基础信息类，对它内部结构进行扩展或者外部添加进行建模以满足各异的数据需求。基础信息类如下图所示：



病人基本信息类：描述患者的人口统计学信息，作为信息的主体，支持“以病人为中心”的信息管理。

问诊信息类：描述患者某次就诊的相关信息，将就诊相关信息分成一般性概念的三类：观察：指对于患者健康相关的客观描述和测量数据，代表医生问诊获取的信息；诊断结论：包含CDSS诊断的建议和医生的确定性的诊断结论；诊疗方案：指根据患者当前情况采取的

医疗行为。

病史信息类：指不随就诊次序变化的个人史、家族史等病史信息。反映病人以往的健康状况。

根据以上抽象类的定义，设计MongoDB的存储方案如下图所示

接口层提供对于各个基本信息类的操作访问接口，包括增删查改基本操作。服务层向上层提供索引服务，由于患者就医的过程中，医生往往根据姓名或结合就诊日期查询已有的病人信息，因此建立指定字段的索引可以提高查询速度，缩短系统反应时间，不影响医生的工作流程。通信层完成MongoDB客户端与MongoDB服务的通信。存储层提供数据的分布式存储结构，利用主从复制构建自动故障转移的存储节点，通过自动分片集群简化数据存储水平扩展的管理。

2.2.3 基于可视化组件的录入展示模块

在疾病诊断过程中，医生是对于病人的信息进行综合处理，这个过程是由三个阶段的循环来表示的：观察（Observation）,诊断（Diagnosis）和治疗（Therapy），如下图所示。在过程中，医生通过回忆或搜索相关的医学知识和经验，向病人询问他/她相关的问题，如临床症状、相关病史、疾病诱发原因、疾病发作情况以及病患家族史等，搜集信息后集合疾病的知识进行诊断推理，并得出一个结论，然后给出治疗的方案。

在这个过程中，如果在问诊过程中，对于病人的相关信息了解越充分，关于疾病的不确定性会越小，因此诊断决策的准确性往往取决于问诊过程中信息的获取程度。然而不同疾病对于病人的信息获取有着相当大的差异，医生如果不了解需要获取的信息类型，很难得出正确的诊断结论。对于系统来说，也就是疾病问诊流程部分的数据获取必须随着疾病的需求而变化，否则不能起到辅助决策的作用。因此，针对疾病流程的动态性，采用可视化组件将可变的界面存为配置文件，并通过数据交互模块完成数据的传输功能。

2.2.3.1可视化组件概述

随着信息技术在医疗领域的应用逐渐广泛，在诊疗过程中产生的诊疗数据往往以电子化医疗文档的形式存在于医疗信息系统中。根据临床需求，国内开发了一套“所见即所得”的结构化医疗文档系统，该系统基于医疗文档的表达方法建立可视化对象，并提供了医疗文档的编辑工具、医疗文档模板的设计工具。利用这套系统，可以在医疗文档模板设计工具编辑诊疗界面，得到文档模板后，再通过医疗文档编辑工具在Web页面显示界面并且提供数据的编辑展示功能。通过这种方式，可以解决诊疗页面需求多样易变带来的工作量大的问题，提高系统的开发效率，缩短更新周期。

2.2.3.2数据交互过程设计



流程说明

采用Ajax技术，同步调用，将数据传往服务端

* SetDataIntoJson: 将界面数据从组件接口取出转换成Json数据格式
* GetDataFromJson : 从服务器传回的Json解析得到数据，并通过组件接口显示数据。

