以下为AI生成的图文笔记的内容

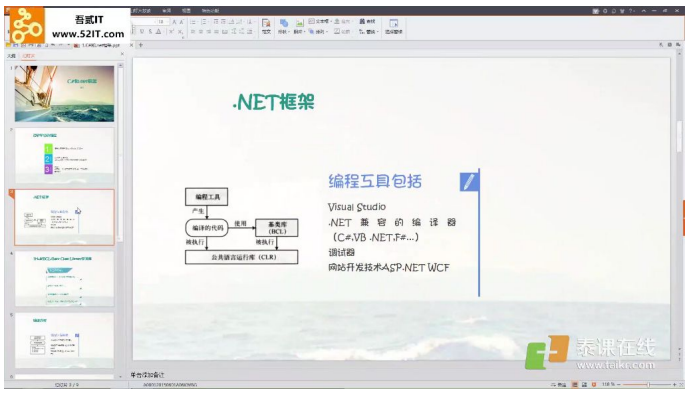
一、*C#基础篇00:04*

1. C#和.net框架介绍 00:12

1）微软平台的编程 01:34

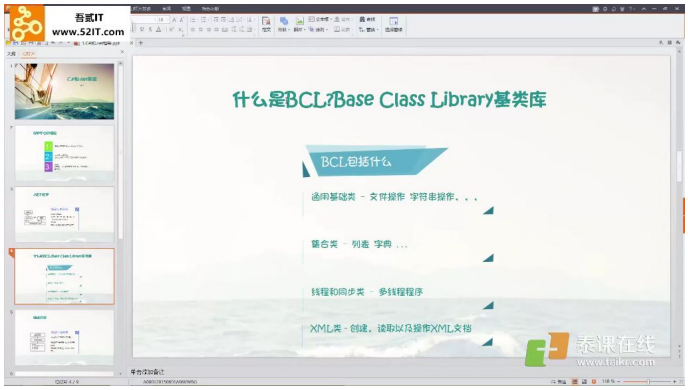
- **发展历程**:
  - 早期使用Visual Basic和C++进行编程
  - 1990年代出现Win32 API、MFC和COM(组件对象模型)
  - 2002年发布第一个.NET框架
- **.NET框架特点**:
  - 多平台支持
  - 行业标准
  - 安全性增强

2）.Net框架 02:31



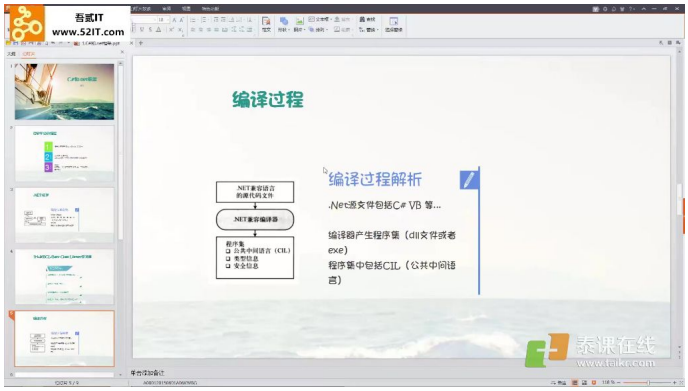
- **组成要素**:
  - **编程工具**: 包括Visual Studio和.NET兼容编译器($C\#$,$VB.NET$,$F\#$等)
  - **基类库(BCL)**: 提供基础功能支持
  - **公共语言运行库(CLR)**: 程序执行环境
  - **网站开发技术**: 如ASP.NET和WCF
- **工作流程**:
  - 通过编程工具编写源代码
  - 编译器将源代码编译为程序集
  - 程序集在CLR环境中执行

3）BCL介绍 04:40



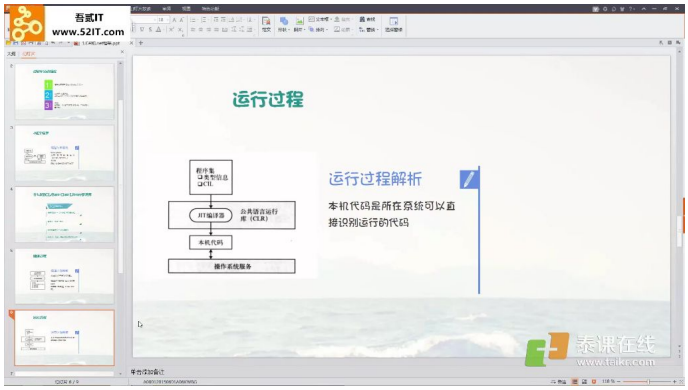
- **定义**: Base Class Library的缩写，提供系统预定义类
- **包含内容**:
  - 通用类型系统(如String类)
  - 文件操作类
  - 集合类(列表、字典等)
  - 多线程相关类
  - XML处理类
- **使用方式**: 开发者可以编写自己的类库，同时调用BCL中的类

