1 Einführung

1.1 Warum Webanwendungen mit SAP?

Das ist eine gute Frage! Ihr Unternehmen hat funktionierende SAP-Anwendungen im Einsatz; das Customizing wurde auf Ihre Geschäftsprozesse angepasst, einige, möglicherweise nicht unbeträchtliche, Modifikationen mussten gemacht werden. Die Anwender mussten im Gebrauch des SAP Frontend (SAP GUI)[1] geschult werden, auf den Arbeitsplätzen war das SAP-Frontend-Programm zu installieren. Nun sind Sie auf einem bestimmten Releasestand produktiv, vielleicht noch auf SAP Anwendungsrelease 4.7, 4.5 oder gar 4.0.

Warum nun Webanwendungen? Und warum sollte ausgerechnet das SAP-System selbst als Web-Server auftreten? Kann es das überhaupt? Wäre der Aufwand, es webfähig zu machen, für das Unternehmen überhaupt finanzierbar?

Die Anforderungen, Geschäftsprozesse über das Internet abzuwickeln, sind kein neumodischer Spleen der IT-Abteilungen, sondern kommen vom Geschäft selbst. Auch nach dem Zusammenbruch der ».com«-Euphorie und der übertriebenen Erwartungen an die »New Economy« und das Internet in den späten neunziger Jahren bleibt genug Substanz, um mit Hasso Plattner von einer »*New* New Economy« zu sprechen. Die durch das Internet geschaffene globale Vernetzung hat zwar nicht, wie manche erwarteten, die klassichen Gesetze der Marktwirtschaft ausgehebelt; dennoch hat die Möglichkeit, weltweit enorme Datenmengen in Sekundenbruchteilen zu übermitteln, für die Wirtschaft den Rang einer Basisinnovation. Das Internet bleibt weiterhin eine zukunftsträchtige Plattform zur Abwicklung von Geschäftsprozessen! Und somit gibt es einen wachsenden Bedarf an Geschäftsanwendungen, die auf dem Internet basieren.

Mag unternehmensintern weiterhin die Benutzeroberfläche des SAP Frontend verwendet werden,[2] auf die die Anwender nun einmal geschult sind – und für

1. SAP GUI ist der SAP-eigene Rich Client, eine proprietäre Benutzeroberfläche des R/3-Systems, die über ein schlankes Protokoll mit dem SAP-System kommuniziert.
2. Auch wenn sich hier mit dem SAP-Portal eine neue, interessante Alternative auftut.

einen SAP Power User sind sie sicher noch für lange Zeit das Richtige – so ist für die Kontakte mit den externen Geschäftspartnern – den Kunden, den Lieferanten, den Logistik-, den Service-Partnern, usf. – das Internet die geeignete Infrastruktur. Der große Vorteil der »kollaborativen Transaktionen«, der Einbindung der Geschäftspartner übers Internet in die eigenen Geschäftsprozesse, liegt in der Zeitnähe und der allgemeinen Verfügbarkeit dieses Netzes, sowohl im geografischen (globale Präsenz, vor allem bei Hinzunahme der drahtlosen Kommunikationsmöglichkeiten) als auch im technischen Sinne (geringe Systemanforderungen).

Wie viel Zeit wird beispielsweise bei einem papierbasierten Bestellwesen verschwendet: Da müssen Kontrakte in Papierform hin- und hergesendet, im Hause durch verschiedene Abteilungen zur Freigabe eingereicht werden und schließlich von beiden Parteien unterschrieben und an den jeweiligen Partner versendet werden. Auch jede einzelne Bestellung muss wieder von Hand angestoßen werden. Wieviel einfacher könnte das Leben sein, wenn der Lieferant mit einem Internet-Zugriff direkt aufs System gehen, die für ihn relevanten Bestelldaten einsehen und seine eigene Unterschrift mittels digitaler Signatur direkt online im SAP-System leisten kann!

Der Lieferant kann – wie andere Geschäftspartner auch, etwa der Kunde – ein kleines oder sehr kleines Unternehmen sein. Das ist für webbasierte Geschäftsanwendungen kein Problem. Denn außer einem PC mit Webbrowser und Internet-Anschluss wird auf der Seite des Geschäftspartners keine weitere Software benötigt: Die gesamte Applikations- wie auch Präsentationslogik wird vom Server übernommen. Dies sichert eine weite Verfügbarkeit übers Internet abgewickelter Geschäftsprozesse.

