撰写：万宽红

计算机控制与应用小组 | 2016年6月

**JDBC-Web开发文档**

JDBC介绍、核心APi、模块插件、WEb框架、前端交互、维护及扩展

目录

[1.JDBC介绍 3](#_Toc452626693)

[2.核心API 3](#_Toc452626694)

[2.1封装CoreService 3](#_Toc452626695)

[2.2查找遍历节点API 3](#_Toc452626696)

[2.3增删节点API 4](#_Toc452626697)

[2.4查询类插件API 4](#_Toc452626698)

[2.5数据类插件API 4](#_Toc452626699)

[2.6插件管理API 4](#_Toc452626700)

[3.模块插件 5](#_Toc452626701)

[3.1数据插件（DataPlugin） 5](#_Toc452626702)

[3.1.1统一接口 5](#_Toc452626703)

[3.1.2固定采样率的波形信号节点 5](#_Toc452626704)

[3.1.3 6](#_Toc452626705)

[3.2查询插件（QueryPlugin） 6](#_Toc452626706)

[3.2.1统一接口 6](#_Toc452626707)

[3.2.2路径查新插件PathQueryPlugIn 6](#_Toc452626708)

[3.2.3 标识符查询插件IdQueryPlugIn 6](#_Toc452626709)

[4.Web框架 7](#_Toc452626710)

[4.1 Web.config 7](#_Toc452626711)

[4.2程序入口 8](#_Toc452626712)

[4.3 WebAPI控制器 8](#_Toc452626713)

[4.3.1路由 8](#_Toc452626714)

[4.3.2 Entity API控制器 8](#_Toc452626715)

[4.3.3 Data API控制器 9](#_Toc452626716)

[4.3.4 Stream API控制器 9](#_Toc452626717)

[4.4 Web视图控制器 9](#_Toc452626718)

[4.3.1 路由 9](#_Toc452626719)

[4.3.2控制器 9](#_Toc452626720)

[4.4 API测试工具 9](#_Toc452626721)

[4.5自动化Web文档 9](#_Toc452626722)

[4.6可视化工具 9](#_Toc452626723)

[5.前端交互 9](#_Toc452626724)

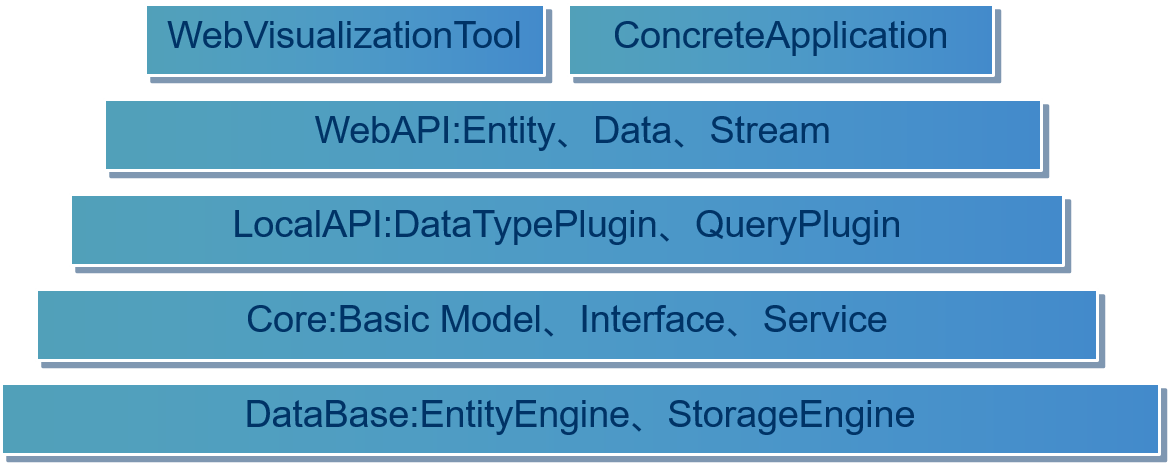
[5.1界面设计 9](#_Toc452626725)