4）编译过程解析 05:53



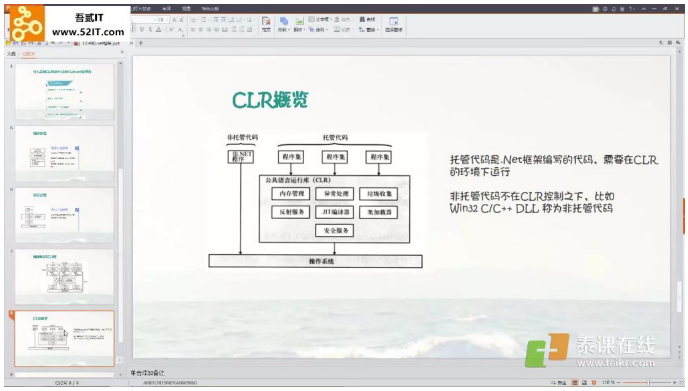
- **编译步骤**:
  - 编写.NET兼容语言源代码($C\#$,VB等)
  - 使用对应编译器编译
  - 生成程序集(dll或exe文件)
- **程序集内容**:
  - CIL(公共中间语言)
  - 类型信息
  - 安全信息
- **学习建议**: 初学者不必深究细节，先掌握基本使用

5）运行过程解析 07:51



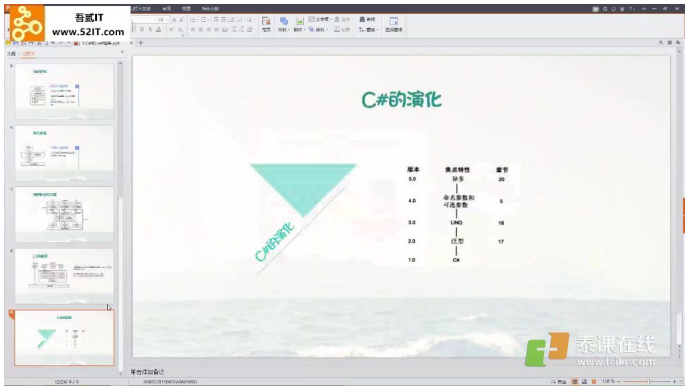
- **运行机制**:
  - 程序集被加载到CLR中
  - JIT(即时)编译器将CIL转换为本机代码
  - 本机代码直接与操作系统交互
- **关键概念**:
  - **本机代码**: 特定操作系统可直接执行的指令
  - **JIT编译器**: 负责在运行时将中间语言转换为本机代码

6）CLR概览 10:28



- **托管代码**:
  - 由.NET框架编写的代码
  - 必须在CLR环境下运行
- **非托管代码**:
  - 如Win32 C/C++ DLL
  - 直接与操作系统交互
- **CLR核心功能**:
  - 内存管理
  - 异常处理
  - 垃圾回收
  - 类加载
  - 安全服务

7）C#的演化 12:33



- **版本发展**:
  - 1.0: 基础语法
  - 2.0: 引入泛型
  - 3.0: 增加LINQ
  - 4.0: 命名参数和可选参数
  - 5.0: 异步编程支持
- **学习建议**:
  - 2.0版本后语言特性趋于稳定
  - Unity主要支持2.0版本特性

二、知识小结

知识点	核心内容	考试重点/易混淆点	难度系数
C#与.NET框架关系	C#是.NET框架下的编程语言，依赖.NET基类库（BCL）和公共语言运行时（CLR）	**CLR与JIT编译器的作用**（托管代码 vs 非托管代码）	★★
.NET发展史	从VB/C++到Win32/MFC，2002年推出.NET 1.0，支持多平台、安全性等特性	**版本演进关键节点**（如.NET 2.0稳定性）	★
编程工具链	Visual Studio（IDE）、.NET兼容编译器（支持C#/VB/F#）、调试器、ASP.NET等	**程序集（DLL/EXE）与CIL（公共中间语言）的关系**	★★★
编译与运行流程	源代码→编译器→程序集（含CIL）→JIT编译→本机代码→操作系统交互	**JIT即时编译与跨平台原理**	★★★★
基类库（BCL）	提供系统级功能（字符串处理、文件操作、多线程、XML等）	**自定义类库与BCL的调用层级**	★★
CLR核心功能	内存管理、垃圾回收、异常处理、反射服务	**托管代码与非托管代码的执行差异**	★★★

C#语言演化	从泛型、LINQ到异步编程，.NET 2.0语法趋于稳定	**Unity支持的C#版本限制**	★★
学习方法论	初学阶段“不求甚解”，先实践后复盘，逐步理解底层机制	**编程指令与机器代码的转换逻辑**	★