一、课程的出品方

MyBatis 中文官网

二、网址

www.mybatis.cn

三、简介

MyBatis 中文官网是一个非盈利性组织。

不管你是在地里长大，不管你是吃肯德基长大，MyBatis 中文官网就想做一个给所有人传递价值和正能量的网站。

本次课程的所收费用都将归入网站基金。网站基金是网站和社群发展的基础保障。

课程目的

探索 Context 写作手法的来源，让大家吃透 Context，最后会使用 Context，让大家自己写的代码看起来更有深度，有艺术美感，而不是像白开水一样平淡。

一、Context：广泛存在

稍有编程经验的人，都见过 Context，Context 表示上下文。

Context 是广泛存在，并且与 Container 是有一定的关系的。

（1）在 Tomcat 里面，每一个 Tomcat 上下文都表示一个 Web 应用。

在源码层面来说：Context 接口继承 Container 接口，StandardContext 类是 Container 组件的默认实现，它继承 ContainerBase 基类并实现了 Context 接口。

（2）Spring 有两个核心接口：BeanFactory 和 ApplicationContext。

其中 ApplicationContext 是 BeanFactory 的子接口。它们都可代表 Spring 容器，Spring 容器是生成 Bean 实例的工厂，并且管理容器中的 Bean。

（3）MyBatis 里面有 ErrorContext，记录本次执行过程中异常的上下文信息。

（4）在 Android 里面，Context 是个抽象类，是其他三个重要类（Activity、Service、Application）的父类。

二、Context 写作手法：熟悉而又陌生

Context 写作手法广泛存在，而且很重要。但是为什么我们在写代码的时候，不会自己潜意识的想到它，也不能自由地使用呢？我觉得：是因为我们对它没有深刻的认识。

硬生生地塞给你一个东西，一个概念，一个写作技巧，但是来龙去脉你不清楚的话，你很难将它运用自如。

三、Context 的来源和科学气息

Context 的来源，我只追溯到 Shell。Context 的根源，应该始于编译器理论。但是，Shell 里面已经包含了编译器的功能。所以，追溯的过程止步到 Shell 就可以了。

bash 是大多数 Linux 系统以及 Mac OS X 默认的 shell，它是在 1978 年前后编写。Java 诞生的时间比它晚的多。

科学家，更有科学气息，更有艺术范，“上下文”这个概念是不是比“子进程”，“容器”，“堆栈调用”，更显得高大上一点啊。其实，上下文跟上面提到的三个东西，意思都相近。

四、Shell 中的上下文：小黑和小小黑

打开 shell，呈现出一个黑色的窗口，这是一个进程。我称之为小黑。

```
Welcome to Alibaba Cloud Elastic Compute Service !

[root@iz8vb2ropxvc5nnxgzriv2z ~]# echo hello
hello
```

你可以跟小黑打个招呼：echo hello，小黑给你一个回复：hello。

最神奇的地方在于：小黑还可以诞生出一个小小黑，你可以执行：sh xxx.sh，这个时候就会诞生出一个小小黑。至于它是不是黑的，不好说，但是可以肯定的说，它是小黑诞生的。**准确的叫法是：子进程。**

只要在小黑里面执行 sh xxx.sh 命令或者./xxx.sh，小黑就会诞生出一个小小黑，既然这么容易分身，那么就有人开始担心了：我在小黑那里存的东西，小小黑是否知道呢？

确切的说，在小黑那里存放的东西，小小黑是不知道的。用户存放的东西，叫用户变量，只在当前小黑中可见，小小黑是看不见的。要想让小小黑也能看到用户存放的东西，必须设置为导出，也就是 export。天知道小黑能诞生出多少个小小黑呢，所以凡是用 export 设置的变量都统一称为环境变量。

