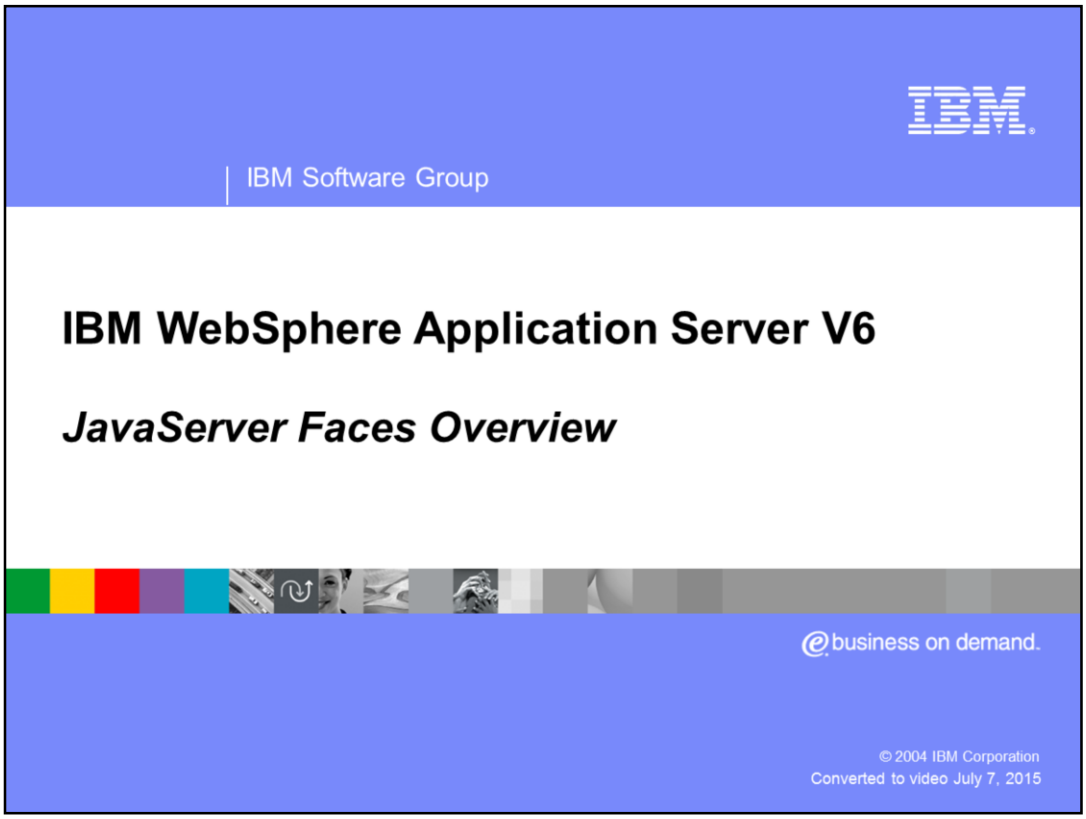
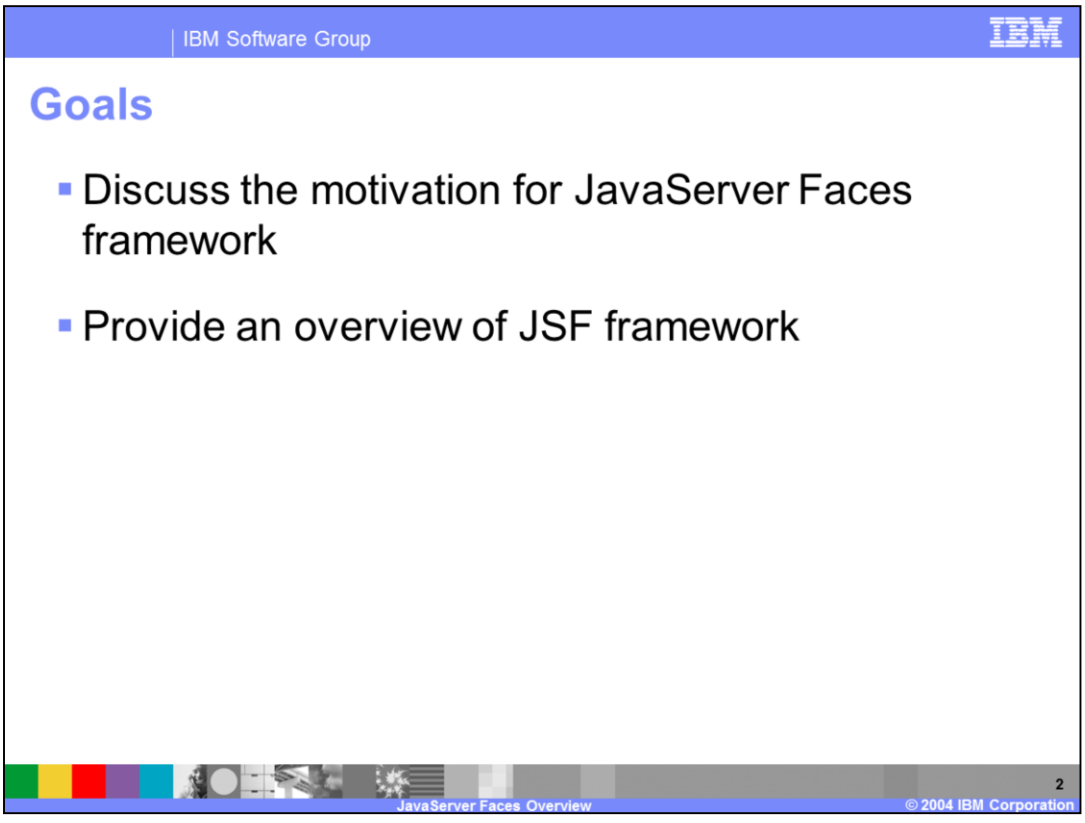
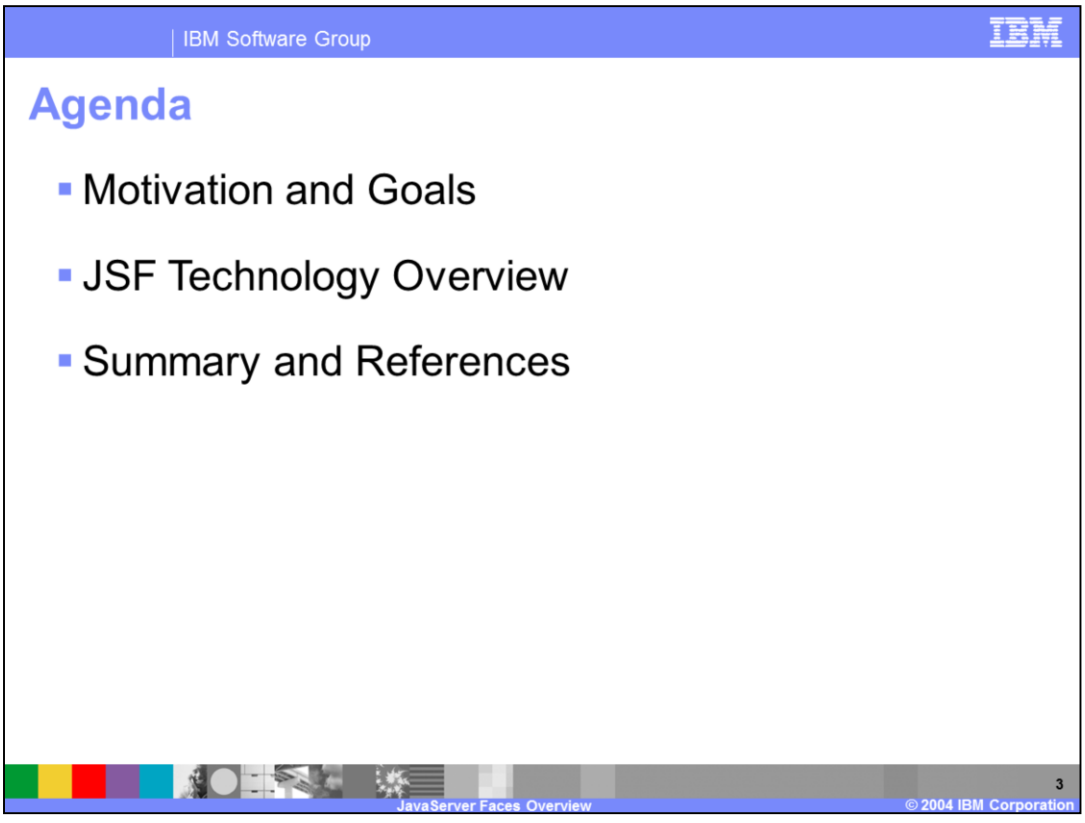