[5.2节点操作 9](#_Toc452626726)

[5.3数据显示 10](#_Toc452626727)

[6.维护及扩展 10](#_Toc452626728)

# 1.JDBC介绍



# 2.核心API

## 2.1封装CoreService

将CoreService和插件列表作为公有属性（对于高级用户，亦可直接调用底层服务），在此基础上对CoreService原有方法进行封装，同时提供插件模块，对上层提供一个一致化的API调用接口。

整体采用单例模式进行实现，将构造函数和静态CoreApi实例设置为私有，提供一个公有的获取实例的静态方法GetInstance()，保证上层代码中只会出现一个CoreApi实例，避免浪费内存、提高代码复杂度、数据处理不一致问题。

## 2.2查找遍历节点API

|  |
| --- |
| 根据Id异步查找JDBCEntity  async Task<JDBCEntity> FindNodeByIdAsync(Guid id)  根据ChildId异步查找Parent  async Task<JDBCEntity> FindParentByChildIdAsync(Guid childId)  根据路径异步查找Node  async Task<JDBCEntity> FindOneByPathAsync(string path)  根据ParentId异步查找所有的Node  async Task<IEnumerable<JDBCEntity>> FindAllChildrenByParentIdAsync(Guid parentId)  根据ParentId异步查找所有的NodeId  async Task<IEnumerable<Guid>> FindAllChildrenIdsByParentIdAsync(Guid parentId)  根据ParentId和ChildName异步查找Node  async Task<JDBCEntity> FindChildByParentIdAndChildNameAsync(Guid parentId, string childName) |

## 2.3增删节点API

|  |
| --- |
| 清空节点数据库  void clearDb()  异步更新Node实例  async Task<JDBCEntity> UpdateNodeAsync(Guid id, Dictionary<string, UpdateEntity> list)  异步修改Node名称（相应修改Path和子节点Path）  async Task<JDBCEntity> ReNameNodeAsync(Guid id, string name)  异步添加Node实例到Experiment  async Task<JDBCEntity> AddOneToExperimentAsync(Guid parentId, JDBCEntity entity)  异步移动节点  async Task<JDBCEntity> MoveJdbcEntityAsync(Guid parentId, Guid entityId)  异步删除entity  async Task DeletAsync(Guid jdbcId, bool isRecuresive = false)  异步清空entity子项  async Task<int> RemoveAllChildrenAsync(Guid jdbcId, bool isRecuresive = false)  异步复制节点  async Task<JDBCEntity> DuplicateAsync(Guid sourceId, Guid parentId, string newName, bool isRecuresive = false, bool duplicatePayload = false) |

## 2.4查询类插件API

|  |
| --- |
| 通过uri异步查找Node实例  async Task<IEnumerable<JDBCEntity>> FindNodeByUriAsync(Uri uri) |

## 2.5数据类插件API

|  |
| --- |
| 创建一个Singal并返回  Signal CreateSingal(string dataType, string name, string initString)  获取节点的属性信息  Dictionary<string, object> GetInformation(JDBCEntity entity)  查询得到数据  async Task<object> GetDataByUriAsync(Uri uri, string format) |

## 2.6插件管理API

|  |
| --- |
| 添加一个DataTypePlugin  void AddDataTypePlugin(IDataTypePlugIn plugIn)  添加一个QueryPlugin  void AddQueryPlugin(IQueryPlugIn plugIn)  从动态库文件加载QueryPlugin  void LoadQueryPlugin(string assemblyFile)  从动态库文件加载DataTypePlugin  void LoadDataTypePlugin(string assemblyFile)  获取最适合处理数据类型dataType的DataTypePlugin  IDataTypePlugIn GetBestDataTypePlugin(string dataType)  获取最适合处理路径uri的QueryPlugin  QueryPlugIn GetBestQueryPlugin(Uri query) |

