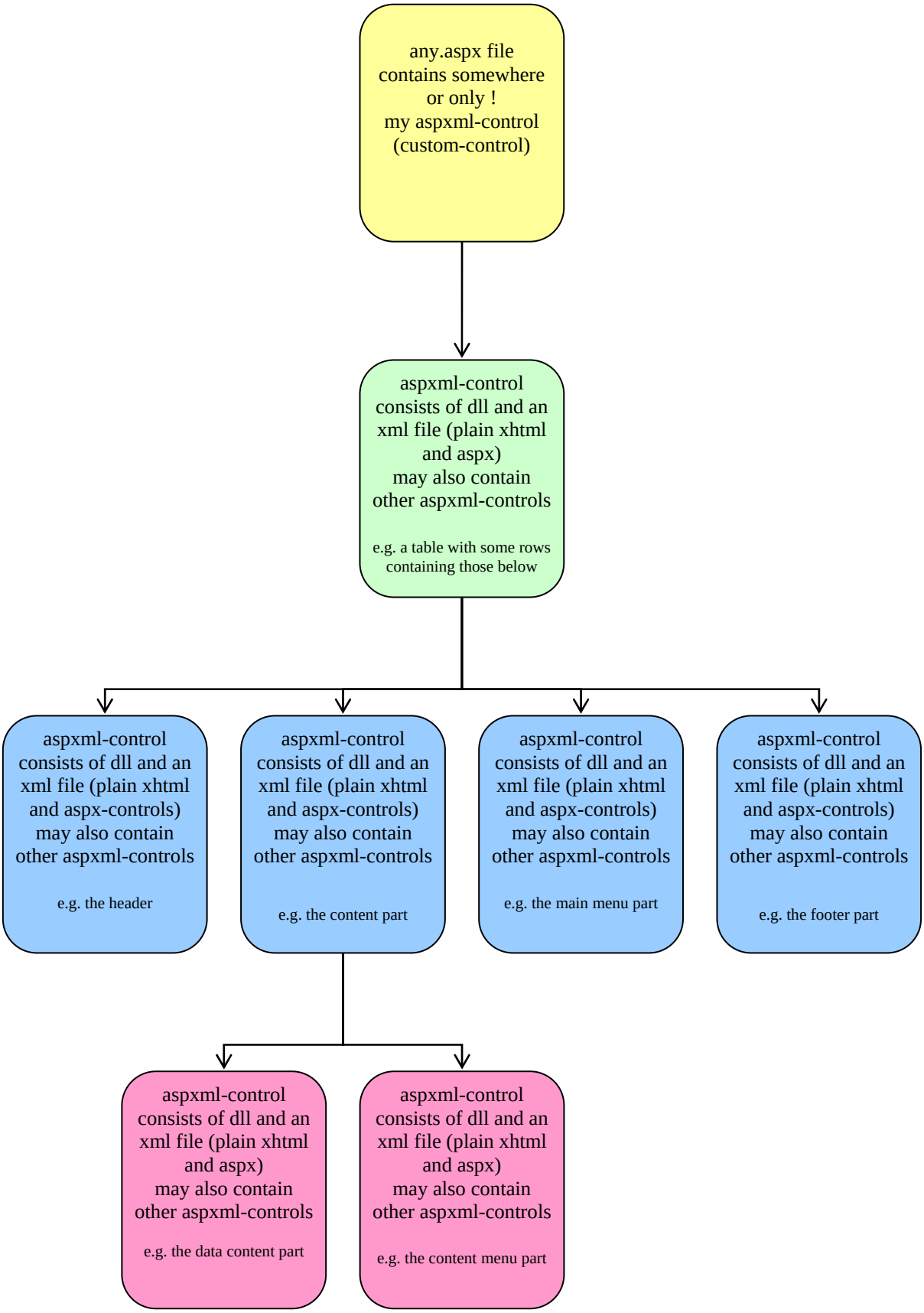
template engine for *asp.net-portals*

templating mechanism scheme:



Recursive parsing process:

The **aspxml-control parses recursive its xml definition file, which is plain xhtml and aspx**. The control uses a Html-XSD and a Aspx-XSD (status asp.net 1.1 atm) and additional information to check correctness, if any file isn't valid x(ht)ml, **the control is able to convert invalid html into valid xhtml**.