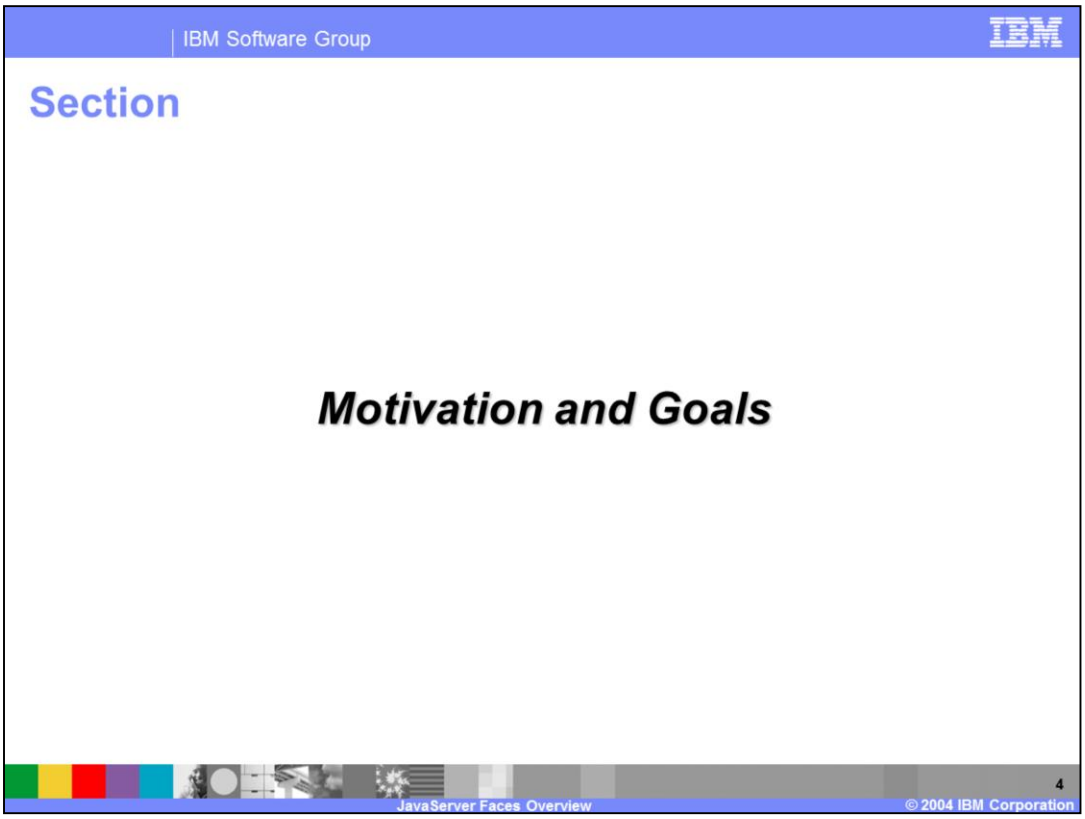
An overview of JavaServer Faces is the focus of this presentation.



