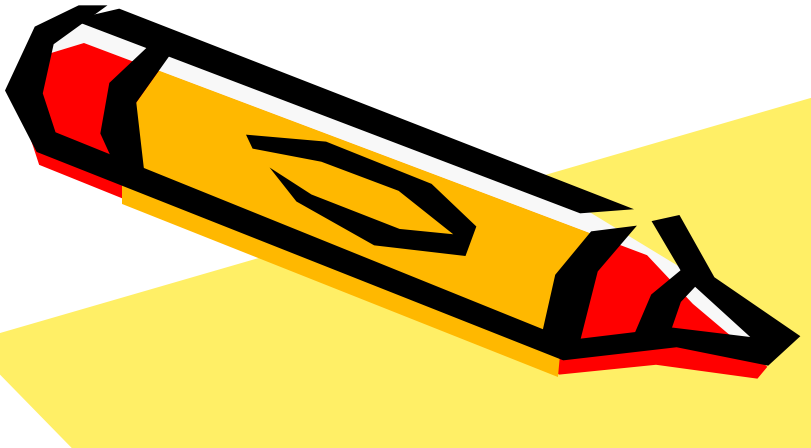
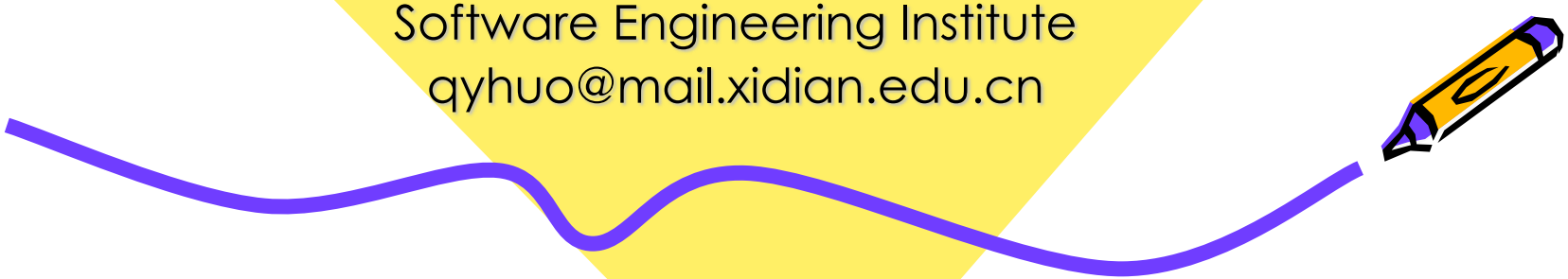
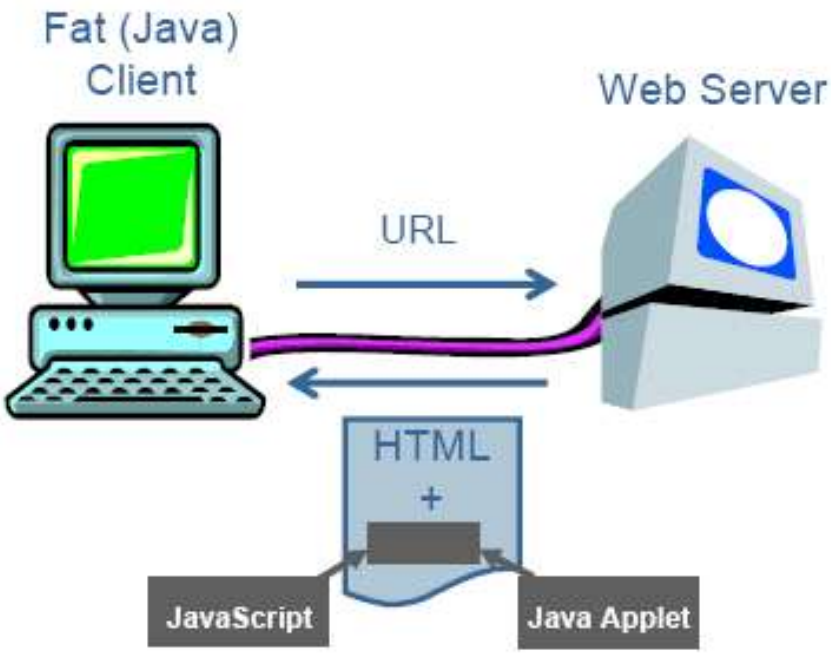