Aber nicht nur auf der Beschaffungs-, sondern besonders auch auf der Verkaufsseite des Unternehmens spielt das Internet eine wachsende Rolle. Die Zahl der Internetbenutzer wächst ständig, und mehr und mehr Haushalte wie auch Unternehmen informieren sich im Internet über interessante, passende Angebote und nutzen es auch für den Einkauf. Ein Internetauftritt für den Verkauf – anfangs eher zur Imagepflege betrieben – gewinnt zunehmend Bedeutung als Umsatzquelle.

Die Notwendigkeit von Webanwendungen einsehend, bleibt allerdings die Frage: Warum sollte ausgerechnet das SAP-System auch als Web-Server verwendet werden? Sollte man sich nicht darauf beschränken, das SAP-System zur Verbuchung von Anwendungsbelegen zu verwenden und die Webanwendungen mit einem separaten, eigens darauf spezialisierten Server entwickeln?

Selbstverständlich wäre es technisch problemlos möglich, einen vorgelagerten Server einzurichten, der für das Internet zuständig ist und beispielsweise mittels Remote Function Calls mit der Geschäftslogik des SAP-Systems kommuniziert. Aber abgesehen von den Kosten für Anschaffung und Betrieb des zusätzlichen Servers lässt ein externer Web-Server meist den gewohnten Komfort des SAP Software Deployments vermissen. Da gibt es erstmal keine Transportwege, kein

Entwicklungs-, Test- und Produktivsystem, und das Verteilen der eventuell zu ändernden HTML-Ressourcen ist mühsame Handarbeit. Das gilt vor allem, wenn es bei zunehmender Netzlast nicht bei einem Server bleibt, sondern mehrere Server sich mittels Load Balancing die Beantwortung der eingehenden Requests teilen. Von Dingen wie einer Versionsverwaltung ganz zu schweigen, mit der sich die letzten Änderungen des HTML-Codes oder der verwendeten Scripts verfolgen ließen. All dies lässt sich selbstverständlich einrichten, aber es ist nicht – wie bei SAP – out of the box verfügbar, sondern bedeutet zusätzlichen Aufwand.

Daher bietet SAP selbst seit Release 3.1, also bereits seit 1996 (!), einen derartigen Server, den Internet Transaction Server (ITS) an. Die Ressourcen, die auf diesen oder diese Server durch das so genannte Publizieren verteilt werden, sind gewöhnliche Objekte des SAP Repository und können daher wie eine ABAP-Source transportiert werden. Somit können Änderungen an Web-Objekten synchron mit den eventuell dazugehörigen Änderungen an der Business-Logik, an Funktionsbausteinen, Klassen etc. in die nachgeordneten Systeme – Test, Schulung und Produktion – importiert werden. Der Internet Transaction Server verarbeitet so genannte *HTML-Templates*, Fragmente von vollständigen HTML-Seiten, die mit einer eigenen kleinen Programmiersprache namens »Business HTML« angereichert werden können. HTML-Templates können wie eine ABAP-Source in der Workbench bearbeitet und danach auf den ITS publiziert werden.

Im Gegensatz zum ABAP-Dynpro oder -Report hat der Entwickler also bei einer ITS-basierten Webanwendung die *volle Kontrolle über die Präsentationsschicht*: Die am Front-end ausgeführte Logik spielt sich im Browser ab und setzt sich aus HTML, CSS und Java-script zusammen. All diese Bestandteile können vom Entwickler angepasst werden. Das ist gerade für öffentliche Internetauftritte wichtig: Kein Unternehmen will auf seinen Webseiten Werbung für SAP machen oder für jeden Kenner sofort als SAP-Kunde erkennbar sein, indem es die SAP-Layouts und Templates verwendet. Der Internetauftritt ist – genau wie bei einer Privatperson – eine Visitenkarte, die nach individuellen, unternehmensspezifischen Anforderungen gestaltet werden soll. Dennoch ist es natürlich ein Vorteil, dass bereits fertig ausgelieferte Webanwendungen wie der SAP Online Store oder der SAP Retail Store zur Verfügung stehen, so dass nur noch die HTML-Templates gemäß unternehmensspezifischen Anforderungen angepasst werden müssen.