**2.3 系统框架实现**

2.3.1 控制层

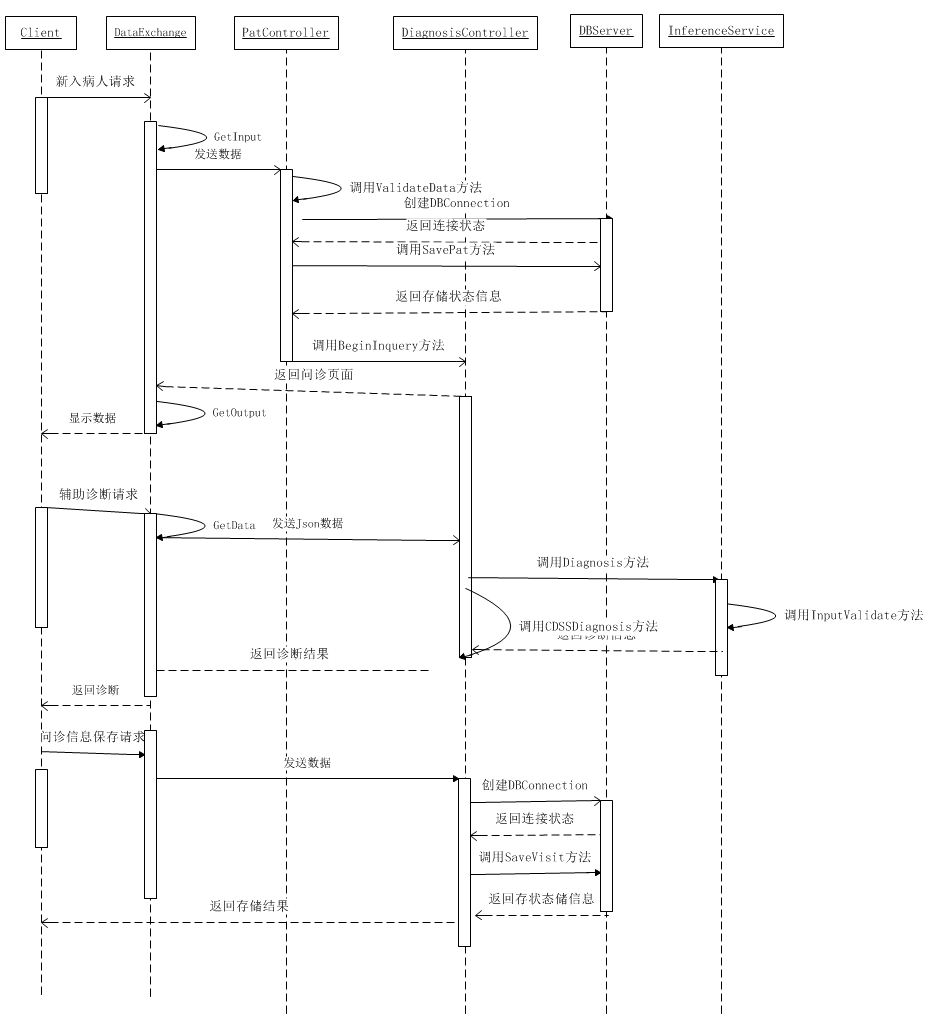
框架的控制器 Controller 是整个 Web 应用程序的控制中心。它承担的主要功能如下：接收浏览器客户端的处理请求，根据用户的请求和需要执行的业务逻辑进行处理，然后获取业务逻辑的处理结果，并将返回的结果组织成Json 文件的形式，通过视图层将其进行转换并显示给最终用户。

首先根据业务需求实现了以下模块

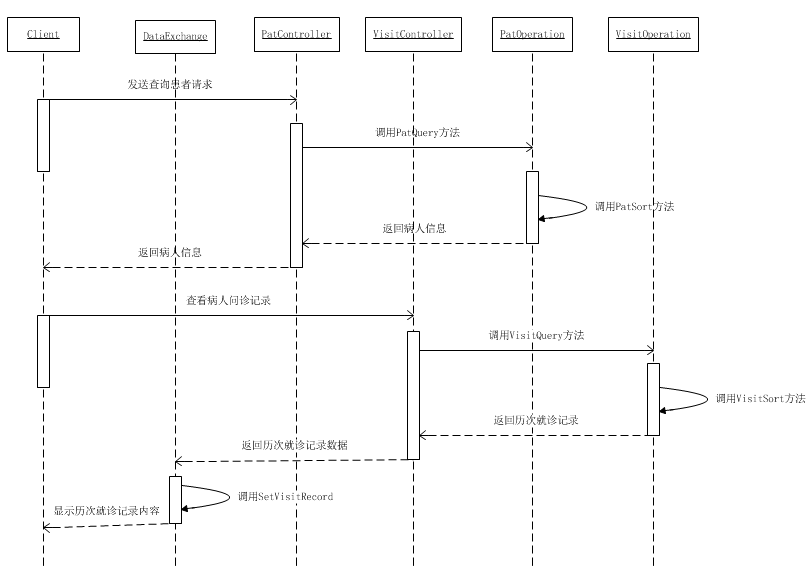
* AccountController：主要实现与用户身份验证的操作流程。
* PatInforController：实现病人信息相关的操作，包括保存新入病人信息和根据姓名等条件查询病人信息。
* VisitRecordController：实现问诊记录操作方面的操作，包括问诊记录的保存，历次问诊记录的查询展示，问诊记录的修改及删除等操作。
* DiagnosisController：实现推理服务的调用操作，将推理所需的数据传送到推理引擎接口，并接收返回的诊断结果。