用户变量也好，环境变量也好，所不同的只是变量所处的环境不同，总共分三种情况：在小黑里面，在小小黑里面，同时在两者里面。

小黑的东西，小小黑是看不到的，那问题来了，小小黑的东西，小黑能看到吗？答案是：不能的，因为仍然是用户变量，是被上下文所隔离开的。

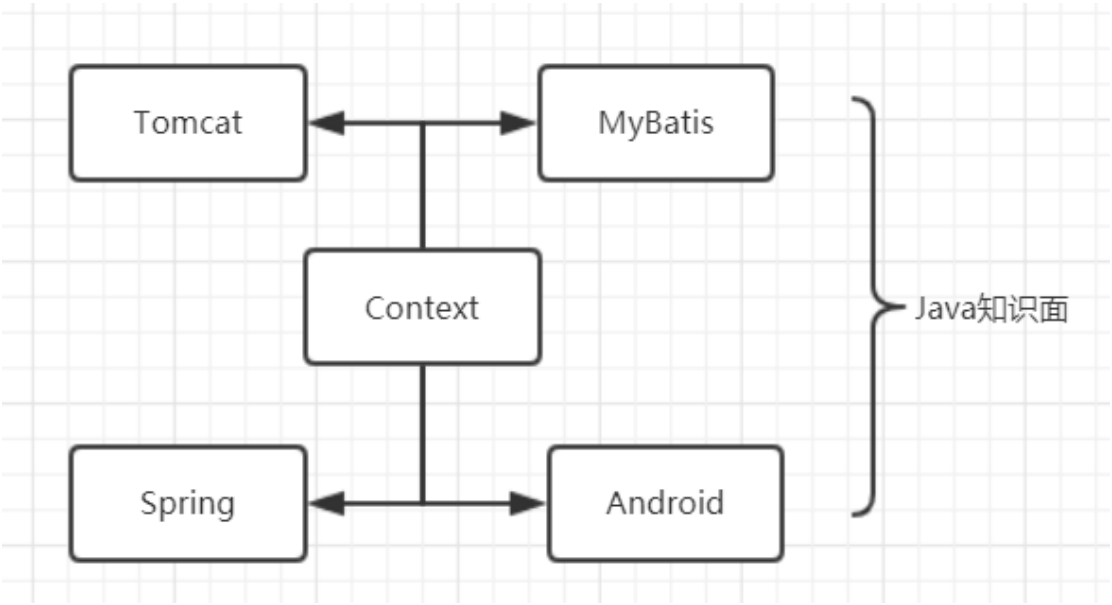
那有用户想让小小黑给小黑传话，怎么办？办不了啊，干嘛非得多此一举呢，让用户直接跟小黑对话吧，越过小小黑，用 source 的方式，也就是 source xxx.sh 的方式，这个时候就不会再平白无故冒出个小小黑了。