Ein weiterer großer Vorteil ist, dass der Internet Transaction Server beginnend mit Basisrelease 6.40 in das SAP-System eingezogen ist. Ab diesem Releasestand entfällt also die Notwendigkeit eines zusätzlichen Internet Transaction Servers: Die gesamte ITS-Logik wird unmittelbar auf den Applikationsservern, ausgeführt. Wenn Sie also noch auf einem recht niedrigen Releasestand (bis 4.6) sind und daher innerhalb des SAP-Systems keine andere Wahl haben, als ITS-basierte Webanwendungen zu schreiben, so bleibt Ihnen diese Lösung bei einem zukünftigen Releasewechsel nicht nur erhalten, sondern wird sogar kostengünstiger, da Ihre ITS-Serverfarm entfallen kann.

Der Internet Transaction Server fällt, wenn man die genannten Vorzüge bedenkt, mit Sicherheit kostengünstiger aus als die Alternative, ein vollständig selbst gebauter externer Web-Server, der via Remote Function Calls mit dem SAP-System kommuniziert. Wir werden jedoch sehen, warum es im Allgemeinen noch größere Vorteile bietet, mit Business Server Pages zu arbeiten.

1.2 Warum Business Server Pages?

Bei all den beschriebenen Vorteilen des Internet Transaction Servers im Vergleich zu anderen Lösungen mit eigenen Web-Servern könnte man meinen, mit dem ITS sei nun das Thema webbasierter Applikationen ein für allemal erledigt. Für ältere Releasestände, bis Anwendungsrelease 4.6, drängt sich dieses Thema auf, da es bis zu diesem Release von SAP keine Alternative gab.

Aber der ITS hat – bei allen erwähnten Vorteilen – auch einige Nachteile. Dass die Web-Serverfunktionalität an ein eigenes System delegiert wird, ist ein Moment der Inflexibilität. Das SAP-System sollte *selbst* als HTTP-Server (und ggf. auch als HTTP-Client) fungieren und in der Lage sein, alle nötigen Schritte vom Eingang des HTTP-Requests bis zum Zurücksenden der Antwort zu übernehmen.

Hierzu bedurfte es einer neuen Komponente innerhalb des SAP-Systems, des *Internet Communication Managers (ICM)*. Mit Einführung des ICM zum Basisrelease 6.10 wurde das SAP-System zu einem vollwertigen Webserver. Ein separater Server für die HTTP-Kommunikation ist schlicht unnötig geworden. Webanwendungen sind seitdem als Business Server Pages vollständig in das SAP-System integriert.

Dadurch ist es möglich geworden, die Präsentations-, Applikations- und Steuerungslogik (View, Model und Controller) auf klar definierten Kanälen innerhalb der ABAP-Welt zu implementieren und zu koordinieren.

Ein View ist ein Fragment der HTML- (oder XML-) Antwort. In Business Server Pages ist er als ABAP-Objekt realisiert, das von einem anderen ABAP-Objekt, einem Controller, instanziiert und aufgerufen wird und seine Informationen über den aktuellen Anwendungszustand aus weiteren ABAP-Objekten, den Models erhält.

Präsentationslogik in den Views (Scripting) kann entweder in ABAP oder mit serverseitigem JavaScript realisiert werden. Komplexere Präsentationslogik kann mittels *Tag Libraries*, die in der BSP-Welt Extensions heißen, in spezielle Klassen ausgelagert werden. Tag Libraries stellen einen gewaltigen Schritt in Richtung Wiederverwendbarkeit dar – und helfen darüberhinaus bei der Gestaltung eines applikationsweit einheitlichen Look & Feel. Komplexere Präsentationslogik kann in Hilfsklassen implementiert werden, die wir als *Kontextklassen* bezeichnen, und die dem View in Form eines Seitenattributs zur Verfügung gestellt werden.

Mit all diesen Features sind Business Server Pages das ABAP-Äquivalent zu Java Server Pages. Sie stellen dem ABAP-Entwickler ein modernes technisches Framework dynamischer Webseitengestaltung zur Verfügung.