本系统框架包含的主要业务逻辑如下：

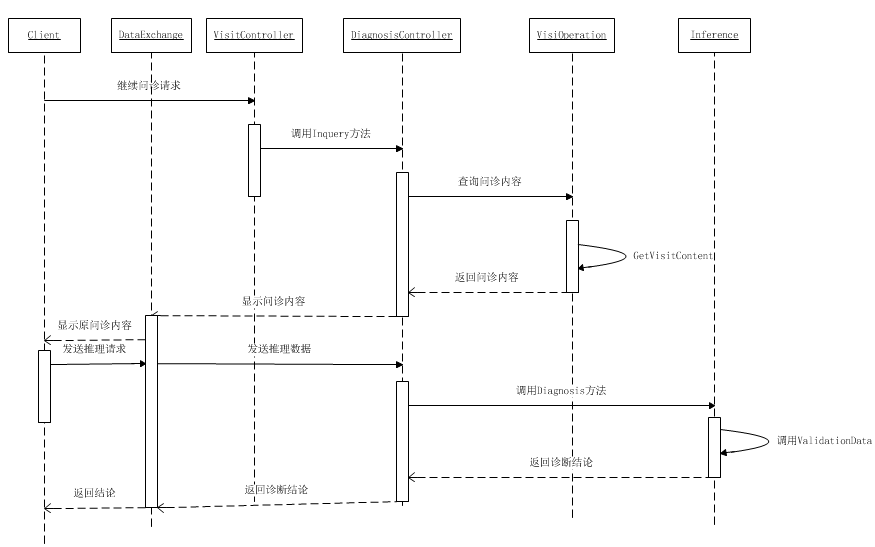
（1）新入病人的问诊流程



（2）已就诊过的病人信息查询



（3）继续某次就诊记录进行就诊



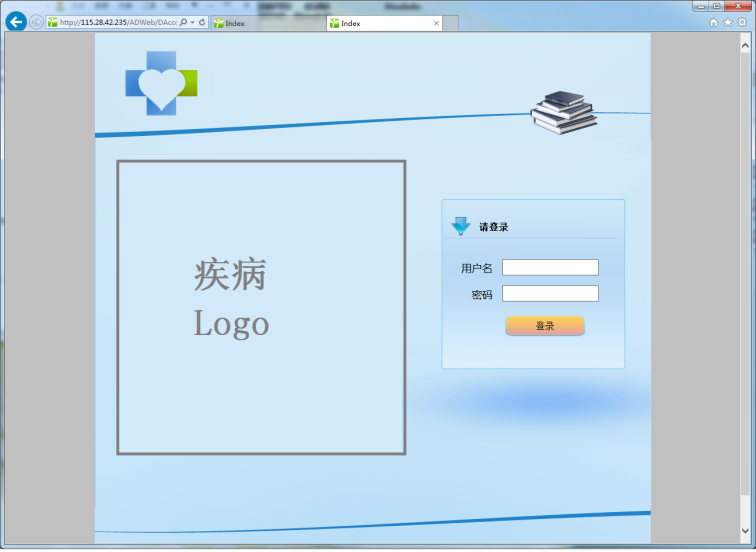
2.3.2 视图层

视图层是主要负责为浏览器客户端提供动态页面显示，视图接收用户输入的数据，并能将最后的数据显示给用户，但它并不接收任何业务逻辑处理。在本系统开发框架中，使用 数据交互接口来转换控制层Controller返回的Json形式的结果集并最终展示给用户