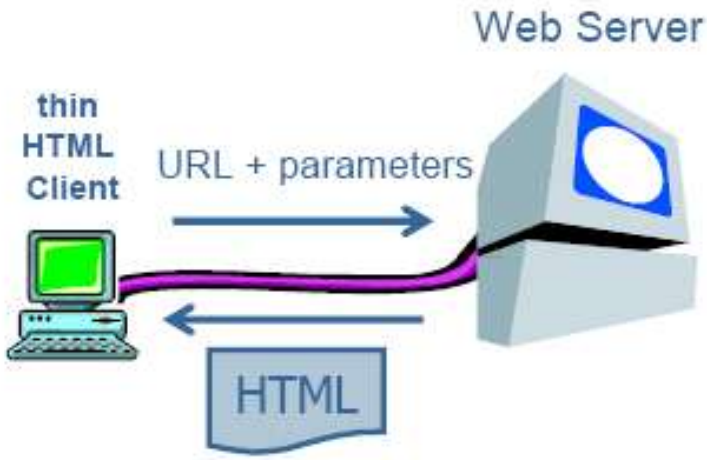
Web应用构建与部署

Qiuyan Huo 霍秋艳
Software Engineering Institute
qyhuo@mail.xidian.edu.cn



Client & Server





client-side technologies: dynamic content

- scripting language
  - JavaScript
- Java Applet
- XML
- AJAX
- ……

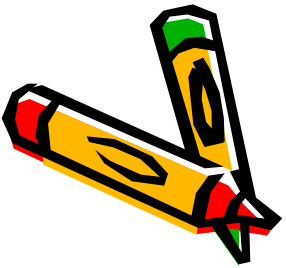
protocals

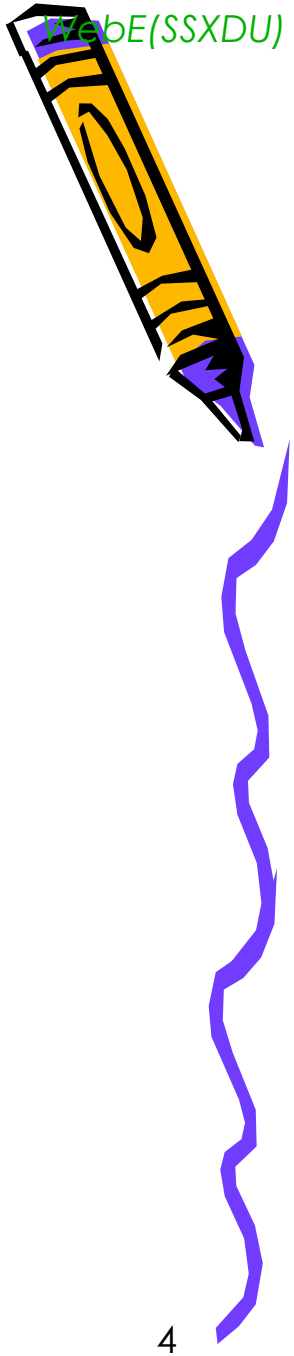
server-side technologies: dynamic HTML

- scripting languages
  - "CGI"-languages
  - PHP
  - ASP
- Java servlets + JSP
- XML + XSLT, CSS
- JSF

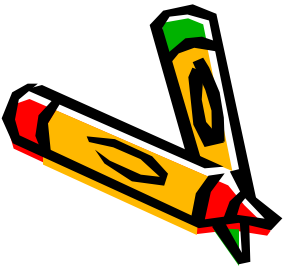
Web应用构建与部署

- Web应用构建原则
- Web应用通信协议
- Web客户端技术
- Web服务器端技术
- Web应用开发框架
- Web应用构建工具
- Web应用部署
- 总结与展望



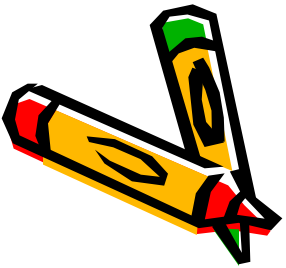


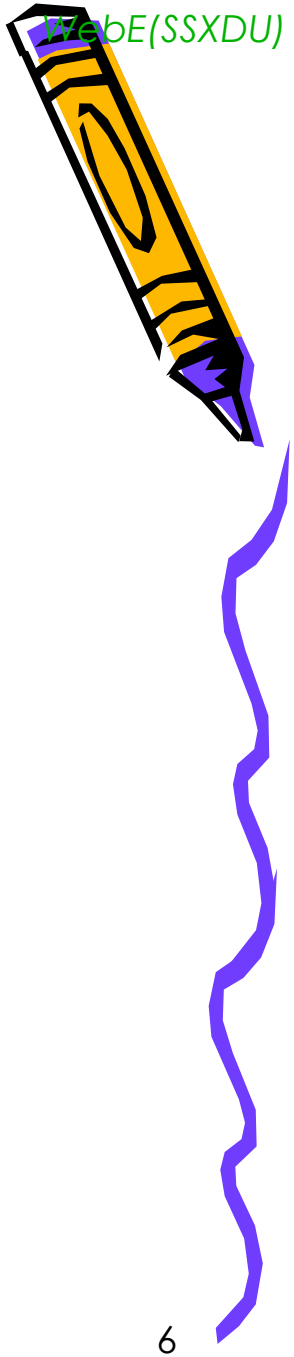
WEB应用构建原则



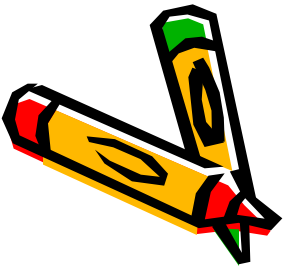
Web应用构建原则

(1) 准备原则
(2) 选择原则
(3) 编码原则
(4) 内容管理
(5) 创作原则
(6) 集成原则
(7) 重构原则
(8) 测试原则



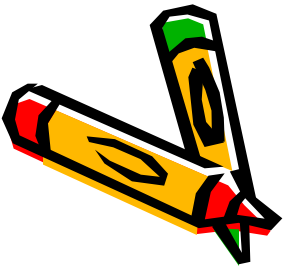


WEB应用通信协议

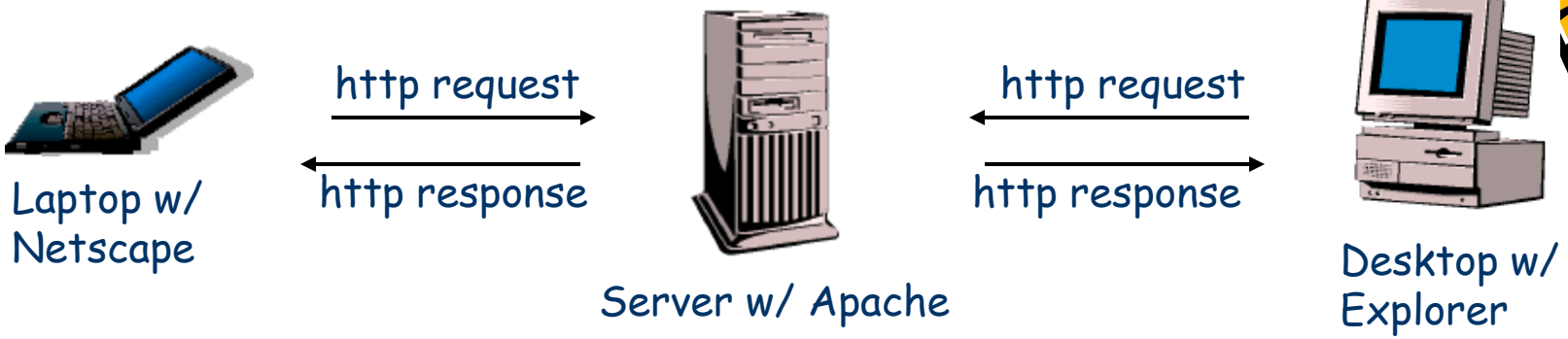


Web应用通信协议

- HTTP
- MIME
- RTP/RTSP
- MMS
- FTP
- SMTP/POP3
- RIP/NFS/DNS



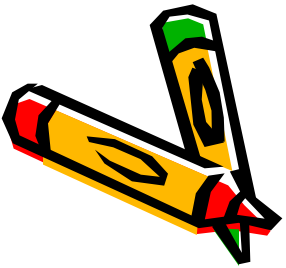
Web应用通信协议-HTTP协议



- HTTP（HyperText Transfer Protocol）超文本传输协议
- 支持客户/服务器模式(Client/Server):
  - Client: browser that requests, receives, displays object
  - Server: receives requests and responds to them
- Protocol consists of various operations
  - Few for HTTP 1.0 (RFC 1945, 1996)
  - Many more in HTTP 1.1 (RFC 2616, 1999)