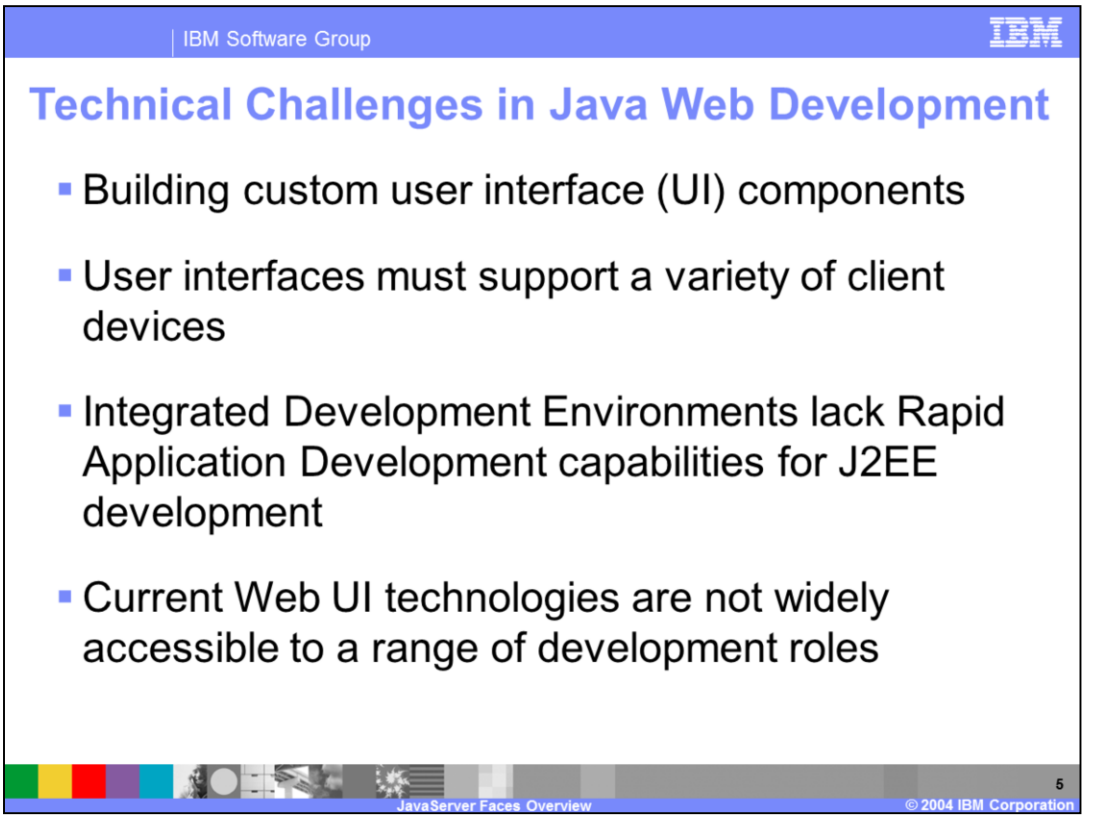
The goals for this presentation are to provide and overview of the JavaServer Faces framework and to provide some insight into the motivation for developing this framework.



This presentation starts with an overview of the motivation and goals of the JSF framework. The second part will focus on providing an overview of the JSF technology.

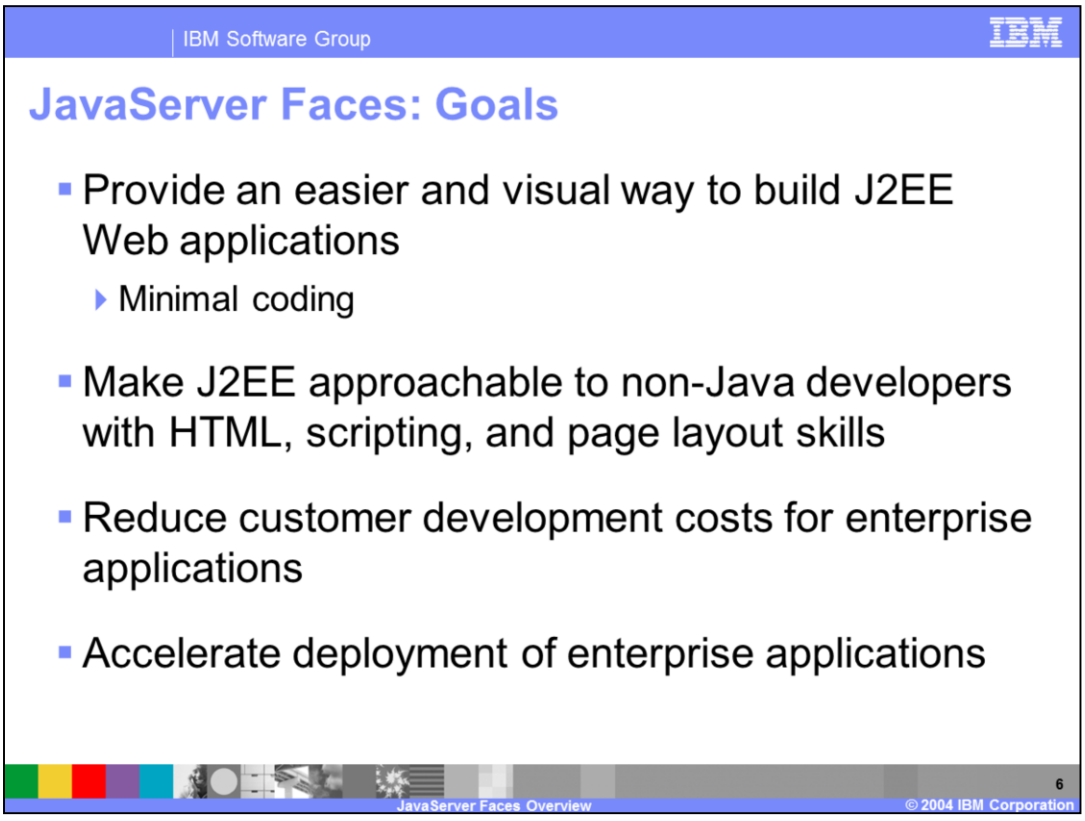


The next section will discuss the motivation for, and the goals of, the JavaServer Faces framework.