1.3 Web Dynpro vs. offenes UI

Für SAP bilden die Business Server Pages noch nicht den Schlusspunkt in der Reihe der Entwicklungswerkzeuge für webfähige Anwendungen. SAP hat weitere große Pläne. Für einen Entwickler von »Standardsoftware« oder »Best Practices«, wie es heute etwas bescheidener heißt, steht nun eine Standardisierung des User Interface auf der Agenda. Wie man im klassischen ABAP-Dynpro mit dem Screen Painter eine Benutzeroberfläche erstellen konnte und dem SAP GUI dessen Darstellung überließ, so kann man ab Basisrelease 7.0 mit dem *Web Dynpro* aus einer Palette standardisierter UI-Elemente Oberflächen erstellen und der Web Dynpro Runtime das HTML-Rendering überlassen.

Für manche ist das eine gute Nachricht: Man muss kein HTML kennen, um Web Dynpros zu erstellen. Die Definition eines Web Dynpro kann in der ABAP Workbench erfolgen und führt zu einer Beschreibung der Oberfläche in Form von XML-Metadaten in einem SAP-proprietären, nichtöffentlichen Format. Die Runtime (sowohl im Java- als auch im ABAP-Stack) übersetzt diese Metadaten in HTML- und JavaScript-Code und spielt diesen in die HTTP-Antwort ein.

Aber selbst wenn er wollte, *könnte* ein Entwickler den vom Web Dynpro erzeugten HTML-Code auch nicht mehr verändern. Das ist beabsichtigt. SAP-Anwendungen sollen ein einheitliches Aussehen haben, und das lässt sich in den Augen der Web-Dynpro-Entwicklung nur dadurch erreichen, dass die SAP-Basis die Hoheit über fast den gesamten HTML- und JavaScript-Code hat, der an den HTTP-Client gesendet wird.[3] Die vielfältigen Möglichkeiten, die ein Webdesigner zur Gestaltung von HTML-Seiten kennt, werden dem Entwickler, ob er will oder nicht, abgenommen, und an ein Team innerhalb der SAP delegiert.[4]

Das mag für die von SAP ausgelieferten Anwendungen, die »Best Practices« das richtige Vorgehen sein, damit nicht jede SAP-Anwendung anders aussieht. Anwendungsentwickler sollen sich darauf konzentrieren, die Geschäftslogik für ihre Prozesse zu realisieren. Die Feinheiten der Oberfläche überlassen sie besser der Basis oder ihrem UI-Team, das um ein SAP-weit einheitliches Look & Feel bemüht ist.

---

3. Mit CSS gibt es immerhin auch im Web Dynpro noch einen eingeschränkten Rahmen der Gestaltung. Man darf mal eine Hintergrundfarbe oder Schriftart ändern. Im Übrigen ist man aber auf das von SAP vorgedachte Look & Feel eingeschränkt.
4. Dem Aussehen der Oberflächenelemente nach dürfte es sich übrigens um dasselbe Team handeln, das auch in der BSP-Welt bereits die SAP-eigene »HTMLB« Tag Library entwickelt hatte.

Beim *Kunden* und bei kundeneigenen Applikationen aber sieht die Sache anders aus. Der Internetauftritt eines Unternehmens ist eine sehr individuelle Angelegenheit. Um ein mit der eigenen Corporate Identity konformes Design zu erreichen, kommt man nicht damit aus, in einer bestehenden SAP-Anwendung ein paar Bildchen auszutauschen und in einem vorgegebenen Stylesheet-Konzept Schriftfarben und -größen zu ändern. Das Applikations-Framework muss dem Webdesigner vielmehr die vollständige Kontrolle des User Interface ermöglichen. Das bedeutet, alle an den Client gesandten Ressourcen, der HTML-Quelltext ebenso wie das CSS, Applets, JavaScript etc. müssen durch die Anwendungsentwicklung steuerbar sein. Für Anwendungen dieser Art ist das Web Dynpro nicht geeignet. Business Server Pages bleiben hierfür das richtige Framework.