Web应用通信协议- HTTP协议

- 简单快速
- 资源灵活多样
- 无连接
- 无状态(stateless)
- 双向传输
- 支持高速缓存
- 支持代理



Session Tracking

- *Session* 和Cookie类似，设计Session的目的也是为了在一个访问期间在不同的页面间传输数据以解决HTTP协议无状态的问题，但Session更加简单、更加安全。
  - 一种在客户端与服务器之间保持状态的解决方案
- Session的意义在于：在浏览器没有关闭的情形之下，一个Web应用的开始和结束。一个session可以包括数次HTTP的请求和应答.
  - 比如经常把Java里提供的javax.servlet.http.HttpSession简称为session

URL Rewriting (重写)

- URL重写实际上就是把session ID放在URL里面，一般如果browser开启cookie那么session ID就存放在cookie里面，但如果cookie被禁用那么session ID就会放在URL里面，这就是URL重写.
- 比如当你登陆http://host/path/file.html表单时进行某些操作，那些URL额外的数据就附加到表示该会话的每个URL上,并且服务器把这个标识符与关于会话所有储存的数据相关联. 重写后为 http://host/path/file?jsessionid=12345
  附加会话信息为jsessionid=12345
  即使浏览器不支持cookie或用户禁用cookie时，这种方法也能起作用.
- Drawback: URLs encoded in HTML pages have to be dynamically adapted for each session.

Cookies

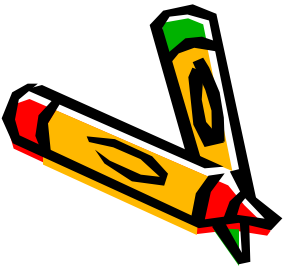
- Small text files used to store server information (e.g., session ID) on the client computer.
- Name-value pairs:

```
    Name: PREF
 Content: ID=5cecd6fe234baadb:NW=1:TM=1175654483:LM=1175654483:S=le5vRJhjrxl9XgZ1
  Domain: .google.com
    Path: /
Send For: Any type of connection
 Expires: Monday, January 18, 2038 3:14:11 AM
```

- Cookie classified as either *session* or *permanent(永久的)*
  - Exchange information transparently - easily implement session tracking
  - No major effort
  - Only a session ID generated by the server has to be transmitted.

Usage Scenarios – URL Rewriting or Cookies

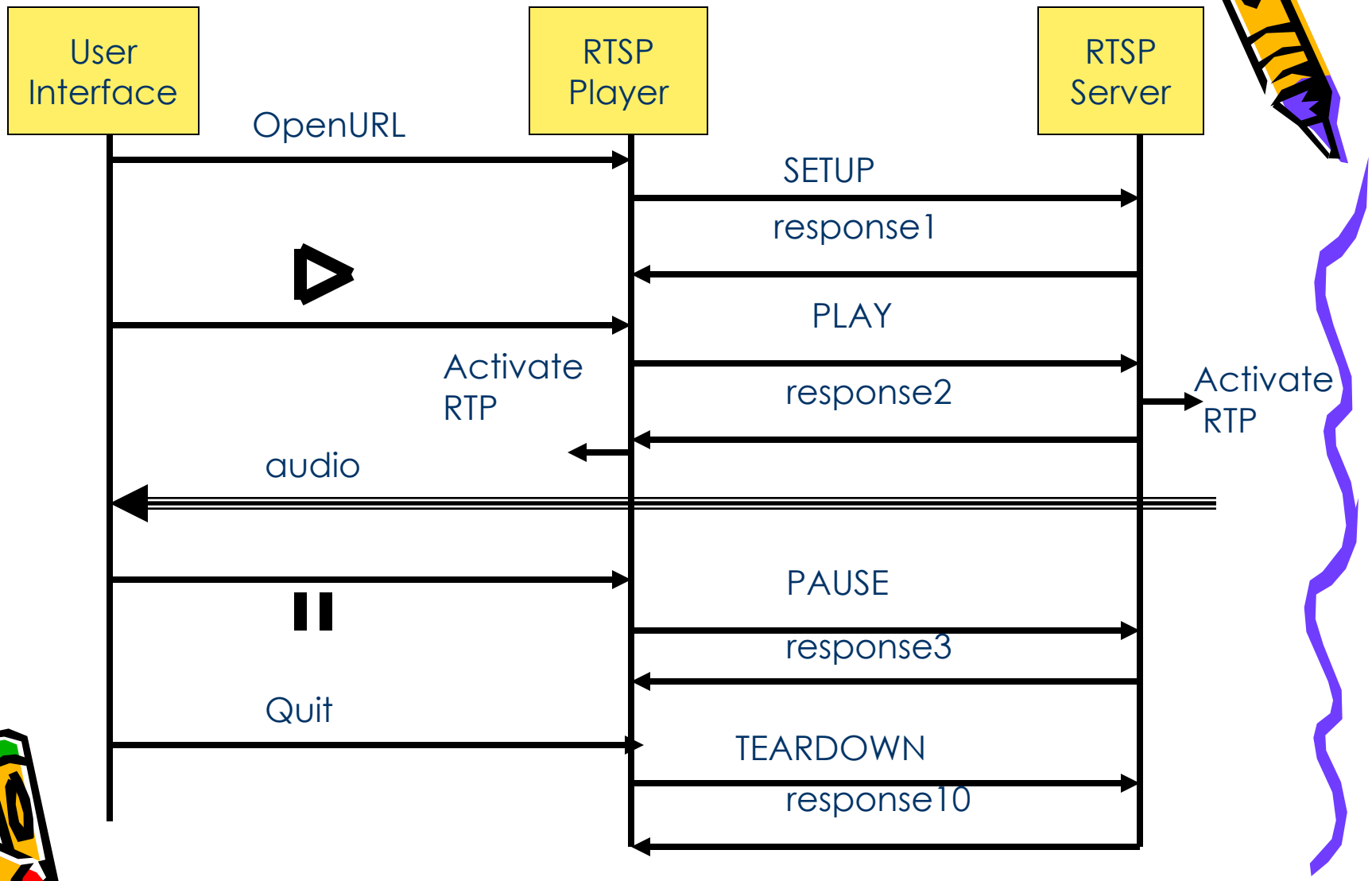
- URL 重写和Cookies的使用依赖于具体情况
- Ideally we want to leave session information on the server side and merely use a safe session ID.
- Combined
  - using a session ID
  - Using URL rewriting for browsers that won't accept cookies.
  - E.g., http://host/path/file?jsessionid=XYZ to http://host/path/XYZ/file
    - XYZ represents a unique key for the session which should be hard to guess.