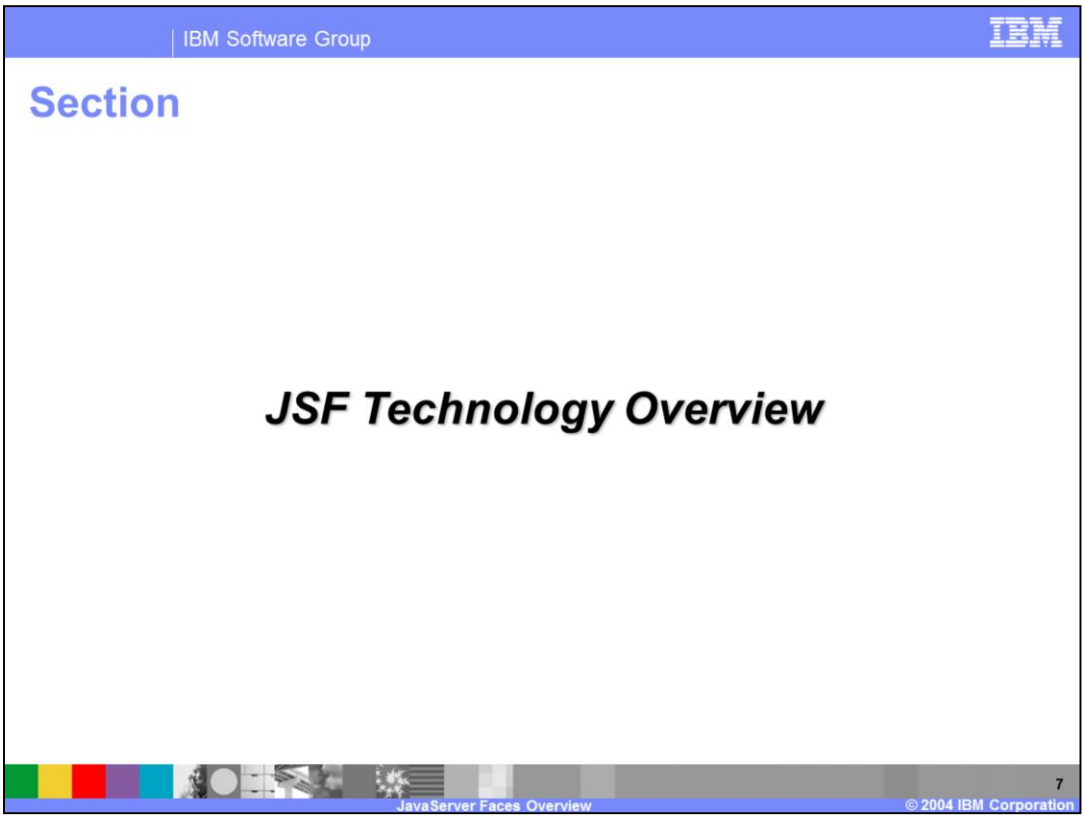


To better understand the motivation behind JavaServer Faces, start by looking at some of the challenges of building a web based application using J2EE technologies. The first challenge is the task of building custom user interface components that are necessary for building any significant web based UI. For example, consider a graphical user interface that uses a tree view, or a tabbed notebook panel. UI components would need to be designed and built to support this type of interface. Furthermore, the design of these components needs to reflect that they must be extensible and reusable to support future growth and maintenance of the product. In the absence of a standard framework for building and reusing custom UI components, each development team ends up duplicating work instead of leveraging pre-existing components. The lack of a standard framework also makes it difficult for Tool providers to build IDEs that provide developers with an easier development experience and make J2EE technologies more accessible to a wide variety of development roles.

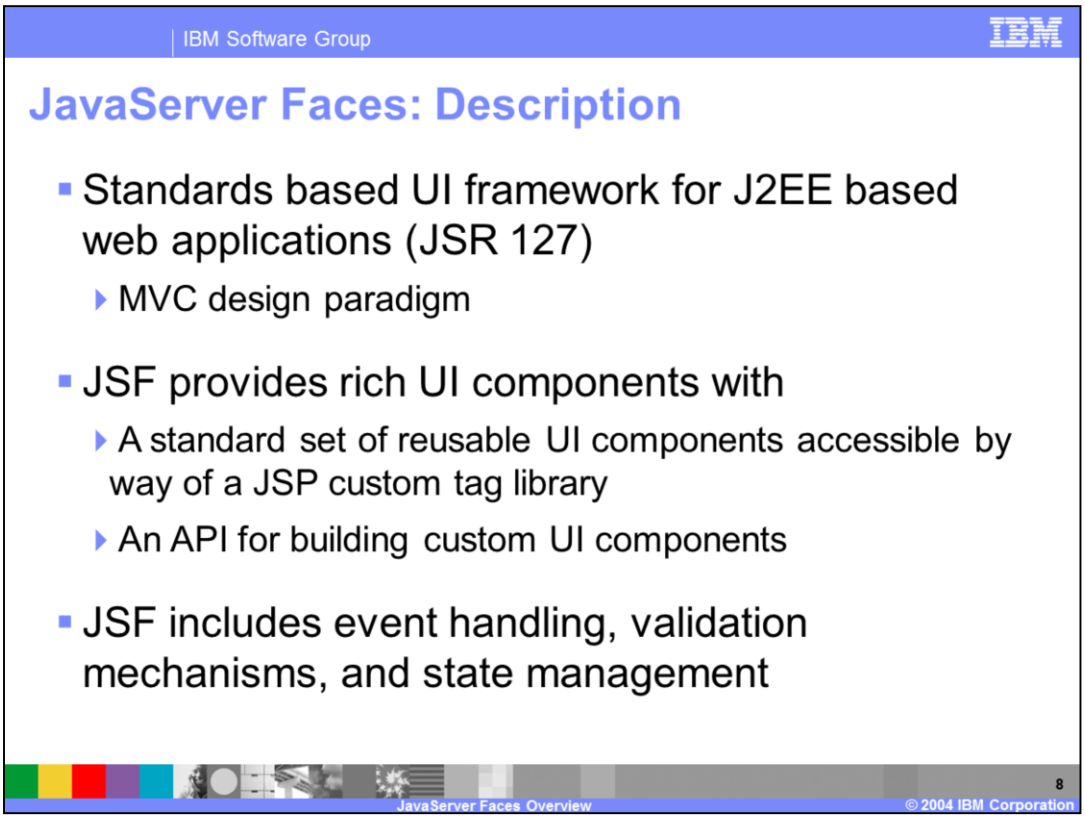
One final challenge is the increased number of client devices that must be supported by web based applications. Building user interfaces that support various client devices is challenging, and will become increasingly so as various client devices become more widely used. Because of this, there is a need to build a framework that will decrease the development cost and resources required to build applications that support various client devices.