Entgegen seinem Namen bedeutet das Web Dynpro einen *Abschied von der klassischen Webanwendung*. Durch die Einführung der XML-Metadaten macht sich SAP unabhängig von einer konkreten Technologie, mit der das User Interface dargestellt wird. Mittlerweile gibt es verschiedene Rendering Engines für das Web Dynpro: Das Web Dynpro lässt sich nicht nur in einem Browser darstellen – wahlweise in HTML oder in Flash – sondern auch mit einem eigenen Web Dynpro Client, unabhängig vom Browser. *Webanwendungen im klassischen Sinne stellen für SAP nur eine kurze Ära dar*, ein Übergangsglied auf dem Weg vom Dynpro zum Web Dynpro.

Die Vereinheitlichung der Benutzerschnittstelle steht für die vielen, von SAP selbst entwickelten Applikationen natürlich im Vordergrund, und sie erreicht dieses Ziel, indem sie dem Anwendungsentwickler die Präsentationsschicht wegnimmt und ihn auf ein grafisches Tool verweist, mit dem er nunmehr seine Benutzeroberflächen nach klar definierten Spielregeln »malen« darf. Dieses Vorgehen ist für ein Unternehmen, das sich einmal zum Ziel gesetzt hat, Standardsoftware zu entwickeln, durchaus sinnvoll. Ob aber seine Kunden oder andere Softwareanbieter es übernehmen sollten, ist eine andere Frage.

In diesem Buch soll es nicht um das *Web Dynpro*, sondern um *Webanwendungen* gehen; zu deren Wesen gehört es, dass auch die Präsentationsschicht dem Entwickler für Anpassungen und Erweiterungen zur freien Gestaltung offensteht. Die ideale Umgebung für Webanwendungen im SAP-Bereich stellen die Business Server Pages dar.

1.4 Über dieses Buch

Dieses Buch, aus der Praxis für die Praxis geschrieben, richtet sich vor allem an Entwickler und Berater, die mit dem Thema Webanwendungen zu tun haben und die Möglichkeiten kennenlernen wollen, mit der ABAP-Entwicklungsumgebung und der BSP-Laufzeit Anwendungen auf Basis des SAP Web Application Servers zu erstellen. Wenn Sie dieses Buch gelesen haben, können Sie auf ein BSP-Frame-

work zurückgreifen, das in zahlreichen Entwicklungen gewachsen ist. Es kann Ihnen helfen, selbst stabile und erweiterbare Anwendungen getreu dem Model-View-Controller-Architekturmuster zu entwickeln.

Der *Retail Store*, den ich Ihnen im nächsten Kapitel kurz vorstelle, war Ausgangspunkt des BSP-Frameworks, auf dem dieses Buch aufbaut. Der Retail Store ist eine Sammlung von Anwendungen, deren Geschäftslogik durch BAPIs und APIs des SAP-Standards abgewickelt wird. Die Retail-Store-Anwendung, die auf dieser Geschäftslogik aufsetzt, ist eine MIGROS-Eigenlösung, deren visuelle Oberflächenelemente (Tabellen, Drucktasten) mit dem Retail Store des SAP-Standards identisch sind. Die Anwendungen laufen jedoch – und das ist das Besondere – direkt über den Web Application Server; es wird nicht wie im SAP-Standard ein vorgelagerter Internet Transaction Server benötigt. Darüberhinaus sind fast alle Anwendungen – hochkomplexe Anwendungen wie den SD-Kundenauftrag eingeschlossen – *zustandslos* implementiert: Es werden keine Sitzungsdaten im Hauptspeicher vorgehalten; stattdessen wird mit serverseitigen Cookies gearbeitet. Die Performance kann sich sehen lassen und zeigt, dass dieses Konzept tragfähig ist.

Im dritten Kapitel geht es um bewährte *Programmierparadigmen*. Diese gewähren hilfreiche Orientierung, um *wartbare, robuste und erweiterbare* Programme zu produzieren. Sie gelten sinngemäß auch für andere Programmiersprachen. Die Features von ABAP Objects helfen in besonderer Weise bei der Verwirklichung der Wiederverwendbarkeit, der Modularisierung und der Entkopplung. In diesem Kapitel stelle ich Ihnen auch eine *Objektfabrik* vor. Fabriken dienen dazu, die Definition von Objekten (in Schnittstellen oder abstrakten Klassen) vollständig von der Implementierung zu trennen.