Web应用通信协议

- MIME协议
  - MIME（Multipurpose Internet Mail Extensions）--多功能因特网邮件扩充服务
- RTP/RTSP协议
  - RTP（Real-time Transport Protocol）--实时传送协议
  - RTSP（Real Time Streaming Protocol）--实时流传输协议
- MMS协议
  - MMS（Microsoft Media Server Protocol）--微软媒体服务器协议
- FTP协议
  - FTP（File Transfer Protocol）--文件传输协议

RTSP Media Player-Server Sequence Chart



RTSP

- RTSP(实时流协议)建立并控制一个或几个时间同步的连续流媒体，如音频和视频。
  - 尽管连续媒体流与控制流交叉是可能的，RTSP 本身并不发送连续流。换言之，RTSP 充当多媒体服务器的网络远程控制。
  - RTSP 提供了一个可扩展框架，实现实时数据（如音频与视频）的受控、按需传送。数据源包括实况数据与存储的剪辑。
  - RTSP 用于控制多个数据发送会话，提供了选择发送通道（如 UDP、组播 UDP 与 TCP 等）的方式，并提供了选择基于 RTP 的发送机制的方法。
- 该协议支持如下操作：
  - 从媒体服务器上检索媒体：用户可通过 HTTP 或其它方法提交一个演示描述请求；
  - 媒体服务器邀请进入会议： 媒体服务器可被邀请参加正进行的会议，或回放媒体，或记录部分或全部演示；
  - 将新媒体加到现有演示中：如服务器能告诉客户端接下来可用的媒体内容，对现场直播显得尤其有用。

IETF（www.ietf.org）

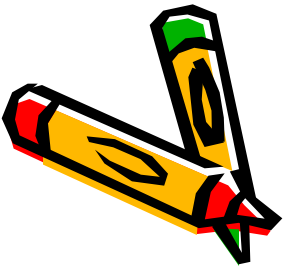
Example of a RTSP Description File

```
<title>Twister/title>
<session>
    <group language=en lipsync>
    <switch>
        <track type=audio
                    e="PCMU/8000/1"
                    src= "rtsp://audio.example.com/twister/
                            audio.en/lofi">
        <track type=audio
                    e="DVI4/16000/2" pt="90 DVI4/8000/1"
                    scr="rtsp://audio.example.com/twister/
                            audio.en/hifi">
    </switch>
    <track type="video/jpeg"
            scr="rtsp://video.example.com/twister/video">
    </group>
</session>
```

back

Web应用通信协议

- SMTP协议和POP3协议
  - SMTP（Simple Mail Transfer Protocol）--简单邮件传输协议
  - POP3（Post Office Protocol V3）--邮局协议版本3
- RIP协议、NFS协议、DNS协议



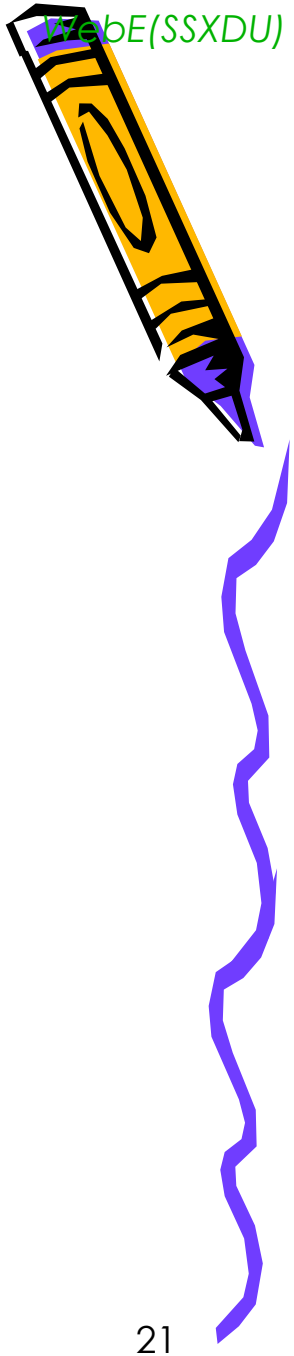
SMTP & POP3

- SMTP 是一种提供电子邮件传输的协议，是建模在 FTP 文件传输服务上的一种邮件服务，主要用于传输系统之间的邮件信息并提供来信有关的通知。SMTP 独立于特定的传输子系统，且只需要可靠有序的数据流信道支持。
- 结合POP3 (Post Office Protocol:邮局协议) 或IMAP (Internet Message Access Protocol:因特网信息访问协议)进行Email的收发。
- SMTP进一步用于基于SOAP的异步消息交换。

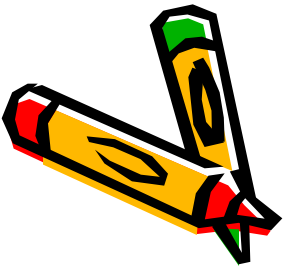
SMTP 由 IETF（www.ietf.org）在 RFC2821中定义。

```
public void postMail( String recipients, String subject,  String message , String
    from) throws MessagingException    {
      //Set the host smtp address
       Properties props = new Properties();
       props.put("mail.smtp.host", SMTP_HOST_NAME);

       props.put("mail.smtp.auth", "true");

      Authenticator auth = new SMTPAuthenticator();
      Session session = Session.getDefaultInstance(props, auth);
      session.setDebug(debug);
      // create a message
      Message msg = new MimeMessage(session);

      // set the from and to address
      InternetAddress addressFrom = new InternetAddress(from);
      msg.setFrom(addressFrom);
      InternetAddress addressTo = new InternetAddress(recipients);
      msg.setRecipient(Message.RecipientType.TO, addressTo);

      // Setting the Subject and Content Type
      msg.setSubject(subject);
      msg.setContent(message, "text/plain");
      Transport trans = session.getTransport("smtp");
      trans.send(msg);
      trans.close();
  }
```

Example of sending Email using SMTP in Java

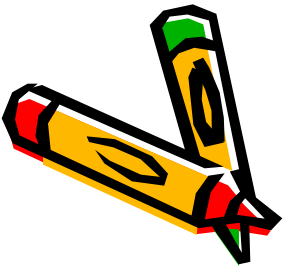


WEB客户端技术



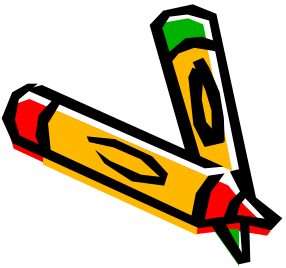
Web客户端技术

- HTML/HTML5
  - HTML（HyperText Markup Language）--超文本标记语言
- XHTML
  - XHTML（The Extensible HyperText Markup Language）--可扩展超文本标识语言
- DHTML
  - DHTML（Dynamic HTML）--动态HTML
- CSS
  - CSS（Cascading Style Sheets）--层叠样式表
- Flash/Flex技术