JavaServer Faces, or JSF, was developed to ease development challenges discussed on the previous slide. It is frequently the case that web applications are built by teams of developers that are not all advanced J2EE programmers. Typically, these developers are specialized in skills such as page layout, form design, scripting languages, and HTML. An important goal of JSF is to better meet the needs of these developers with an easier and visual way to build web applications with rich user interfaces that leverage the J2EE platform. Along with this goal, the JSF framework also aims to provide the ability to build and deploy applications more quickly, providing a faster time-to-value and lower cost of development for enterprise applications.



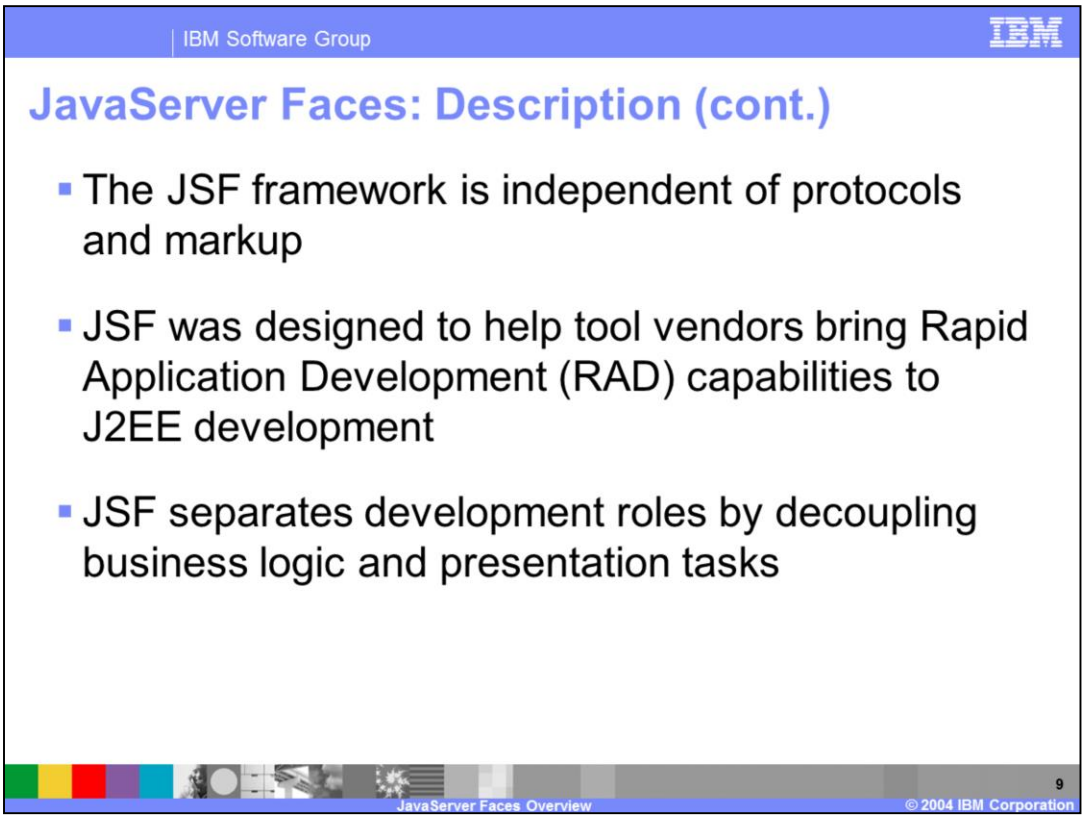
The next section will discuss an overview of the JavaServer Faces technology.



JavaServer Faces (JSF) is a standards-based user interface framework that is used to build J2EE web applications. The JSF framework follows the familiar Model-View-Controller design paradigm. The JSF specification is defined by JSR 127, and can be found on the Sun website; the URL is in the References section at the end of this presentation.

