# XML解析によるBean定義およびMyBatis SQL情報の取得

## 1. XML解析の概要

XML（Extensible Markup Language）は、構造化データを記述するための言語であり、JavaのSpringフレームワークやMyBatisなどで設定ファイルとして利用される。

XMLを解析することで、以下の情報を取得できる:  
・SpringのBean定義  
・MyBatisのnamespace, SQL ID, SQL内容

## 2. Bean定義の取得

SpringのXML設定ファイルにはBeanの定義が含まれる。これを解析することで、Bean名やクラス名を取得可能。

例（XML）:  
<beans>  
 <bean id="userService" class="com.example.service.UserService"/>  
</beans>

解析のポイント:  
・beanタグのid属性 → Bean名  
・beanタグのclass属性 → クラス名

## 3. MyBatisのnamespace, SQL ID, SQL内容の取得

MyBatisのマッピングファイル（\*.xml）には、namespaceごとにSQLが定義されている。

例（XML）:  
<mapper namespace="com.example.mapper.UserMapper">  
 <select id="getUserById" parameterType="int" resultType="User">  
 SELECT \* FROM users WHERE id = #{id}  
 </select>  
</mapper>

解析のポイント:  
・mapperタグのnamespace属性 → Namespace  
・各SQLタグ（select, insert, update, delete）のid属性 → SQL ID  
・SQLタグの内容 → SQL文

## 4. 解析フロー

1. XMLをパース（解析）  
2. ノードを探索し、必要な情報を取得  
3. 結果を整理して出力

## 5. Pythonによる解析例

Pythonのlxmlライブラリを用いて、XMLを解析し、必要な情報を取得する。

サンプルコード（Bean定義の取得）:  
```python  
from lxml import etree  
tree = etree.parse("beans.xml")  
for bean in tree.xpath("//bean"):  
 print(f"Bean ID: {bean.get('id')}, Class: {bean.get('class')}")  
```

## 6. Javaによる解析例

JavaのDOMパーサを用いた解析方法。

サンプルコード（MyBatis SQL ID取得）:  
```java  
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  
DocumentBuilder builder = factory.newDocumentBuilder();  
Document doc = builder.parse(new File("mapper.xml"));  
NodeList sqlNodes = doc.getElementsByTagName("select");  
for (int i = 0; i < sqlNodes.getLength(); i++) {  
 Element sqlElement = (Element) sqlNodes.item(i);  
 System.out.println("SQL ID: " + sqlElement.getAttribute("id"));  
}  
```