本文使用HTML和CSS开发了以下视图模板，为控制层提供人机交互接口。

1. 系统登录视图模板

系统初始登录页面，提供权限验证功能，如图



1. 病人信息查询视图模板

新录病人信息或者查询已就诊病人信息，如图



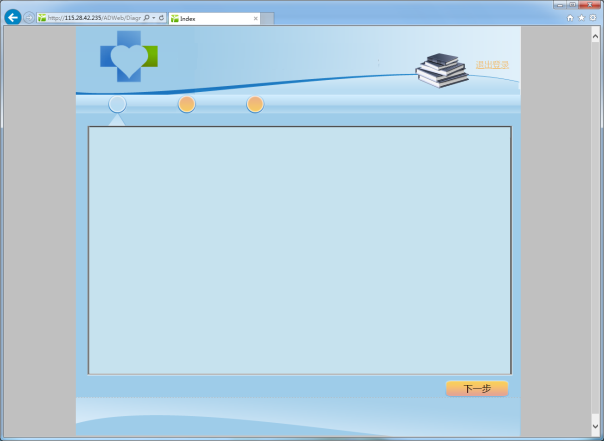
1. 病人问诊记录查询视图模板

根据问诊日期显示病人历次就诊记录信息，如图



1. 问诊流程视图模板

提供问诊向导的功能，结合可视化组件可达到问诊流程可配置的效果，如图



蓝色区域为可视化组件显示区域，通过读入不同的页面配置文件，可以展现不同的问诊流程。

2.3.3 模型层

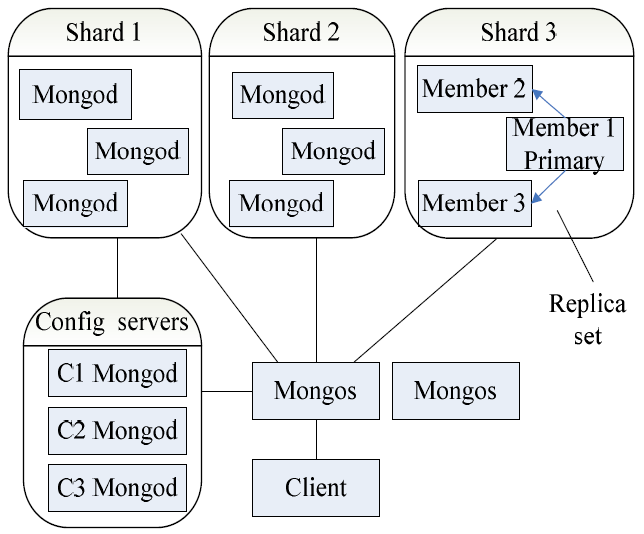
模型是应用程序的主体部分，它被用于封装与应用程序的业务逻辑相关的数据以及对数据的处理方法。一个模型可以被多个视图提供据，所以提高了应用的可重用性。本系统框架中模型层主要分为两部分---数据操作和推理服务

2.3.3.1 数据持久化

（1）分布式集群搭建

MongoDB 集群包括一定数量的 mongod（分片存储数据）、mongos（路由处理）、config

server（配置节点）、clients（客户端）、arbiter（仲裁节点）。 MongDB 集群典型结构如图所示：



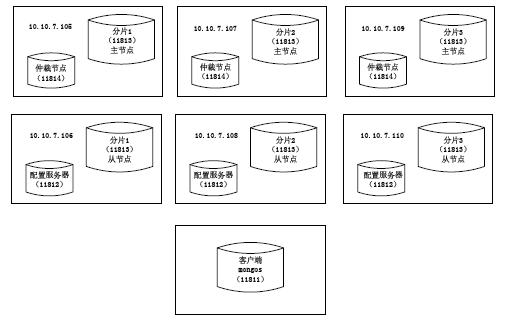
1．分片：一个分片为一组mongod，通常一组包括两台mongod，这两台mongod互为主从模式。而且存储的数据相同，互为数据备份。一个完整的数据按照一定的顺序进行分割，分别存储到不同的分片中。每个分片上的数据都是一个范围内的连续数据，所以支持指定分片的范围查询。当数据块超过指定容量大小时，数据块将会分成两块，当某个分片的数据过多，或者新的分片加入集群，数据将自动迁移，均衡到整个集群系统中。

2．mongos：mongos作为一个路由处理节点，类似集群系统的控制中心，负责路由和集群的协调工作，使得集群成为一个整体。mongos启动的时候，从配置服务器获取分片的信息，当客户端请求存储服务时，mongos将请求路由到各个分片上，各个分片操作完毕之后，再将汇总结果返回给客户端。

3．config server：存储集群的信息，包括分片和块数据信息。主要存储块数据信息，每个config server上都有一份所有块数据信息的拷贝，以保证每台config server上的数据的一致性。

MongoDB服务器分布式部署有很多种方式。在部署的时候，可以将每个config、mongos、mongod都单独运行在独立的服务器上。也可以将config、mongos与mongod服务器共存。后者更为常用，节省了服务器资源。

下图为物理机共享的集群。config和mongos与mongod共享同一台服务器。 在实验室环境下利用三台电脑，组成一个分片。一共有三个分片。每台分片有一个replica（副本集），每个replica由两台虚拟机组成，有一个主节点，一个从节点。两个节点的数据是一样的，形成数据备份。



（2）数据操作类实现

MongoDB数据库提供了基于各种语言的驱动，在本论文中，主要使用 C#开发，所以采用了 MongoDB 的 C#驱动。C#驱动文件中有两个文件：MongoDB.Bson.dll 和 MongoDB.Driver.dll。分别是两个类库：Bson 类库和 Driver 类库。