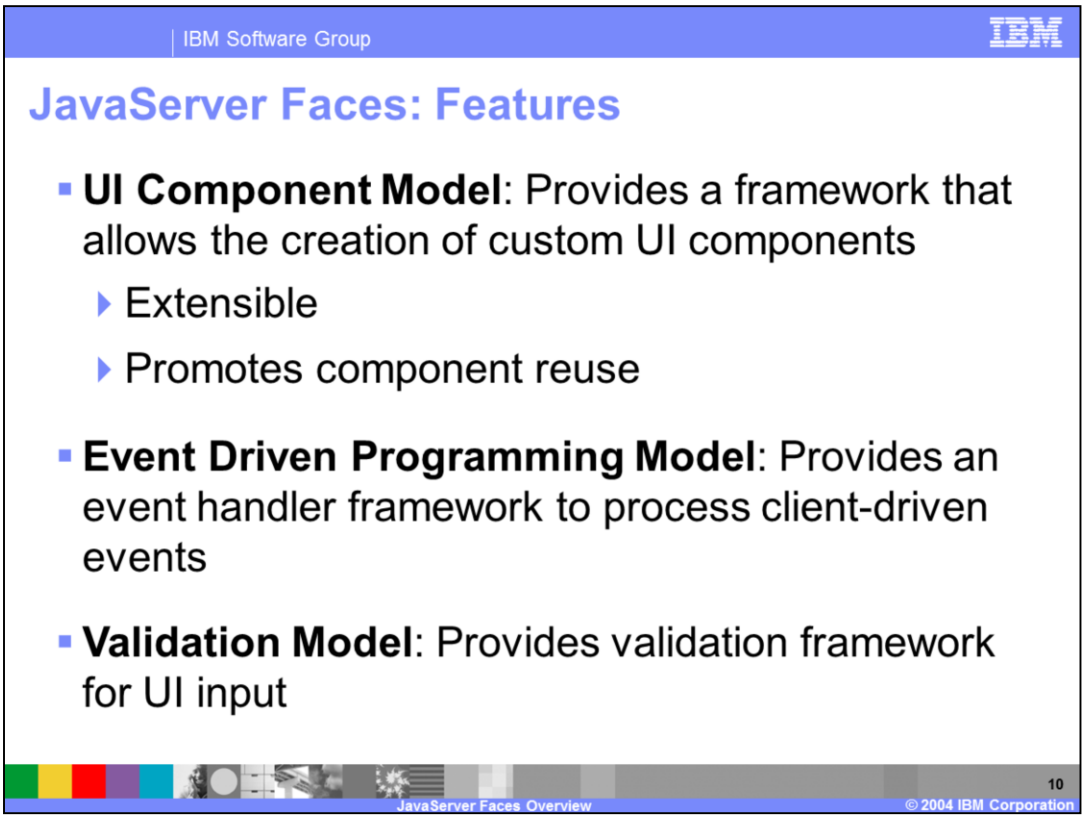
One of the primary features of JSF is its support for a standard set of UI components. This support comes in the form of a custom JSF tag library that includes a set of standard HTML-based UI components, and an API for building custom UI components that can be easily built and reused.

The JSF framework also provides a server side model that enables handling of client events and the validation of user input. The framework also automatically manages the session state in between user requests. This support is often referred to as the JSF runtime.



Another important feature that would be of particular interest to development teams building applications targeted for a variety of client devices is the fact that JSF has been designed to be independent of protocols and markup. Within the JSF architecture, this is done by decoupling the individual component definitions from the way in which they are rendered.

JSF has also been developed to be easily integrated with IDE tools to facilitate Rapid Application Development, or RAD. Because of this, JSF technologies can enable complex web UIs to be built much more easily even for developers who may have little Java or J2EE development experience. The standard UI framework provided by JSF enables tool providers to bring Rapid Application Development capabilities to J2EE. Finally, the reusable and extensible nature of the JSF components make growing and maintaining web based applications more feasible. As a result of this, you will often here JSF spoken of in the context of Rapid Application Development.

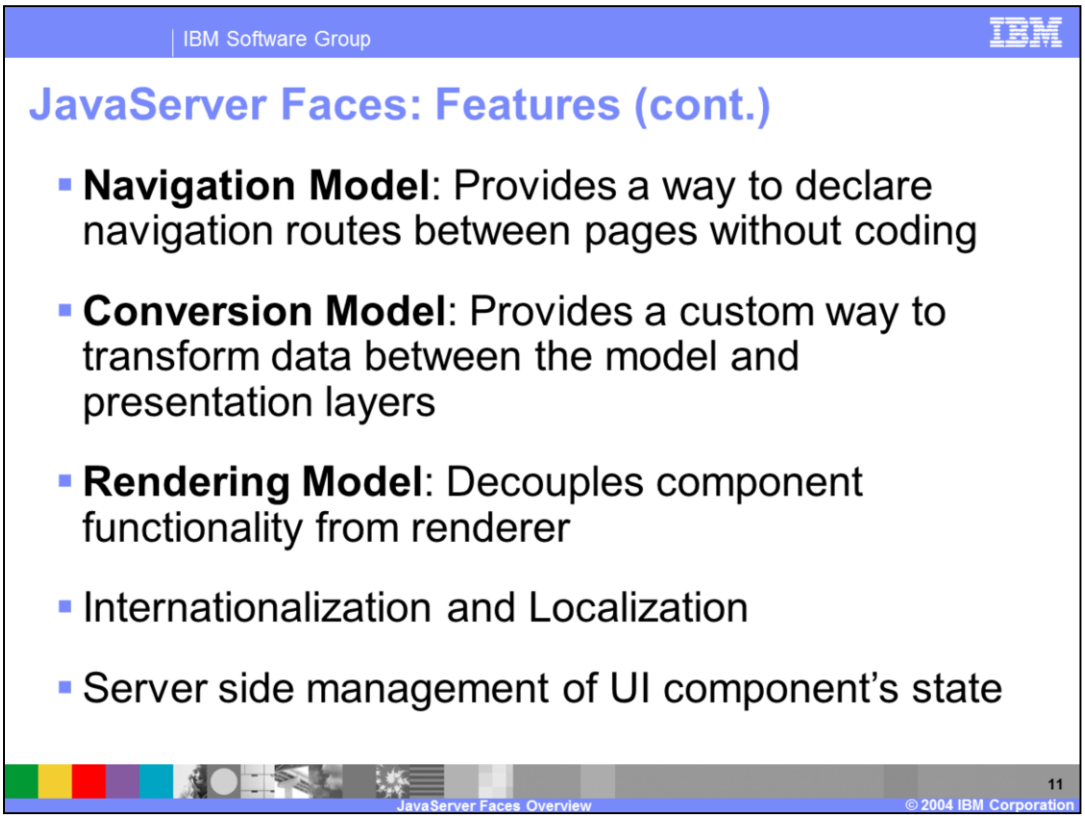


Each JSF feature discussed on this slide and the next addresses a particular development problem that typically needs to be handled when developing even a simple web based GUI application.