Web客户端技术

- DOM
  - DOM（Document Object Model）--文件对象模型
- JavaScript/AJAX
  - AJAX（Asynchronous JavaScript and XML）--异步JavaScript和XML
- ActiveX/Silverlight
- Applet/JavaFX
- VRML与X3D
- XML



Java Applets

- 由Java写的小应用程序被装载到客户机browser中的“sandbox”中运行，browser需要有JVM运行环境。Applets被编译为bytecode,因此可以运行于任何具有JVM的platforms。
  - 不持久存储
  - Java小程序的源文件有三种，后缀名分别是.java, .class和.jar。
    - 编辑.java文件
    - 运行.class文件
    - 运行.jar文件，如果你有它，并在<applet>中加上archive="*.jar"，则可以在最新的浏览器中加快载入速度。

ActiveX Controls

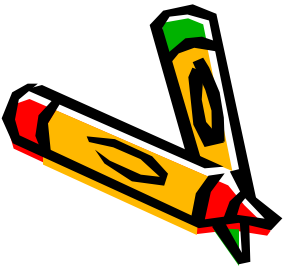
- ActiveX的整体技术是由Microsoft的COM (Component Object Model,组件对象模型)构筑的.
- ActiveX Controls (控件)是标准的COM组件,用来提供特定接口．构筑包括从与用户交互和适应COM的事务处理监视器到Web服务器、全部实现自动化的机构.
  - 存在在浏览器特定的缓存目录(cache directory)
  - Binary code
  - As a Plug-in or helper
  - Can access all system areas and functions of the user who owns the security context it runs in.
    - Security risk!
      - MS allows vendors to use crypto (*密码术的*) method to sign these components.
  - Developed in an arbitrary(*任意的*) language, including Java, VB, C++, as long as the language's compiler meets the required COM specifications.

AJAX

- “Asynchronous JavaScript + XML”
- 使用AJAX可以构建更为动态和响应更灵敏的Web应用程序.
- 对浏览器端的JavaScript、DHTML和与服务器异步通信的组合.
- Ajax包含：
  - 基于XHTML和CSS标准的表示；
  - 使用DOM进行动态显示和交互；
  - 使用XMLHttpRequest与服务器进行异步通信；
  - 使用JavaScript绑定一切。

Silverlight, Flex and JavaFX

- approaches to next-gen RIA development
- Silverlight brings the power of the .NET framework and XAML directly to the browser.
- Flex builds upon its already strong ActionScript foundation by providing a way to declaratively create user interfaces with their new XML based language, MXML.
- JavaFX ditches XML all together, using a new scripting language and classes dedicated to vector graphics and animation.



XML

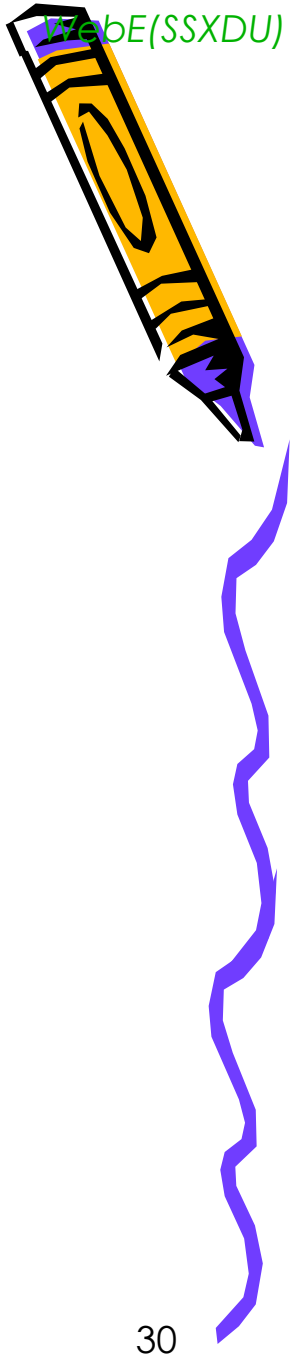
- Markup：文本标记
- XML比较适合于标记文档
- E.g.：用户名、密码、所在部门、性别、年龄

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <user>
3       <name>张三 </name>
4       <password>123456</password>
5       <department>技术部</department>
6       <sex>男</sex>
7       <old>30</old>
8 </user>
```

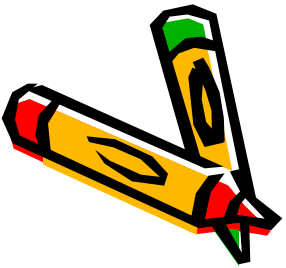
JSON

- JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，基于文本，Unicode编码。
- JSON却更适合于时行数据交换处理。
- E.g.：用户名、密码、所在部门、性别、年龄