Model-View-Controller (MVC) ist ein Architekturmuster, das für alle Anwendungen mit Benutzerschnittstelle mit Gewinn eingesetzt werden kann. Es unterstützt die Einhaltung der erwähnten Programmierparadigmen, indem es die Software in drei Schichten trennt: Zwei dieser Schichten, View und Controller, dienen der Anbindung der Benutzerschnittstelle – der große Rest, Model genannt, enthält die eigentliche Anwendungslogik. Im Kapitel *Anwendungen mit Benutzerschnittstelle* bringe ich Ihnen dieses Konzept in seinen Grundzügen nahe.

Bevor Sie das MVC-Architekturmuster auf den konkreten Fall von Web-Applikationen anwenden, erläutere ich Ihnen im fünften Kapitel die wichtigsten Eigenschaften des Internet Communication Frameworks (ICF), auf dem auch die Business Server Pages aufbauen. Sie werden die Möglichkeiten kennenlernen, mittels eines *Requestbehandlers* das SAP-System als HTTP-Server einzusetzen, Services in den ICF-Servicebaum einzuhängen und Aliaspfade zu definieren. Sie lernen auch das Minimum, das Sie über Anmeldungen wissen müssen. Schließlich schauen wir uns noch das MIME Repository und den BSP-Handler etwas genauer an.

Nun haben Sie das notwendige Handwerkszeug, um das MVC-Konzept für Webanwendungen zu erschließen. Ich stelle Ihnen ab dem 6. Kapitel ein Framework für BSP-Anwendungen vor, das Sie Schritt für Schritt kennenlernen werden – von der obligaten *Hallo Welt!*-Anwendung über den Begriff des *Panels* und der *Flow Logic*, den im 7. Kapitel erläuterten Mechanismus der Datenbindung bis zu einem komplexen, realistischen Beispiel: Im 8. Kapitel erarbeiten wir eine Anwendung zum Anlegen, Ändern und Verwalten von SAP-Servicemeldungen.

In vertiefenden Kapiteln geht es um die Themen

- *Sitzungsdaten* – die verschiedenen Möglichkeiten, in einer zustandslosen Webanwendung Kontext aufzubewahren. Unser MVC-Framework arbeitet zu diesem Zweck mit einer eigenen Klasse `zcl_mvc_sessiondata`.
- *Message Handler* – Bei der Trennung der Benutzerschnittstelle von der Anwendungslogik bekommt er eine eminente Wichtigkeit: Meldungen können nicht dort *ausgegeben* werden, wo sie *auftreten*. Die Klasse `zcl_messages`, Komponente unseres MVC-Frameworks, wird diskutiert.
- Ausführlich möchte ich Ihnen dann *Tag Libraries (*auch bekannt als *BSP-Extensions)* vorstellen – also die Möglichkeit, wiederkehrende Präsentationslogik in Form von selbstdefinierten Elementen zu kapseln. Ich erläutere Ihnen auch den Pool von UI-Elementen, wie sie bei der Realisierung des Retail Store verwendet wurden. Ich zeige Ihnen anhand ihrer Funktionen auch die Schwierigkeiten, die es bei der Implementierung komplexerer Elemente wie eines Eingabe- oder Tabellentags zu beachten gibt.
- *Implementierungsdetails* des MVC-Frameworks. Möglicherweise wurde Ihre Neugier durch die Beispiele geweckt. Das Kapitel *Das MVC Framework im Detail* diskutiert Besonderheiten des Entwurfs und der Implementierung.
- *Texte*. – Um sie übersetzungsrelevant zu gestalten, sollte man Ablagen wie das Online Text Repository (OTR) verwenden. Die Klasse `zcl_bsp_util` bietet weitere Möglichkeiten, sich aus dem reichen Textpool des SAP-Standards zu bedienen.
- *Clientseitige Logik* – Hierzu gehören nicht nur Abläufe wie Eingabeprüfungen, Cursorsteuerung und Hilfefenster, sondern auch – und das ist ein besonders vielversprechendes Thema – das clientseitige Rendern mit XSLT. An einem detaillierten Beispiel zeige ich Ihnen, wie man es einrichten kann, dass eine Applikation den Client nur noch über ihre dynamischen Anteile informiert – während die statischen Teile in Form einer XSLT-Transformation im Puffer eingelagert werden. Neben einer Reduktion der Netzlast liegt ein Vorteil dieses Ansatzes in einer noch klareren Trennung von User Interface und Anwendungsmodell. Sie müssen hierbei nicht auf Tag Libraries verzichten: Eine Extension ZX hilft Ihnen, Ihre UI-Elemente so zu gebrauchen, wie Sie es von einer »klassischen« BSP-Anwendung gewohnt sind.