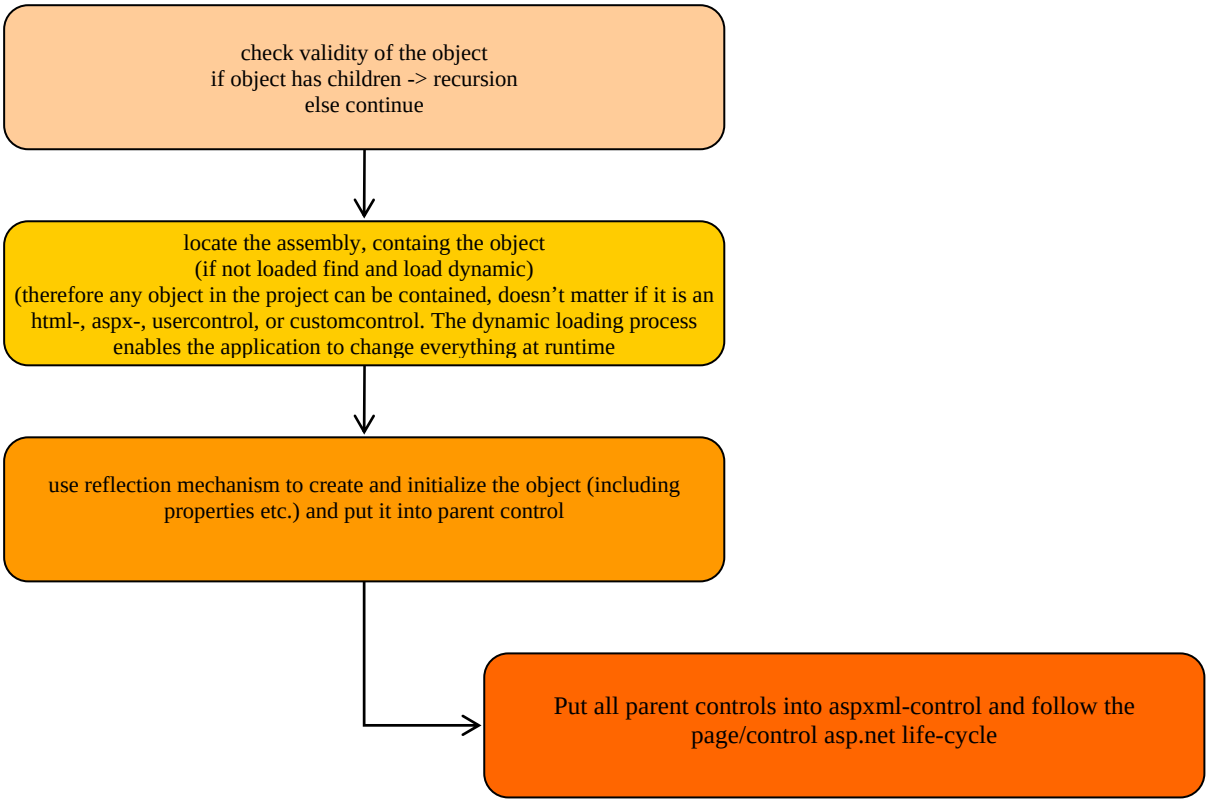
This means **you can develop the xml-file template(s) as a user/custom-control with Visual-Studio or anything else and just upload it**. At the server a tool could cut the body part to an xml file and add the xml-headers for the html-XSD and aspx-XSD and copying the dll to a directory.(a real simple tool could be able to do the same local).
You can easily write a tool, that is able to change the xml definition files and inserts objects (xml-aspx-tags) enabling you and other developers to **upload any control and deploy it at runtime into your application !**

The aspxml-control defines itself completely through this xml-file, which get cached and automatically updated on changes (This enables the mentioned changes at runtime, like uploading an custom-control or an aspxml-control replacing or extending another one !!!).

The aspxml-control **allows you to load different definition xml-files by setting the source path** (which can be an uri or a resource in an dll, you can also load the information from a database, store it in your application-cache and proceed) of the xml file as a property (e.g. depending on authorisation issues). Also localisation might be realized through this, but in my opinion it is more convenient to do this inside code and database.

If you want an aspxml-control that consist only of static content (e.g. footer), simply set a flag and enforce the asp.net control caching mechanism (to ensure speed of the parsing process you should separate static and dynamic content objects into different aspxml-controls wherever possible, or implement your user/custom controls with the build-in asp.net caching mechanism).

Each found (sub)Tag (object) in the xml file, will start the recursive process:



Easy Sample for an definition file (as you see it is simply using an extended aspxsd and can be mixed up with xhtml)

```
<asp:aspxmlcontrol xmlns:asp="http://www.manuelcavallaro.com/AspNet/Controls"
	xmlns="http://www.manuelcavallaro.com/AspNet/HtmlBody">

<asp:TextBox>Welcome</asp:TextBox>

<asp:Table style="Z-INDEX: 101; LEFT: 5px; POSITION: absolute; TOP: 5px" BorderWidth="1px"
BorderColor="#000033" BorderStyle="Solid" Width="90%" EnableViewState="true" ID="vmenu1">
	<asp:TableRow EnableViewState="true">
		<asp:TableCell BorderWidth="1px" BorderColor="#000000" EnableViewState="true">
			Yo !
		</asp:TableCell>
		<asp:TableCell BorderWidth="1px" BorderColor="#000000">
			check this out !
		</asp:TableCell>
	</asp:TableRow>
	<asp:TableRow>
		<asp:TableCell BorderWidth="1px" BorderColor="#000000">
			<a href="AppInfo.aspx">AppInfo<asp:Label ForeColor="#cc0000" Font-
			Bold="true" BackColor="#66cccc">HelloWorld</asp:Label></a>
			<asp:Button ID="Button1" Width="100px" EnableViewState="true"
			Text="losgehts"/>
				<asp:TextBox>Hello</asp:TextBox>
		</asp:TableCell>
		<asp:TableCell BorderWidth="1px" BorderColor="#000000" EnableViewState="true">
			<asp:DropDownList OnSelectedIndexChanged="Handler.Button1_Click"
			ID="DropDownList2" Width="100px" EnableViewState="true"
			AutoPostBack="true">
			<asp:ListItem Value="hi"></asp:ListItem>
			<asp:ListItem Value="Fred" Selected="true"></asp:ListItem>
				</asp:DropDownList>
			</asp:TableCell>
	</asp:TableRow>
</asp:Table>
</asp:aspxmlcontrol>
```

The project is working, but far from Release. The administration controls (user-friendly administration of the whole process) aren't done atm, I stopped the project December 2003, because a part of the functionality will get implemented with Masterpages in Asp.net 2.0 (announced for 2005).

The complete portal engine is far to heavy just to do it for fun (contains more than this templating mechanism, a portal engine, url rewriting engine, localization engine, dynamic menus, caching mechanisms and many more)

If the new Masterpage-concept in Asp.net 2.0 does not meets requirements, i'll continue, else it will go open-source.

Please contact me for further details.