The UI Component Model is the framework used to create custom UI components. The significant piece to this model is the Java API that allows developers to build custom UI components. This is an extensible model, that promotes component reuse.

The event driven programming model is a framework that is used to process various client events. An example of a client event might be changing the value of a text field or clicking a submit button on a form. Although there is support for client side event handling, when this event driven programming model is discussed it will primarily focus on server side handling of client events.

The validation model is provided to support server side validation of user input. JSF provides a number of standard validators and an extensible API that can be used to create custom validators.

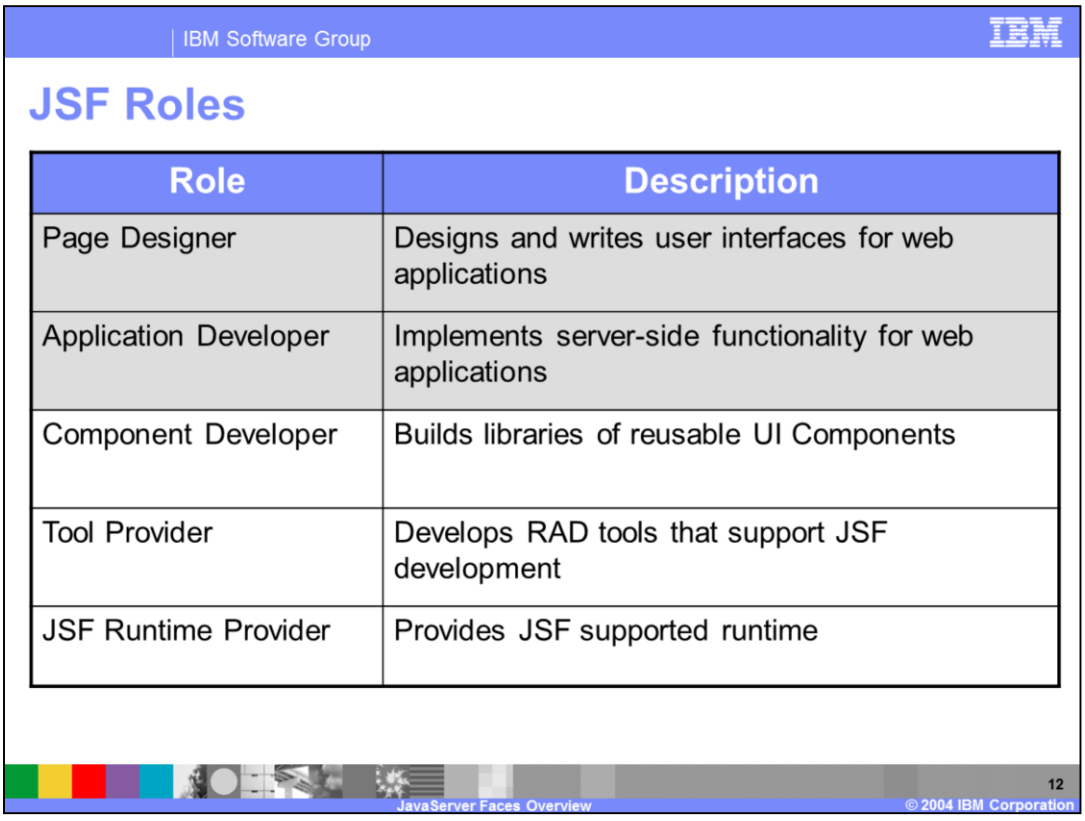


Most web applications are composed of many pages, and one of the challenges of web developers is how to manage page navigation between these resources. The Navigation Model was designed to address this issue. This model provides a way to define navigation routes between pages without the need to write specific code to handle this.

Another common problem that web applications need to solve is converting string data (from text fields) to the underlying business object representation. The conversion model in JSF provides several standard converters, and also provides and API for creating custom converters.

The rendering model provides the framework that is needed to decouple component functionality from the way in which it is rendered.