五、从 Java 到 Shell 建立的是 Context 的知识空间

上面我的追溯过程是从 Java 到 Shell，其实，这个过程就是建立了一个 Context 的知识空间。很多人可能没有看懂，下面我再明确的说一下。

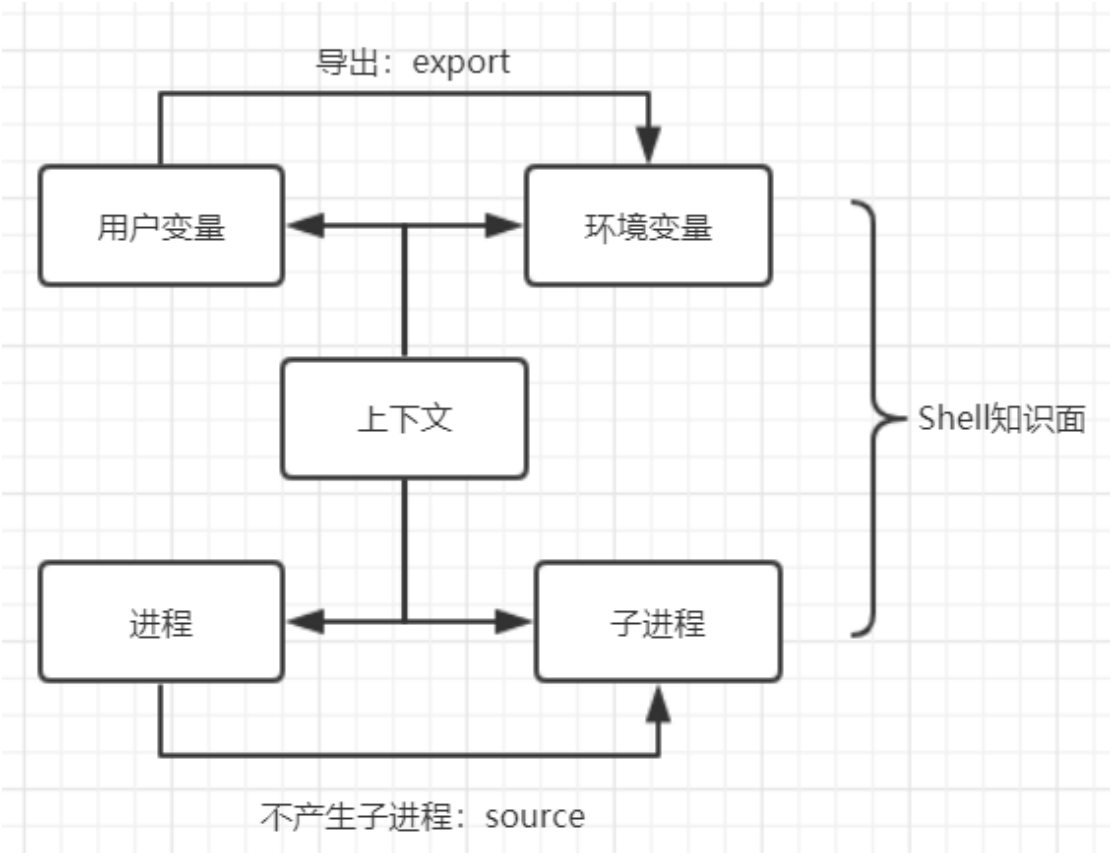
Context 的知识空间的建立分为三步：

第一步：从点到面，建立 Java 知识面。



跑来，绕去，但是仍然在 Java 里面，并没有什么厚度和深度而言。

第二步：从点到面，建立 Shell 知识面。



从 Java 延伸到 shell，是不是变的更有是深度，更有厚度了？

第三步：上面的两个知识面，连起来就是一个知识空间。

知识空间，表示知识掌握的厚度和深度。

六、应用举例

硬生生地塞给你一个东西，一个概念，一个写作技巧，但是来龙去脉你不清楚的话，你很难将它运用自如。但是，经过上面的 Context 的追溯，我相信大家对 Context 这个东西有了更深刻的认识了。

一个概念，一个术语，看似很简单，但是背后代表着一种思想，一种境界。至于是什么样的思想，什么样的境界，因人而异，只可意会不可言传，希望大家在空余时间认真去思考。

下面举个场景来说明一下 Context 的使用。

大早上，两个人见面了。**【注意：上下文就是“早上”和“见面”】**

小 A 问小 B：你吃过饭了吗?

小 B 回答：我吃过早饭了。

小 A 又问小 B：你吃的什么呢?

小 B 回答：我吃的是油条。

大概的编码手法如下所示：

首先定义一个接口：

```
public interface MeetContext
{
    public void meet();
}
```

接着，实现接口，完成早上的问候功能：

```
public class Morning implements MeetContext
{
    //提问者
    private Asker asker;
    //回答者
    private Answer answer;
    //事件队列
    private List<String> event = new ArrayList<String>(1);

    public Morning()
    {
    }

    /**
     * 启动
     */
    public void meet()
    {
```

```
        String msg = "Did you have breakfast?";
        asker.ask(msg);
        answer.answer();
    }

    public void setAsker(Asker asker)
    {
        this.asker = asker;
    }

    public void setAnswer(Answer answer)
    {
        this.answer = answer;
    }

    /**
     * 提问者
     */
    class Asker
    {

        public void ask(String msg)
        {
            event.add(msg);
            System.out.println("Asker: " + msg);
        }
    }

    /**
     * 回答者
     */
    class Answer
    {

        public void answer()
        {
            String msg = event.get(0);
            System.out.println("Answer get Asker question: " + msg);
        }

    }

    public static void main(String[] args)
```

```
    {

        Morning morning = new Morning();

        Asker asker = morning.new Asker();
        Answer answer = morning.new Answer();

        morning.setAsker(asker);
        morning.setAnswer(answer);

        morning.meet();

    }

}
```

Context 是一个概念，也是一个术语，也是一种编程思想，希望大家能吃透它，让自己写的代码有深度，有灵性。

七、学习的要求

什么是学习呢?

学就是从不了解到理解的过程；习就是学过后再反复地学，使熟练。

听众里面有我的徒弟，对于徒弟们的要求是：背过。只有背过之后，我才认为你学会了我的讲的东西。如果没有背过，我不认可，我觉得你没有学到东西。其实，这些东西并不难背啊。但是你把它记在心里，对你的成长非常有帮助的。

八、致谢

谢谢大家的参与，谢谢大家对 MyBatis 中文官网的支持！谢谢大家对我个人的支持!

我希望花费时间做出的东西，能给大家带来价值。没有带来价值，对你，对我都不是一件好事。