# 3.模块插件

## 3.1数据插件（DataPlugin）

### 3.1.1统一接口

|  |
| --- |
| 插件名称，应保证其唯一性  string Name { get; }  对某一数据类型进行评分，上层API根据分值决定使用哪一个数据类插件  int ScoreDatatype(string dataType);  获取信号节点属性信息  Dictionary<string, object> GetInformation<T>(Signal signal);  按需异步获取信号数据  Task<object> GetDataAsync<T>(Signal signal, string fragment, string format);  按需异步获取信号数据游标  Task<ICursor> GetCursorAsync<T>(Signal signal, string dataFragment);  将数据异步写入信号节点中  Task PutDataAsync<T>(Signal signal, string fragment, object data);  根据初始信息创建一个信号节点  Signal CreateSignal(string datatype, string name, string iniString); |

### 3.1.2固定采样率的波形信号节点

|  |
| --- |
| 节点标识符  Guid Id { get; set; }  父节点标识符  Guid ParentId { get; set; }  节点名称  string Name { get; set; }  节点类型  JDBCEntityType EntityType { get; set; }  节点路径  string Path { get; set; }  节点元数据  Dictionary<string, object> ExtraInformation { get; set; } |
| 数据类型，指明了数据存储结构、采样类型、元数据等信息  string DataType { get; set; }  采样类型  string SampleType { get { return DataType.Substring(DataType.LastIndexOf('-')+1).ToLower(); } } |
| 返回相应的数据类插件实例，如为空则通过核心API获取最佳处理插件并缓存  IDataTypePlugIn MyPlugIn  起始时间  double StartTime { get; set; }  截止时间  double EndTime { get; set; }  采样间隔  double SampleInterval { get; set; }  时间单位  string Unit{ get; set; }  构造函数，对相关属性进行初始化  FixedIntervalWaveSignal(string name, string dataType, string initString): base(name)  异步获取某段数据的游标  async Task<ICursor> GetCursorAsync(string fragment)  异步获取某段数据  async Task<object> GetDataAsync(string fragment, string format)  获取信号节点属性  Dictionary<string, object> GetInformation()  异步写入某段数据  async Task PutDataAsync(string fragment, object data) |

### 3.1.3

## 3.2查询插件（QueryPlugin）

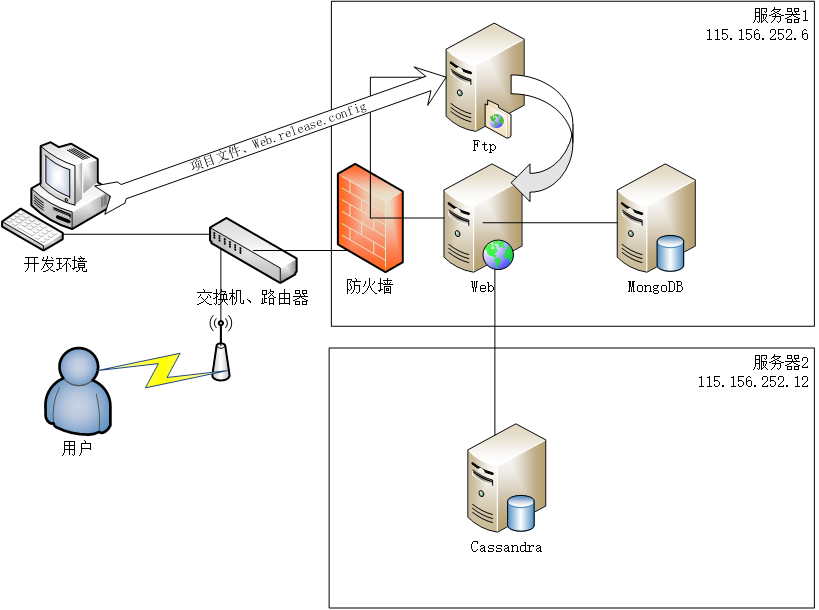
### 3.2.1统一接口

|  |
| --- |
| 插件名称，应保证其唯一性  string Name { get; }  对查询类型进行评分，上层API根据分值决定使用哪一个查询类插件  int ScoreQuery(string query);  异步查找符合条件的节点  Task<IEnumerable<JDBCEntity>> FindJdbcEntityAsync(string query); |