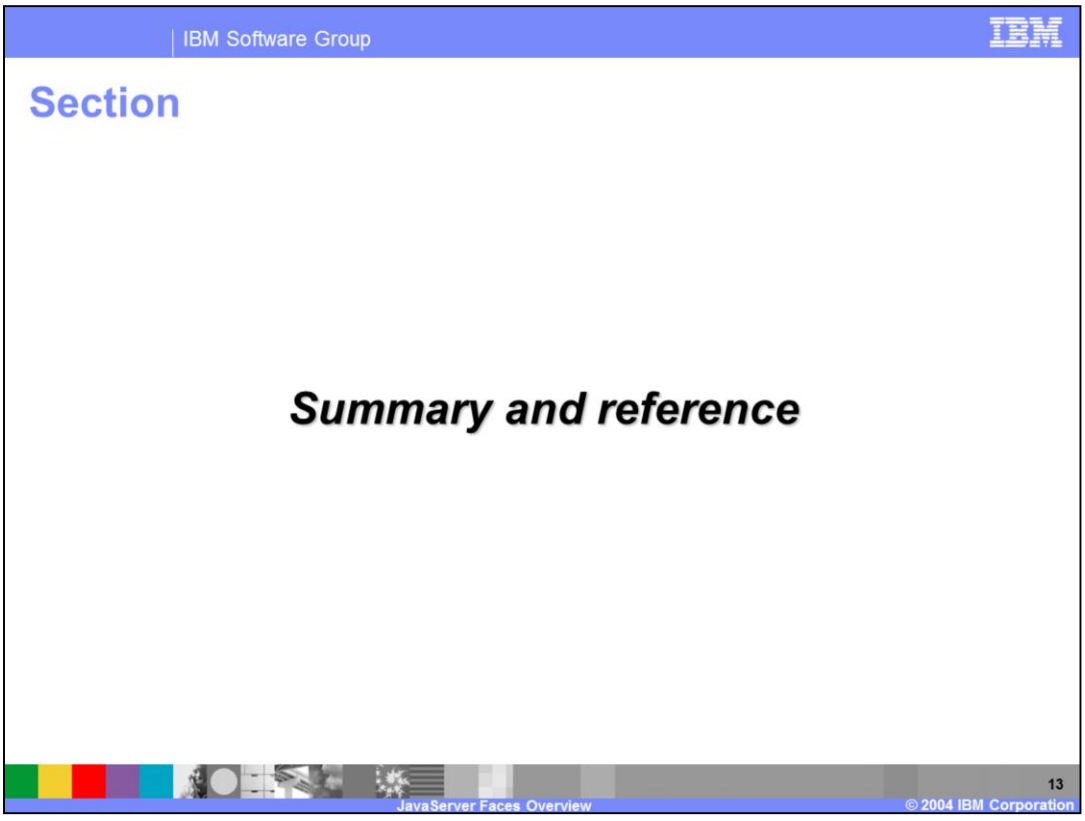
Any non-trivial web application needs to be able to manage UI component state between user requests. The JSF runtime automatically takes care of this state management so there is no need to build your own infrastructure for state management.



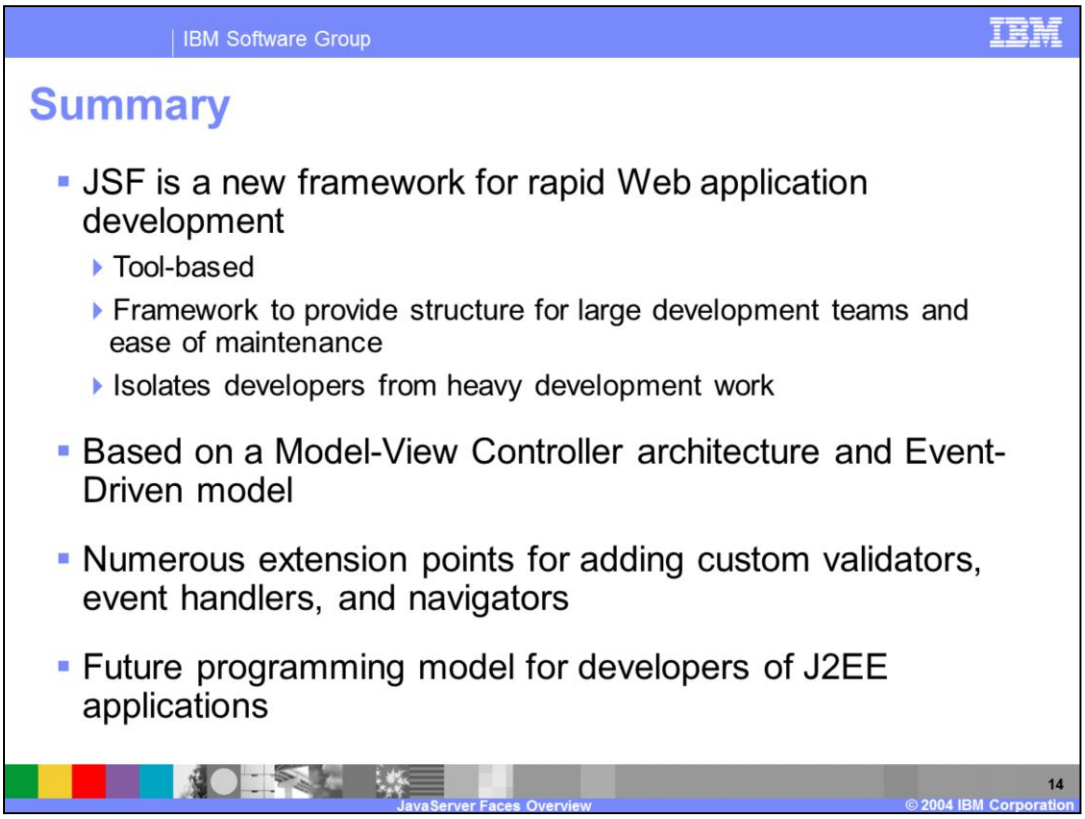
There are several developer roles that would be interested in JSF technologies. However, the largest number of developers interested in JSF fall into the categories of Page Designer and Application Developer.

Page Designers are typically skilled in HTML, JavaScript, and a particular type of rendering technology, like JSPs. With respect to JSF, this group will typically be using custom components that are developed by component developers.

Application Developers are typically skilled in Java and J2EE technologies. The Applications Developer's role when building a JSF application will primarily be in implementing server-side functionality, including custom UI components, custom validators, event handlers, managed beans, and so on.



The next section will provide a summary and references for this presentation.



In summary, this presentation has provided an overview for the JavaServer Faces framework. This technology is intended to be incorporated with tool support to enable rapid application development in J2EE web development.

The JSF framework is based upon a Model-View-Controller based design paradigm, and also includes an event driven model that provides an easy framework to handle events when building web based applications.

The JSF framework is extremely flexible and includes several extension points for adding such things as validators, event handlers, and converters.

References

- JSR 127
  - http://java.sun.com/j2ee/javaserverfaces/
- JSF and Struts
  - http://www-106.ibm.com/developerworks/java/library/j-integrate/