```
1 {
2       "name":"张三",
3       "password":"123456",
4       "department":"技术部",
5       "sex":"男",
6       "old":30
7 }
```

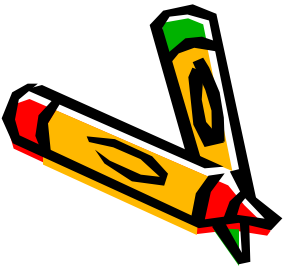


WEB服务器端技术



Web服务器端技术

- Web应用服务器端开发技术
- 中间件技术
- Web服务

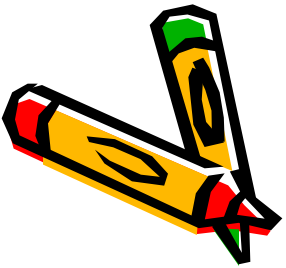


Web应用服务器端开发技术

- CGI
  - CGI（Common Gateway Interface，公共网关接口）
- ISAPI
  - ISAPI（Internet Server Application Programming Interface，因特网服务器应用程序接口）
- PHP
  - PHP（PHP：Hypertext Preprocessor，超级文本预处理语言）
- ASP/ASP.NET
  - ASP（Active Server Page，动态服务器页面）

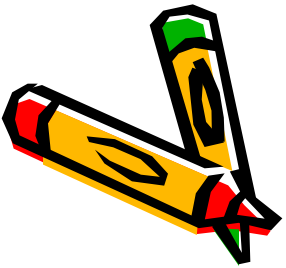
Web应用服务器端开发技术

- Servlet
- JSP
  - JSP（Java Server Page，Java服务器页面）
- Perl/Python/Ruby
- ColdFusion



中间件技术

- 应用服务器
  - IIS（Internet Information Server，因特网信息服务器）
  - Apache
  - Oracle Weblogic Server
  - IBM WebSphere
- EJB
  - Session Bean（会话bean）
  - Entity Bean（实体bean）
  - Message Driven Bean（MDB，消息驱动bean）
- 消息系统



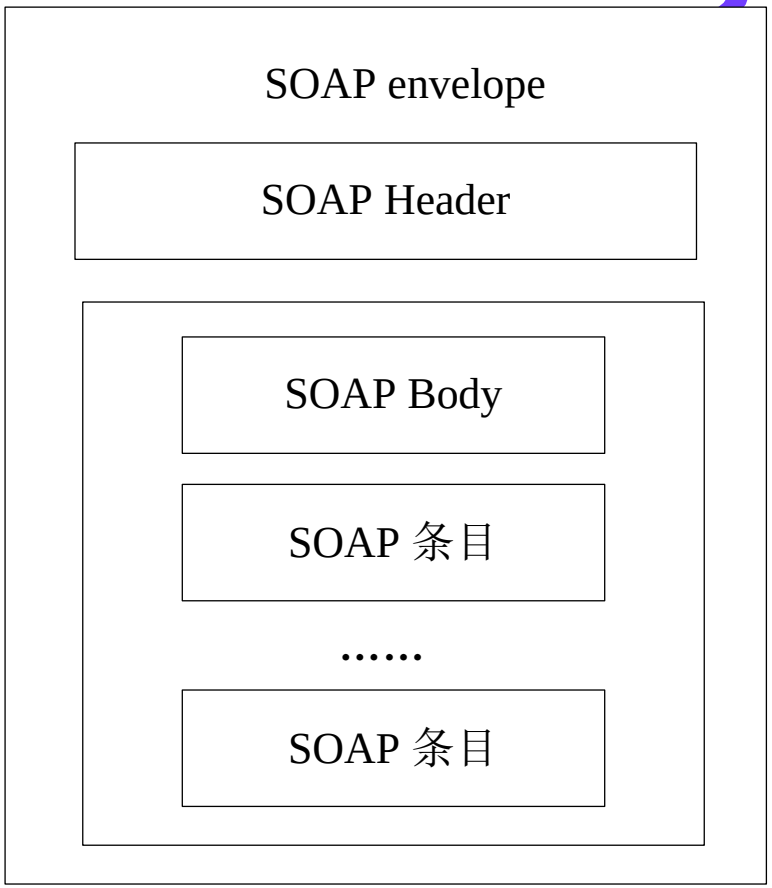
Web服务

Web服务协议栈

Process	BPEL4WS
Universal Description, Discovery, and Integration	UDDI
Services Description	WSDL
Messaging	SOAP
Extensible Markup Language	XML
Transport Protocols	HTTP、SMTP等

Web服务

- SOAP（Simple Object Access Protocol，简单对象访问协议）
- 内容
  - SOAP封装
  - SOAP编码规则
  - SOAP RPC表示
  - SOAP绑定
- SOAP消息结构



SOAP 实例

- SOAP 请求

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>

</soap:Envelope>
```

http://www.w3school.com.cn/soap/soap_example.asp

SOAP 实例

- SOAP 响应

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>

</soap:Envelope>
```

http://www.w3school.com.cn/soap/soap_example.asp

Web服务

- WSDL（Web Services Description Language，Web服务描述语言
- 服务接口和服务实现元素

分类	名称	作用
抽象定义	消息（Message）	抽象定义了在通信中使用的消息的数据结构。
	类型（Type）	数据类型的容器，包含了所有在消息定义中需要的XML元素的类型定义。
	端口类型（PortType）	定义了一种服务访问入口的类型，包含若干操作。
	操作（Operation）	每个操作代表访问入口支持的一种类型调用，WSDL支持包括单向请求、单向响应、请求响应、响应请求四种访问入口调用模式。
具体定义	服务（Service）	描述一个具体的被部署的Web服务所提供的所有访问入口的部署细节，一个服务可包含多个服务访问入口。
	绑定（Binding）	定义某个端口类型的具体协议和数据格式规范的绑定。
	端口（Port）	描述服务访问入口的细节，包括地址、消息调用模式。

WSDL

- WSDL 实例

```
1  <message name="getTermRequest">
2     <part name="term" type="xs:string"/>
3  </message>
4
5  <message name="getTermResponse">
6     <part name="value" type="xs:string"/>
7  </message>
8
9  <portType name="glossaryTerms">
10   <operation name="getTerm">
11     <input message="getTermRequest"/>
12     <output message="getTermResponse"/>
13   </operation>
14 </portType>
```

http://www.w3school.com.cn/soap/soap_example.asp

WSDL

- E.g