### 3.2.2路径查新插件PathQueryPlugIn

### 标识符查询插件IdQueryPlugIn

# 4.Web框架



## 4.1 Web.config

Config文件为XML格式，是在部署环境下的服务器配置文件，除定义网站功能、程序集外，还可以存储用户自定义配置信息（如数据库连接字符串等）。

由于JDBC还未开发完毕，需要同时在开发环境、测试环境和生产环境中使用，为了便于数据库的切换，分别定义了不同的config文件。

|  |
| --- |
| **Web.config** |
| <connectionStrings>  <add name="MongoDB" connectionString="host =mongodb://127.0.0.1:27017 &amp; database = JDBC-web &amp; collection = Payload"/>  <add name="CassandraDB" connectionString="host =115.156.252.12 &amp; database = jdbc\_debug"/>  </connectionStrings> |
| **Web.Debug.config** |
| <connectionStrings>  <add name="MongoDB"  connectionString="host = mongodb://127.0.0.1:27017 &amp; database = JDBC-web &amp; collection = Payload" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>  <add name="CassandraDB"  connectionString="host = 115.156.252.12&amp; database = jdbc\_debug" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>  </connectionStrings> |
| **Web.Release.config** |
| <connectionStrings>  <add name="MongoDB"  connectionString="host =mongodb://127.0.0.1:27017 &amp; database = JDBC-web &amp; collection = Payload" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>  <add name="CassandraDB"  connectionString="host =115.156.252.12 &amp; database = jdbc\_web" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>  </connectionStrings> |

使用VS集成的发布功能，可以方便地选择config文件，使用文件系统或ftp进行网站发布。

## 4.2程序入口

|  |
| --- |
| public class WebApiApplication : System.Web.HttpApplication  {  /// <summary>  /// 注册功能区、路由等  /// 运行初始化Business代码  /// </summary>  protected void Application\_Start()  {  AreaRegistration.RegisterAllAreas();//注册应用程序中的所有区域  GlobalConfiguration.Configure(WebApiConfig.Register);//注册Web API路由  FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);//注册全局过滤器  RouteConfig.RegisterRoutes(RouteTable.Routes);//注册MVC路由  BundleConfig.RegisterBundles(BundleTable.Bundles);//注册捆绑  BusinessConfig.ConfigBusiness();//执行初始化逻辑代码  }  } |

## 4.3 WebAPI控制器

WebAPI控制器是指继承System.Web.Http.ApiController的控制器，主要特征是返回json、xml等形式的数据。

### 4.3.1路由

在WebApiConfig的Register方法中，对Web API路由进行了配置。路由分为EntityApi（前缀为Entity/）、DataApi（前缀为Data/）、StreamApi（前缀为Stream/）三类，每类又根据HTTP method的不同提供Get、Post、Delete、Put四种方法进行处理。

### 4.3.2 Entity API控制器

### 4.3.3 Data API控制器

### 4.3.4 Stream API控制器

## 4.4 Web视图控制器

Web视图控制器是指继承System.Web.Mvc.Controller的控制器，主要特征是返回html文档。

### 4.3.1 路由

Web视图路由分别在RouteConfig的RegisterRoutes方法和各个VisualizationAreaRegistration的RegisterArea方法中进行配置。

### 4.3.2全局控制器

### 4.3.3 区域控制器

## 4.4 API测试工具

## 4.5自动化Web文档

## 4.6可视化工具

# 5.前端交互

## 5.1界面设计

## 5.2节点操作

## 5.3数据显示

# 6.维护及扩展