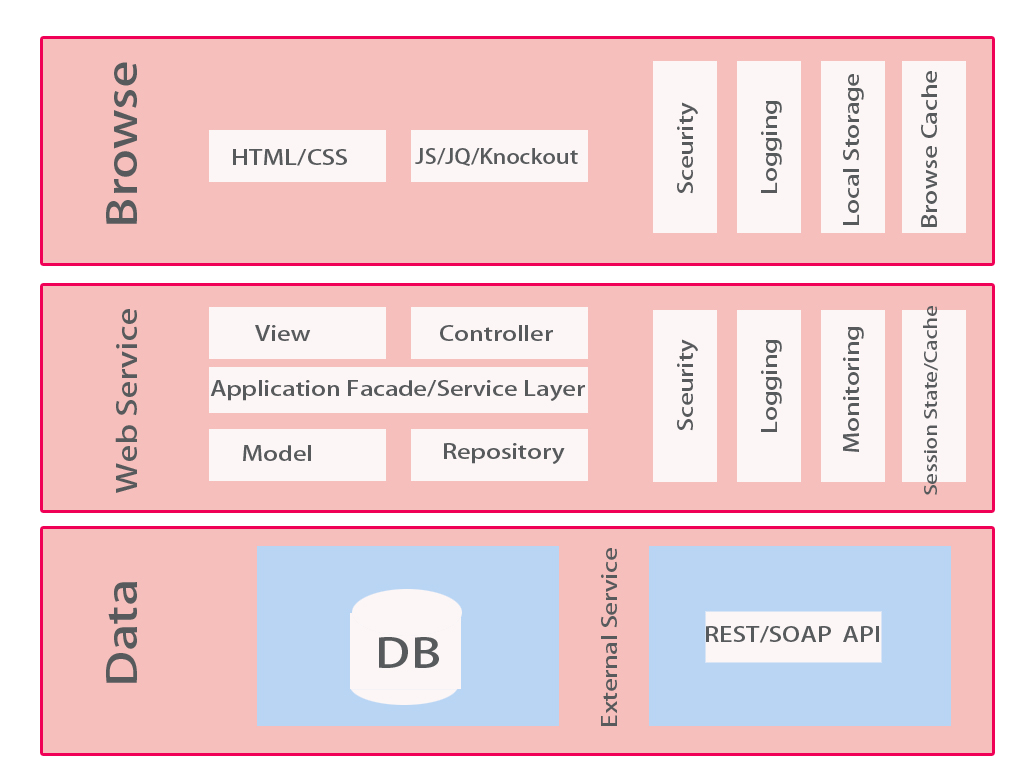
web开发三层架构： 表示层（UI）、业务逻辑层(BLL)、数据访问层(DAL)

架构设计图



一、系统各层次职责

1.UI（UserInterface）层的职责是数据的展现和采集，数据采集的结果通常以Entityobject提交给BL层处理ServiceInterface侧层用于将业务或数据资源发布为服务（如WebServices）。

2.BL（BusinessLogic）层的职责是按预定的业务逻辑处理UI层提交的请求。

（1）BusinessFunction子层负责基本业务功能的实现。

（2）BusinessFlow子层负责将BusinessFunction子层提供的多个基本业务功能组织成一个完整的业务流（Transaction只能在BusinessFlow子层开启。）

3．ResourceAccess层的职责是提供全面的资源访问功能支持，并向上层屏蔽资源的来源。

（1）BEM（BusinessEntityManager）子层采用DataAccess子层和ServiceAccess子层来提供业务需要的基础数据/资源访问能力。

（2）DataAccess子层负责从数据库中存取资源，并向BEM子层屏蔽所有的SQL语句以及数据库类型差异。

DBAdapter子层负责屏蔽数据库类型的差异。

ORM子层负责提供对象－关系映射的功能。

Relation子层提供ORM无法完成的基于关系（Relation）的数据访问功能。

（3）ServiceAccess子层用于以SOA的方式从外部系统获取资源。 注：ServiceEntrance用于简化对Service的访问，它相当于Service的代理，客户直接使用ServiceEntrance就可以访问系统发布的服务。ServiceEntrance为特定的平台（如Java、.Net）提供强类型的接口，内部可能隐藏了复杂的参数类型转换。

（4）ConfigAccess子层用于从配置文件中获取配置object或将配置object保存倒配置文件。

4．Entity侧层跨越UI/BEM/ResourceManager层，在这些层之间传递数据。

二、Aspect

Aspect贯穿于系统各层，是系统的横切关注点。通常采用AOP技术来对横切关注点进行建模和实现。

1．SecurtiyAspect：用于对整个系统的Security提供支持。

2．ErrorHandlingAspect：整个系统采用一致的错误/异常处理方式。

3．LogAspect：用于系统异常、日志记录、业务操作记录等。

三、规则

1．系统各层次及层内部子层次之间都不得跨层调用。

2．Entityobject在各个层之间传递数据。

3．需要在UI层绑定到列表的数据采用基于关系的DataSet传递，除此之外，应该使用Entityobject传递数据。

4．对于每一个数据库表（Table）都有一个DBEntityclass与之对应，针对每一个Entityclass都会有一个BEMClass与之对应。

5．有些跨数据库或跨表的操作（如复杂的联合查询）也需要由相应的BEMClass来提供支持。

6．对于相对简单的系统，可以考虑将BusinessFunction子层和BusinessFlow子层合并为一个。

7．UI层和BL层禁止出现任何SQL语句。

四、错误与异常

异常可以分为系统异常（如网络突然断开）和业务异常（如用户的输入值超出最大范围），业务异常必须被转化为业务执行的结果。

1．DataAccess层不得向上层隐藏任何异常（该层抛出的异常几乎都是系统异常）。

2．要明确区分业务执行的结果和系统异常。比如验证用户的合法性，如果对应的用户ID不存在，不应该抛出异常，而是返回（或通过out参数）一个表示验证结果的枚举值，这属于业务执行的结果。但是，如果在从数据库中提取用户信息时，数据库连接突然断开，则应该抛出系统异常。

3．在有些情况下，BL层应根据业务的需要捕获某些系统异常，并将其转化为业务执行的结果。比如，某个业务要求试探指定的数据库是否可连接，这时BL就需要将数据库连接失败的系统异常转换为业务执行的结果。

4．UI层(包括Service层)除了从调用BL层的API获取的返回值来查看业务的执行结果外，还需要截获所有的系统异常，并将其解释为友好的错误信息呈现给用户。

五、应用框架

·前端框架：vue + element-ui组件 ·后端框架：springboot+mybatis+mysql