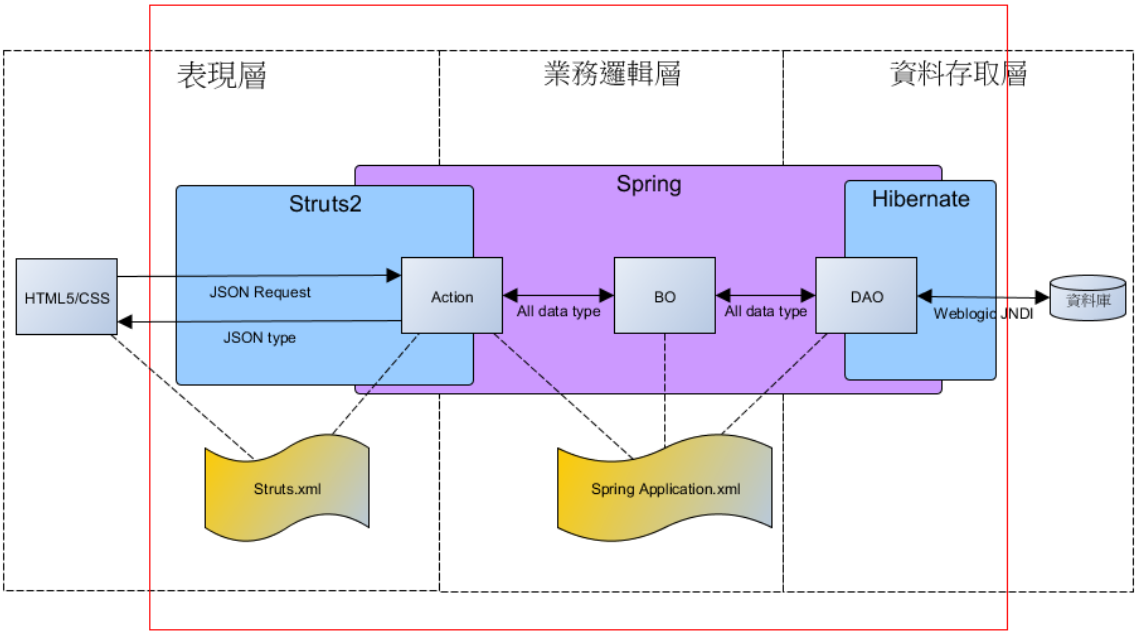
壹、系統處理流程與概要設計



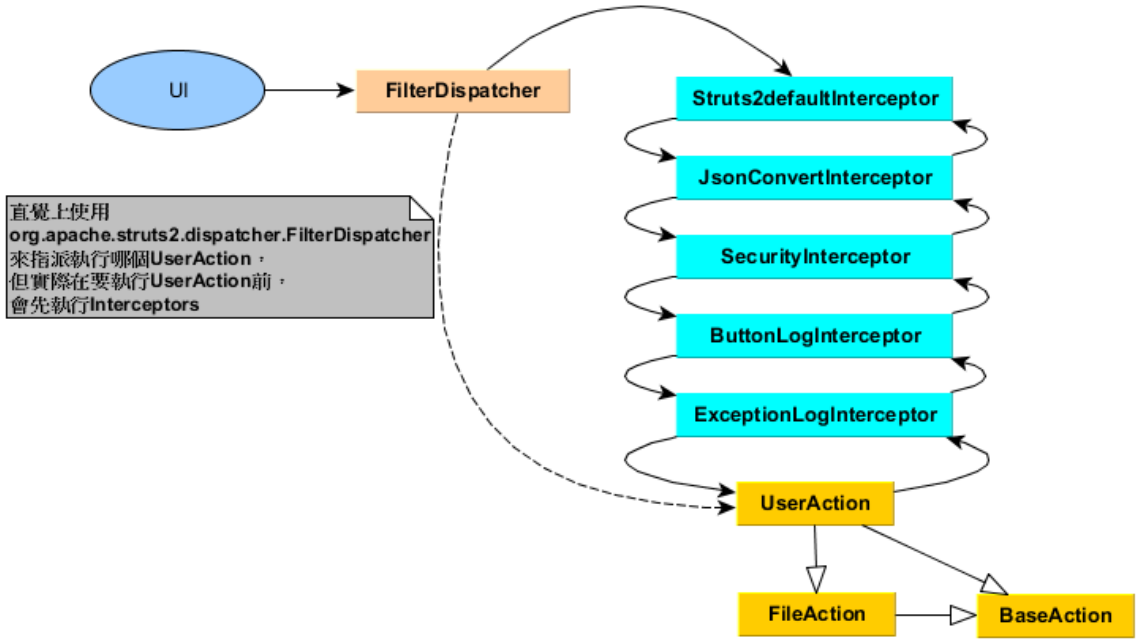
此系統主要套用 Struts2 Framework，Spring Framework，Hibernate Framework 等框架，各負責的工作如下

Framework	功能	勾稽物件
Struts	提供 MVC 結構化處理	Action
Spring	為基底框架，Struts2 的 Action 及 Interceptor 物件允許由 Spring Framework 托管，且 Spring 對 Hibernate 支援良好，因此主要運作的物件都交由 Spring 管理	Action, VO, BO, DAO,其他 common Object
Hibernate	ORM 框架，將 DB table 物件化，省略 SQL 語法，直覺操作 table 欄位	DAO

主要運作物件種類

類型	功能
Action	主要負責收集 request 資訊與提供頁面資料
Action VO	存放 request 資料物件，每個 Action 都擁有自己的 VO 物件
BO	商業邏輯物件，主要負責業務邏輯資料的實現
DAO	資料存取物件，主要負責與資料庫溝通