```
<message name="getTermRequest">
   <part name="term" type="xs:string" />
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string" />
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
      <input message="getTermRequest" />
      <output message="getTermResponse" />
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
     soapAction="http://example.com/getTerm" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
</binding>
```

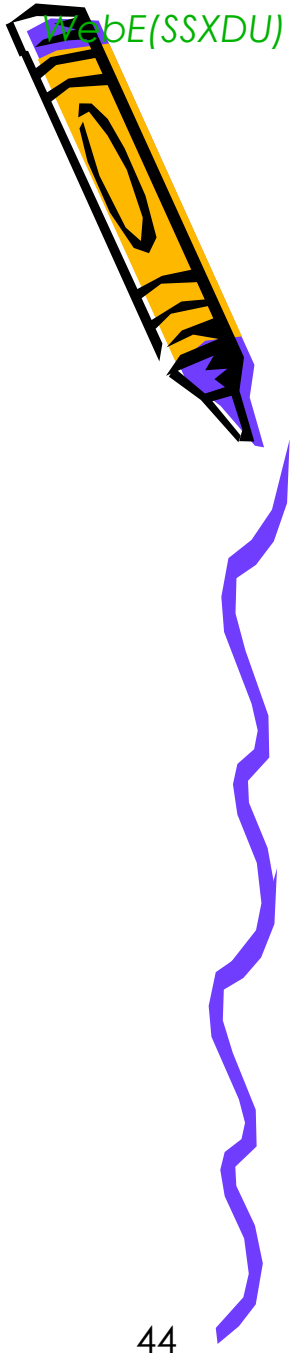
Web服务

- UDDI（Universal Description Discovery and Integration）--统一描述、发现和集成服务
- UDDI注册提供的信息包含:
  - 白页：UDDI注册者提供的基本信息，包括联系地址、联系人和相关的联系标识符；
  - 黄页：根据标准分类法进行的行业分类类别；
  - 绿页：服务发布者提供的公开大众的服务技术信息，是服务使用者所需要的全部内容。

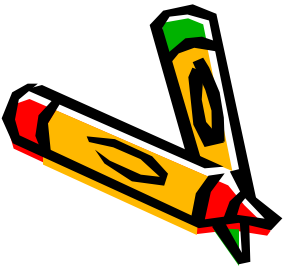
Web服务

- BPEL4WS（Business Process Execution Language for Web Services）--Web服务的业务流程执行语言，也被称为BPEL或BPELWS
- 特性
  - 灵活性
  - 嵌套组装
  - 关注点分离
  - 会话状态和生命周期管理
  - 可恢复性

Web服务开发环境

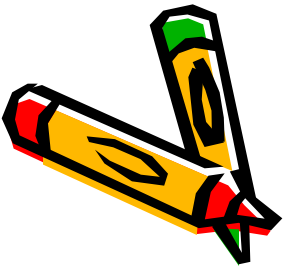


WEB应用开发框架



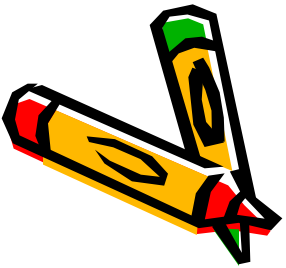
Web应用开发框架

- Java EE开发框架
- .NET框架
- Web层开发框架
- Ruby框架
- Python框架
- Web服务开发框架
- Web应用开发框架的选择



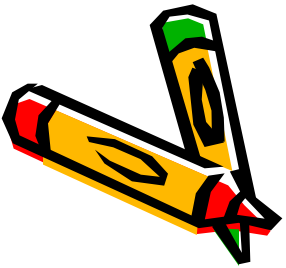
Java EE开发框架

- SSH（Spring、Struts和Hibernate）
- WebWork
- Tapestry
- JSF
- Turbine
- Maverick框架
- OPS



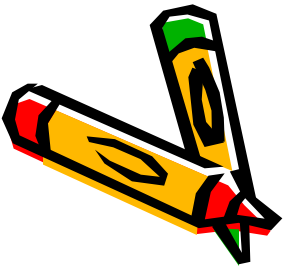
.NET框架

- 两个关键组件
  - 公共语言运行时 (Common Language Runtime，CLR) 和 .NET框架类库
- 提供了运行时环境，和庞大的框架类库。
  - 框架类库可以充分地降低将开发人员编程的难度，让开发人员更为轻松地完成开发工作。



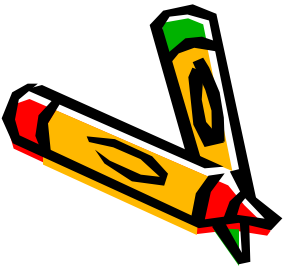
Web层开发框架

- WebPage3.0
- Flex框架
  - Cairngorm
  - PureMVC
  - Model-Glue:Flex
  - Foundry
  - Guasax
  - ARP
  - Flest



Web层开发框架

- AJAX框架
  - Prototype
  - jQuery
  - Mootools
  - DOJO
  - Ext JS
  - Ajax.NET
  - AFLAX

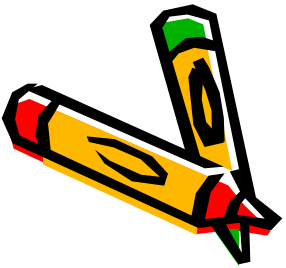


AJAX 应用框架

- AJFORM：一个极易上手的AJAX框架，被用来编写入门级的AJAX代码。
- Tibet：提供了大量的易移植和完整的JavaScript API，通过这些可以快速生成大量的客户端代码，Tibet自称是企业级AJAX。
- qooxdoo：一个发展迅猛的应用框架，提供广泛的UI支持，正在开发基础架构等特性。
- Open Rico：一个支持Ajax架构和用户交互的多用途框架。
- DOJO：提供完整的轻量级窗口组件和浏览器-服务器消息映射支持。