Bson类库：BSON 是类似 JSON 的一种二进制形式的存储格式，简称 Binary JSON，它和 JSON 一样，支持内嵌的文档对象和数组对象，但是 BSON 有 JSON 没有的一些数据类型，如 Date 和 BigData 类型。它是 MongoDB 文档数据库内部的数据存储方式。

Driver 类库：Driver 类就是通过 C#来访问 MongoDB 的驱动类。该类中基本上包括了所有的 MongoDB 数据库的操作。主要是 MongoDB 数据库的连接、服务端控制、数据库操作如增删改查等等。

实现 MongoDB 数据库操作类，就是基于 MongoDB 的 C#驱动的基础上，再次根据具体的需求，将其中的一些功能封装起来。方便对 MongoDB 数据库进行操作。该数据库操作类主要包括 MongoDB 数据库的连接、MongoDB 数据库的管理以及 MongoDB 数据库中的Collection 表的操作管理，主要分为下面三类

1. MongoConnectionManage类

数据库操作池类。主要功能是管理数据库的操作连接。减少客户端与数据库之间的连接次数。在这个连接池类里面，限定了数据库的连接数量，超过连接数据量，数据库将不支持连接。这样可以防止过度的连接导致数据库的服务故障。该类的主要函数如下表：

1. MongoOperation 类

MongoDB 操作类，该类主要负责 MongoDB 数据库的操作接口。比如数据库的文档的创建、增添、修改及删除，主要函数如下表：

1. MongoCommend类

MongoDB 的驱动中有一些功能可能没有集成，但是由于 MongoDB 采用 Javascript 语言来解释命令，可以通过执行一些 Javascript 命令来进行一些数据库的操作来进行对MongoDB集群的管理。主要函数如下表：

2.3.3.2 推理服务

（1）基于规则的推理实现

CLIPS（C Language Integrated Production System）是一个公共领域的软件工具，用于构建专家系统。这个名字是 “C语言集成产生式系统”的缩写。CLIPS的第一个版本开始于1985年美国航空航天局约翰逊航天中心（作为现有系统的技术ART\*Inference替代）。

CLIPS可能是最为广泛使用的专家系统工具。CLIPS集成了一个完整的面向对象语言COOL编写的专家系统。 COOL结合了面向过程，面向对象和逻辑（定理证明）语言的编程范式。由于具有高移植性、高扩展性和强大的知识表达能力和编程方式，CLIPS被广泛应用于政府、工业和学术界的专家系统。CLIPS基本的组成部分包括事实表、知识库和推理机内核，通过这些组成部分，可根据已有的事实和规则推理出所需的结论。

CLIPS规则通常由5个部分组成xxx

本论文中将使用CLIPS推理机内核作为规则推理的实现工具。将知识库作为外部文件来源，针对引擎输入的需求，编写输入和输出的数据解析接口。

（2）基于数据的推理实现

WEKA是新西兰Waikato大学开发的全面的数据挖掘系统，它不仅提供了多种数据挖掘方法(分类、聚类、关联规则等)的多种常用算法进行知识发现，还提供了适用于任意数据集的数据预处理功能，以及算法性能评估的多种方法。WEKA是由JAVA语言实现的开放性平台，具有非常良好的扩展性和兼容性，用户可以根据具体需要将个性化的算法封装进系统，达到数据处理及算法性能评估的目的，正是由于WEKA具有良好定义的数据结构和基本的统计接口,将它作为基于数据的推理方式的实现工具。

WEKA

通过对于WEKA的标准接口Classifier的封装，读取外部的算法模型文件以及数据

**2.4专科疾病诊断决策支持系统构建方案**

**2.5小结**

**3.头痛诊断决策支持系统实现**

**3.1 系统需求分析**

3.1.1疾病特征分析

3.1.2临床知识的特性

**3.2 基于框架的系统开发流程**

3.2.1推理引擎选择

3.2.2 数据模型设计

信息类表

3.2.3 问诊界面配置

3.2.4 数据交互层实现

**3.3 系统实现**

**3.4小结**

**4.老年痴呆症诊断决策支持系统实现**

**4.1 系统需求分析**

4.1.1疾病特征

4.1.2临床知识的特性

**4.2 基于框架的系统开发流程**

4.2.1推理引擎选择

4.2.2 数据模型设计

4.2.3 问诊界面配置

4.2.4 数据交互层实现

**4.3 系统实现**

**4.4小结**

**5.总结与展望**

**5.1 总结**

**5.2 展望**