Im Kapitel *Andere Requestbehandler* zeige ich Ihnen die Möglichkeit, XML-basierte HTTP-Requests zu schreiben. Um möglichst allgemeingültig zu bleiben, fiel mir als Beispiel nichts Besseres ein als ein XML-basierter Data Browser – eine SE16 via HTTP. Sie zeigt jedoch die wesentlichen Techniken und Mechanismen, die Sie zur Verarbeitung von XML für ein- und ausgehendes HTTP verwenden können.

Nicht nur als Server ist das SAP-System zu gebrauchen: In seiner umgekehrten Rolle als *HTTP-Client* erschließt sich ihm die weite Welt des Internets: Wo immer draußen im Web ein nützlicher Service existiert, lässt er sich mit wenigen Handgriffen auch für SAP-Anwendungen dienstbar machen. Zum Basisrelease 6.40 wurde das Vorgehen zum Einbinden von Web-Services stark vereinfacht. Aber auch in früheren Releaseständen ist es kein Hexenwerk. Um zu sehen, wie Web-Services eingebunden werden, werden wir einen Primzahlservice aus dem Web aufrufen. Ein in ABAP programmierter *RSS Newsreader*, der beliebige Feeds einlesen und in ABAP darstellen kann, zeigt Ihnen, wie einfach es ist, einen HTTP-Client durch ein koordiniertes Zusammenspiel von ABAP und XSLT zu realisieren.

Mit »*Ajax im Einsatz*« lernen Sie eine weitere aktuelle, XML-basierte Technik für Webanwendungen kennen. In diesem Kapitel entwickele ich mit Ihnen zusammen Schritt für Schritt eine Ajax-Anwendung zur Verwaltung von Job-Attributen. Wir gelangen schließlich zu einer aus *Serversicht* statischen Weboberfläche, die aber auf dem Client hochdynamisch ist: Die Präsentationslogik wird mit XSLT und JavaScript im Browser realisiert. Das `XMLHttpRequest`-Objekt ruft für die Anwendungslogik XML-basierte Services auf. Das sind Techniken, die Sie kennen sollten: auch wenn Sie aus Ajax kein Dogma machen müssen, bereichern Ajax-Techniken Ihre Webanwendungen.

Damit ist der Reigen der Web-bezogenen Artikel beendet. Es folgen noch zwei wiederverwendbare Softwareteile, die Sie sicher in Ihren nächsten ABAP-Entwicklungen mit Vorteil verwenden können. Sie sind vom MVC-Framework völlig unabhängig:

- Eine Klasse `ZCL_TABLE_HANDLER`, die das Problem der Alt-/Neustandsverwaltung von Daten in Pflegetransaktionen ein für allemal löst.
- Eine Klasse `ZCL_PARAM` zur Pflege von applikationsweiten Parametern.

Unter *http://www.mits.ch/bsp* können Sie sich den Quelltext des Frameworks sowie sämtliche Beispiele dieses Buches ansehen. Sie haben dort auch die Möglichkeit, einen Transport mit dem Framework direkt in Ihr SAP-System einzuspielen.

1.5 Danksagungen

Ich bin meiner Frau Katinka Penert und meinen Kindern David und Clara zu tiefem Dank verpflichtet, dass sie mir für dieses Buchprojekt so viel Zeit geschenkt haben.

Ein großes Dankeschön geht auch an meine Vorgesetzten Oliver Jung, Erwin Moser und Marcel Schaniel, dafür, dass sie das Vorhaben auch im Namen der MIGROS ohne Wenn und Aber unterstützt haben.

Meinem Verleger Michael Barabas danke ich für die gute und unkomplizierte Zusammenarbeit!