- BackBase：一个完整的浏览器端框架，提供了丰富的浏览器操作功能，以及对.NET和JAVA平台的集成。
- Bindows：一个通过DHTML、JavaScript、CSS和HTML等技术强劲联合起来的一套完整的Windows桌面式的WEB应用程序解决方案。

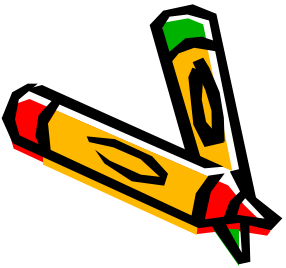
Ruby框架

- 包括RoR、Camping、Merb、Nitro、RoR、Ramaze、Sinatra和Bowline GUI框架
- RoR（Ruby on Rails）是一个用于编写Web应用的框架，基于编程语言Ruby。
- RoR使Web应用的开发人员有了一种新的选择，给开发人员带来的感觉不仅仅是一个开发工具。



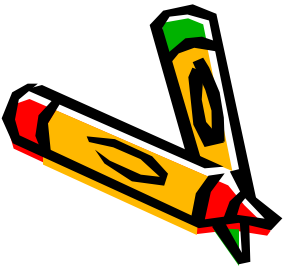
Python框架

- 包括CherryPy、CubicWeb、Flask、Pylons以及最新发布的BlueBream1.0等等。
- Zope是一种让具备不同技能的开发人员一起构建Web应用的开源框架。



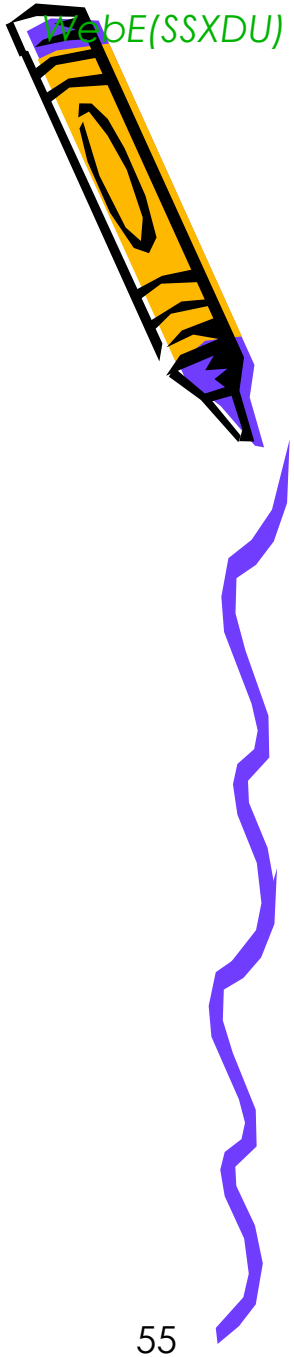
Web服务开发框架

- Axis(2)
- .NET FrameWork
- Xfire
- Apache CXF
- ActionWebService
- Python Web服务框架

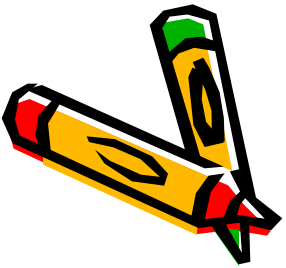


Web应用开发框架的选择

- 在选择开发框架的过程中需要注意：
  - 选择能够对开发过程提供更多、更好帮助的Web应用开发框架；
  - Web应用开发框架的学习要简单，上手要快；
  - 要有很好的技术和文档支持；
  - Web应用开发框架结合其它技术的能力要强；
  - Web应用开发框架的扩展能力要强；
  - Web应用开发框架最好能提供可视化的开发和配置；
  - Web应用开发框架的设计结构要合理；
  - Web应用开发框架要是运行稳定的，运行效率高的；
  - Web应用开发框架要能很好的结合目前公司的积累；
  - 注意判断应用的场景和开发框架的适用性。

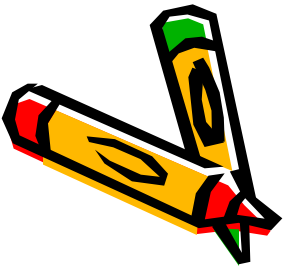


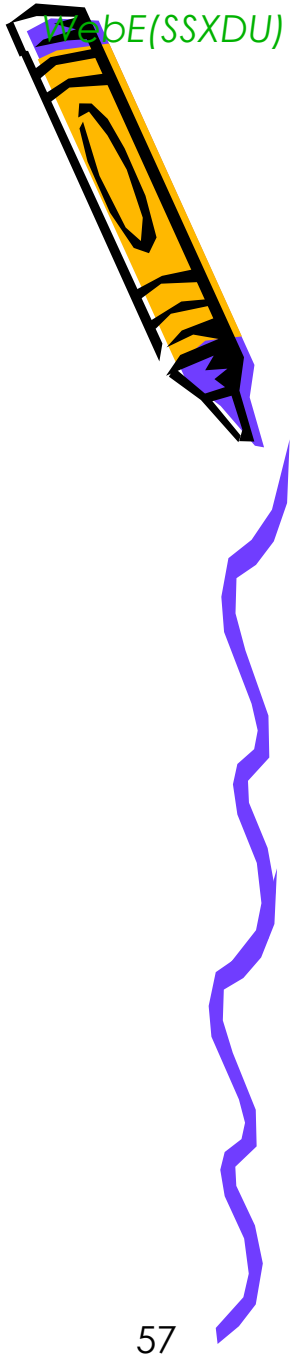
WEB应用构建工具



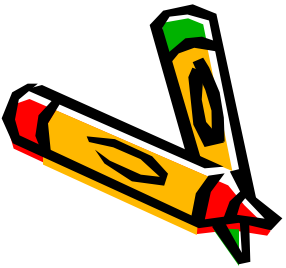
Web应用构建工具

- Visual Studio
- Eclipse
- Netbeans
- Jbuilder
- WebDB
- Zend PHP Studio
- AppEngine





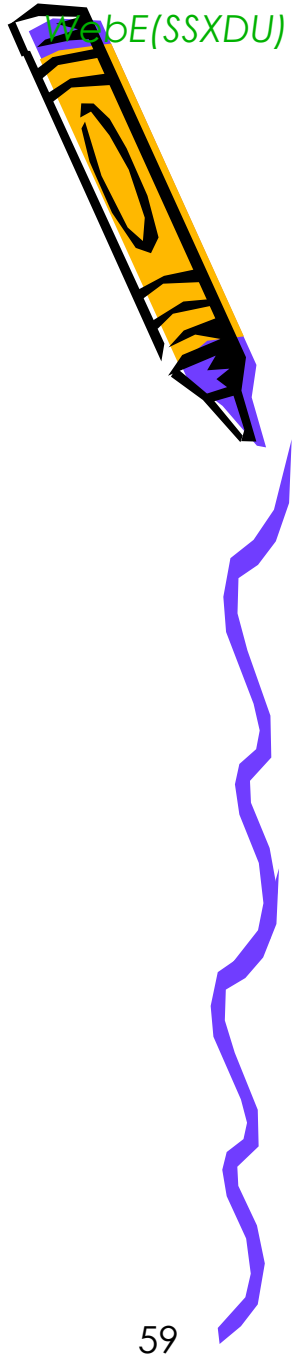
WEB应用部署



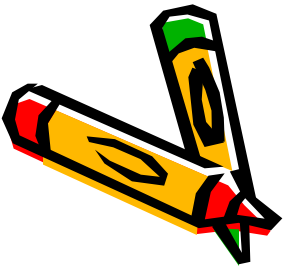
Web应用部署

- 部署粒度
- 部署原则
  - 管理客户对Web应用增量的期望；
  - 安装与测试交付包；
  - 交付前建立支持制度；
  - 先改正有缺陷的Web应用，然后再交付。
- 部署环境
- 版本控制和CMS

生产环境——压缩代码

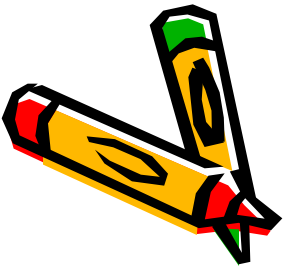


总结与展望



总结与展望

- Web应用开发涉及到Web应用通信协议、客户端开发技术、服务器端开发技术、开发框架、开发工具等内容，选择合适的开发技术和开发框架可以提高Web应用开发的效率和质量。
- 而Web应用的发布与部署的粒度特性，使其需要遵循一些原则，了解部署环境，做好版本和内容的管理与控制。
- 很难说未来Web工程中会占据主导地位的技术是什么，不过Web技术一定会像更强大更易于使用的方向发展。



Project Task：Task7

- Web应用构建
  - 选择适当的技术，完成Web应用（部分功能）构建。
    - Your Web application must have at least 5 webpages, counting the homepage.
    - Try to make each webpage about something different
  - 部分功能要包含架构的各种组件。如：架构中包含数据库，就需要实现访问数据库存储的功能。
- Web应用部署
  - 在Web应用构建过程中，选择适当粒度进行部署。