Struts2 基礎結構圖



來源	元件名稱	功能行為
Struts2	FilterDispatcher	UI 發出 request 請求，會被此 filter 攔截，若符合所設定的 URL pattern，如/xxx.action，Strus2 會依照設定檔指派處理此 request 的 Action 物件
Struts2	Struts2DefaultInterceptors	所有 Struts2 預設的 Interceptors，詳細請參考 Struts2 官方文件
自訂	JsonConvertInterceptor	1. 負責處理 request JSON 字串轉換至 Action VO 物件。 2. 統一自動化回覆成功訊息
自訂	SecurityInterceptor	1. Token 生成與檢查 2. IP 來源檢查 3. 使用權限限制 4. 註冊流程導頁控制
自訂	ButtonLogInterceptor	自動化網站日誌紀錄
自訂	ExceptionLogInterceptor	1. 應用程式 Exception handle 2. 統一自動化回覆錯誤訊息
自訂	UserAction	功能 Action 物件
自訂	FileAction	上傳檔案 Action 物件
自訂	BaseAction	系統基礎 Action 物件

此專案採用 Token 機制，在 servlet context 建立一個 Listener 提供存取 token，配合 SecurityInterceptor 檢核 token 的合法性。

1. TokenListener
   TokenListener 實作 javax.servlet.ServletContextListener，藉由此 Interface 來快速建立 Token 機制，當 AP 啟動時立即在記憶體建立一個存放 Token 物件的容器，並同時建立一個 Timer 負責清除 Token。

2. Token
   Token 物件屬性

屬性	型態	說明
tokenID	String	Token 唯一辨識值
datas	Map	資料存放物件，讓 Token 有 store 資料的功能
effectDate	Date	Token 過期時間
isExpired	Boolean	是否已過期

Spring 說明
在 AP 啟動時使用 Spring 提供的 org.springframework.web.context.ContextLoaderListener，來建立主要運作物件，ContextLoaderListener 會讀取 web.xml 環境變數 contextConfigLocation 設定的檔案位置，將設定檔所設定的指定的物件，註冊在 Spring Application context 中，成為其中的 Bean Object。

web.xml

```
<!-- spring setting -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        classpath:conf/appctx-*.xml,
        classpath:conf/*/appctx-*.xml
    </param-value>
</context-param>

<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

為了減少設定檔編寫的複雜度，使用 Annotation 來協助註冊的步驟，要利用 annotation 需在設定檔設定如下，scan 的範圍為 com.neweb package 下所有的物件。

```
<bean class="org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor"/>
<context:component-scan base-package="com.neweb.*" />
```

使用 Annotation 註冊物件

1. Spring 提供@Repository/@Service/@Controller/@Component 標籤，來進行註冊，四種標籤皆可註冊到 Application context 中，為了物件語意的表達分別使用不同的標籤，分配如下

標籤	層別	對應物件
@Controller	表現層	Action
@Service	業務邏輯層	BO

@Repository	資料存取層	DAO
@Component	anywhere	共用元件及 Action VO

2. @Scope("prototype")此標籤說明，若向 Spring application context 取得 Bean 時，需要建立一個新的 instance，在預設的情況下註冊的 Bean 皆為 singleton，即為單一的 instance

何時使用 prototype

一個 request 如同一個 Thread 依照此系統結構會呼叫到一個 Action Object，並使用到 Action VO，而不同的 request 應該使用各自的 Action 及 Action VO object，才不會造成 non-thread-safe 的狀況，所以全部的 Action 及 Action VO 都該設定為 prototype。

範例 Action

```
@Controller
@Scope("prototype")
public class MemberAction extends BaseAction<MemberVO>{
```

範例 Action VO

```
@Component
@Scope("prototype")
public class MemberVO {
```

範例 BO

```
@Service
public class MemberBOImpl implements IMemberBO {
```

範例 DAO

```
@Repository
public class MemberMasterDao extends BaseDaoHibernate {
```

使用 Annotation 完成 IOC 動作

標籤	注入行為
@Resource	為 javax.annotation.Resource，此標籤依名稱來找尋相對應的 Application context 中的 bean
@Autowired	org.springframework.beans.factory.annotation.Autowired，此標籤依 type 來找尋相對應的 Application context 中的 bean

@Resource 範例

SecurityInterceptor.java

```java
public class SecurityInterceptor extends AbstractInterceptor {

    private static final long serialVersionUID = 1L;
    private static Logger logger = Logger.getLogger(SecurityInterceptor.class);

    @Resource
    private List<String> excludeSystemVerify;    //不需檢查使用者資訊 或者 是否登入
```

在 appctx-base.xml 中定義為 excludeSystemVerify

```xml
<!-- 系統排除登入資訊檢核 -->
<bean id="excludeSystemVerify" class="java.util.ArrayList">
    <constructor-arg>
</bean>
```

@Autowired 範例

EditAddrAction.java

```java
@Controller
@Scope("prototype")
public class EditAddrAction extends BaseAction<EditAddrVO> {

    private static final long serialVersionUID = -409582671462219019L;
    private static Logger logger = Logger.getLogger(EditAddrAction.class);

    @Autowired
    private IEditAddrBO editAddrBO;
```

EditAddrBOImpl.java使用 @Service註冊在Application context中，EditAddrAction的屬性editAddrBO使用@Autowired進行注入，其type為IEditAddrBO在Application context，而EditAddrBOImpl實作IEditAddrBO，因此可對應到EditAddrAction.editAddrBO會被注入EditAddrBOImpl此實體物件

```java
@Service
public class EditAddrBOImpl implements IEditAddrBO {
```